CREXDATA

Critical Action Planning over Extreme-Scale Data

D5.2 – Final Report on Explainable AI, Visual Analytics and Augmented Reality

Version 1.0

Documentation Information

Contract Number	**101092749**
Project Website	https://crexdata.eu/
Contractual Deadline	M36, December.2025
Dissemination Level	PU - Public
Nature	R - Report
Author	Gennady Andrienko (FR), Natalia Andrienko (FR), Denise Botrini (CNR), Eleonora Cappuccio (CNR), Despoina Dimelli (TUC), Bahavathy Kathirgamanathan (FR), Georgios Lamprinakis (TUC), Katerina Mania (TUC), Jan Pfeifer (UPB), Jens Pottebaum (UPB), Salvatore Rinzivillo (CNR), Ioannis Safranoglou (TUC), Alexios Stavroulakis (TUC)
Contributors	
Reviewers	Miguel Ponce de Leon (BSC), Arnau Montagud (BSC), Isabel Martínez (BSC)
Keywords	Explainable AI, Visual Analytics, Uncertainty, Augmented Reality

CREXDATA has received funding from the European Union's Horizon Europe programme under grant agreement number 101092749.

Change Log

Version	Author	Date	Description Change
V0.1	S. Rinzivillo	20-10-2025	Initial ToC
V0.2	S. Rinzivillo	20-10-2025	Importing content from previous deliverable
V0.3	Alexios Stavroulakis	10-11-2025	Added AR section
V0.4	Natalia Andrienko	15-11-2025	Revision and contribution to VA sections
V0.5	S. Rinzivillo	23-11-2025	Final polish and integration
V0.6	Natalia Andrienko	08-12-2025	Edits after internal review
V0.9	S. Rinzivillo	15-12-2025	Final version for coordinator
V1.0	A. Deligiannakis	24-12-2025	Approved by the Coordinator

Contents

List of Figures

Executive Summary

This deliverable reports on the research activities conducted within Work Package 5 (WP5) of the CREXDATA project, focusing on the integration of Explainable AI (XAI), Visual Analytics (VA), and Augmented Reality (AR) to enhance human understanding of complex AI models and support decision-making under uncertainty. Building upon the initial findings presented in Deliverable D5.1, this report presents the final outcomes achieved by Month 36, emphasizing methodological advances, tool development, and their application across CREXDATA use cases.

The central objective of WP5 has been to develop human-centered approaches that bridge the gap between black-box AI models and end-users. Our work is grounded in the recognition that model interpretability requires not only transparent algorithms but also alignment with domain knowledge, human cognitive capacities, and situated decision contexts.

Key contributions include: (1) Development of domain-aware local rule-based explanations through extensions to the LORE (LOcal Rule-based Explanation) framework, incorporating custom neighborhood generators that respect expert knowledge and domain constraints; (2) Visual analytics techniques supporting interactive model creation, feature engineering, and explanation exploration, enabling experts to embed domain knowledge throughout the ML pipeline; (3) A conceptual framework for representing and visualizing uncertainty, facilitating communication between data-driven models and human mental models; (4) Augmented reality prototypes demonstrating how immersive visualization can support spatial-temporal data exploration and uncertainty-aware decision-making; (5) a methodology for elicitation of expert knowledge in a conceptual model via LLM-based interactions.

These advances have been validated and refined through application to diverse CREXDATA use cases spanning health crisis management, maritime surveillance, and firefighter safety monitoring. Close collaboration with WP2 ensured alignment with user requirements, while integration with WP3 enabled deployment within the CREXDATA platform. Coordination with WP4 ensured coherence between model training and explanation techniques. The deliverable demonstrates that effective AI explainability emerges from the synergistic combination of algorithmic methods, interactive visualization, and domain-grounded design principles.

The document is structured as follows:

- Section 2 presents the research activities on Explainable AI, with a focus on enhancing exisiting post-hoc explanation methods to incorporate domain knowledge with custom neighborhood generators.
- Section 3 gives an overview of the Visual Analytics methods developed to support the exploration of complex domains and models thorught interactive visualizations of rules and their relationships with the data.
- Section 4 discusses the strategies to exploit Visual Analytics to support the analysis of uncertainty in the models and to extract the knowledge of the expert to be incorporated in the models.
- Section 5 presents the advances on Augmented Reality, addressing the specific challenges emerged from the use cases, with the support for uncertainty analysis on the

models.

- Section 6 presents the advances for the conceptual modeling of complex and uncertain domains, with a focus on the elicitation of expert knowledge through LLM-based interactions.

1 Introduction

Understanding of computer models by users has critical importance for their adoption for practical use. The research in eXplainable AI (XAI) develops algorithmic approaches intended to help users understand model decisions. However, these approaches alone are insufficient for two main reasons. The first issue is the cognitive mismatch between algorithmically generated explanations and human mental models and logic. A model is truly explained if a domain expert accepts it based on both empirical evidence of satisfactory accuracy and compliance with the domain knowledge and reasoning. The second issue is ignorance of human cognitive capacities. While certain types of models, which are often used as mimic models to represent the work of "black boxes", are deemed "transparent" or "interpretable" by design, they may be incomprehensible in practice due to their size and complexity.

In CREXDATA WP5 we strive to address these problems by developing human-centred approaches to creation and explanation of ML models. This includes

- supporting interactive exploration of model recommendations in a dialog with an XAI system,
- employing visual analytics techniques for creating human-intelligible models,
- involving easily perceivable and interpretable visual representations in explanations,
- designing interactive visualizations to facilitate understanding of model prediction uncertainties.

Quoting the CREXDATA project proposal, we address the MO3.2 objective:"**Visual analytics coupled with XAI for understanding complexity and reasoning under uncertainty**. Helping users to comprehend the space of possible situations, future worlds and choose a reasonable decision strategy. Giving user access to the internal rationale of AI models, enabling a Human-Machine conversation to improve the quality of the decision process, leveraging a data-driven approach to exploit expert knowledge."

As a general result of the activities we have designed a pipeline (synthetized in Figure 1) to integrate the human expert knowledge in ML and XAI: (i) during the ML process, domain knowledge is used to prepare appropriate training data and refine the model; (ii) in the explanation phase, this knowledge is further leveraged to ensure explanations align with user perspectives.

During **model training**, domain experts incorporate their knowledge through feature engineering, supported by interactive interfaces and visual representations. While feature engineering is a standard part of ML workflows [17], our contribution lies in offering UI components and visualization techniques that enable experts to (a) apply and manage domain-informed data transformations, (b) assess the representational adequacy of engineered features for the modelling task, and (c) explore and compare model outputs across iterations to identify and address performance issues. The preliminary version of this framework was presented in our previous deliverable D5.1. Here we present the formalization and refinement of the framework that is organized around a set of modular operations:

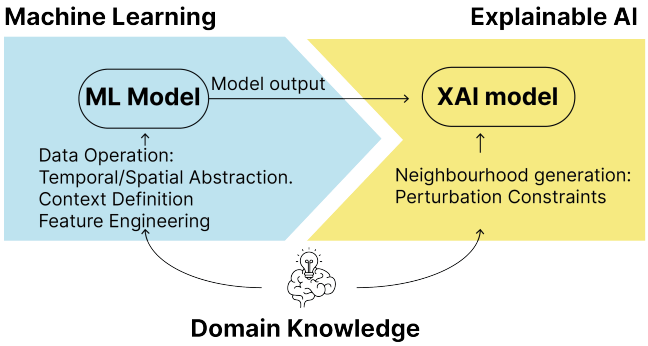


Figure 1: Integrating human expertise into the ML pipeline. Knowledge is embedded through data structuring and realistic neighbourhood generation, supporting model interpretability and explanation plausibility.

Step 1: Data abstraction – defining analytical units Raw data often consist of elementary observations tied to individual objects, timestamps, or locations. The first step is to define higher-level units of analysis suitable for the modelling task, e.g., grouping time points into intervals, spatial points into regions, or individuals into groups. The framework provides interactive interfaces to configure such aggregations and view their effects on the data distribution (see, for example, [5, 6]).

Step 2: Context encoding Many predictive tasks depend not only on the internal properties of a unit but also on the context in which the unit occurs. Experts can define context windows, e.g., preceding time intervals, neighbouring regions, or connected nodes of a network, and derive features that summarise relevant surrounding conditions. Our tools support this through parametrised transformations and visual representations allowing users to inspect the relevance and informativeness of contextual features (e.g., [42]).

Step 3: Synoptic feature construction To capture the internal structure or dynamics of a unit, the system provides functions for computing statistical summaries, trends, variation measures, or custom aggregates. For example, experts can summarise speed variations over a time interval or density distributions within a region. Visualisations help users to verify whether the derived features align with domain-relevant behaviour patterns (e.g., [3]).

Step 4: Iterative model refinement After model training, experts can compare predictions across model versions using side-by-side views and performance overlays. This feedback loop allows users to detect where features are insufficient or misleading, and return to previous steps to refine transformations or add new features.

This iterative process allows domain knowledge to shape the model indirectly through improved data representation. Rather than encoding knowledge explicitly in rules or ontologies, experts interactively refine how data are structured, contextualised, and summarised to better capture relevant behaviour patterns. Our framework supports this process through interfaces designed for expert-guided data transformation and responsive visual feedback.

In our work, we focus on generating *local explanations* telling how a model obtains a specific prediction for a given instance. We build on the rule-based explanation method `LORE` (LOcal

Rule-based Explanation) [25], which approximates the model's decision boundary in the local neighbourhood of the instance via a transparent surrogate model. The explanation takes the form of logical rules: (i) a **factual rule**, which identifies the key factors leading to the prediction, and (ii) a set of **counterfactual rules** describing minimal changes in feature values that would alter the prediction.

Explanations are more effective when they reference concepts familiar to the user. By relying on features engineered through expert-driven abstraction and contextualisation (see D5.1 and [4]), we expect that our explanations align closely with domain knowledge and thus can be well understood by users.

To ensure that generated explanations are meaningful to domain experts, we extend `LORE` with domain knowledge through neighbourhood constraints, guiding the synthetic data generation process (see Section 2.1 for details). This constraint-driven extension allows the generated neighbourhood to better align with the expert's mental model and domain expectations.

In our research activities we have focused on four main direction of research:

- Explainable AI (XAI) methods, with a focus on post-hoc explanations, to make accessible the internal processes of Machine Learning models. This activity mainly focused on the study and development of explanation models that can be directly applied to black-box models of the use cases of the project (see Section 2)
- Uncertainty in models, with a focus on the development of a conceptual model of uncertainty to map the concepts and relationships of the domain and the user toward the data-driven models (see Section 6)
- Visual Analytics (VA) methods to support the accessibility of the human to the complexity of models. The VA methods are designed to support the creation of ML models, incorporating expert knowledge, and the presentation of explanations to users (see Section 3). They also include solutions to assist the analysis of uncertainty in the models (see Section 4)
- Augmented Reality (AR) methods, with a focus on supporting the use cases of the project. The AR methods are designed to support the interaction of the user with the models and the data, and to provide a more immersive experience in the exploration of the models (see Section 5). They are also designed to support the analysis of uncertainty in the models.

2 Explainable AI (XAI)

This section will provide an overview of the explainable AI methods developed within the project. Inspired by [13], we are following two parallel strategies to make the AI models more interpretable and explainable. The first strategy is to use inherently interpretable models, such as decision trees, which are easy to understand and explain. The second strategy is to use post-hoc explainability methods, which can be applied to any black-box model to provide explanations for their predictions. This general approach should also take into account the specific requirements of the application domains of CREXDATA, such as the need for explanations to be consistent with domain-specific knowledge and be compatible with the time-sensitive decision making processes of the CREXDATA users. Besides, the explanation methods should also be able to handle different types of data, ranging from tabular data to unstructured data, such as text, images, and time series.

2.1 Custom generator for Domain-specific rules

Exploiting the new flexible architecture of LORE introduced in Deliverable D5.1, we focus on generating local explanations that clarify how a model produces a specific prediction for a given instance. Starting from the approach that is presented in Deliverable 5.1, we extended the set of neighborhood generators to include a formalization of the domain knowledge provided by experts. This methodology is based on the definition of constraints guiding neighbourhood generation:

- **Hard constraints:** Synthetic instances must satisfy structural properties defined during data abstraction and contextualisation (e.g., temporal consistency). Violations are discarded.
- **Soft constraints:** Feature-level weights reflecting expert knowledge bias the mutation process toward more meaningful changes.

These constraints help generate neighbourhoods that better match expert expectations and domain logic.

2.1.1 Decision tree-based generator

This generator selects which features to perturb based on domain-specific relationships that are too complex for explicit expert rules. Instead, we extract this knowledge from the training data using decision trees (DTs). For each class c, we train a DT that distinguishes instances of class c (label 1) from all others (label 0). Each tree thus encodes the feature relationships that characterise class c.

At generation time, the black-box model predicts the class of an instance x. The corresponding DT is then used to retrieve the features along its decision path. These features are given higher perturbation probability, implementing **selective perturbation** as a form of **soft constraint**. Hard constraints ensure valid relationships among speed features (e.g., $\texttt{SpeedMinimum} \leq \texttt{SpeedQ1} \leq \texttt{SpeedMedian} \leq \texttt{SpeedQ3}$). Non-selected features may still be perturbed, but with lower probability. The computational cost remains close to the baseline generator, aside from the one-time DT training.

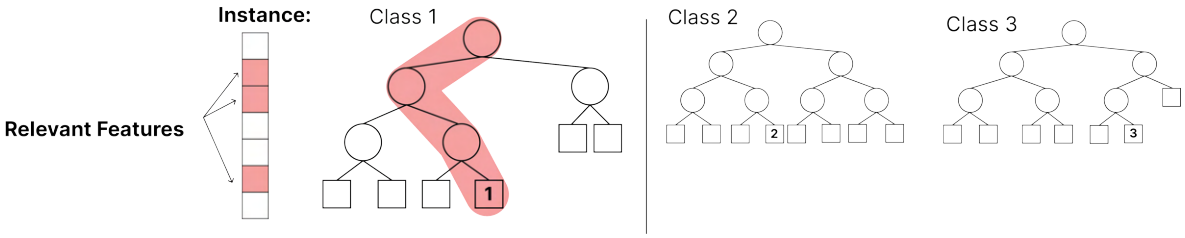


Figure 2: A forest of DTs used to determine feature-perturbation probabilities. The DT corresponding to the predicted class identifies the features (in red) most relevant for perturbation.

Genetic optimization of the DT-based generator We further refine the generator via a genetic optimization procedure. While keeping LORE's fitness functions, mutation and crossover operations are restricted by the same hard and soft constraints. This improves the generation of opposite-class instances and enhances neighbourhood quality.

2.1.2 LLM-based generation of synthetic neighbourhoods

We conducted experiments using large language models (LLMs) to generate synthetic neighbourhoods for both COVID-19/mobility prediction and vessel movement tasks, illustrating their ability to incorporate domain knowledge interactively.

For the COVID-19 case study, GPT-4 Turbo (ChatGPT-4o) was used. We remind that the random forest model was trained on the real data with the numbers of COVID-19 cases and the numbers of internal trips within 52 provinces of Spain in the period from April 2020 to May 2021. The original continuous time series were divided into weekly intervals and discretized into 4 levels of disease incidence (relative number of cases to the province population) and 4 levels of population mobility (relative number of trips in respect to the pre-pandemic number in each province). Each event of changing the COVID-19 level was complemented by context data representing the levels of COVID-19 and mobility in the preceding 6 weeks. The model was trained to predict the next COVID-19 incidence level based on the prior 6-weeks context and the absolute time passed from the pandemic onset.

Through iterative dialogue, we specified requirements emphasizing both randomness and domain consistency. Using the expert instructions and the training data, the LLM produced Python code integrating transition probabilities between different levels of COVID-19 and and between different levels of population mobility levels. After initial testing, we requested increased diversity to ensure coverage of all target classes. The generator was evaluated on 15 real instances, producing 100 synthetic neighbours each. UMAP visualizations showed that synthetic neighbours clustered around the originals while maintaining sufficient diversity to produce varied model predictions, confirming plausible and informative data variations. Challenges included translating informal domain knowledge into precise prompts, validating and debugging code, and requiring coding expertise.

For the vessel movement case study, Google Gemini v2.5 Flash was used. Given a textual prompt describing domain constraints and feature characteristics, the LLM generated Python code to produce class-targeted, constraint-compliant synthetic neighbours. The generator incorporates domain knowledge through:

- **Feature-constrained perturbation:** Values are perturbed within reasonable ranges while preserving structural relationships. Speed statistics (`SpeedMinimum`, `SpeedQ1`, `SpeedMedian`, `SpeedQ3`) maintain ascending order and remain below the empirical upper bound (22.0 knots). Curvature-related features (`Log10Curvature`, `DistStartTrendAngle`, `Log10DistStartTrendDevAmplitude`) remain within observed ranges. High speed and high curvature cannot co-occur. Speed- and distance-related features stay non-negative, and `Log10MinDistPort` remains above -3.05.
- **Class-tendency-guided generation:** Synthetic instances are biased toward each of the six behaviour classes to support counterfactual exploration and class separation analysis.
- **Balanced neighbourhood construction:** For each input, a user-specified number of neighbours is generated, balanced across all six class tendencies to ensure diversity.

Overall, both experiments demonstrate that LLMs can act as interactive coding assistants to create domain-informed data generators, producing realistic, diverse, and class-aware synthetic neighbourhoods. Visual analytics techniques were essential for evaluating instance quality, though additional tools are needed to systematically assess feature realism.

2.1.3 Alternative neighbourhood generators

To evaluate the impact of constraint-driven generation, we compare with two baselines:

- **Random Generator:** The `LORE` default, independently perturbing features within valid ranges.
- **kNN Generator:** Our baseline selecting the k nearest neighbours from the training data. It relies on existing labelled samples rather than generating synthetic ones and is applicable only when training data are available.

2.1.4 Evaluation of synthetic neighbourhoods

To assess the impact of our methodology, we compared the outputs of different neighbourhood generators to evaluate how well each generator aligns with expert knowledge. As a proxy, we used the distribution of the training data: if a generator produces neighbourhoods consistent with expert intuition, its output should also resemble the training distribution, and the resulting explanation rules should reflect both.

We performed two types of evaluation: (1) an **analytical comparison**, using quantitative metrics to assess locality, compactness, and plausibility; and (2) a **visual qualitative comparison**, examining how the marginal distributions of synthetic features match the training data and how effectively each generator diversifies feature values.

Analytical comparison Neighbourhood generators aim to produce perturbed versions of an instance x that (i) remain realistic—i.e., consistent with the training distribution—and (ii) stay close to x. To measure compactness, we use the **instance distance** defined in [25]:

$$D(x, N) = \frac{1}{|N|} \sum_{z_i \in N} d(x, z_i),$$

where $d(x, z_i)$ is the distance between x and each generated instance z_i (Euclidean distance in our setup).

To assess realism, we use the **Energy Distance** [55] and **Maximum Mean Discrepancy (MMD)** [24], two standard metrics for comparing probability distributions. Both quantify how closely the generated neighbourhood resembles the training data.

For each instance, we generated a neighbourhood of size $n = 2000$ per generator and repeated this process multiple times. In all cases, the hypothesis that the generated and training distributions were identical was rejected ($p < 0.001$).

Generator	Instance distance	Energy	MMD	Time (s)
Random	119.1400 ± 12.0031	111.8990 ± 13.7530	0.0178 ± 0.0014	0.0358 ± 0.0042
Baseline	9.6637 ± 4.8497	5.0910 ± 0.0000	N/A	0.0024 ± 0.0001
Custom	2.2196 ± 0.2118	13.5748 ± 0.0430	0.0399 ± 0.0004	6.5345 ± 0.1702
Genetic	6.0275 ± 4.0885	12.9352 ± 0.1566	0.0575 ± 0.0009	13.4913 ± 0.2668
Custom genetic	1.2714 ± 0.9461	15.0236 ± 0.6921	0.0587 ± 0.0053	52.7897 ± 0.7902
LLM	11.9267 ± 1.5013	8.1266 ± 0.0198	0.0083 ± 0.0002	0.0500 ± 0.0021

Table 3: Vessel movements: Comparison of neighbourhood generators across instance distance, Energy, MMD, and computation time (mean $\pm$ standard deviation).

Table 3 reports the results for the Vessel case study. The *Custom* and *Custom Genetic* generators produced the most compact neighbourhoods (lowest instance distance). The *LLM* generator also performed well, achieving both compact neighbourhoods and the lowest Energy and MMD scores. The *Random* generator generated the most dispersed data, while the *Baseline* (training data) naturally obtained the lowest Energy value. Overall, the *Custom* and *LLM* generators offered the best balance between realism and locality. The *LLM* generator was also the fastest, whereas the *Custom Genetic* generator incurred substantial computational overhead.

Generator	Instance distance	Energy	MMD	Time
Random	6.2837 ± 0.1386	0.3761 ± 0.0375	0.0052 ± 0.0003	0.3078 ± 0.0400
Baseline	6.2837 ± 0.1386	0.0000 ± 0.0000	N/A	0.0002 ± 0.0000
Custom	6.5898 ± 0.0760	1.7483 ± 0.0187	0.0203 ± 0.0003	0.0482 ± 0.0039
Genetic	6.7421 ± 0.0567	3.2248 ± 0.0500	0.0336 ± 0.0005	20.2679 ± 0.3779
Custom Genetic	5.8902 ± 0.0929	4.3411 ± 0.0647	0.0391 ± 0.0004	41.5833 ± 1.0018
GPT	6.4904 ± 0.0657	2.0216 ± 0.0204	0.0253 ± 0.0004	0.4225 ± 0.0440

Table 4: COVID-19 and Mobility: Comparison of generator variants across instance distance, Energy, MMD, and computation time (mean $\pm$ standard deviation).

Table 4 shows the results for the COVID-19 and mobility dataset. Since most features are

categorical, perturbations are limited, leading to similar instance distance values across generators. No single generator clearly outperformed the others, and the analytical evaluation provides only a partial assessment. Therefore, in the next section we complement these findings with a qualitative comparison of the feature distributions in the generated neighbourhoods.

2.1.5 Visualization-based qualitative comparison of feature distributions

To assess the quality of the generated neighbourhoods, we conducted a qualitative comparison of the feature distributions of the synthetic instances produced by each generator. We developed an interactive visual dashboard that displays, for each feature and generator, rows of histograms showing the marginal distributions of the generated values. These distributions are directly compared to those of the training data, which serve as a reference for the expected behaviour of each feature. This visual inspection helps evaluate how well each generator captures the underlying structure of the domain.

The analysis is performed around a specific instance x. For example, Figure 3 illustrates the feature distributions for an instance from the *Trawling* class in the Vessel dataset, characterized by the following feature values: *SpeedMinimum*=0.91, *SpeedQ1*=2.37, *SpeedMedian*=2.49, *SpeedQ3*=2.78, *Log10DistanceStartShapeCurvature*=0.39, *DistanceStartTrendAngle*=-0.01, *LogDistStartTrendDevAmplitude*=0.70, *MaxDistPort*=27.55, *LogMinDistPort*=1.30.

The rightmost columns display the training data distributions. Since the generators are expected to produce local instances, it is particularly relevant to compare their outputs with the *Baseline* generator, which samples training instances located in the neighbourhood of x.

The *Random* generator yields distributions that diverge substantially from the training data because it samples uniformly from the entire feature space. All other generators produce instances that remain closer to the expected distributions. The first row of histograms shows the class distributions in the generated neighbourhoods. The *Random* generator produces instances for all classes, whereas the *Custom*, *Genetic*, and *Custom Genetic* generators focus on classes near the target instance. The *LLM* generator generates a broader class spread to increase neighbourhood diversity.

The other rows show the distributions of individual features. For speed-related features (*SpeedMinimum*, *SpeedQ1*, *SpeedMedian*, *SpeedQ3*), the customized generators reproduce internal feature relationships and constraints effectively, resulting in narrow ranges. The *LLM* generator produces distributions more similar to the training data. A similar pattern is observed for shape-based features (*Log10DistanceStartShapeCurvature*, *DistanceStartTrendAngle*, *LogDistStartTrendDevAmplitude*), where the *LLM* generator aligns closely with the *Baseline*. For port proximity features (*MaxDistPort*, *LogMinDistPort*), the customized and genetic generators stay tightly localized around x, while the *LLM* generator explores a wider region of the feature space. The *Random* generator again produces distributions that deviate significantly from the training data, presenting a uniform spread across the feature ranges.

2.2 Integration of LORE in the CREXDATA platform

We exploited the new implementation in a specific use case involving three Workpackages: WP2, WP3, and WP5. The use case focuses on predicting the exhaustion levels of firefighters

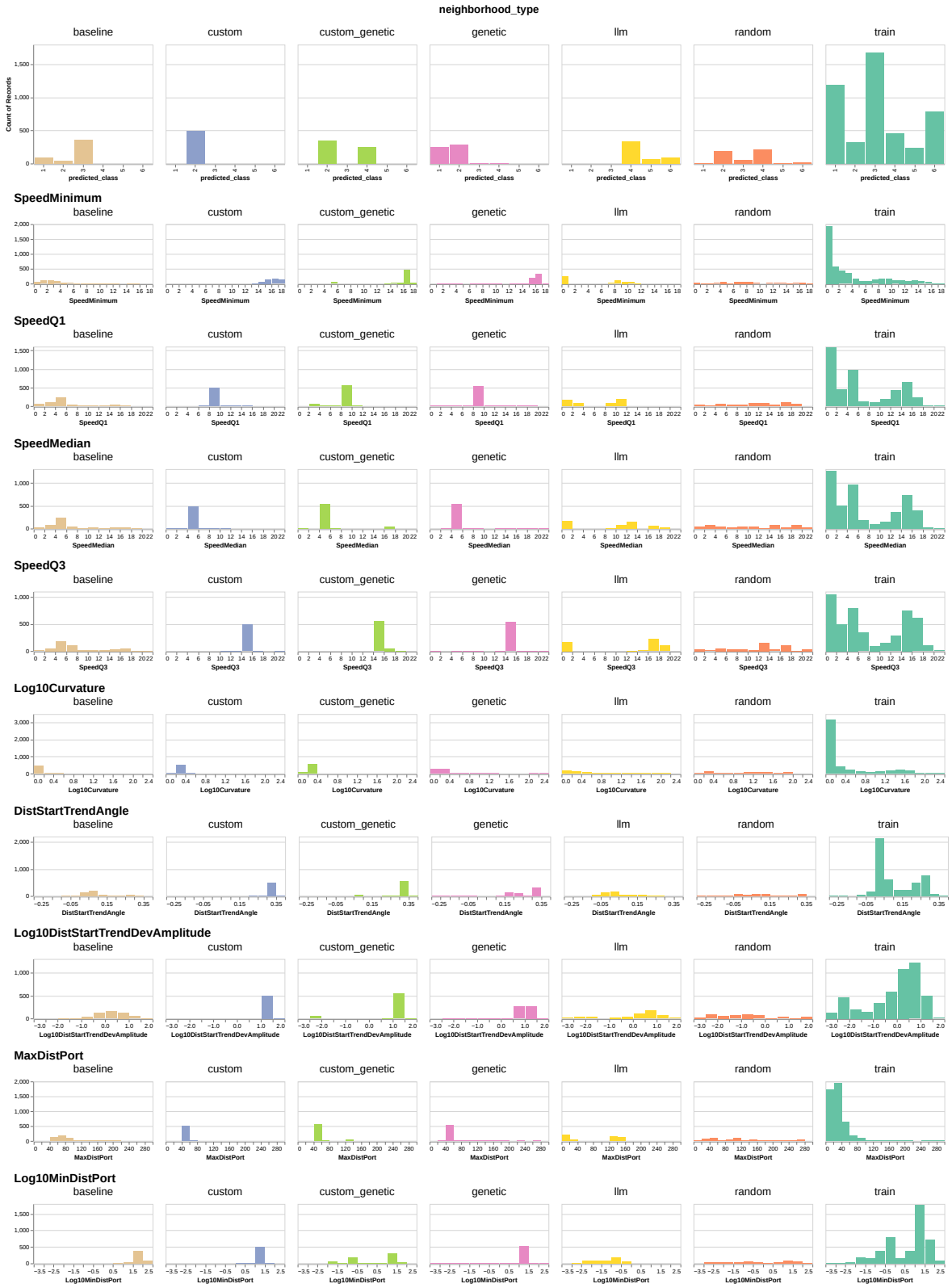


Figure 3: Dashboard for comparing and exploring neighbourhoods generated around an instance x in the Vessel dataset. The first row of histograms shows the class distributions in the generated neighbourhoods. The other rows show the distributions of individual features. The rightmost columns display the training data distributions.

during their operations, based on sensor data collected from wearable devices. The goal is to provide real-time predictions of exhaustion levels, along with explanations that help understand the factors contributing to these predictions. The details on the integration of LORE in the CREXDATA platform were reported in Deliverable D3.2.

2.2.1 Use case of firefighter exhaustion prediction

Data Collection. The data was collected from January 2025 to July 2025, involving only adult male participants aged 18 and older. During the training sessions, firefighters were asked to put on a weareable device for tracking the biometric parameters. After each training session, all participants had to fill out a questionnaire. They were asked to: (i) provide personal information (age, sex, weight, height), (ii) rate how intense they perceived the physical activity, whether very light, light, moderate, intense, or very intense, (iii) and indicate the level of exhaustion they reached on a scale from 1 to 5, with 1 being very low and 5 being very high.
Data preprocessing. All data was organized into a dataset for the second part of the analysis. As reported in Figure 4, each row of the dataset had the following attributes: date and time (ts), skin temperature (Tskin), environment temperature (Tamb), ambient relative humidity (amb_RH), skin relative humidity (skin_RH), sweating rate (SWR), heart rate (HR), heart rate confidence (HR_confid), activity estimate (ACT_est), and the target variable, exhaustion.

	ts	Tskin	Tamb	ambient_RH	skin_RH	swr	HR	HR_confid	ACT_est	exhaustion
0	2025-06-11 12:00:00+00:00	27.180000	27.730000	21.330000	24.220000	34.000000	144.780000	42.200000	0.910000	1
1	2025-06-11 12:01:00+00:00	27.090000	27.630000	21.340000	24.270000	34.400000	149.820000	40.110000	0.370000	1
2	2025-06-11 12:02:00+00:00	27.020000	27.550000	21.310000	24.280000	34.740000	172.810000	35.380000	0.640000	1
3	2025-06-11 12:03:00+00:00	26.960000	27.480000	21.340000	24.260000	34.040000	175.800000	39.170000	1.430000	1
4	2025-06-11 12:04:00+00:00	26.930000	27.440000	21.390000	24.280000	33.500000	117.560000	37.960000	1.240000	1

Figure 4: Dataset structure for firefighter exhaustion prediction.

We then enriched the dataset by computing for each ID, moving averages and standard deviations of each column, considering the current row and up to five previous rows. The size of the windows is parametric. This specific preprocessing step was implemented as a custom operator in the CREXDATA platform. To enforce the unicity of the training sessions, we grouped the observation by session ID and sorted them by the temporal variable 'ts'. We implemented a personalized slipt criterion to get the training and test sets, based on a percentage threshold over the temporal span of each session: despite the lenght of each session, the training set contains all observations up to the threshold, while the test set contains the remaining ones.

Model training and explanation. We tested several types of models for learning, ranging from transparent models such as decision trees to more complex models such as random forests and neural networks (we used an implementation of ROCKET). We found that random forests provide the best trade-off between accuracy and efficiency.

The model performed particularly well in the first three classes, achieving a very high precision score for class '3' (0.91), and quite good scores in the same metric for class '1' and '2' (respectively, 0.74 and 0.68). Table 5 reports the complete classification metrics for each class, along with the global metrics (accuracy, macro average, and weighted average). In contrast, classes '4' and '5' showed a notable imbalance, characterized by pretty low precision (respectively, 0.36 and 0.41) but perfect recall (1.00), suggesting that the model led to many

false positives but no false negatives. This notable imbalance could be caused by the few examples available for these classes (33 for class '4' and only 15 for class '5') in contrast to the higher support for the rest of the classes. Overall, the global metrics allowed us to conclude that our model was pretty consistent, with quite good generalization skills.

Class	Precision	Recall	F1-Score	Support
1	0.74	0.82	0.78	481
2	0.68	0.80	0.73	454
3	0.91	0.65	0.76	741
4	0.36	1.00	0.53	33
5	0.41	1.00	0.58	15
Accuracy		0.75		**1724**
Macro Avg	0.62	0.85	0.68	1724
Weighted Avg	0.79	0.75	0.75	1724

Table 5: Classification metrics for exhaustion level prediction model.

Explanation. To explain the predictions of the random forest model, we implemented and compared two different methods: LORE and SHAP. We used SHAP library to analyze the contribution of each variable in the classification. As shown in the Figure 5, it visually highlights which features had the greatest influence on the predictions. The feature 'Tskin_rolling_std' had the highest average impact on the model's output across all classes, being particularly influential for class '3' (blue), class '1' (purple) and class '2' (fuchsia). 'Tamb_rolling_mean' and 'ambient_RH_rolling_mean' also had a significant impact for these three classes. In contrast, class '5' (green) appeared to be the least predicted, as also confirmed by the performance metrics discussed above. Overall, the most influential features identified by the model were those derived from rolling means and standard deviations computed *after* the data collection phase, highlighting the critical role of the temporal dimension in the model's predictions.

We then used LORE library to extract the logical rules followed by our model. As shown below, the feature 'Tskin_rolling_std' emerged as a major splitting variable across multiple rules, confirming its relevance as previously highlighted by the SHAP analysis. Variations in its threshold, in combination with other features such as 'ACT_est', 'HR', and 'swr_rolling_std', resulted in different classifications.

```
rule premises:

Tskin_rolling_std <= 11.758705139160156
ACT_est <= 7.369220018386841
```

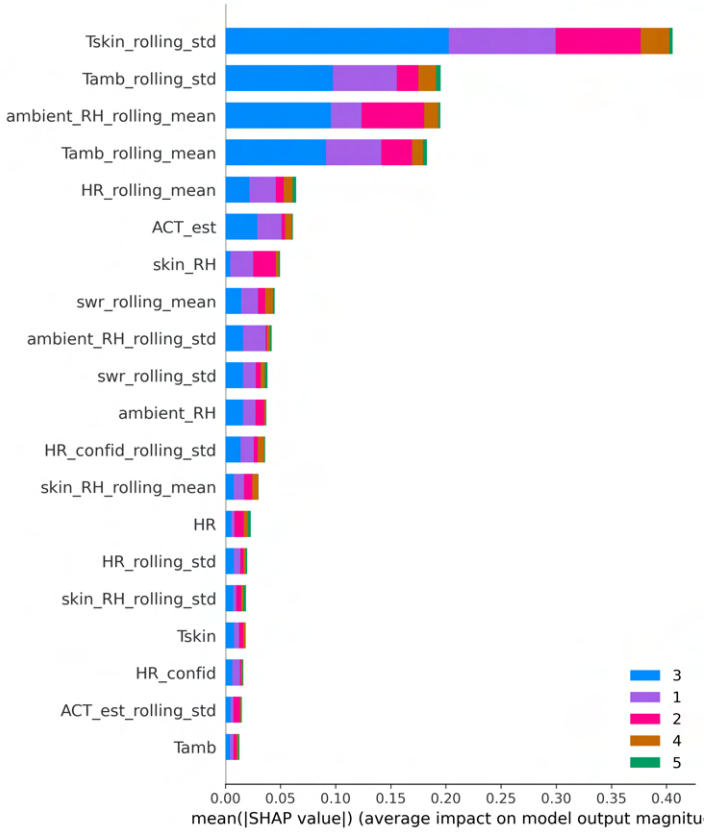


Figure 5: In this chart, we can see what are the most relevant features used by the Random Forest Classifier for the prediction of each single label.

```
HR > 51.99660110473633
swr_rolling_std <= 335.98008728027344
consequence: exhaustion = two

premises:
Tskin_rolling_std > 11.758705139160156
HR_confid_rolling_mean <= 93.74883651733398
skin_RH > 88.28531646728516
ambient_RH_rolling_mean <= 71.52997589111328
consequence: exhaustion = three

premises:
Tskin_rolling_std <= 11.758705139160156

ACT_est > 7.369220018386841
consequence: exhaustion = one

premises:
Tskin_rolling_std <= 11.758705139160156
```

```
ACT_est <= 7.369220018386841
HR > 51.99660110473633
swr_rolling_std > 335.98008728027344
consequence: exhaustion = four
```

The whole system was implemented as a workflow in the CREXDATA platform, integrating data ingestion, preprocessing, model training, and explanation generation. Custom operators are described in Deliverable D3.2.

2.3 Lesson learned and next steps

The modular architecture of the explainer LORE enabled the development of custom neighbourhood generators that incorporate domain knowledge through hard and soft constraints. Although this approach proved to produce more realistic neighbourhoods, a few limitations were identified:

- The surrogate model learned by the new synthetic neighbourhoods not always reflects the expected relevance of some features according to the domain experts. Of course, this is mainly due to the simplification that the surrogate model introduces. This misalignment is more evident when the data space has a low dimensionality or discrete features. We noticed that in the COVID-19 case study, where the features are mostly categorical with few possible values. We plan to address this issue by exploring alternative types of surrogate models, e.g. logic-based models.
- The computational cost of some custom generators, especially those based on genetic algorithms, is higher than the baseline random generator. We worked to optimize the code and the algorithmic implementation, to better exploit parallelization of modern processors. However, there is still room for improvement, especially when dealing with large datasets. Indeed, this is a limitation if the explainer is used in an interactive setting, where the explanation should react to user inputs in real time. A possible solution is to pre-compute the neighbourhoods for a set of representative instances, and then interpolate the explanations for new instances. A preliminary proof of concept of this approach was published in [48].

3 Visual Analytics for Supporting XAI

There are two main uses of visual analytics (VA) for supporting XAI:

- Employ VA in the process of model building to create models incorporating expert knowledge, which makes them better understandable to users. The knowledge extracted in the course of model building can, potentially, be used in generating user-oriented explanations of the model logic and decisions.
- Employ VA techniques in presenting explanations to users and enabling them to explore and thereby better understand the model work.

In CREXDATA WP5, we develop VA approaches aiming to attain both objectives: help domain experts and data scientists create well performing and appropriately explainable models and generate easily understandable explanations of model work for end users.

Our work in the second period has been focused on designing and developing visual analytics (VA) techniques to enable inspection and understanding of ML models at the global level.

3.1 Interactive Visual Exploration of Rule-Based Model Logic

Rule-based machine learning models, including those derived from decision trees or forests, are often considered inherently interpretable. However, human understanding is hindered by model size, rule complexity, and interdependencies between features. Moreover, rule sets extracted from ensemble models can contain contradictory, incomplete, or counterintuitive logic, even when the overall model achieves high predictive accuracy. We have developed and tested a visual analytics methodology for supporting systematic exploration of rule-based model logic and its alignment with domain knowledge. Our approach integrates overview visualizations, interactive filtering, contradiction analysis, and topic modeling. This enables analysts to detect illogical or implausible rules, assess their potential impact, and refine the model to improve its interpretability and trustworthiness. A key distinction of our methodology is its ability to support reasoning about model behavior both with and without access to labeled data. We demonstrate the approach through two studies related to the CREXDATA use cases: evaluating logical consistency in a vessel movement classifier and analyzing feature relationships in a COVID-19 prediction model. For this purpose, we have implemented our approach in an exploratory prototype software named **RuleSense**.

3.1.1 Introduction

In high-stakes domains (e.g., healthcare, policy, infrastructure), validating that model logic aligns with domain reasoning is as important as test accuracy. Our goal has been to support logic-oriented exploration of rule-based models (which can be derived from decision trees/forests, possibly, trained as surrogate models mimicking the behavior of black box models) by: (1) providing overview-to-detail visual summaries of rule-condition distributions; (2) enabling interactive filtering and drill-down by outcome, features, and value intervals; (3) detecting contradictions and redundancies; (4) uncovering co-occurrence patterns among feature conditions.

Unlike instance-focused explainability tools, our exploratory prototype system RuleSense is

designed for model-wide logic inspection. We used RuleSense to test how our methodology can help analysts verify model soundness, extract insights, and assess acceptability for deployment. Importantly, model analysis can be done without involving test data, which may not be available to the analyst (e.g., relevant data may be sensitive or proprietary). However, in a case when suitable test data are available, they can be used for testing a model after modifications.

3.1.2 Summary of related work

Interpretability for rule-based models focuses on improving comprehensibility through simplified structures and explanations. Visual analytics for model interpretation provides matrix and glyph-based views, projections, and interactive exploration for ensembles, often emphasizing performance and instance-level explanations. Topic modeling has been widely applied beyond text to reveal latent structure but has not been used before to analyze ML rule sets. Our work builds on prior research in interpretable machine learning, rule-based modeling, and visual analytics. However, it stands apart by:

- prioritizing logic-focused *model-wide* understanding (domain alignment, contradictions) over instance-level performance explanation;
- operating without requiring data access (data-optional audit);
- offering explicit contradiction/subsumption analysis of the relationships between rules; and
- applying topic modeling to rule conditions to reveal recurring logic patterns.

The main distinctions are summarized in Table 6.

Table 6: Key distinctions between our approach and SOTA.

Aspect	Existing Systems	Our Approach
Primary Goal	Performance inspection, local explanation	Evaluate logic, align with domain reasoning
Data Dependence	Strong (instance-based metrics, coverage, etc.)	Works independently of data (data-optional)
Target Audience	Model developers	Domain experts, model reviewers
Rule Reduction	Optimized for fidelity or performance	Optional, logic-driven cleaning and simplification
Contradiction Detection	Rarely addressed	Explicitly supported
Use Cases	Debugging, instance-level analysis	Post-hoc audit, model understanding, trust support

3.1.3 Problem Statement

Background. Decision trees can be transformed into rule sets (one rule per root-to-leaf path). Random forests yield large, diverse rule sets with redundancies and contradictions, complicating interpretation.

Objectives. Support understanding of a model's internal logic and its alignment with domain knowledge. Core tasks (data-optional):

- T1 overview visualization;
- T2 interactive filtering and drill-down;
- T3 analysis of feature relationships;
- T4 detection of logically inconsistent rules.

When labeled data are available:

- T5 apply rules to labeled data;
- T6 assess impact of rule edits on performance.

3.1.4 Methodology

Overview visualization We provide (Figs. 6, 8) class-wise and feature-wise heatmaps of condition frequencies over discretized value intervals. The rows of the matrices correspond to the features and the columns to equal-length intervals of feature values. The colour intensity represents how often each interval of feature values appears in rule conditions for the corresponding class. Gray bars on the left of the heatmap rows indicate the total number of rules for a class; blue bars show the fraction of those rules containing the corresponding featı

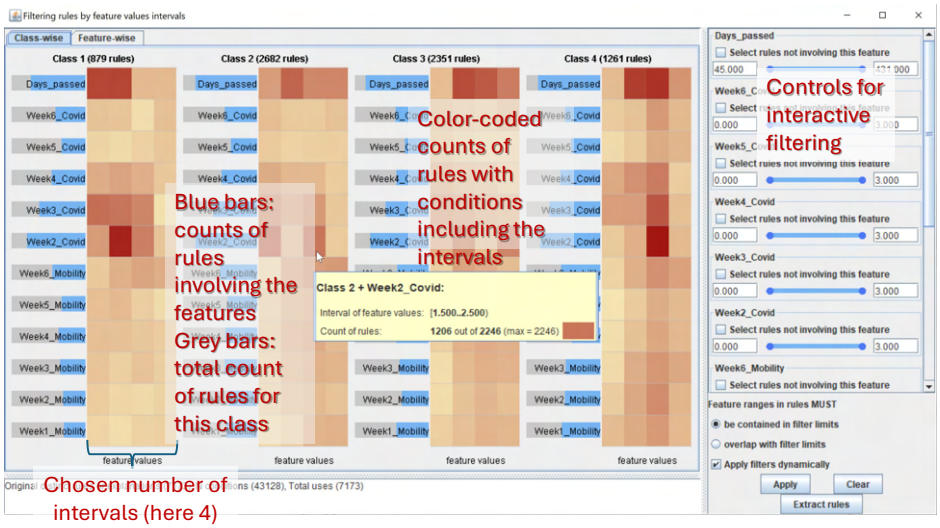


Figure 6: Overview of the distributions of the features and their value intervals across rule subsets, with interactive controls for filtering.

Interactive filtering and subset extraction Filters support: selecting outcomes; including/excluding features; constraining value ranges; extracting/removing subsets for inspection or refinement. The overview updates accordingly with a reference view for comparison.

Computational rule analysis We detect:

- *Contradictions*: more-general and more-specific rules leading to different outcomes;
- *Subsumed rules*: strictly less general rules with the same outcome;
- *Groups of similar rules*: merging significantly overlapping conditions to reduce complexity while preserving semantics.

Topic modeling over rule conditions To uncover recurring condition patterns, we discretize feature ranges (e.g., terciles/quartiles) and encode each condition as a string consisting of the feature name followed by a binary code over intervals: 1 if the condition includes or overlaps with a given interval and 0 otherwise. Rules become "documents" and encoded conditions "terms". Non-negative Matrix Factorization (NMF) extracts topics (frequently co-occurring condition sets). Topic counts are chosen empirically; we report them but omit extended selection procedures as requested.

Task support

- **T1**: heatmaps for global usage and value distributions;
- **T2**: filters for features/outcomes/intervals; subset isolation;
- **T3**: co-occurrence exploration via heatmaps, filters, and topics;
- **T4**: contradiction and subsumption analysis; filters to isolate suspicious rules;
- **T5–T6**: optional application to labeled data to validate edits and assess impact.

3.1.5 Case Study 1: Vessel Movement Patterns

Domain, data, and model We analyze a random forest trained to classify movement segments of fishing vessels into (Fig. 7):

1. forward movement,
2. trawling,
3. port enter/exit, and
4. anchoring.

Nine features capture speed statistics, trajectory shape, and port proximity (some log-transformed). The forest (100 trees) is transformed into 9,939 unique rules; after contradiction and subsumption removal, 9,515 remain.

Computational cleaning We removed contradictory and subsumed rules without harming accuracy, yielding a cleaned rule set (9,515 rules).

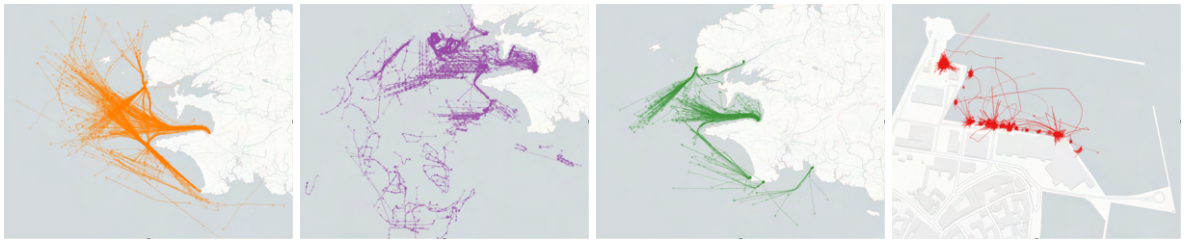


Figure 7: Characteristic shapes of segments of vessel trajectories corresponding to 4 types of activities: forward movement, trawling, port entering or existing, and anchoring.

Feature value distributions To perform task **T1**, we create an overview display (Fig. 8) with dividing the feature value ranges into 10 equal-length intervals. The class- and feature-wise heatmaps highlight:

- Class 4 (anchoring) is strongly identified by very low speeds and proximity to ports; topic participation and usage align with expectations.
- Class 1 (forward) emphasizes higher speeds and larger trend angles, with low trajectory curvature.
- Class 2 (trawling) is heterogeneous, suggesting subpatterns.
- Rules for different classes often share many intervals, indicating complex boundaries that rely on feature combinations.

Interactive filtering and drill-down Filters exposed rules contradicting domain expectations, e.g., class 3 (port enter/exit) rules omitting port proximity features; class 1 rules without speed constraints; or implausible speed thresholds for trawling. An example of exploration by means of interactive filtering is shown in Fig. 9. In this example, we explore whether the classes (e.g., class 1 for forward movement) can be distinguished without relying on speed-related features. In the left image, rules involving any speed attribute are hidden to examine what non-speed-based rules remain. The filter leaves a small number of rules, many of which rely instead on features like `DistStartTrendAngle`, though not always within plausible value ranges. On the right, `DistStartTrendAngle` is further restricted to low values, highlighting rules for class 1 that do not comply with our expectations.

These exploratory operations do not modify the rule set but help identify potentially illogical or implausible patterns for further investigation. In the next phase, we evaluate and edit the rule set based on these findings.

Rule editing and evaluation (optional data use) To assess the impact of modifying the rule set, we apply the rules to a labeled dataset containing 4,798 instances **(T5)**. This allows us to evaluate how rule removal affects model performance and to determine whether simplification or logic refinement can be achieved without sacrificing predictive accuracy **(T6)**.

Using the labeled set for validation, we removed rule groups inconsistent with domain logic (e.g., class 3 without port distance constraints; overly broad angle ranges; implausibly high trawling speeds). Edits had no or negligible impact on accuracy, while improving logical clarity.

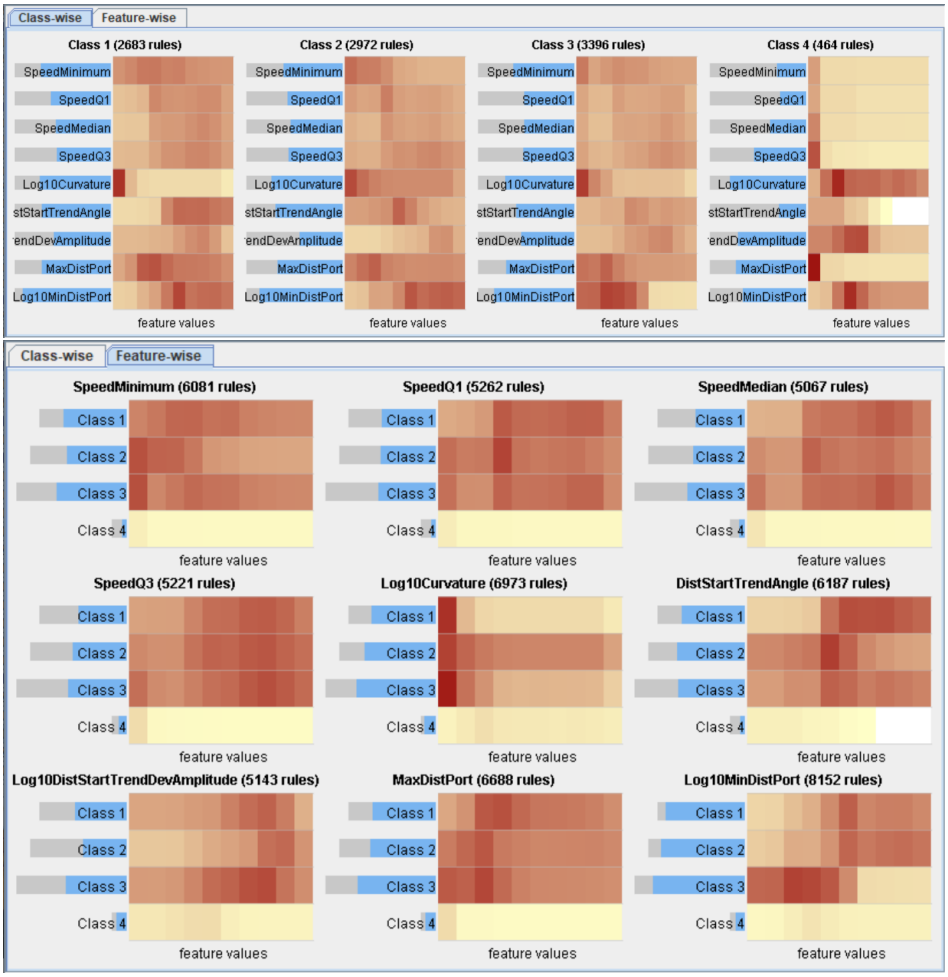


Figure 8: Class-centered (top) and feature-centered (bottom) overviews. Gray bars: total rules per class; blue bars: rules using the feature.

The original cleaned rule set consisted of 9,515 rules; after the refinement process, 8,903 rules remain.

Feature interrelationships Feature interrelationships **(T3)** can be explored interactively using the filtering tools. By restricting the value range of one feature and observing how the distribution heatmaps of other features respond (as illustrated in Fig. 10), users can identify patterns of co-occurrence and conditional dependence across the rule set. The filtering interface also supports restricting the value ranges of multiple features simultaneously.

However, while informative, such purely interactive exploration is time-consuming and does not scale well for large number of features. To address this limitation and streamline the discovery of complex feature interactions, we complement manual filtering with topic modeling (TM) aiming to reveal recurrent combinations across the rule set. To encode the rule conditions in a suitable format, we discretized the value ranges of the features into quartiles. Each condition was represented by a four-digit binary code prefixed with the corresponding feature name. We used an empirically selected topic count to extract interpretable patterns.

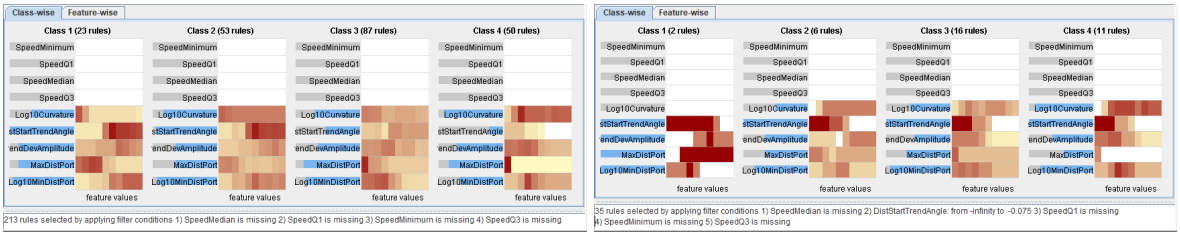


Figure 9: Examples of interactive filtering of rules. Left: Class-wise distributions after hiding rules including any of the speed attributes. Right: Result of limiting `DistStartTrendAngle` to values below 0.

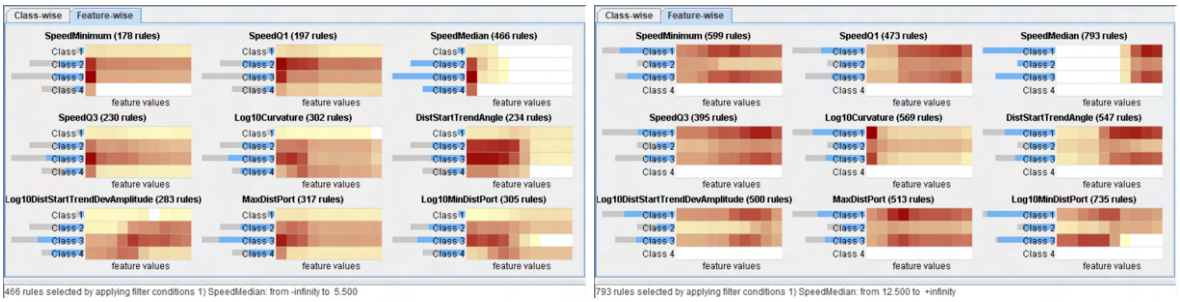


Figure 10: Exploration of the relationships between `SpeedMedian` and the other features by limiting the value range of `SpeedMedian` to low (left) and high (right) value intervals.

The topic–term matrix for the 13-topic model is visualized in Fig. 11. In this display, rows correspond to features, columns to topics, and cells contain visual representations of the binary codes indicating the presence or absence of quartile-based intervals in rule conditions. Within each code, darker and lighter shades represent interval presence and absence, respectively. To maintain clarity, only terms with weights exceeding a user-defined threshold (adjustable via a slider below the table) are shown. The opacity of the symbols is proportional to the weight of the respective term.

Topic interpretation. Topic 00 links higher `SpeedMinimum`/`SpeedQ1` with low `Log10Curvature` and higher `DistStartTrendAngle`; it is unexpectedly tied to low–medium port distances, while topics 01, 02, 08, and 12 show similar high-speed/low-curvature patterns at medium–large distances. Topic 04 combines low `SpeedMinimum` with medium–high `SpeedQ1` and low curvature, typically far from ports. Low-speed topics (05, 10, 11) align with low `DistStartTrendAngle` and high `Log10Curvature` or high `Log10DistStartTrendDevAmplitude`, as expected for zigzagging; topics 08 and 12 also pair high amplitude with low curvature, which is less intuitive.

To clarify how feature associations (topics) relate to predicted classes, Fig. 12 shows rules represented by line connecting rule weights across the 13 topics (horizontal: topics 0–12; vertical: weights), colored by class. Each class spans multiple dominant topics; none is class-exclusive. The exception is class 4 (anchoring), which is predominantly associated with topic 10, indicating a distinctive behavioral pattern.

For a clearer view, we also use per-class quantile plots of topic weights (Fig. 13). Axes match the line chart: topics 0–12 (horizontal) and weights (vertical). Each plot shows 20 quantiles; alternating bands indicate increasing quantiles. Missing lower bands reflect near-zero weights,

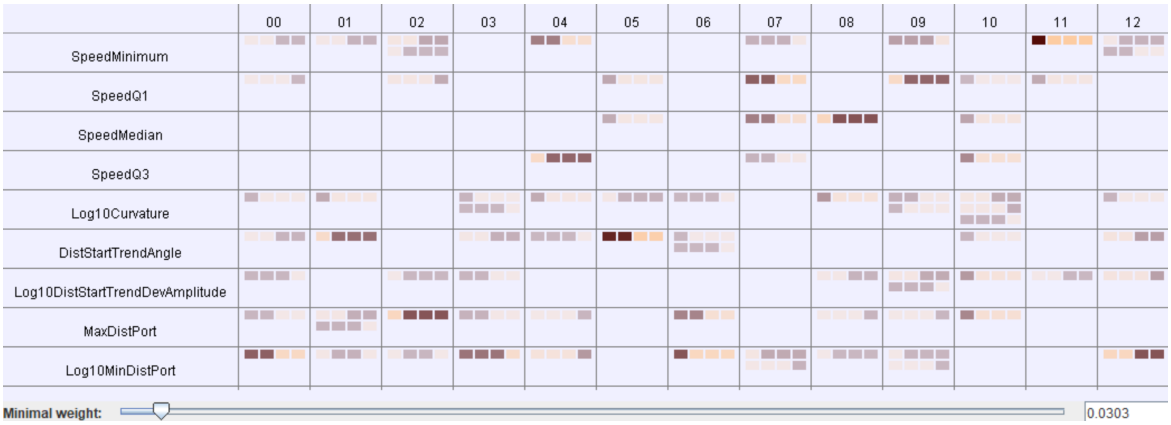


Figure 11: Topic–term matrix: Columns represent topics, rows correspond to features, and cells contain sequences of light and dark rectangles encoding the binary representations of feature conditions.

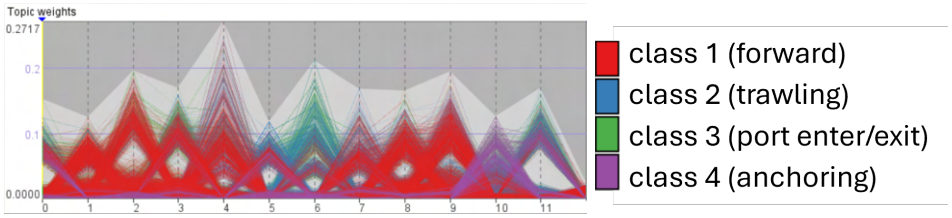


Figure 12: Topic weight profiles per rule, colored by predicted class.

so the number of visible bands per class–topic indicates the share of rules where the topic is substantial.

The quantile plots indicate:

- Class 1 (forward): topics 1–4, 8, 9 (high speed, low curvature).
- Class 2 (trawling): mix of straight-line (topics 1, 4) and low-speed/zigzag patterns (5, 7, 8, 9, 11).
- Class 3 (port enter/exit): near-port topics 0, 3, 6 dominate, but far-port topics 2, 4, 8, 9 also appear, indicating a logic mismatch.
- Class 4 (anchoring): clear dominance of topic 10 (low speeds, high `Log10Curvature`, low `DistStartTrendAngle`, small port distance), consistent with expected behavior.

Topic modeling has proven valuable in our investigation by uncovering feature interdependencies and frequently co-occurring conditions across the rule set. The extracted topics reveal key patterns of feature interaction and their associations with different predicted classes. As such, topic modeling complements the synoptic overview and interactive filtering by enabling analysts to move beyond the examination of individual features toward a more holistic understanding of the logic and structure underlying the model's behavior.

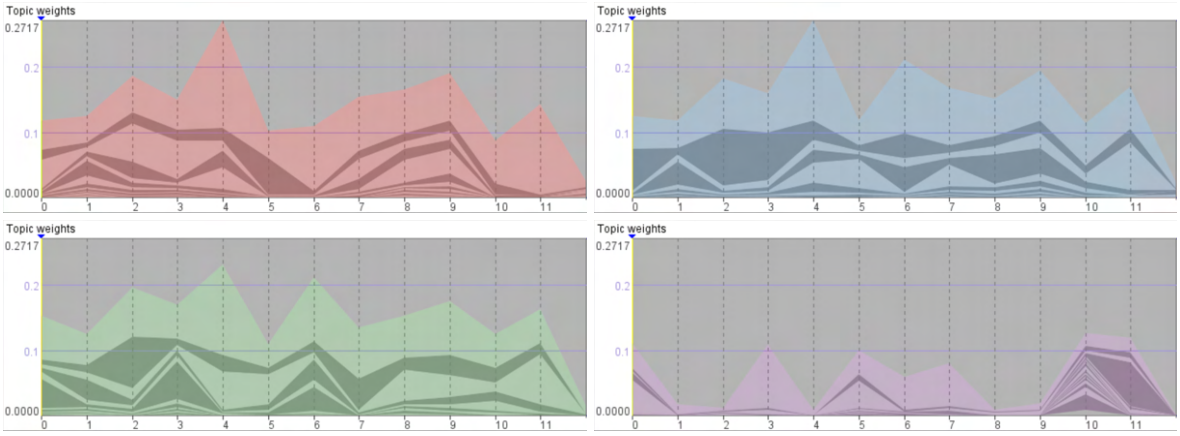


Figure 13: Quantile plots of topic weights per class (forward, trawling, port enter/exit, anchoring). The plots use desaturated versions of the colors introduced in Fig. 12.

Summary In this case study we tested how our approach can enable analysts to evaluate alignment of a model with domain logic and expectations. Through overview visualizations, interactive filtering, and rule removal, we were able to identify and eliminate logically flawed rules without harming model performance. The use of labeled data, when available, served as a validation mechanism but was not necessary for detecting inconsistencies. More importantly, ensuring that the model's logic is sound increases the likelihood that it will behave reasonably when applied to previously unseen inputs, whereas high accuracy on test data does not necessarily guarantee good performance in such situations.

3.1.6 Case Study 2: Pandemic–Mobility Relationships

Model and data A random forest model was trained on publicly available data describing daily COVID-19 incidence and population mobility across 52 Spanish provinces. The original daily time series of the numbers of disease cases and the numbers of internal trips within provinces were transformed into sequences of discrete events lasting 1 to 21 weeks, where an event represents a stable disease incidence level (1–4). Each event was characterized by features reflecting disease and mobility levels in its temporal context, covering the six preceding weeks [1]. For the disease level, only weeks -6 to -2 were used, excluding week -1 to account for the typical delay in implementing mobility policies in response to rising or falling case numbers. An additional feature, days since the pandemic onset, was included to account for pandemic evolution. Since the first wave had distinct characteristics (high incidence and lockdown), data before April 2020 were excluded from model training. The trained model yields 7,173 rules; after cleaning, 4,097 remain.

As the model is a classifier, each of the four COVID-19 incidence levels is treated as one of the classes. In the following text, the string "class X" is equivalent to "level X", where X ranges from 1 to 4.

Overview and rule grouping Figure 14, top left, shows a projection of the rules based on condition similarity with dots colored by predicted class (incidence level from 1 in blue to 4 in red). The projection shows distinct clusters for incidence levels 1 and 4, with levels

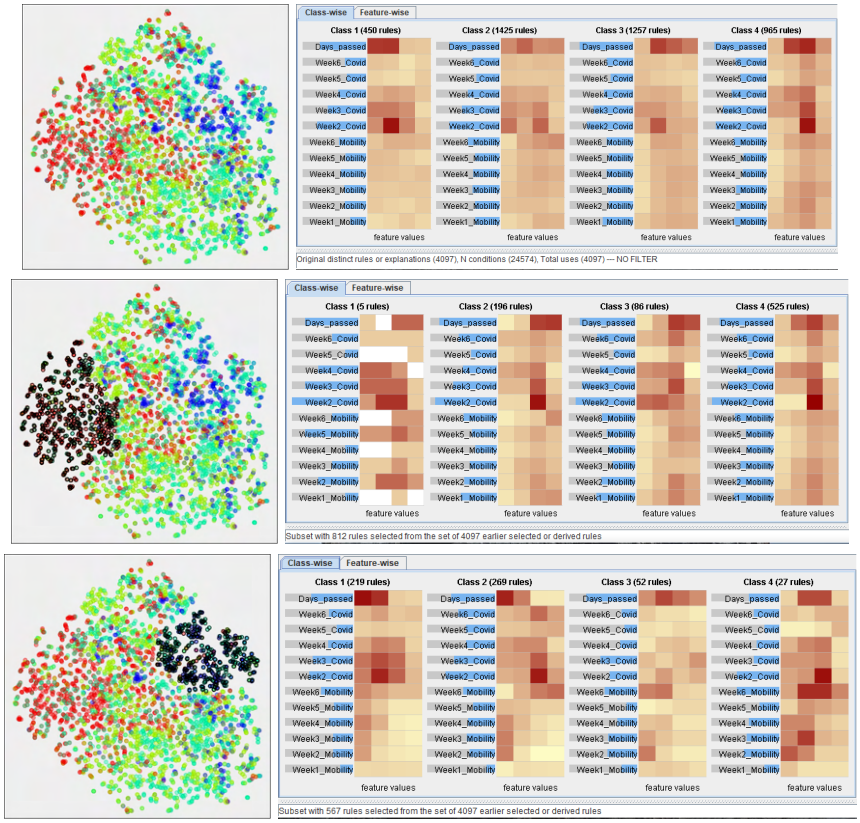


Figure 14: Exploration of the COVID-19 and mobility model. Left: projection of rule set by similarity of rule conditions. Right: feature value distributions for the entire rule set (top) and for two selected clusters (middle and bottom).

2 and 3 intermixing. On the right of the projection, a feature distribution display shows that rules predicting class 1 largely refer to the first two quarters of the study period (roughly until the end of September 2020), whereas most class 4 rules pertain to the period from July 2020 to the end of January 2021. Rules for class 3 have the highest frequency for the second quarter, and it gradually decreases in quarters 3 and 4. Here we refer to the quarters of the pandemic time span because the value ranges of all features, including the feature `Days_passed`, have been consistently divided into 4 equal intervals. Hence, each rectangle of the topmost heatmap row represents one quarter of the period from April 2020 to May 2021. An illustration of how to interpret the heatmap display is provided in Fig. 15.

Notably, the distributions indicate that features representing disease levels in the preceding weeks, especially in week -2, are more influential for predictions than mobility features.

To interpret rule clusters visible in the projection, we interactively select regions in the projection containing the clusters. In Fig. 14 (middle row), the selection contains many class 4 rules (see the class 4 heatmap on the right). The heatmap display tells us that these rules mostly refer to the time from August 2020 onward and describe situations where a high pandemic level (e.g., level 3 in week -2) combined with high mobility led to further increase (class 4). This is visible from high color intensity of rectangle 3 in the heatmap rows corresponding to the mentioned features. In contrast, rules predicting class 2 (second

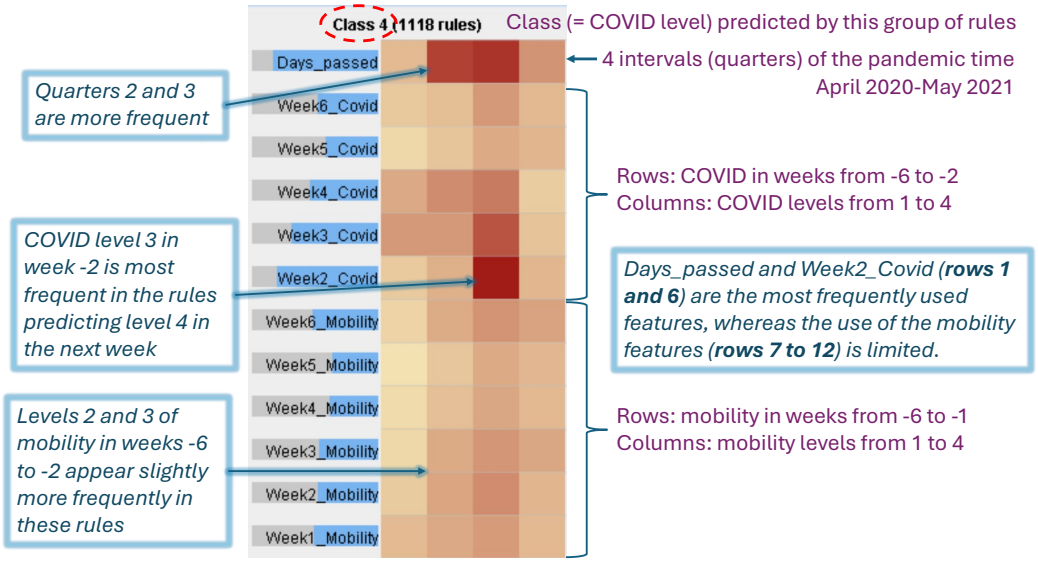


Figure 15: Explanation for interpreting the heatmap displays.

heatmap) in this period often featured lower mobility, consistent with expectations. In the bottom row, the selection contains mostly rules referring to spring–summer 2020 and capturing declines (2→1 or 3→2) typically preceded by decreasing mobility, consistent with early control dynamics.

Investigating the role of Days_passed. The distribution overview in Fig. 14 showed that `Days_passed`, the temporal progression feature, is among the most frequently used conditions across all classes. To explore how the model behaves without this feature, we selected only the rules not involving `Days_passed` (784 rules). As shown in Fig. 16, the distributions of other features remain similar to the full rule set; however, mobility features, particularly for class 2, exhibit more distinct usage patterns. This suggests that mobility features can play non-negligible role in determining trends.

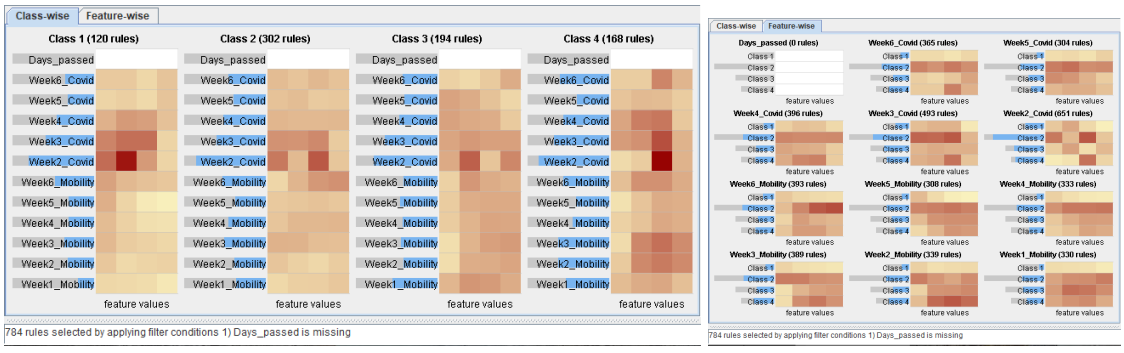


Figure 16: Feature distributions over rules that do not involve Days_passed.

We further segmented the rule set into four periods based on the values of `Days_passed` to examine temporal shifts in model behavior (Fig. 17). The first segment (days 45–134, roughly April–June 2020) is dominated by class 1 rules, corresponding to early post-lockdown recovery

Figure 17: Feature distributions in rules grouped by different ranges of `Days_passed`.

and decreasing incidence. In the second period (days 135–211, July–mid-September), class 2 and 3 predictions increase, while mobility also rises. The third segment (days 212–349, mid-September 2020 to January 2021) marks the peak of the second wave, with class 4 dominating and class 1 disappearing entirely. The final segment (days 350–431, February–May 2021) shows a decline in class 4 and a resurgence of class 2 and 3 predictions, despite continued high mobility. These patterns illustrate how the model adapts to pandemic evolution and shifting behavioral norms over time.

Feature interrelationships via topics We utilize NMF-based topic modeling to uncover patterns of feature co-occurrence and potential interdependencies. Using an empirically selected topic count, we receive the set of topics shown in Fig. 18, left. the histogram in Fig. 18 depicts how often each topic is dominant by predicted level. Topic 0 is rare, indicating infrequent patterns with minor predictive impact.

Most topics capture within-group relationships (COVID or mobility), with weak cross-group links. Topics 6 and 7 encode stable incidence at levels 4 and 1, respectively, over five weeks (late vs. early period). Topic 6 (sustained high incidence, varied mobility at weeks $-5/-4$) most often predicts level 3, less often level 2. This is visible from the relative sizes of the orange and yellow segments of the 7th bar of the histogram in Fig. 18, right (note that the first, very low bar corresponds to topic 0). Topic 7 frequently precedes transitions to level 2 (the yellow segment representing level 2 is the largest in the 8th bar corresponding to topic 7), but also appears with levels 3 (orange segment) and 1 (green segment).

Topics 1, 2, 3, 4, 8, and 9 reflect temporal COVID trends, which is visible from the positions of the darker rectangles in the glyphs corresponding to the COVID features, which represent

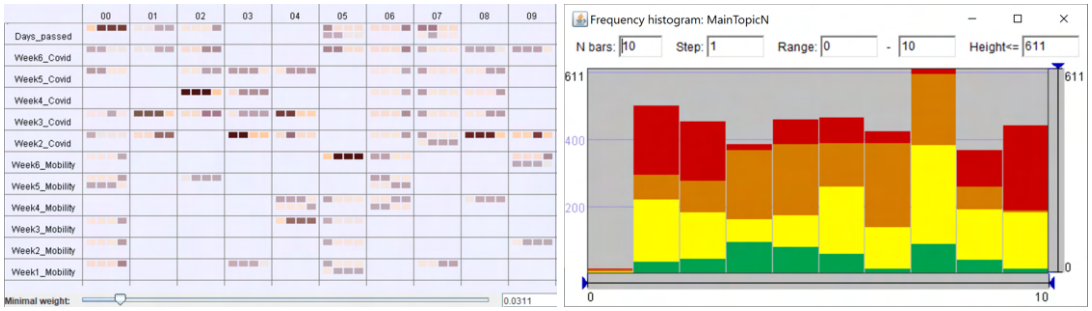


Figure 18: Left: Topic–term matrix for the COVID-19 rule set. Right: Histogram of dominant topic counts across rules, segmented by predicted disease level: 1 (green), 2 (yellow), 3 (orange), and 4 (red).

the COVID l;evels in weeks from -6 to -2. Topics 1 and 2 show high incidence at week -6, a temporary dip, then a rise, appearing in rules for levels 2 and 4. Topics 3/8 rise from weeks -6 to -4 then slightly decline by week -2. Topic 9 pairs high incidence at week -2 with lower levels at week -6. Overall, similar temporal patterns can lead to different outcomes.

Only topics 0 and 5 involve mobility relationships; with topic 0 rare, topic 5 (drop to mobility level 1, early period) is more relevant. Its rules mostly predict level 2 but also levels 3, 1, and 4, indicating mobility reduction is not strongly predictive.

For extreme outcomes, level 1 rules often show decreasing incidence (topics 3/4), consistently low incidence (topic 7), or low mobility (topic 5). Level 4 predictions align with increasing incidence (topics 1, 2, 9), though topic 8 (slight recent decline) can also lead to level 4.

In sum, the model primarily captures temporal dynamics in COVID incidence, with weak links to mobility. Topic patterns highlight decision-logic complexity and complement direct rule filtering and visual inspection.

Summary The model logic reflects evolving pandemic phases: mobility had a larger role early on, while recent disease levels dominated later. Topics capture temporal dynamics (sustained levels, rebounds) and clarify that similar trajectories can lead to different outcomes depending on context.

3.1.7 Lessons Learned

- Model-wide logic inspection is essential. High accuracy can mask implausible or contradictory rules; auditing the rule set directly improves trustworthiness.
- Data-optional auditing adds value. RuleSense supports third-party or restricted-data settings by focusing on inherent rule semantics.
- Domain knowledge guides effective cleanup. Removing rules that contradict fundamental domain logic can simplify models without harming performance.
- Visual overviews accelerate sensemaking. Class- and feature-wise heatmaps quickly reveal missing or overused features and implausible value ranges.

- Topic modeling complements interaction. Topics succinctly capture recurrent condition patterns and heterogeneity that are hard to see via manual filtering alone.
- Heterogeneous classes need structure-aware views. For diverse behaviors (e.g., trawling), topic- and projection-based views expose subpatterns better than single metrics.
- Iterative, human-in-the-loop refinement is practical. Lightweight cycles of filter–inspect–edit–validate can produce clearer, more defensible models.

3.2 Rules vs. SHAP: Complementary Model Understanding

Rule-based and feature-attribution explanations offer different but complementary views on model behavior. Rules expose the model's logical structure; SHAP quantifies per-feature influence on predictions. Used together in a visual analytics workflow, they enable cross-validation of insights, faster detection of inconsistencies, and more trustworthy refinement of model logic.

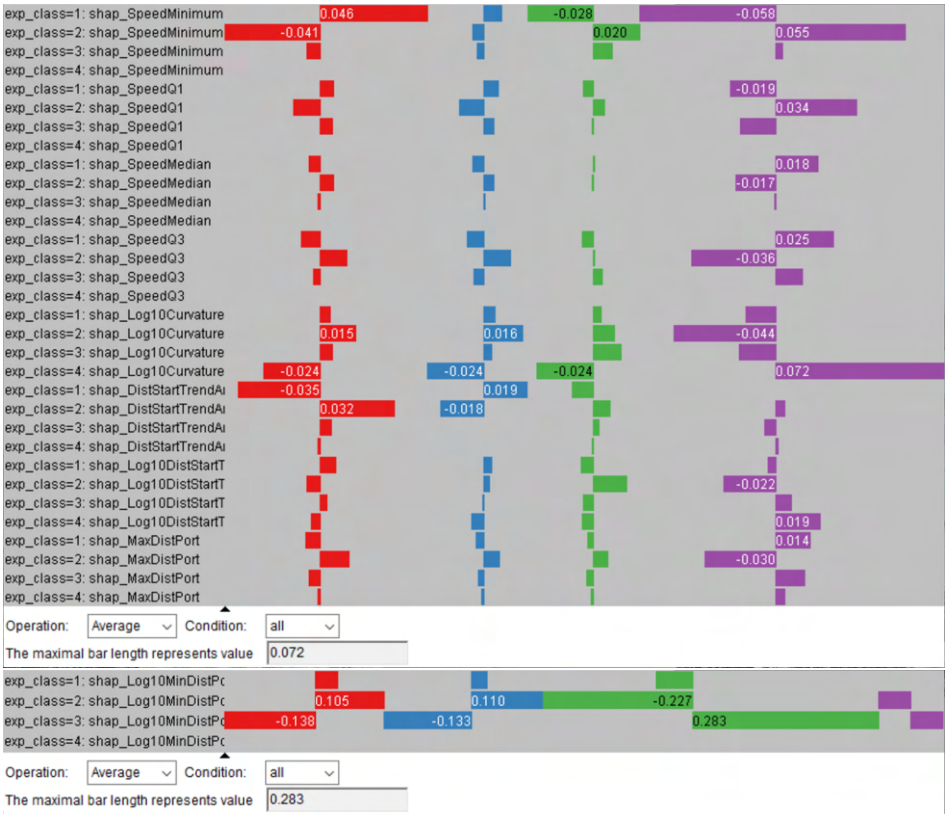


Figure 19: Aggregated multi-class SHAP values by true class. The colours correspond to the behavior classes as specified by the legend in Fig. 12.

3.2.1 Approach (case study: vessel movement).

We extract rules from a random forest (four classes; nine features) and visualize feature-interval distributions (Fig. 8). Using a labeled test set, we compute multi-class SHAP values per instance and aggregate them to: (i) compare class-wise SHAP patterns (Fig. 19); (ii)

inspect pairwise SHAP differences for critical features (Fig. 20); (iii) review SHAP summary plots for Classes 1 and 2 (Fig. 21).

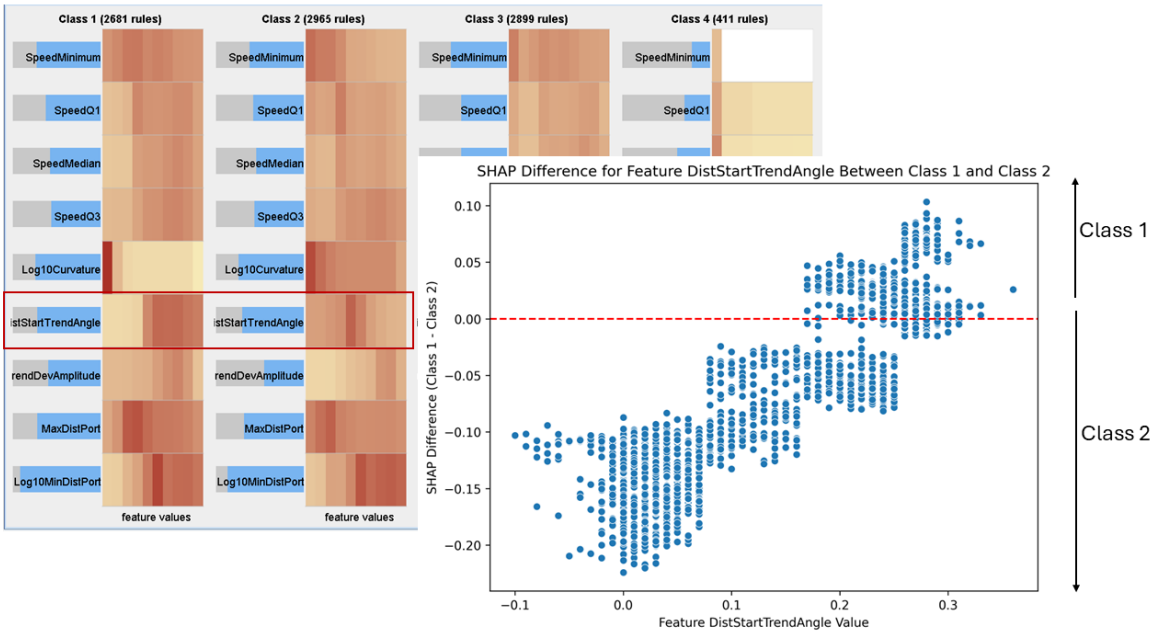


Figure 20: Pairwise SHAP difference (Class 1 minus Class 2) for DistStartTrendAngle vs. feature values.

3.2.2 Key observations.

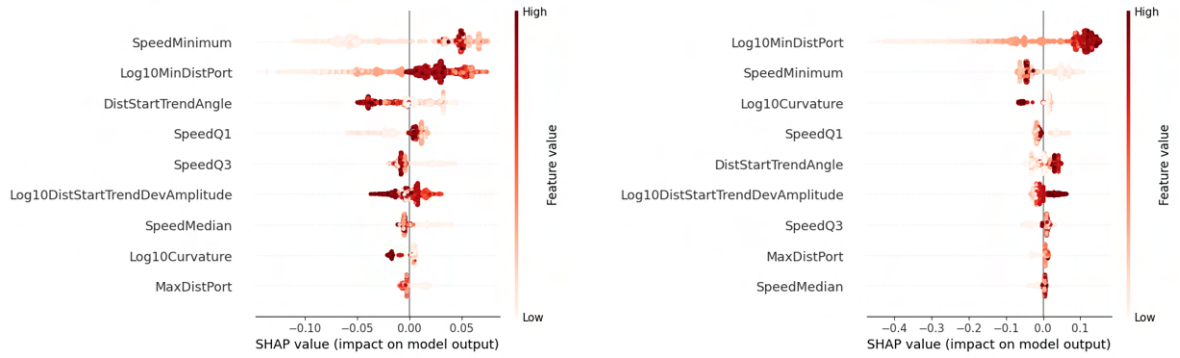
- Broad alignment: port-distance features show consistent importance across rules and SHAP.
- Notable discrepancy: DistStartTrendAngle appears negative for Class 1 in aggregated SHAP bars but rules favor higher values. Pairwise SHAP differences clarify that low values favor Class 2 while higher values favor Class 1; mixed regions suggest strong feature interactions.

Implication: single-method explanations can mislead; combining rules and SHAP reveals nuance and supports logic verification.

3.2.3 Discussion

We have designed an interactive workflow that combines rule-based and SHAP-based explanations to exploit their complementarity: rules expose the model's logical structure, while SHAP quantifies feature influence on predictions. Relying on a single method can mislead. For example, DistStartTrendAngle shows contradictory global vs. local SHAP patterns, reflecting correlated features and non-linear interactions. Because SHAP isolates marginal contributions, it can assign counterintuitive signs under dependence; conversely, rules may overgeneralize or obscure interactions.

Integrating both views supports:



(a) Class 1 **(b) Class 2**

Figure 21: SHAP summary plots (per-instance contributions and feature values). The rows are ordered based on the feature importance for the respective classes.

- **Consistency validation:** agreement raises confidence in explanations.
- **Complexity identification:** disagreement flags interaction effects and regions needing deeper analysis.
- **Bias detection:** discrepancies can reveal hidden biases or oversimplified assumptions.

For model developers, the composite view enables targeted diagnostics: features frequent in rules but with low SHAP impact may be overused; alignment with domain knowledge can be checked explicitly. The interactive interface supports iterative refinement (e.g., pruning rules tied to unstable or counterintuitive attributions), improving transparency without harming performance. This stance aligns with critiques that SHAP, while mathematically sound, may depart from human intuitions; embedding SHAP in a multi-method interface helps users understand its limits and interpret results more critically.

3.2.4 Lessons Learned.

Combining rule-based analysis with SHAP in a visual analytics workflow improves interpretability and trust: single-method explanations can mislead, and disagreements often reveal feature interactions, correlations, or weakly defined decision boundaries. Model-wide, data-optional rule audits and interactive filtering help remove illogical or redundant rules without harming performance; topic modeling exposes recurrent condition patterns and class heterogeneity. Cross-validating rules with SHAP supports consistency checks, complexity discovery, and bias detection. Iterative, human-in-the-loop refinement aligns model logic with domain knowledge, yielding clearer and more defensible models.

3.2.5 Open Problems.

Challenges remain in stabilizing SHAP under correlated features and non-linear interactions; automating constraint checks for domain logic (e.g., contradiction/subsumption detection) at scale; selecting and validating topic counts and discretization schemes; defining quantitative consistency metrics between rule- and attribution-based explanations; extending the workflow beyond tree ensembles to deep models; ensuring reproducibility under data shifts; integrating

causal reasoning to move beyond correlational attributions; and supporting privacy-aware, federated auditing where data access is limited.

4 Visual Analytics for Decision Making under Uncertainty

The objective of this task is to help users of simulation and ML models to gain awareness and understanding of uncertainties involved in model forecasts. In the second period of the project, the focus of the work was on developing methods for detecting and investigating cross-impact patterns in bivariate time series. This work was performed in close collaboration with WP2, Task T2.2 "Health Crisis Use Case".

This chapter presents a visual analytics workflow for detecting stable cross-impact patterns in time series pairs. A sliding window technique computes multiple impact measures, including a novel Kendall's tau variant that tolerates minor fluctuations. Evaluating these measures across various time lags reveals dynamic relationships between time series. An interactive Ikat plot facilitates the exploration of impact distributions, helping identify intervals where specific cross-impacts remain stable (e.g., trends in one series followed by similar or opposite trends in another after a lag). These intervals are extracted as events, whose temporal (and, when applicable, spatial) distributions can be analyzed to uncover broader patterns across multiple time series pairs and over extended time spans. This includes identifying co-occurring cross-impacts and variations in cross-impact presence or type across different periods and data subsets. Experiments on real-world datasets demonstrate the framework's ability to isolate robust patterns, providing a scalable and interpretable approach to analyzing complex temporal dynamics.

This work supports decision making under uncertainty by extracting stable, interpretable evidence from noisy, nonstationary, and lagged time series relationships, and by quantifying how conclusions change under different parameter choices.

- Reduces ambiguity in data:
  - Tolerance-based Kendall's τ treats minor fluctuations as ties, focusing on substantive trends rather than noise.
  - Sliding windows and explicit lag exploration reveal time-varying lead–lag structures without assuming stationarity.
- Produces decision-relevant evidence:
  - Identifies intervals where one series' trends reliably precede another's (positive or negative impacts), informing timing and targeting of actions.
  - Extracts impact events and summarizes their temporal and (when available) spatial distributions to prioritize periods/regions with predictable responses.
  - Provides significance guidance and practical τ thresholds to convey confidence levels.
- Enables robustness and transparency:
  - Ikat plots give an at-a-glance overview of impacts across time and lags, supporting visual validation and hypothesis generation.

  - Sensitivity analysis over tolerance, window length, lag range, and impact thresholds helps assess the stability of findings and avoid overfitting to transient noise.

In the case study, the approach identified periods and regions where mobility changes were most likely to affect COVID-19 incidence (early waves), and where impacts weakened later. Such evidence can inform adaptive policy, resource allocation, and risk management under evolving conditions.

4.1 Introduction

In many real-world scenarios, dependencies between time series often manifest with varying lags due to economic, physical, or behavioral delays. For instance, gold and crude oil prices exhibit dynamic relationships influenced by inflation expectations, geopolitical events, and macroeconomic policies, where oil often responds with a lag to gold price shocks. Similarly, interest rates and stock market indices demonstrate lead-lag effects, as monetary policy changes influence equity markets over different time horizons. In meteorology, temperature and electricity consumption show seasonal dependencies, with energy demand lagging behind temperature shifts. In epidemics, lagged correlations have been found between the intensity of population mobility and the rate of new infections [9]. Identifying lagged relationships and assessing their strength is crucial for forecasting, risk management, and decision-making in various domains [59].

Detecting and understanding cross-impacts between time series in both temporal and spatial contexts is a fundamental challenge in data analysis. Identifying these relationships requires not only the computation of statistical impact measures over time but also careful consideration of different time lags. However, short-term fluctuations, periodic variations, and irregular trends complicate this process, necessitating robust methods for preprocessing and analysis.

We address these challenges by introducing a systematic approach that computes multiple impact measures over a sliding window with varying time lags. In addition to traditional measures, such as Pearson correlation, rank correlation, and Kendall's tau, we have introduced a novel variant of Kendall's tau that is less sensitive to minor temporal deviations in value sequences and thus more robust against small fluctuations.

These computations form a key part of our analytical workflow for detecting, extracting, and analyzing stable cross-impact patterns in datasets containing multiple bivariate time series. To facilitate informed selection and testing of impact measures and parameter settings, we developed an interactive visual interface including the novel Ikat plot, which is controlled by query widgets. This interface is primarily used in the initial phase of analysis. By experimenting with a few sample time series pairs, analysts can immediately see how detected patterns vary depenfing on different parameter settings. Once suitable settings are identified using the samples, i.e., the settings that yield patterns aligned with the analysis objectives, they are applied to the full dataset (i.e., all time series pairs) to detect similar patterns across all pairs. For example, the analyst may initially use the data from Madrid, Barcelona, and Seville to determine the appropriate settings and then use these settings to locate the patterns of interest in the data from all Spanish provinces. The resulting patterns are then extracted and represented as "impact events", which undergo further analysis as outlined in our proposed workflow (see Fig. 22). In this way, the Ikat plot acts as a tool for hypothesis generation

and parameter sensitivity analysis, enabling analysts to visually assess the stability and characteristics of cross-impacts before proceeding to automated event extraction.

Our main contributions thus include

- A novel, robust variant of Kendall's tau tailored for time series analysis that tolerates minor temporal fluctuations.
- The Ikat plot, an interactive visualization supporting parameter exploration and pattern detection in the initial workflow stage.
- An event extraction methodology applicable to large sets of time series pairs.
- A comprehensive event extraction and analysis workflow (Fig. 22) to identify significant cross-impact patterns.
- A proof-of-concept case study on epidemiological data, demonstrating the utility of the approach.

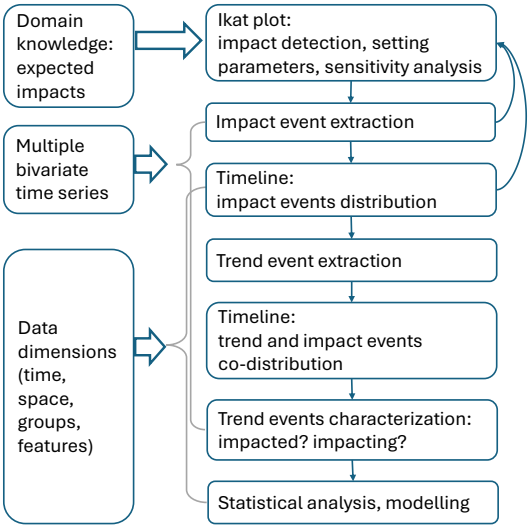


Figure 22: General workflow for detecting and analyzing impact events from multiple bivariate time series.

The remainder of this chapter is organized as follows. Section 4.2 introduces the base part of the methodology, including the computation of impact measures, the design of the interactive visualization, and the event creation process. Section 4.3 presents the entire workflow within a case study. Finally, Section 4.4 critically discusses the approach and outlines directions for future research.

4.2 Method

We begin this section by describing methods for quantifying the impact between pairs of time series of equal length. Next, we explore how to apply these measures to different segments of the time series with varying lags, effectively shifting one series relative to the other. We

Pair (i, j)	(x_i, x_j)	(y_i, y_j)	$x_i - x_j$	$y_i - y_j$	Standard Sign (s_1, s_2)	Standard Result	Below Tolerance?	Modified Sign (s_1, s_2)	Modified Result
(0,1)	(1.0, 2.5)	(1.1, 2.0)	-1.5	-0.9	(-1, -1)	Concordant	No	(-1, -1)	Concordant
(0,2)	(1.0, 2.9)	(1.1, 3.2)	-1.9	-2.1	(-1, -1)	Concordant	No	(-1, -1)	Concordant
(0,3)	(1.0, 4.2)	(1.1, 3.9)	-3.2	-2.8	(-1, -1)	Concordant	No	(-1, -1)	Concordant
(1,2)	(2.5, 2.9)	(2.0, 3.2)	-0.4	-1.2	(-1, -1)	Concordant	Yes (x within tol.)	(0, -1)	Ignored (Tie)
(1,3)	(2.5, 4.2)	(2.0, 3.9)	-1.7	-1.9	(-1, -1)	Concordant	No	(-1, -1)	Concordant
(2,3)	(2.9, 4.2)	(3.2, 3.9)	-1.3	-0.7	(-1, -1)	Concordant	Yes (y within tol.)	(-1, 0)	Ignored (Tie)

Table 7: Pairwise comparisons for standard and tolerance-based Kendall's Tau, assuming example time series $X = [1.0, 2.5, 2.9, 4.2]$, $Y = [1.1, 2.0, 3.2, 3.9]$, and tolerance thresholds $\epsilon_x = \epsilon_y = 0.8$.

then discuss how to visualize the results to provide an overall overview of the distribution of impact values across the entire time interval and within the desired range of lag values.

4.2.1 Kendall's Tau with Tolerance

Let's consider a simple example of two monotonically increasing time series $X = [1.0, 2.5, 2.9, 4.2]$ and $Y = [1.1, 2.0, 3.2, 3.9]$. For these data, the Pearson correlation is equal to 0.961, and Spearman rank correlation is 1, indicating perfect monotonicity. Pearson correlation measures linear relationships, and Spearman rank correlation assesses monotonicity based on rank differences. While Pearson correlation can be sensitive to outliers and Spearman correlation may not capture certain nonlinear patterns, Kendall's Tau [33] offers a more robust alternative by evaluating the concordance of paired observations. It compares the number of concordant and discordant pairs, where concordant pairs maintain the same relative order in both variables, while discordant pairs do not.

Kendall's Tau is particularly useful in time series analysis, as it is more resistant to noise, non-stationarity, and lagged dependencies, which are common issues in real-world data. In contrast, more complex measures, such as Granger causality [23], impose strict assumptions like stationarity, making them harder to apply and interpret. Formally, given a time series of length T, if c is the number of concordant pairs and d is the number of discordant pairs, Kendall's Tau is computed as:

$$\tau_n = \frac{c - d}{c + d} = \frac{2(c - d)}{T(T - 1)}$$

The value of Kendall's Tau ranges from -1 (perfect disagreement) to +1 (perfect agreement), with 0 indicating no clear relationship. For the given toy time series, all 6 pairs are concordant (Table 7, column "Standard result"), and therefore Kendall's Tau is equal to 1.

Kendall's Tau was originally designed for rankings, assuming that all distinct values differ significantly. However, when applied to continuous values, even small, often random fluctuations can be treated as distinct ranks, potentially distorting the results. To address this sensitivity to minor fluctuations, we introduce tolerance thresholds ϵ_x and ϵ_y for the X and Y series respectively. Differences smaller than these thresholds are treated as ties, effectively ignoring fluctuations that may be due to noise rather than meaningful trends.

The original Kendall's Tau algorithm and our extension are detailed in Algorithm 1, with modifications highlighted in gray. For example, in Table 7, pairs (1,2) and (2,3) have at least one difference below the tolerance threshold (0.8), causing them to be treated as

ties in the modified calculation. This yields a final value of $4/6 = 0.667$ (Table 7, column "Modified result").

The tolerance thresholds ϵ_x and ϵ_y can be specified in two ways: as absolute values (e.g., $\epsilon_x = 14.293$) or as percentages of the series value range. For percentage-based specification, the thresholds are computed as $\epsilon = (max - min) * percentage/100$, where min and max represent the minimum and maximum values of the respective time series. To ensure robust threshold calculation in the presence of outliers, the system optionally uses percentile-based range estimation (e.g., 5th to 95th percentiles) instead of absolute min-max values, preventing extreme values from distorting the tolerance calibration.

Algorithm 1 Compute Kendall's Tau with Tolerance

Require: Two input arrays X and Y of equal length T
Require: Tolerance parameters ϵ_x and ϵ_y
Ensure: Kendall's Tau coefficient with tolerance

if $n \leq 1$ **then**
  return 0
end if
$concordant \leftarrow 0$, $discordant \leftarrow 0$ ▷ Initialization
for $i \leftarrow 0$ to $T - 2$ **do**
  for $j \leftarrow i + 1$ to $T - 1$ **do**
    $diff_x \leftarrow X[i] - X[j]$
    $diff_y \leftarrow Y[i] - Y[j]$
    $tied_x \leftarrow |diff_x| < \epsilon_x$
    $tied_y \leftarrow |diff_y| < \epsilon_y$
    if $tied_x$ **and** $tied_y$ **then** ▷ Both are within tolerance
      continue ▷ Ignore the tied pair
    end if
    $sign_x \leftarrow$ sgn$(diff_x)$ if not $tied_x$ else 0
    $sign_y \leftarrow$ sgn$(diff_y)$ if not $tied_y$ else 0
    if $sign_x = sign_y$ **then**
      $concordant \leftarrow concordant + 1$
    else if $sign_x \neq 0$ **and** $sign_y \neq 0$ **then**
      $discordant \leftarrow discordant + 1$
    end if
  end for
end for
$totalPairs \leftarrow \frac{T(T-1)}{2}$ ▷ Compute total number of possible pairs
$\tau \leftarrow \frac{concordant - discordant}{totalPairs}$ ▷ Compute Kendall's Tau
return τ

$$\sigma_\tau = \sqrt{\frac{2(2T+5)}{9T(T-1)}} \tag{1}$$

In statistics, the significance thresholds for Kendall's Tau were determined using the asymptotic

variance of the statistic for large T under the null hypothesis of independence. In this asymptotic case the distribution for τ, under the null hypothesis, converges to a normal distribution around $\tau = 0$ with a variance given by Equation (1). Using, for example, a significance level of $p = 0.05$ (two-tailed), values of τ above or below approximately $\pm 1.96\sigma_\tau$ are statistically significant.

The introduction of tolerance thresholds ϵ_x and ϵ_y in our modified Kendall's Tau affects the calculation by effectively increasing the number of 'tied' pairs. When differences between data points fall within these tolerances, they are treated as ties rather than strict increases or decreases. This increased presence of ties in the data has a known statistical effect on Kendall's Tau: it leads to a larger standard error (σ_τ) and, consequently, requires larger absolute values of τ to reach statistical significance. In essence, with tolerance, the measure becomes more conservative, requiring stronger evidence of a monotonic relationship to be deemed significant [34, 53, 52]. While a precise analytical formula for the variance of Kendall's Tau with arbitrary tolerances can be complex and is beyond the scope of this paper, our experimental evaluation empirically accounts for this effect by observing the behavior of the metric and adjusting our interpretation of significance thresholds accordingly based on the specific tolerance values used.

Based on Equation 1, Figure 23 illustrates the critical values for $|\tau|$ across a range of time series lengths (T) from approximately 5 to 50, and for different significance levels ($p = 0.01, 0.05$). These thresholds represent the minimum absolute Kendall's Tau value required for a correlation to be considered statistically significant at the respective p-value, assuming no ties in the data.

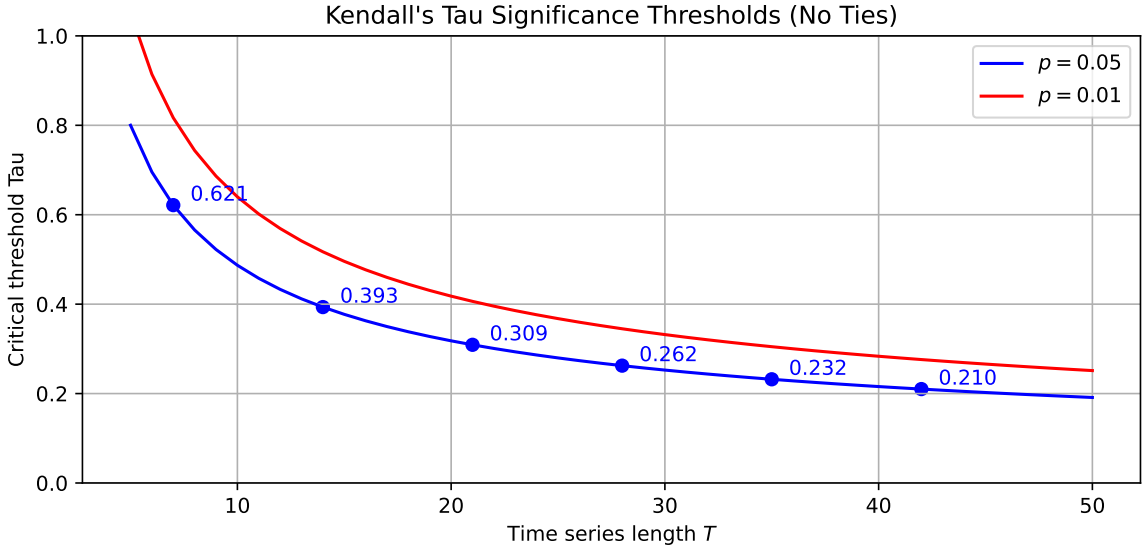


Figure 23: Critical thresholds of Kendall's Tau for $p = 0.05$ and $p = 0.01$ without tolerance. Marked points correspond to selected T values of $7, 14, 21, 28, 35, 42$ on the $p = 0.05$ line.

It is important to note that the critical values presented here serve as general indicators of desired statistical significance at given p under assumptions such as approximate normality

Parameter	Description	Selection Guidelines	Impact on Results
Tolerance ε_x, thresholds ε_y	Minimum difference to treat as meaningful change rather than tie	Start with 3–5% of series range; Increase for noisy data; Decrease for subtle pattern detection	Higher values $\rightarrow$ more conservative, fewer significant patterns
Time window length T^W	Number of consecutive time points used for τ calculation	Align with natural cycles (e.g., 7, 14 days); Balance smoothing vs. temporal resolution; Consider domain-specific periodicities	Larger $T^W \rightarrow$ smoother patterns, less sensitivity to short-term fluctuations
Maximum lag L	Maximum time delay considered between base and response series	Set based on expected influence delays; Consider domain knowledge; Align with temporal cycles	Larger $L \rightarrow$ increased computational cost, potential for spurious long-term patterns
Significance threshold $\lvert\tau\rvert$	Minimum absolute tau value for statistical significance	Use Fig. 23 as starting point; Adjust upward when tolerance creates many ties; Validate empirically with data	Lower thresholds $\rightarrow$ more detected patterns, higher false positive risk

Table 8: Summary of Key Parameters for Kendall's Tau with Tolerance

of Kendall's Tau distribution. In practical applications involving real data, the distribution of τ values may deviate from theoretical assumptions due to factors like non-normality, autocorrelation, or complex dependence structures. Therefore, analysts need to assess empirically the distribution of τ in their specific context and adjust significance thresholds accordingly. Interactive exploration of different tolerance values, as demonstrated in our implementation, helps identify appropriate thresholds that balance noise reduction with pattern sensitivity.

Generally, we analyze a pair of time series of length T, denoted as $X = \{x_0, x_1, \ldots, x_{T-1}\}$ and $Y = \{y_0, y_1, \ldots, y_{T-1}\}$, using a time window of length T^W and considering time lags l in the range $0, 1, \ldots, L$.

The selection of the time window T^W should be guided by domain knowledge and respect relevant temporal cycles inherent in the data. For instance, many human activities exhibit weekly patterns due to work schedules, business hours, and social routines, which are reflected in corresponding time series data. Financial markets demonstrate similar cyclicality with trading patterns that vary between weekdays and weekends, while epidemiological data may show weekly reporting cycles due to administrative practices. Therefore, choosing T^W to align with these natural periodicities - such as multiples of 7 days for weekly cycles - helps ensure that the computed impact measures capture meaningful relationships rather than artifacts of temporal misalignment.

The maximum lag L should reflect the anticipated range of influence delays as estimated by domain experts, representing the longest reasonable time period over which one series might

affect another. In many applications, L should be aligned with the length of relevant temporal cycles; for example, in economic data with weekly patterns, setting L to multiples of 7 days ensures that lagged relationships are captured across complete cycles. This domain-informed selection prevents both computational overhead from examining irrelevantly long lags and the risk of missing meaningful delayed impacts that occur within expected timeframes.

Table 8 summarizes the key parameters and provides guidance for setting these parameters. The same information is presented in the user interface for guiding parameters tuning, see an example in Fig. 24.

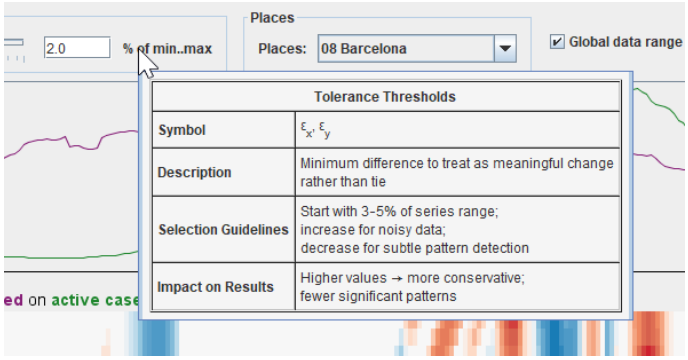


Figure 24: An example of a tooltip-based presentation of parameter guidance directly in the user interface, aligned with Table 8.

After setting appropriate parameter values, we compute the matrix

$$\tau = \{\tau_{tl}\} = \{\tau([x_t, x_{t+1}, \ldots, x_{t+T^W-1}], [y_{t+l}, y_{t+l+1}, \ldots, y_{t+l+T^W-1}])\}$$

where l varies within the interval $[0, L]$ and t ranges from 0 to $T - T^W - L$. This is achieved by shifting Y back in time by l time steps, applying a sliding window of length T^W over both X and shifted Y, and computing the τ value for the time series segments contained in the window. We refer to X as the **base** time series and Y as the **response** time series, emphasizing that Y is expected to react to earlier changes in X.

In the following subsection, we present the method with an illustrative example using daily time series of two financial indicators: crude oil futures *"CL=F"* and gold futures *"GC=F"* over the years 2023–2024. These data were extracted from finance.yahoo.com using the `yfinance` Python library.

4.2.2 Interactive Visualization of Impact Distribution

After computing impact measures over the selected time window for all possible time steps and lags, we aim at analyzing their distributions and identifying meaningful patterns. These objectives lead to the following desiderata for the interactive visualization of impact values:

- **D1**. Provide an overview of dynamics of τ over the timeline and across different lags to assess the stability and depth of patterns.
- **D2**. Visually link impact values computed over the time window to corresponding fragments of the base and response time series for contextual understanding.

- **D3**. Validate the results of τ calculations by showing the dynamics of the values involved in the computation.
- **D4**. Enable identification of periods characterized by specific properties, such as value ranges of the base and response time series, their dynamics, and impact values.
- **D5**. Support the extraction of events that contain identified behavioral patterns of interest, i.e., creation of standalone data items that can be handled separately from the original time series.

In our illustrative example, we consider gold as the base time series and crude oil as the response time series. These time series are shown in the top line plot in Fig. 25. Below it are two images of the ***Ikat plot*** representing the $T \times L$ matrices of τ values computed over the same T^W using the original Kendall's Tau and our tolerance-adjusted variant, with $\epsilon_x = \epsilon_y$ set to 3% of the variables' value ranges.

The Ikat plot displays the distribution of impact patterns over time and across various lags (**D1**). Negative impact values are shown in shades of blue, positive in shades of red, and impact absence in white. Introducing tolerance reduces the intensity of colors in regions of minor changes or fluctuations in the time series, as small variations are treated as ties when computing τ. This results in lower impact values, replacing saturated red and blue by white or near-white shades.

The results of impact calculation depend on the chosen time window T^W. To determine the optimal window for specific data and analysis objectives, multiple Ikat plots need to be visually compared. Figure 25 demonstrates the influence of T^W by presenting plots for $T^W = 7, 14, 21, 28$ chosen according to the weekly cycles of business activities. For $T^W = 7$, the plot exhibits small, scattered patterns, likely caused by fluctuations in the unsmoothed daily time series. As T^W increases, color variations become smoother across the matrix, revealing patterns at different scales.

All variants of the Ikat plot reveal an interplay between two basic geometric patterns: vertical and diagonal stripes. Vertical stripes appear when a short trend in the base time series corresponds to a prolonged trend in the response time series, resulting in sustained high impact values across an extended range of time lags (i.e., along the vertical axis). Diagonal stripes emerge when the response time series exhibits a short-lived trend. In this case, the impact weakens as the time lag increases, causing the depth of the effect along the Y-axis to diminish with each step along the X-axis. Vertical interruptions within diagonal patterns occur when the base time series has a flat segment or undergoes a trend reversal. A typology of patterns that can be observed in the Ikat plot is presented in Table 9.

Figure 29 presents the user interface designed to support the required functions. The Ikat plot provides an overview of τ dynamics (**D1**), while visual linkage to time periods (**D2**) and corresponding fragments of the base and response time series (**D3**) is supported through coordination with the time graph display (top) and the scatter plot (right). Figure 26 illustrates how the UI explains the computed impact for a matrix cell under the cursor, displaying all relevant details in a popup window and in the scatterplot area.

To enable identification of periods characterized by specific properties (**D4**), we support

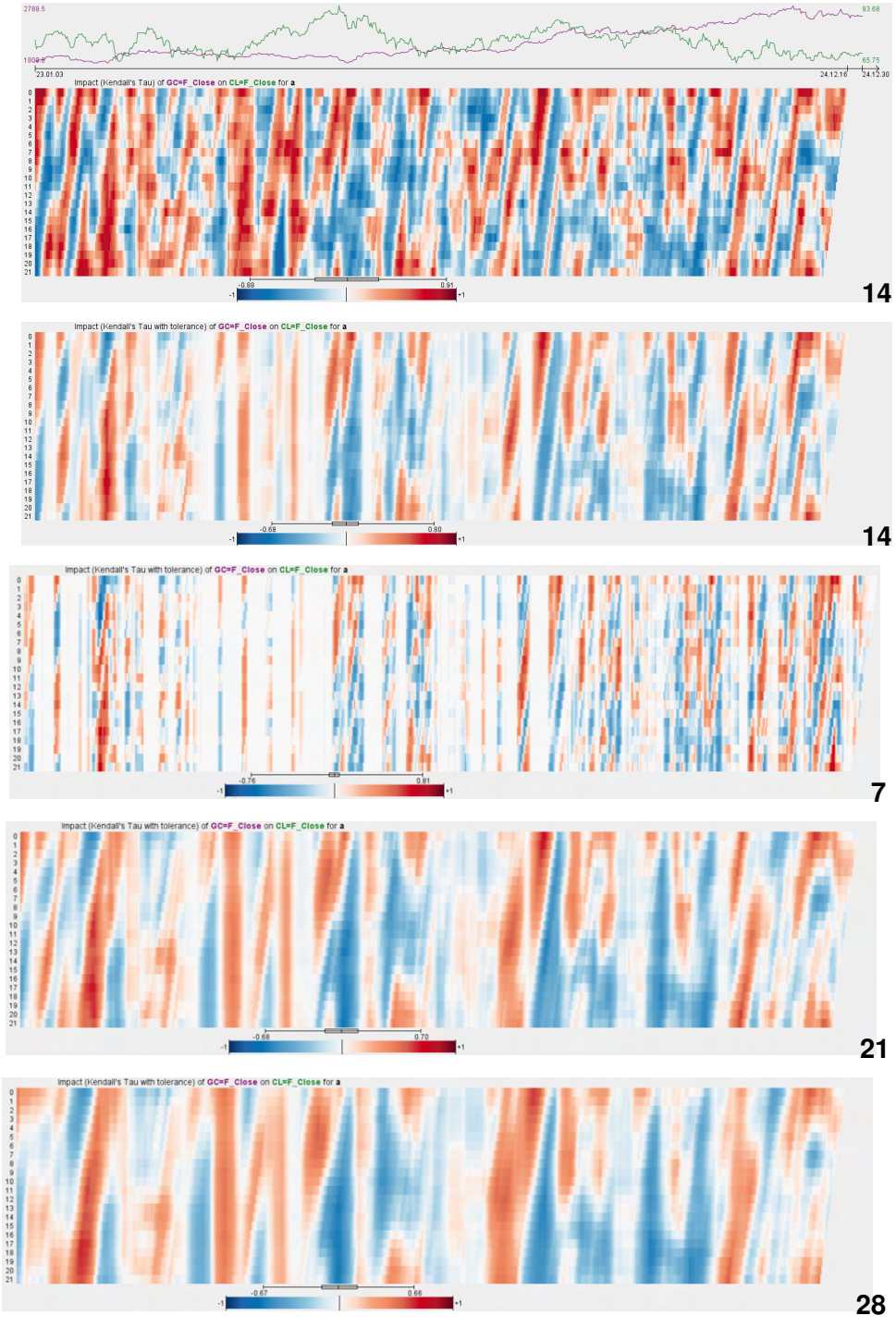


Figure 25: The line plot at the top shows time series of gold and crude oil futures prices. Below it, five Ikat plots represent the dynamics of impact values over time (horizontal axis) and across lags from 0 to 21 days (vertical axis). The top Ikat plot displays the original Kendall's Tau values, and the remaining four show Kendall's Tau with a 3% tolerance. The numbers in the lower right corners are the sliding window lengths in days.

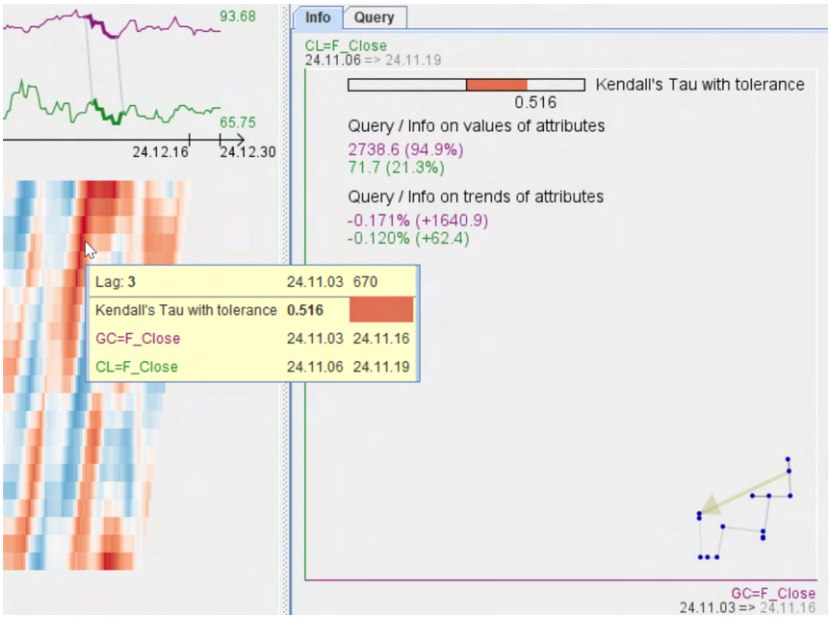


Figure 26: Fragment of the Ikat plot in Fig. 25, demonstrating the impact on November 03, 2024, with $lag = 3$ days. The corresponding segments of the base and response time series are highlighted in bold in the time graph above, and local statistics is provided below. A sequence of points corresponding to the time window is shown in the scatter plot to the right, with gold on the horizontal axis and crude oil on the vertical axis. A directed arrow connects the first and last points in the time window.

Figure 27: Ikat plot (Figs. 25,26) after applying a dynamic query, as shown on the right. Cells of the τ matrix that do not meet the query conditions are visually de-emphasized.

several kinds of queries:

- **Time** queries restrict the time interval and/or time lags.
- **Level** queries restrict the matrix to time intervals when values of base or response time

Figure 28: Different kinds of queries constrain results in different ways.

series fall within the specified range.

- **Trend** queries restrict the matrix to time intervals when trends of base or response time series fall within the specified range. Trend is computed as difference between median values in last and first thirds of the time window:

$$median(\texttt{idx} + \lceil 2/3\, T^W \rceil) - median(\texttt{idx})$$

where $median(\texttt{idx})$ calculates the median of the time series in a window of length $\lfloor T^W/3 \rfloor$ starting from index `idx`.

- **Impact** queries restrict the matrix to cells with impact values within the specified range.

For level and trend queries, values are linearly normalized to $[0, 1]$ using the min and max values of the entire time series, similar to how the tolerance threshold is set. Conditions are specified through dynamic query sliders, possibly using "Not" modifiers. Several types of conditions can be combined. For example, Fig. 27 displays the result of combining one level condition ($gold \geq 50\%$), one trend condition ($trend(gold) \geq 1\%$), and one impact condition $|\text{impact}| \geq 0.3$. Figure 28 explains how different types of query conditions affect the selection of regions in the τ matrix.

Filtered-out cells do not disappear completely from the plot. Instead, they are shown as narrow colored stripes, contrasting with the fully covered cells that satisfy the query. This design choice ensures that the context of filtered-out data remains visible while maintaining a clear distinction between selected and non-selected cells. Unlike alternatives such as opacity or blur, narrow stripes preserve the original color encoding and do not distort the perceived data magnitude [58]. Adjusting opacity can unintentionally change the perceived intensity of color, which is already mapped to quantitative values in our heatmaps, potentially leading to misinterpretation. Similarly, blur introduces spatial uncertainty and can obscure boundaries,

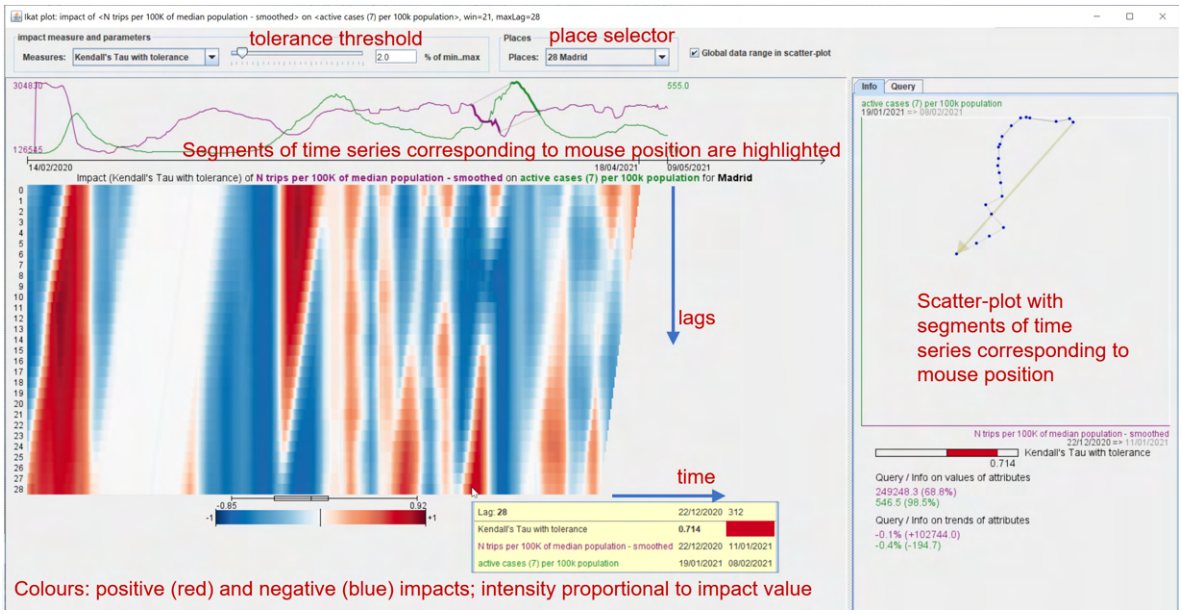


Figure 29: Visualization of impact values. The control panel, located at the top, allows for selecting time series, impact measures to apply, and setting parameters such as the tolerance threshold. The central section features a plot of the base and response time series, aligned with the impact matrix. This matrix displays impact values along the timeline and across various time lags in the form of an Ikat plot. The plot is named after a Southeast Asian dyeing technique used to pattern textiles. For any selected cell in the Ikat plot, the corresponding time series fragments are highlighted in bold on the timelines above the plot and presented on the scatter plot to the right, with a directed arrow indicating the progression from the first to the last point.

Pattern Type	Visual Representation	Inferred Meaning	Example
Sustained Impact	Vertical stripes across multiple lags.	A specific event or trend in the base time series is followed by a prolonged, consistent trend in the response series.	A sustained marketing campaign leads to a prolonged increase in sales over many weeks.
Transient Impact	Diagonal stripes, often fading with increasing lag.	A short-lived trend in the base series leads to a short-lived trend in the response series. The impact is strongest at a specific lag.	A sudden weather event (e.g., a cold snap) causes a spike in energy consumption that lasts only for a few days.
Lag-Specific Impact	Horizontal bands at a particular lag value.	The relationship between the time series is stable and consistent over a long period, but only for a very specific time delay.	A factory production increase consistently leads to a downstream delivery increase at a fixed lag of 5 days.
Episodic Impact	Localized "hotspots" or small, isolated colored patches.	A transient, short-lived impact that is only significant at a specific time step and for a narrow range of lags.	A single, one-time policy announcement leads to a very strong but short-lived impact on stock market volatility.
Stable Trend-Driven Impact	Large, contiguous areas of uniform color.	A strong, stable, and persistent relationship between the two time series across a wide range of time steps and lags.	A fundamental economic relationship where prices consistently rise and fall together over a multi-year period.
Periods of "Noise"	White or near-white regions, interrupting other patterns.	Indicates a lack of significant impact or a period where the relationship is too weak to be considered significant.	A period of economic stability where fluctuations in one time series have no significant impact on the other.

Table 9: Typology of Cross-Impact Patterns Observable on the Ikat Plot

making it harder to discern fine-grained patterns [41].

Moreover, reducing the occupied area through stripes provides a discrete, pre-attentive visual cue [11, 12] that scales well in dense and sparse regions. This approach avoids the visual blending and overplotting issues that opacity-based techniques can create, particularly in datasets with high data density. By maintaining both the color and shape distinction, the stripes offer a more robust contextual view of the filtered-out data. This is especially useful when investigating query sensitivity, as analysts can still perceive the distribution and density of filtered-out cells relative to the selected cells. Previous work on visualization design emphasizes the importance of preserving context without overpowering the primary data and maintaining perceptual separability between encodings[41, 15]. Our choice of narrow stripes directly supports these principles by allowing users to analyze patterns and gaps in the data without introducing perceptual ambiguity.

4.2.3 Event Extraction

As stated in the introduction, the visual query interface centered on the Ikat plot is used at the initial stage of the analysis workflow (Fig. 22). This enables the analyst to select parameter settings that reveal the kind of impact patterns aligned with the analysis goals. The effects of these parameter settings are tested on several pairs of time series selected from a large dataset. Once the analyst identifies the settings capable of capturing the patterns sought, these settings are used by an automatic method to extract *impact events* from the entire dataset (**D5**).

An impact event is a standalone temporal object defined by the temporal attributes t_1, t_2, l_1, l_2, where $[t_1, t_2]$ denotes the interval in the base time series that exerts an impact, and $[l_1, l_2]$ specifies the corresponding interval in the response time series, expressed as time lags relative to the start time t_1. Geometrically, these four attributes define a rectangular region within the matrix where columns represent time steps in the base time series and rows represent time lags (as in Ikat plot, which is a visual representation of such a matrix).

Algorithm 2 Extract Events

Input: Boolean matrix b
Output: List of events $events$
$events \leftarrow []$ ▷ Initialize an empty list of events
Initialize a 2D boolean array $visited$ with the same dimensions as b
for each row r in b **do**
    for each column c in $b[r]$ **do**
        if $b[r][c] =$ true **and** $visited[r][c] =$ false **then**
            $event \leftarrow$ new Event() ▷ Create a new Event object
            dfs(b, visited, r, c, event) ▷ Depth-First Search
            if isValidEvent$(event, b)$ **then**
                $events \leftarrow events \cup \{event\}$ ▷ Add event to list
            end if
        end if
    end for
end for
return $events$

To find candidate regions for constructing events, the method computes, for each matrix

Algorithm 3 Depth-First Search for Event Exploration

1: **Input:** Boolean matrix b, Visited matrix $visited$, row, column, Event object $event$
2: **Output:** Updated Event object
3: **if** row is out of bounds **or** col is out of bounds **or** visited[row][col] = true **or** b[row][col] = false **then return**
4: **end if**
5: $visited[row][col] = \text{true}$ ▷ Mark the current cell as visited
6: $event.coordinates.add(\text{new Point}(col, row))$ ▷ Add the current cell to the event
▷ Explore neighbors (Up, Down, Left, Right)
7: dfs(b, visited, row - 1, col, event) ▷ Up
8: dfs(b, visited, row + 1, col, event) ▷ Down
9: dfs(b, visited, row, col - 1, event) ▷ Left
10: dfs(b, visited, row, col + 1, event) ▷ Right

cell, a binary value $b[i][j]$ that is $true$ if cell (i, j) satisfies all query conditions. The resulting boolean matrix b is supplied to Algorithm 2, which recursively calls Algorithm 3, to identify contiguous regions containing $true$ values. The function $isValidEvent$ checks whether the detected regions satisfy additional query constraints specifying the minimal event duration (the horizontal extent), minimal depth (the range of lags covered), and numeric characteristics of the event's shape. If so, these regions are used to construct events.

In addition to the temporal attributes, each event is described by several features, including the base and response trend parameters and the value of the chosen impact measure.

Parameter sensitivity analysis As parameter settings for event extraction are typically identified using selected pairs of time series, the analyst may be unsure whether these settings will generalize to all remaining data, or whether small variations in parameter values might lead to significant differences in event extraction results. To address this, an automatic procedure can be run for each parameter, iteratively varying its value v within a range $[v - \eta, v + \eta]$, using an increment δ (where η and δ are specified by the analyst). At each step i, the procedure performs event extraction using the current parameter value v_i, recording the number of extracted events and aggregate statistics of event features. The resulting sequences of numeric indicators can be visualized to assess the effects of parameter variation.

Typically, as thresholds (e.g., tolerance, impact strength, or trend inclination) increase, the counts of extracted events and their lengths and depths decrease. A line graph visualization (as in Fig. 30) can reveal bend points where the rate of change increases. Such bend points indicate critical parameter values that merit further interactive examination using the Ikat plot. Patterns visible in the Ikat plot then guide the analyst in selecting the most appropriate critical value, based on domain knowledge and the specific task requirements.

Figure 30 demonstrates the sensitivity analysis procedure. For the time series used in our running example, we systematically varied the tolerance thresholds ϵ_x and ϵ_y within the intervals suggested by Fig. 23, applying the same value to both, assuming similar behavior in the base and response time series. The line graphs in Fig. 30 represent trends in the number of extracted events, total event size, and coverage metrics as tolerance thresholds increase. The lines suggest that the Ikat plot should be used to explore and compare patterns

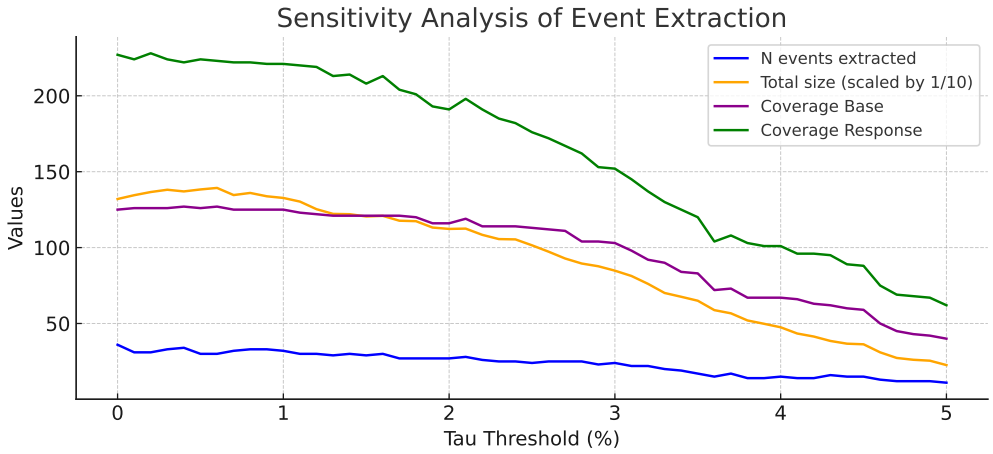


Figure 30: Sensitivity of the event extraction procedure to the tolerance.

at threshold values of 2% and 3% for making the final choice.

4.2.4 Conceptual Comparison with Existing Approaches

Before moving to the case study, we briefly compare our Ikat plot-based methodology to established approaches for analyzing relationships between time series pairs, particularly Cross-Recurrence Plots (CRP) and Dynamic Time Warping (DTW). While these methods have been widely adopted, the Ikat plot approach differs in several essential conceptual ways:

- **From Similarity to Impact.** CRP and DTW focus on quantifying similarity or alignment between time series, typically capturing periods where both series evolve in similar ways. In contrast, the Ikat plot is designed to reveal *impact* relationships, which may be positive (e.g., increases in one series lead to increases in another) or negative (e.g., increases in one series lead to decreases in the other). This allows for the detection of asymmetric and potentially causal relationships, rather than just co-movement.
- **Explicit Time Lag Consideration.** While CRP and DTW can capture temporal misalignments, they do not directly expose or analyze the range of possible time lags at which impacts may occur. The Ikat plot explicitly maps impact strength across a user-defined range of lags, enabling the discovery of lead-lag effects and delayed responses.
- **Reduced Sensitivity to Noise.** Standard similarity-based methods can be overly sensitive to short-term fluctuations and noise, which may obscure meaningful patterns. Our use of a tolerance-based Kendall's tau measure within the Ikat plot framework treats minor fluctuations as ties, focusing attention on substantial, stable impact patterns.
- **Interpretability of Patterns.** The visual encoding in CRP and DTW often emphasizes global structure or synchronization, but may lack direct interpretability for complex lagged relationships. The Ikat plot, by contrast, enables analysts to recognize and interpret local, lag-specific, and temporally stable cross-impacts through intuitive geometric patterns (e.g., vertical, diagonal stripes, or hotspots).
- **Local Patterns versus Global Similarity.** DTW is optimized to find a single global

alignment path that minimizes total distance between two time series, potentially overlooking local but meaningful impact episodes. The Ikat plot supports the identification and extraction of such localized events, which may only occur during specific intervals or at selected lags.

- **Event Extraction for Downstream Analysis.** Unlike CRP and DTW, which primarily produce similarity scores or recurrence matrices, the Ikat-based workflow includes explicit event extraction. This enables downstream quantitative and visual analysis of impact events across multiple time series pairs, supporting further modeling and hypothesis generation.

In summary, the Ikat plot approach shifts the focus from general similarity assessment to interpretable, lag-aware detection of stable impact relationships, providing a foundation for systematic event-based analysis. The following case study (Section 4.3) demonstrates these benefits by applying the workflow to a real-world spatio-temporal dataset. In this demonstration, we do not detail the interactive parameter selection process with the Ikat plot, but instead focus on the subsequent workflow steps (see Fig.22): extracting different types of events from the entire dataset and analyzing their temporal and spatial distributions. This focus illustrates how Ikat-guided exploration supports a comprehensive investigation of stable cross-impact patterns across multiple time series, moving beyond initial parameter tuning to provide insights at the whole dataset level.

4.3 Case study: Mobility and COVID-19 Impact Analysis

To test and demonstrate the entire analytical workflow starting with impact detection (Fig. 22), we apply it to interactions between mobility patterns and COVID-19 incidence rates across the 50 provinces and the two autonomous cities of Ceuta and Melilla using publicly available datasets [46, 45, 44] described in [43]. The study examines two daily time series per province from February 14, 2020, to May 9, 2021: (1) normalized counts of internal trips per 100,000 residents and (2) reported numbers of active COVID-19 cases. To mitigate short-term fluctuations caused by underreporting, reporting delays, and potential errors, as well as strong weekly mobility patterns, we apply a **7-day moving average** for back-smoothing over the preceding week.

4.3.1 Defining Impact Events

Impact events are defined as situations where changes in one time series (e.g., mobility) are followed by systematic changes in another (e.g., COVID-19 cases). We focus on the following types of relationships, as summarized schematically in Fig. 31:

- **Negative impacts**, where increasing disease incidence leads to decreasing mobility and vice versa.
- **Positive impacts**, where increasing or decreasing mobility contributes to a rise or reduction in disease incidence.

Please note that the terms "negative" and "positive" in this context do not denote assessments but only mean that the trend directions in two time series are opposite (negative impact) or same (positive impact).

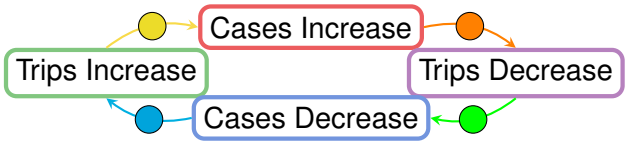


Figure 31: Expected impacts between trends in time series of cases and trips. Nodes represent trend events, and arrows - impact events. Colors correspond to legends in Figs. 32, 33, 34, 35.

To explore and test these impacts, we applied the interactive Ikat plots to pairs of time series from selected provinces, such as Madrid (Fig. 29) and Barcelona (Fig. 32). Impact values were computed over a *7-day* ***sliding window*** with ***time lags*** *ranging from 0 to 21 days*. To capture negative impacts of disease on mobility, the disease series was treated as the base and the mobility series as the response (Fig. 32a). Conversely, to capture positive impacts of mobility on disease spread, the mobility series was used as the base and the COVID-19 series as the response (Fig. 32b).

Based on interactive exploration and parameter sensitivity analysis (Section 4.2.3), we adopted the following settings: ***Kendall's Tau tolerance*** $\epsilon = 2\%$ of the value range; ***minimum lag*** *= 7 days*; ***trend thresholds*** = $\pm 3\%$ for both base and response variables. The impact value was further constrained according to the specific type of event:

- **Cases increase** $\rightarrow$ **Trips decrease** (impact ≤ -0.3)
- **Cases decrease** $\rightarrow$ **Trips increase** (impact ≤ -0.3)
- **Trips increase** $\rightarrow$ **Cases increase** (impact ≥ 0.3)
- **Trips decrease** $\rightarrow$ **Cases decrease** (impact ≥ 0.3)

In addition to the impact events, we also extracted **trend events** from each time series using the same ***sliding window*** *of 7 days* and $\pm 3\%$ ***trend thresholds*** as for the impact event extraction. The extracted event types include:

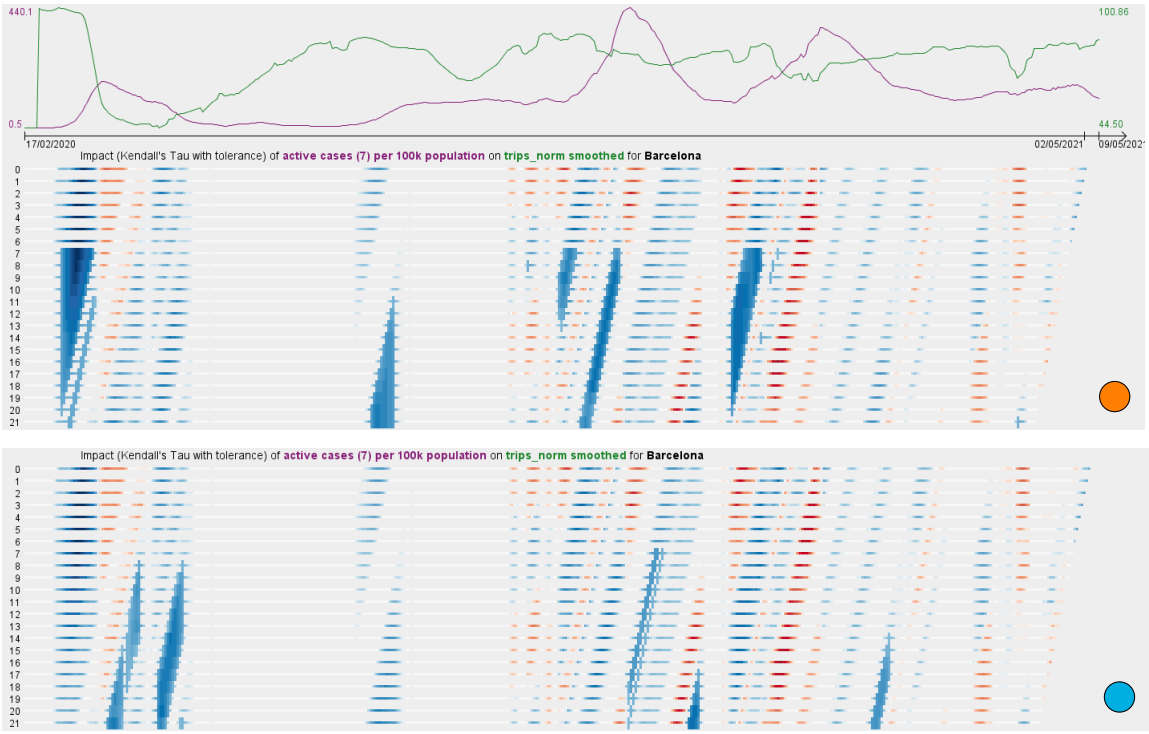
- **Case trends:** intervals of increasing or decreasing case counts.
- **Mobility trends:** increasing of increasing or decreasing numbers of trips.

Finally, for all event types, additional constraints were applied to enforce a ***minimum event duration*** and sufficient ***lag range***, ensuring statistical robustness.

4.3.2 Visualization of Events

We extracted four types of trend events and four types of impact events (Fig. 31). Displaying all eight simultaneously in a single view is difficult due to their frequent temporal overlaps. Interactive filtering allowed us to focus on various combinations of selected event types. To explore these combinations, we used two complementary visualizations: a space–time cube (Fig. 33) and a 2D timeline display (Fig. 34). In both displays, colors encode event types consistently with Fig. 31.

The space–time cube [27, 36, 19] offers an integrated view of temporal and spatial distributions of events, but effective interpretation requires interactive rotation and navigation to reduce



(a) Detection of negative impacts of the disease trends on population mobility.

(b) Detection of positive impacts of the mobility trends on the pandemic development.

Figure 32: Example Ikat plot screenshots used to select and test event extraction parameters in the case study. The colored circle in the bottom-right corner of each image indicates the event type, as defined in Fig. 31.

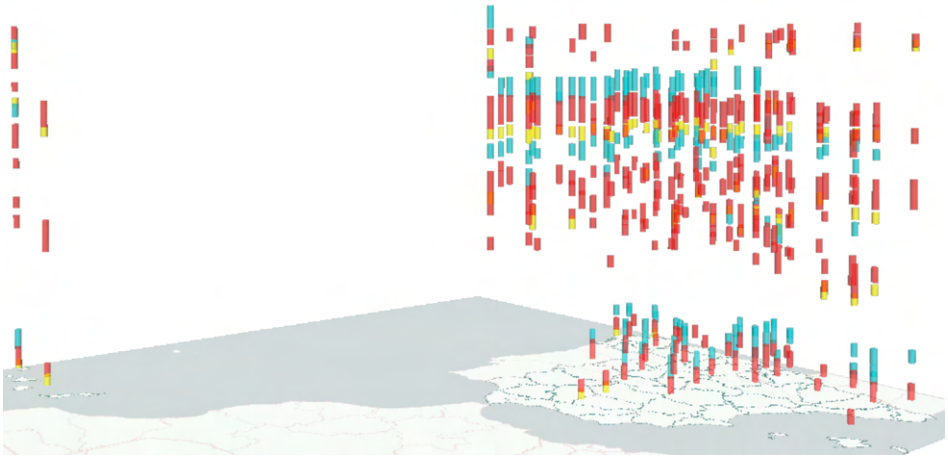
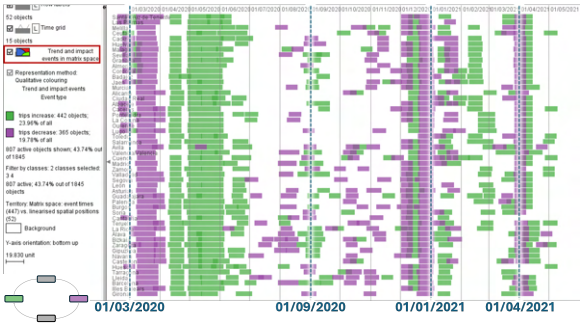


Figure 33: Impact events Cases decrease → Trips increase, Trips increase → Cases increase, and trend events Cases increase are shown in space-time cube.

(a) COVID-19 trend events, with red for increase and blue for decrease.



(b) Mobility trend events with green for increase and purple for decrease.

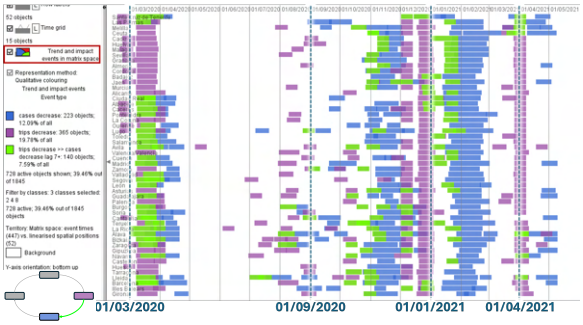
Figure 34: Visualization of the distributions of the extracted trend events along the timeline (horizontal dimension) and over the set of provinces (vertical dimension).

occlusion. For clearer inspection of temporal patterns, we therefore also employ a timeline view, where the horizontal axis represents time, while the vertical axis lists provinces. Each event is shown as a horizontal bar spanning its start and end times, with bar length proportional to event duration. The ordering of provinces along the vertical axis was derived by applying Sammon's mapping [49] to their geographic centroids. This nonlinear dimensionality reduction projects the provinces to one dimension while approximately preserving geographic distances. As a result, neighboring provinces in the original 2D space tend to appear close to each other in the timeline, enabling interpretation of spatially related temporal patterns while maintaining compactness.

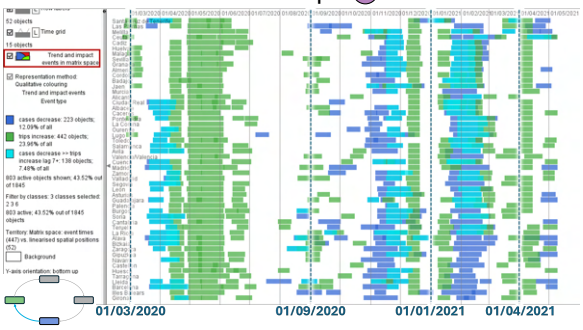
To facilitate interpreting the temporal patterns in the timeline visualizations, the screenshots in Figs. 34 and 35 are additionally annotated with vertical lines marking several anchor dates (March 1, 2020; September 1, 2020; January 1, 2021; April 1, 2021). These markers provide temporal reference points that help approximate when particular patterns occurred.

Figure 34a shows the distribution of COVID-19 trends: red for increases and blue for de-

(a) Increase trend of COVID-19 cases, impact of cases increase on trips decrease, decrease trend of trips.



(b) Decrease trend of trips, impact of trips decrease on cases decrease, decrease trend of cases.



(c) Decrease trend of cases, impact of cases decrease on trips increase, increase trend of trips.

(d) Increase trend of trips, impact of trips increase on cases increase, increase trend of cases.

Figure 35: Visualization of trend and impact event distributions corresponding to expected transitions between trends (Fig. 31). Impact events always partly or fully cover corresponding trend events. For example, an impact event of type "cases increase » trips decrease" in panel A is visually represented as a bar drawn over the corresponding trend events "cases increase" and "trips decrease".

creases. The trends form three major epidemic "waves" that are broadly synchronized across provinces, interspersed with periods of more heterogeneous regional behavior. Figure 34b shows the distribution of mobility trends: green for increases and purple for decreases. Similar to the epidemic curves, mobility exhibits several nationwide phases of reduction and recovery, often coinciding with policy interventions such as lockdowns or easing of restrictions, but also periods where provinces diverge, likely reflecting region-specific measures or behavioral responses.

Impact events can be displayed in the same way. While visualizing all event types together leads to clutter, examining selected combinations of trend and impact events (e.g., Fig. 35) reveals how specific patterns in COVID-19 incidence or mobility translated into measurable impacts, and where trends occurred without clear follow-up effects.

4.3.3 Exploration of trends and impacts

Figure 35 presents four screenshots of the event timeline display, showing different combinations of extracted trend and impact events. The visualizations highlight relationships between increases or decreases in COVID-19 cases and mobility allowing us to assess whether the COVID-19 or mobility trend events had significant impacts or occurred without measurable effects.

Overlaps between trend and impact events. Impact events are derived from trend events and always at least partially overlap with them in time. Each impact event (e.g., cases increase $\rightarrow$ trips decrease) corresponds to a period where a significant trend in one time series (e.g., an increase of cases) is followed by a trend in another series (e.g., a decrease in trips) after a time lag. Since impact events inherently link two trend events, the bar of the impact event is drawn over the respective trend bars.

By analyzing the combined displays of trend and impact events (Fig. 35), we observe several key patterns:

1. Mobility Responses to Pandemic Dynamics

- **Strong initial response to rising cases:** In March and in autumn 2020, increasing COVID-19 incidence was typically followed by decreasing mobility, reflecting imposed restrictions (lockdown has started on March 15) and voluntary behavioral changes. Later in the pandemic (e.g., summer 2020, spring of 2021), increases in cases often did not result in mobility reductions (Fig. 35a).
- **Recovery of mobility after case declines:** When cases decreased, mobility generally increased, likely due to relaxation of restrictions or behavioral changes (Fig. 35c).
- **Seasonal influences:** A notable drop in mobility occurred across all 52 provinces before and during Christmas 2020; in 35 of them the drop was not related to case increases. Similarly, the trips decreased in all provinces around Easter 2021 (beginning of April). The display also shows a synchronous decrease of mobility in 44 provinces in the beginning of December 2020. This trend appeared due to mobility restrictions imposed during several consecutive holidays [32].
- **Regional differences:** The provinces of Andalusia, located in southern Spain and positioned in the upper part of the timeline display, were less affected by the first wave of the pandemic than other regions. This is evident from the absence of COVID-19 increase trends in the upper left area of Fig. 34a). Nevertheless, mobility decreased nationwide during this period (Fig. 34b). Andalusia also experienced lower pandemic impact in the summer of 2020.

2. Impact of Mobility Changes on COVID-19 Incidence

- **Mobility restrictions were most effective early in the pandemic:** The strongest impact of reduced mobility on declining cases was seen in March-April 2020 and again in October-November 2020. However, in later periods, many decreases in mobility did not result in a corresponding reduction in COVID-19 cases. Overall, only 137

out of 365 (37.5%) mobility decrease events 🟣 led to a decline in cases 🔵, but these accounted for 46% (103/223) of all case decreases, indicating that mobility reductions 🟣 played a role in disease control but were not the sole factor.

- **Limited impact of mobility increases on case surges:** While some mobility increases 🟢 were followed by a rise in cases 🔴 (Fig. 35d), this effect diminished over time. Between April and June 2020, widespread mobility increases 🟢 had little measurable effect on COVID-19 incidence. In 2021, this impact further weakened, suggesting moderating factors like vaccination and immunity. Only 89 out of 442 (20%) mobility increase events 🟢 resulted in increased cases 🔴. Conversely, only 30% (70/230) of all case increases 🔴 (excluding the first wave) were preceded by rising mobility 🟢 within a 1–3 week time frame.

3. Asymmetric Influence Between Mobility and Cases. The effect of rising COVID-19 cases 🔴 on reducing mobility 🟣 is stronger and more consistent than the effects of mobility changes on the COVID-19 development. This asymmetry suggests that factors like restrictions and public awareness influence mobility more reliably than mobility influences virus transmission.

4. Variability in Impact Duration. Some impact events are short-lived, while others persist for weeks. This variation may be linked to region-specific factors such as population density, policy changes, and mobility habits.

5. Spatio-Temporal Differences in Responses. Some provinces exhibit strong, repeated correlations between mobility and COVID-19 trends, while others show inconsistent relationships. The presence of trend events without corresponding impacts in certain provinces suggests that interventions or external factors significantly alter mobility-disease interactions. There are also prolonged time periods when trend events (e.g., increasing cases 🔴 or mobility 🟢) do not lead to measurable impacts in most provinces. These cases likely reflect external influences such as vaccination rates, seasonal variations, or local interventions (e.g., mask mandates).

Concluding Remark. The visualization allowed us to detect stable patterns, regional differences, and periods where expected impacts do not occur, highlighting the complex interplay between mobility and pandemic trends. We complement this rough visual assessment with a more rigorous analysis of the trend-impact relationships by quantifying the proportions of trend events linked to impact events, either as results of prior influences or as contributors to subsequent changes.

4.3.4 Quantitative analysis and modeling

To obtain quantitative data for the analysis, we applied the **event context extraction** method [2]. For each trend event, we defined two temporal neighborhoods:

- **Preceding context** (3 weeks before event start) – to check if a prior impact event influenced the trend.
- **Following context** (3 weeks before event end) – to check if the trend contributed to a subsequent impact event.

Each trend event **X trend$_X$** (where *X* is mobility or disease cases, and *trend$_X$* is increase or decrease) was characterized as:

- **Impacted** if an impact event **Y trend$_Y$** $\rightarrow$ **X trend$_X$** existed within the 3 weeks before its start.
- **Had impact** if an impact event **X trend$_X$** $\rightarrow$ **Y trend$_Y$** existed within the 3 weeks before its end.

Based on these definitions, we generated two binary attributes: **"Was impacted"** (*yes*/*no*) and **"Had impact"** (*yes*/*no*). To visualize the results, we used impact transition graphs (Fig. 36), whose layout is consistent with the graph of expected impacts shown in Fig. 31, with nodes representing trend events and arrows signifying impact events.

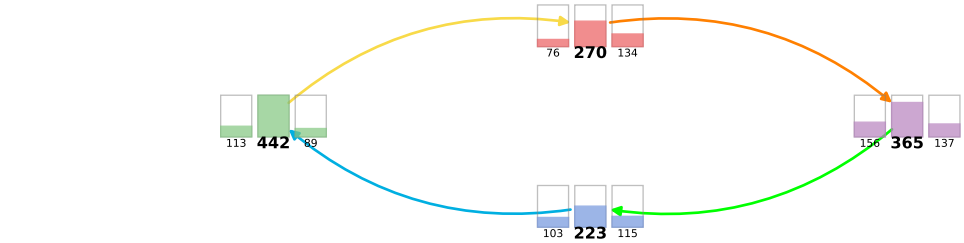
Each graph displays trend-impact relationships through colored bar charts corresponding to different trend event types, encoded by colors. Bar heights are proportional to event counts, with the middle bars indicating the total number of events, the left bars representing impacted events, and the right bars showing impacting events. Different event subsets can be selected using spatial, temporal, and other filters, and the display updates to reflect the corresponding statistics.

The top graph in Fig. 36 shows all events nationwide over the entire study period. The three graphs in the middle row depict trend-impact relationships during the three pandemic waves, while the bottom row presents event subsets for Madrid, Barcelona, and Andalusia. Each graph uses its own scale for the bar charts, with the tallest bar representing the maximum value within the subset and the remaining bars scaled proportionally. This supports the comparison of the proportions across the graphs.

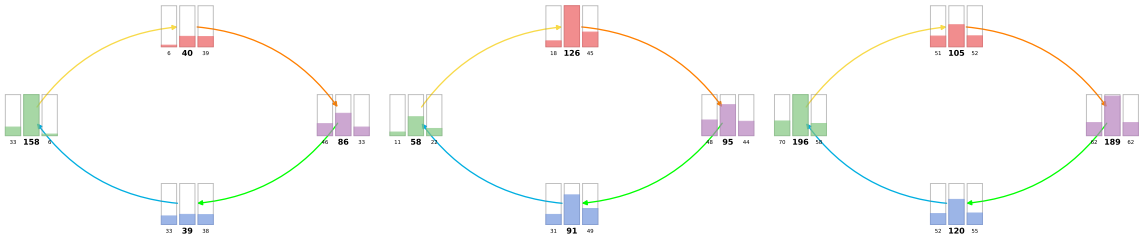
To obtain further insights into the event properties, modeling can be used. Specifically, we are interested which features of an event are predictive for whether it will have an impact on subsequent events. We choose to build an inherently interpretable [26] logistic regression model, as its feature weights describe direct dependencies between the target variable ("having impact") and the input features. In contrast to time-series analysis, features are extracted from the entire event durations and not from fixed time windows in order to keep correspondence with the visually analyzed events.

We model mobility and disease event impacts separately. Given the substantial differences between pandemic waves, we create per-wave models alongside global ones. Results on the balanced model accuracies can be found in Fig. 37, ranging from 70% to 92%. For the first COVID wave, a model for the pandemic event impacts cannot be trained as all COVID-19 events had impact, i.e., there were no such conditions under which an event would have no impact. Hence, none of the features was relevant, so it was impossible to build a model predicting how the result (existence of impact) depends on the features.

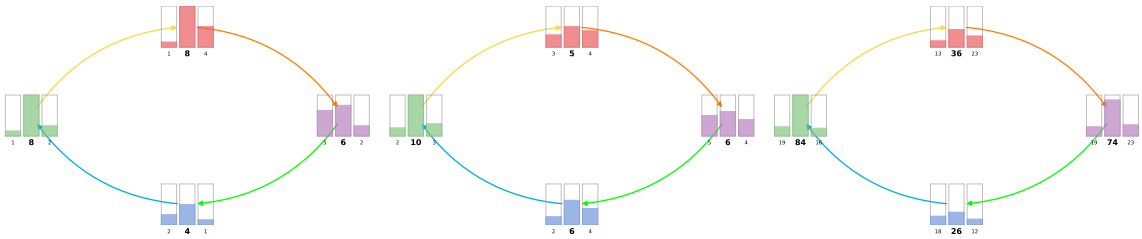
Figure 37 also shows feature importance as absolute model coefficients, revealing, e.g., a spatial distribution for the first wave mobility event impacts. Notably, the importance of features varied significantly between pandemic waves, suggesting that seasonality and evolving mitigation strategies played a decisive role, while other factors were subordinate. The importance scores of the features can guide the queries described in Sec. 4.2.2.



(a) Whole Spain: all times



(b) Whole Spain: 18/02/2020-31/05/2020, 01/06/2020-30/11/2020, 01/12/2020-09/05/2021



(c) Madrid, Barcelona, and Andalusia: all time periods

Figure 36: Impact transition graphs showing trend-impact relationships. The colors represent event types as have been specified in Section 4.3.1. Bar charts represent different trend event types, with bar heights proportional to event counts: total (middle bars), impacted (left), and impacting (right). The top graph summarizes all events nationwide, while the three graphs in the middle row depict relationships during the three pandemic waves. The bottom row presents subsets for Madrid, Barcelona, and Andalusia.

4.3.5 Insights gained

The analysis of trend and impact events provides numerous insights into the dynamics of mobility and COVID-19 incidence. We identified periods with stable cross-impacts, where mobility and COVID-19 cases exhibited consistent dependencies. The strongest linkages existed in early 2020, with mobility reductions playing a key role in controlling disease spread (Fig. 36, middle left). Over time, the influence of mobility on transmission weakened (Fig. 36, middle center and right), likely due to changing public behavior (e.g., wearing masks and social distancing), improved interventions, and vaccination. The persistence of impact events varied across regions (Fig. 36 bottom), indicating heterogeneity in local responses to pandemic conditions. Additional analysis can be done by combining epidemiological data with the dynamics of mobility between provinces, assessing risk of disease transmission, and studying its dynamic impact in space and time.

	Mobility Event				Covid Event			
	Wave 1	*Wave 2*	*Wave 3*	*All*	*Wave 1*	*Wave 2*	*Wave 3*	*All*
Model Accuracy	0.920	0.700	0.710	0.700		0.730	0.720	0.620
Average Mobility	1.056	0.288	0.678	1.339		0.069	0.618	0.080
Trend type (increase/decrease)	0.428	0.023	0.370	0.225		0.181	0.912	0.026
Extreme event value	0.373	0.320	0.041	0.050		0.341	0.029	0.141
Average Cases	1.406	0.455	0.638	1.395	failed to	0.469	0.527	0.589
X coordinate province	0.981	0.100	0.214	0.291	build a	0.086	0.296	0.015
Y coordinate province	0.157	0.361	0.120	0.050	model: all	0.096	0.175	0.072
Event Start Case value	0.783	0.105	0.216	0.360	trends	0.489	0.734	0.270
Event End Case value	2.543	0.806	0.361	0.365	have	0.204	0.150	0.093
Event Start Mobility value	1.555	0.331	0.273	0.130	impact	0.123	0.135	0.034
Event End Mobility Value	0.031	0.062	0.590	1.043		0.320	0.373	0.799
Amplitude of Covid values	0.378	0.571	0.482	0.409		0.073	0.580	0.200
Amplitude of Mobility Values	0.152	0.274	0.177	0.023		0.361	0.458	0.658

Figure 37: Feature weights in logistic regression models predicting the expected impacts from mobility (left) and case trends (right).

4.4 Discussion and Conclusion

A key contribution of our work is a general workflow for detecting and extracting impact events from multiple bivariate time series, followed by visualization and comprehensive analysis of event distributions and relationships across multiple data dimensions (Fig. 22). By leveraging a modified Kendall's tau measure with tolerance, an interactive Ikat plot visualization, and event extraction techniques, we provided an approach for exploring dynamic relationships between time series at varying time lags. This section reflects upon various aspects of our approach and outlines possible future research directions.

4.4.1 Impact Measures and Parameter Sensitivity

Instead of relying on traditionally applied Granger causality, which poses strong data requirements and is difficult to interpret, we propose using an easy-to-understand measure that captures stable dependencies more intuitively. The introduction of tolerance thresholds in Kendall's tau helped mitigate the effect of minor fluctuations, leading to a more robust detection of significant impact patterns. Our study demonstrated that impact measures are sensitive to parameter choices, including:

- The length of the sliding time window: Shorter windows captured more transient dependencies, while longer windows smoothed fluctuations but risked obscuring fine-grained patterns.
- The threshold for Kendall's tau with tolerance: An appropriate cutoff should be determined by analyzing the distribution of values in areas surrounding potential events. This ensures that detected dependencies are not artifacts of local fluctuations.

The computational complexity of our approach remains manageable, as it scales linearly with respect to the length of time series and quadratically with the time window size. The flexibility of the approach can be enhanced by expressing tolerance in terms of standard deviation or

quartile-based measures for reducing influence of outliers.

4.4.2 Event Extraction, Visualization and Interpretation

Our event extraction method transforms abstract statistical dependencies into interpretable events with temporal and, when relevant, spatial references. Once extracted, these events can be analyzed as a standalone dataset. Understanding larger-scale patterns of event occurrence and relationships requires examining the distribution of these events over time and across relevant data dimensions. For this purpose, we primarily use a timeline display, where one dimension represents time and events are visualized as bars positioned along the time axis. The remaining display dimension represents the set of time series from which the events were extracted, i.e., the locations or entities described by the time series. A meaningful linear arrangement of these items along the dedicated display dimension is crucial for detecting variation patterns among groups of entities or regions in space.

Our case study provides an example of achieving a linear ordering by applying a dimensionality reduction method to the geographic positions of the provinces. This approach tends to group geographically neighboring provinces together in the plot. However, since neither local nor global relationships are strictly preserved, it is essential to validate the detected patterns. To do this, we used geographic maps, but due to space constraints, we omit the details.

The ordering can also be derived by applying dimensionality reduction to other attributes present in the original data, which may be particularly relevant for non-spatial data. A viable approach, always available, is to arrange the original time series based on their similarity. When additional attributes are available, such as overall characteristics of patients in medical studies or products in sales analyses, these attributes can also be utilized to obtain meaningful arrangements. Such arrangements enable analysts to determine whether the distribution of events is related to item similarities.

Since a one-dimensional arrangement cannot fully represent similarity relationships without introducing distortions, it is essential to have tools that help verify whether display neighbors are indeed similar and whether similar items are not separated in the display. One possible approach is to highlight the most similar items for the item under the cursor, allowing analysts to quickly assess the validity of the arrangement and identify potential inconsistencies.

4.4.3 Limitations and Generalizability

The metrics we use for impact detection, including the Kendal's Tau, implicitly assume regular temporal sampling of the data and identical temporal resolution across all time series. When this assumption is not fulfilled, or when some values are missing, the data need to be preprocessed by applying resampling and/or value permutation. Presence of measurement errors in the data can also distort the results of impact event extraction and downstream analysis; however, this is a common problem necessitating data cleaning before using any method or analytical workflow.

The proposed approach for impact detection and event extraction is general by design and can be applied to bivariate time series in any domain. However, the following analysis applied to the extracted events may be specific for the application domain and depend on the analysis goals. Our case study should be considered as an example demonstrating a possible purpose

of extracting impact events from a large set of bivariate time series, visualization techniques applicable for exploration of the impact distribution and variation, and a possibility of using the events for impact modeling. While these ideas are transferrable to other application domains, other visualization and/or modeling methods may be more suitable.

4.4.4 Future Research Directions

Several potential extensions of our approach can be explored:

- **Explicit Encoding of Changes**: The Ikat plot always shows the parts of the data satisfying the current query settings and immediately updates in response to any change. The previous state of the display is not visible anymore, and it is hard to understand what has changed. The visualization design could be enhanced by techniques highlighting the changes. For instance, upon user's request, the elements that have changed could start blinking or alternating between the old and new appearance.
- **Comprehensive Threshold Sensitivity Assessment**: Our implementation enables interactive visual exploration of the sensitivity of event detection results to the Kendall's tau tolerance threshold, and helps to obtain a graph showing how the number of detected events and their numerical characteristics change as the threshold varies within a specified range. However, to support more comprehensive analysis and well-grounded decisions during event extraction, it would be beneficial to include computation and visual representation of threshold influence in space and time.
- **Enhanced Multivariate Analysis**: It would be exciting to find approaches to extend the method from bivariate time series to multidimensional data representing more complex systems, such as chains or networks of impacts.

Our study highlights the importance of combining computational analysis with interactive visual exploration to uncover stable cross-impact patterns in time series data. Future work will focus on refining the methodology and extending its applicability to more complex impact patterns, diverse domains, and real-time scenarios.

5 Augmented Reality and Uncertainty visualizationat the Field

In this section we present the work done in Tasks 5.4 and T5.5 that focuses on the development of an Augmented Reality (AR) system architecture for real-time urban flood management, integrating extreme scale data to enhance situational awareness and decision-making during emergency response operations. We show the general architecture of the system and the use cases where it has been applied and tested.

5.1 Eye Tracking & Mental Fatigue Monitoring

Rescue operations often take place in highly demanding environments where responders must make rapid, complex decisions under both physical and psychological stress. In such conditions, mental fatigue can critically impact performance, leading to slower reaction times, reduced situational awareness, and an increased risk of errors. Understanding and mitigating mental fatigue is therefore essential to ensure the safety, effectiveness, and well-being of rescuers in the field.

Recent advancements in eye-tracking technology have opened new possibilities for assessing cognitive load and fatigue in real time. Eye metrics—such as fixations, saccades, blinks, and pupil dilation—provide valuable, non-invasive indicators of mental effort. By monitoring these parameters, and the byproducts derived from them, researchers and field operators can gain objective insights into the cognitive state of rescuers during missions. Integrating eye-tracking systems into field operations can enable the early detection of fatigue, support adaptive task allocation, and improve training protocols. Such integration represents a promising step toward enhancing resilience, performance, and safety in high-stress rescue environments.

5.1.1 Eye Tracking Hardware Implementation

The device is built around a Raspberry Pi 4 Model B (Pi), which performs image acquisition, real-time processing, and H.264 encoding. Integrated 802.11ac Wi-Fi and Bluetooth 5.0 enable remote operation for data streaming, calibration, and headless configuration (e.g., via SSH or a lightweight web interface), as well as over-the-air software updates—useful in wearable or head-mounted deployments. Video is captured with a Raspberry Pi Camera v3 NoIR Wide (Sony IMX708) connected through a short flex extender for flexible placement .Illumination is provided by a single $850\,\mathrm{nm}$ IR LED with a 150° emission angle, powered from the Pi's $5\,\mathrm{V}$ rail (pin 2) through a $180\,\Omega$ series resistor and PWM-driven for adaptive brightness. To reject visible light while transmitting the LED output, an $850\,\mathrm{nm}$ IR long-pass filter is mounted in front of the lens, appropriately cut to size (Fig. 38). Eye safety is ensured by limiting radiant intensity to $<0.008\,\mathrm{W\,cm^{-2}}$, within the IEC 60825-1 Class 1 exposure limit of $0.01\,\mathrm{W\,cm^{-2}}$.

5.1.2 System Architecture & Frame Processing Unit

Our system architecture integrates multiple Raspberry Pi 4 Model B (Pi) devices, each paired with a HoloLens 2 or any AR headset that lacks an integrated eye-tracking camera, to capture eye video and transmit frames to a central Frame Processing Unit (FPU). Incoming frames

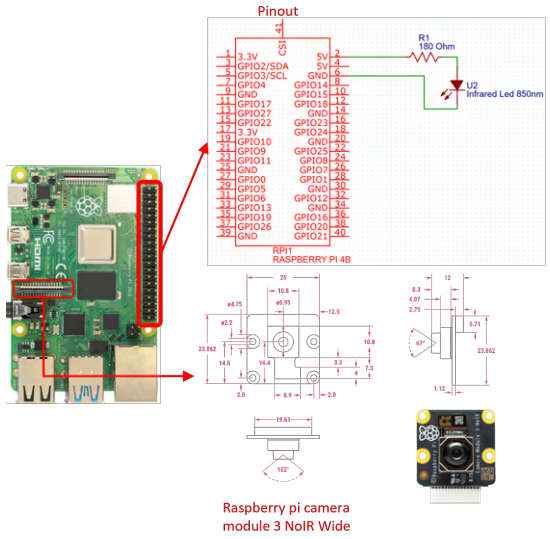


Figure 38: Hardware schematic of the monitoring system

pass through an image-preprocessing pipeline and are then fed to a Convolutional Neural Network (CNN) that estimates pupil area and blink probabilities. From the predicted pupil centers (x, y), the system computes angular distance, velocity, and acceleration—enabling frame-level labeling of saccades and related eye-movement events with defined thresholds and refractory periods for robustness. A sliding-window scheme aggregates these signals into features, which a trained machine-learning model uses to infer mental fatigue once per second. Each inference is packaged with the device identifier and published to an Apache Kafka topic with timestamps for synchronization and auditability. AR clients subscribe to the relevant stream by ID and associate each fatigue estimate with its corresponding headset, enabling real-time, confidence-aware visualization of each user's cognitive state in AR and supporting unobtrusive, scalable monitoring across multiple operators.

Each Raspberry Pi 4 Model B embedded on the Hololens 2 (AR) device establishes communication with the Frame Processing Unit (FPU) (Fig. 39), which operates as a Python-based server. The process begins with the Raspberry Pi opening a socket connection to the FPU and transmitting its unique device identifier (ID). Once the ID is successfully registered by the server, the Raspberry Pi initiates a dedicated TCP connection with the FPU and closes the socket connection. The Raspberry Pi captures video frames at a resolution of 1920×1080 pixels and a frame rate of 50 Frames Per Second (fps) using the Raspberry Pi Camera and streams them to the FPU via TCP for real-time encoding and transmission. The server is designed to handle multiple simultaneous connections from different Raspberry Pi–based AR devices. Since each device submits its unique ID prior to video streaming, the FPU is able to reliably associate incoming frames with the corresponding device. Importantly, the ID information is not embedded within the video data itself but is managed separately through the initial connection setup.

On the FPU, frames were converted to grayscale, cropped to 400×400 around the eye, resized to 128×128, and fed to the pretrained U-net model. We analyzed each video frame using the pretrained U-net model[47] (mDice = 84.0% and real-time efficiency 45.2 FPS,

0.2 GFLOPS, 0.03 M parameters) from MEYE app[40]. The model was trained on 11,897 manually annotated, 128×128 grayscale images of mouse and human eyes[39].

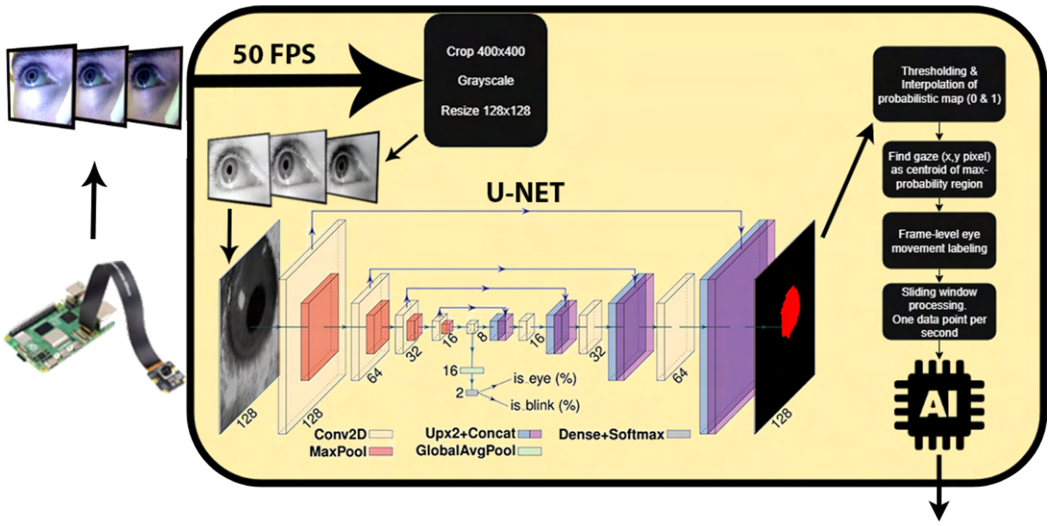


Figure 39: ***Frame Processing Unit***

For each frame, it produced a pupil probability map and eye/blink detection probabilities, from which pupil area was extracted. In continuous processing, frames with blink probability > 0.8 (threshold decided from internal tests) were labeled as blinks and excluded from analysis. For frames not labeled as blinks, pixels with pupil probability above 0.65 (also determined through internal tests) were classified as pupil and converted into a binary mask. Using regionprops from scikit-image, we extracted connected components and kept the largest as the pupil. The mask was then resized to the frame dimensions and refined with internearest interpolation. Finally, the centroid of the resized mask was taken as the eye's gaze point, and the same mask was used to fit a circle to the pupil in the 400×400 cropped frame.

Figure 40: ***Left: Eye Frame cropped 400x400px, grayscaled & resized 128x128. Right: Probabilistic region on the pupil area***

5.1.3 Pixel-to-degree Conversion & Frame Labeling

To derive angular metrics (saccades, fixations), pixel coordinates must be converted to degrees. Following [54], this is done by dividing the horizontal and vertical fields of view by their respective resolutions to obtain degree-per-pixel ratios, which are then multiplied by pixel displacements. However, with our high-resolution camera (1080p), the ratios were

implausibly small, thus even large displacements produced unrealistically small angular changes. As a result, saccade velocities never exceeded angular thresholds. To address this, we implemented an individual geometric calibration procedure for each participant.

Five fixation targets were placed on a wall in a cross configuration at eye height, with one central and four peripheral targets at ± 5 cm horizontally and vertically from the center, at a viewing distance of $D = 65$ cm. The angular separation was computed by trigonometric equation $\Delta\theta = 2\arctan\left(\frac{\Delta X}{2D}\right)$. During calibration, participants fixated each target while pupil centroids were recorded. The pixel displacement between central and peripheral fixations (Δpx) was divided by the known angular separation to obtain the pixel-to-degree conversion factor $k = \frac{\Delta\theta}{\Delta \mathrm{px}}$. Participant-specific calibration yielded conversion factors in both horizontal and vertical directions, which were applied to all subsequent eye movement data to express gaze trajectories, velocities, and accelerations in degrees of visual angle.

Following [54], eye movements were classified per frame using angular velocity, acceleration, and duration thresholds. Saccades were defined as movements exceeding 240 °/s velocity and 3000 °/s² acceleration, and fixations as movements below these values. Duration constraints were applied during buffer processing, where at 50 FPS (20 ms/frame) saccades were confirmed only if $\leq$100 ms ($\leq$5 frames) and fixations if $\geq$100 ms ($\geq$5 frames). Flickers—artifacts with unrealistically high values (e.g., >1000 °/s or >100,000 °/s²) were also detected and filtered. The buffer is triggered when a frame is provisionally labeled as fixation, saccade, or flicker, and runs until terminated by a blink, an out-of-threshold movement, or five consecutive fixations. Upon deactivation, flickers are interpolated, gaze features (angular distance, velocity, acceleration, coordinates) recalculated, and frames relabeled. Events are reclassified by duration: saccades longer than five frames ($\geq$100 ms at 50 FPS) and fixations shorter than five frames are labeled “other” and excluded. To allow interpolation when the buffer begins with a flicker, the last non-blink frame is stored and prepended, ensuring its angular metrics remain available.

5.1.4 Sliding Window

Eye tracking data were recorded at 50 FPS. Every 50 frames corresponding to 1 second of data were translated into eye movement data and sent to the 1-minute processing queue, which holds the most recent 3000 frames (60sec × 50FPS). The system normally streams data directly; however, when the buffer process is triggered, three cases can occur: (1) if fewer than 50 rows of data are returned from the buffer (e.g., 30), the system waits for the remaining frames to arrive before forming a complete 1-second segment; (2) if exactly 50 rows are returned, they are immediately sent to the queue; and (3) if more than 50 rows are returned (e.g., 70), the first 50 are sent to the queue, while the remaining 20 are held until an additional 30 arrive to complete the next 50-frame segment. Once the queue reaches 3000 frames or more, the complete 1-minute window is passed for feature extraction. From each 1-minute window, we first computed the base oculomotor metrics: fixation count, saccade count, and blink count. Using these, we derived the following kinematic metrics: **mean pupil diameter**, the average pupil size across all frames; **mean fixation duration**, the average duration of fixations from fixation-labeled frames and their count; **mean saccade duration**, the average duration of saccades from saccade-labeled frames and their count; **peak saccade velocity**, the maximum angular velocity during saccades; **mean saccade velocity**, the average angular velocity across saccade frames; and **mean saccade amplitude**, the average

angular distance moved per saccade frame. We adopted a sliding-window strategy to capture temporal dynamics of oculomotor behavior, as mental fatigue develops gradually and short-term fluctuations may be uninformative. The resulting feature vectors are input to a machine learning classifier for mental fatigue detection.

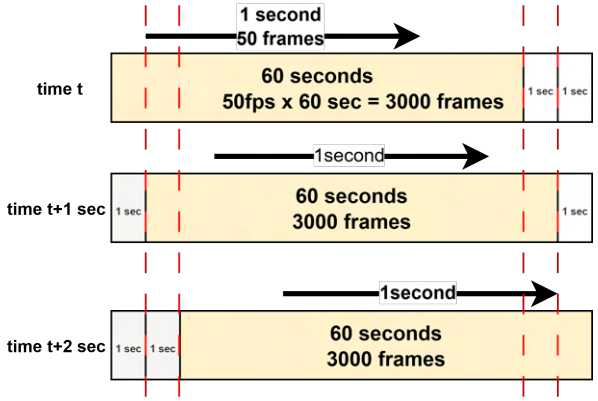


Figure 41: Sliding window of one minute with 1 second step

5.1.5 Server Integration & Data Distribution

To enable real-time collaborative monitoring of mental fatigue in multi-user AR environments, we connect each user to our Node.js-based server that acts as a central middleware for data processing, device pairing and distribution as described in paragraph 5.1.2. This server integrates seamlessly with the FPU, handling predictions from a Machine Learning classifier and facilitating communication across devices. Data dissemination is handled with Apache Kafka, enabling scalable, fault-tolerant streaming of fatigue predictions in JSON format. Upon deployment, each user establishes a WebSocket connection to the Node.js server for bidirectional communication. Users select their corresponding Raspberry Pi ID from a preconfigured list inside the AR app interface, which triggers a mapping request to the server. This pairs the user's AR device with their eye-tracking Pi, enabling differentiated data handling. Pis capture eye data and stream it to the FPU for processing, where the ML model generates per-second mental fatigue predictions (low, mild, moderate, severe). These, along with the device ID and timestamp, are JSON objects that are being published to a dedicated Kafka topic by the FPU. The Node.js server subscribes all connected clients as consumers to this topic, allowing each AR device to retrieve a real-time stream of predictions for all users. Through the pre-established ID mapping, the system highlights the user's data (e.g., via color-coded holographic badges) while displaying aggregated team metrics, visualized in AR. The pipeline operates as shown in figure [42].

5.1.6 Augmented Reality Environment

The AR component was implemented on Microsoft HoloLens 2 which has its own built-in eye tracking system. However, its API exposes only gaze position (x,y pixels) and blink events, but not angular metrics or pupil diameter, due to privacy and calibration constraints. To overcome this limitation, we integrated our custom low-cost eye tracking add-on (Fig. 43), enabling

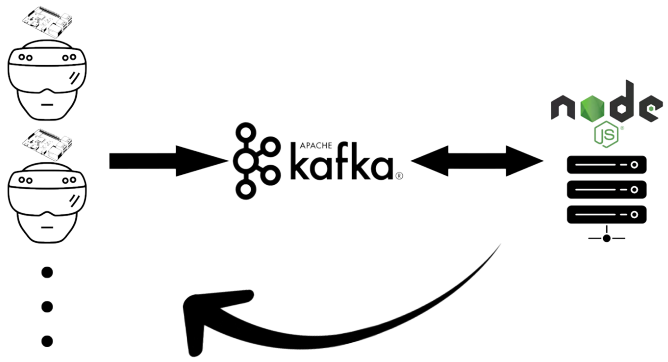


Figure 42: Data and visualization pipeline

access to raw data (pupil size, blink frequency, saccadic metrics), preserving compatibility with the AR ecosystem.

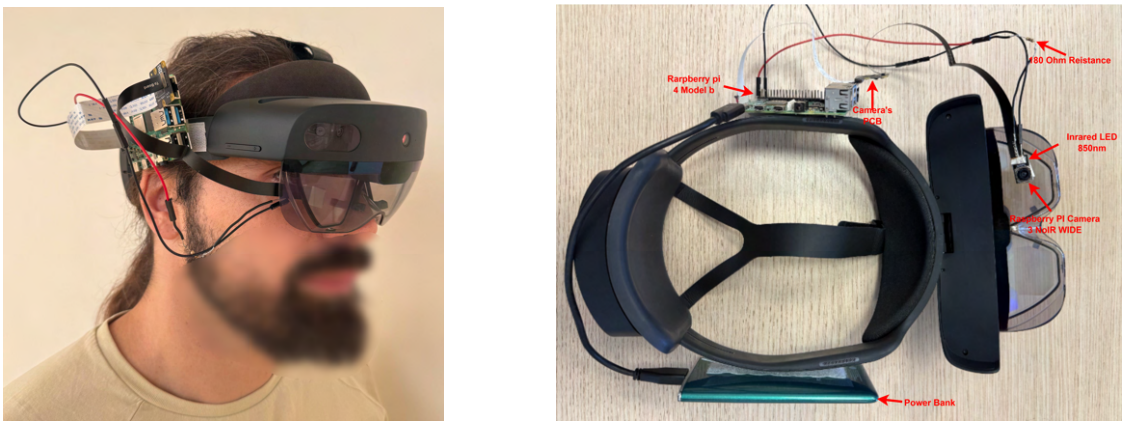
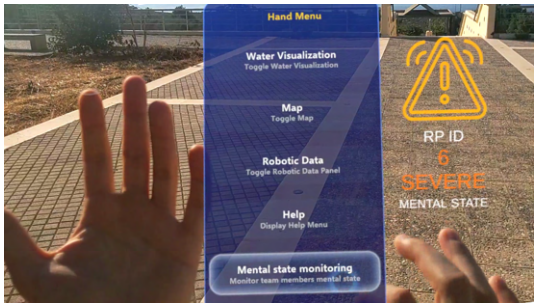


Figure 43: ***a) Side view of Hololens 2 with attached electronics. b) Prototype Top View***

The user interface (UI) viewed via the Hololens 2 is designed for minimal intrusion, displaying fatigue data as holograms tied to the users field-of-view (FOV). Upon initialization, users interact with a holographic menu to select their Pi ID via the monitoring panel for pairing (Fig. 44a). Once connected, the interface renders personalized and team-wide visualizations: individual fatigue levels appear as color-coded badges, e.g., green for low, yellow for mild, orange for moderate, red for severe, overlaid on the user's FOV. Aggregated metrics, such as team average fatigue or individual fatigue states, are presented via a monitoring panel initiated via a hand menu, or dismissed via hand gestures and hand interactions(Fig. 44c). When a user enters a critical severe mental fatigue state, a prominent popup hologram appears in front of teammates' views to notify them of the affected individual's urgent cognitive situation, prompting immediate support or task reassignment(Fig. 44b). This setup facilitates proactive interventions, such as task reassignment prompts when fatigue thresholds are exceeded, thereby enhancing safety and performance in AR-augmented environments.



(a) Hand menu

(b) Alert popup

(c) Monitoring Panel

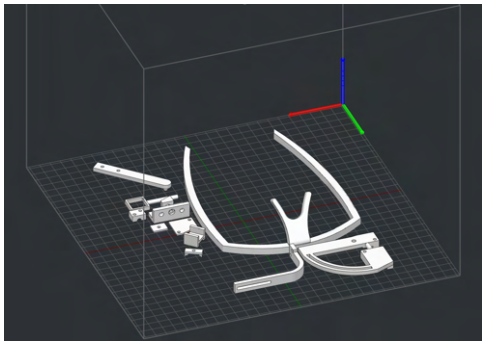
Figure 44: *UI components*

5.1.7 Experimental Procedure

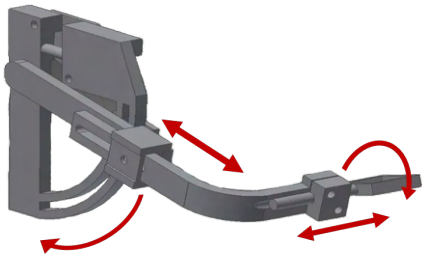
Eye videos were continuously recorded using a lightweight eye tracking system mounted on a 3D-printed frame (Fig. 45). Eight participants (5 men, 3 women; aged 18–36) took part, all well-rested and instructed to abstain from caffeine, alcohol, and strenuous activity beforehand. Prior to the main experiment, they completed a brief training session to familiarize themselves with the tests and response requirements. Six participants completed the full protocol, while two discontinued after two sets.

The experiment assessed sustained attention, working memory, selective attention, and spatial reasoning using four tests: AX-CPT, N-BACK, Visual Search, and Mental Rotation (Fig. 46) [30]. Each session comprised three sets, with all tests (≈10 min each) presented in counterbalanced order and separated by short self-paced breaks. After each test, participants completed the fatigue subscale of the Brunel Mood Scale (BRUMS) [56, 57], consisting of four items (exhausted, sleepy, tired, worn-out) rated on a 5-point Likert scale (0 = “Not at all” to 4 = “Extremely”). The mean of these items yielded a 1–4 fatigue score, chosen for its strong validity as a mental fatigue index and low participant burden.

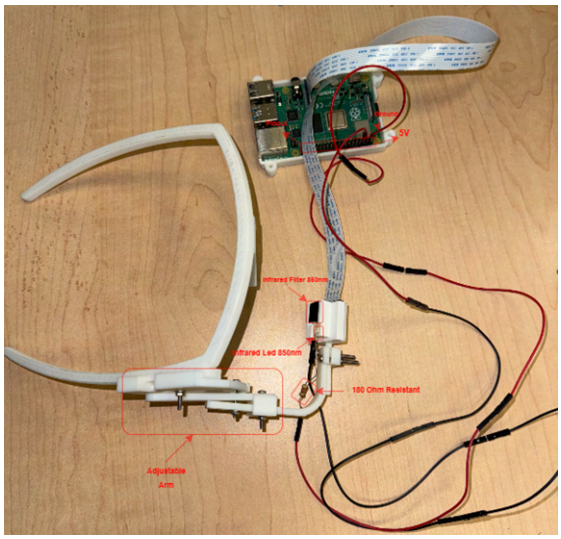
- **AX Continuous Performance Test (AX-CPT)**: The AX-CPT was used to induce mental fatigue [38]. Each trial presented a red cue, two white distractors, and a red probe (see Figure 47). Targets were A–X sequences (70%), and non-targets were B–X, A–Y, or B–Y (10% each). Participants pressed “k” for targets and “d” otherwise. Letters appeared for 300 ms with 1200 ms intervals, and accuracy feedback was given. Trials were pseudorandomized to preserve the 7:3 ratio.



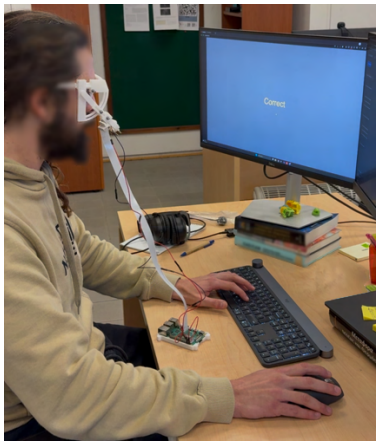
(a) STL files



(b) Adjustable arm



(c) 3D Print model embedded with raspberry system



(d) Participant performing a battery task test

Figure 45

- **N-back Task**: The n-back task [35] was used to assess working memory, with participants completing a 3-back version to maximize cognitive demands. On each trial, they indicated whether the current letter (cue) matched the one shown three trials earlier, responding with button “k” for matches (30% of trials) and “d” otherwise (70%). Letters were presented in white on a grey background for 2000 ms, followed by feedback and a 1200 ms interval (see Figure 48), with trials pseudorandomized to maintain the target-to-non-target ratio.
- **Visual Search Task**: We used a visual search task based on Horowitz and Wolfe [31]. On each trial, participants viewed an array of 11 letters that could be rotated by $0°$, $90°$, $180°$, or $270°$. Letters were positioned on an invisible 4×4 grid, centered on the display. The grid size was normalized so that a uniform border of 25% of the participant’s screen size remained on all sides. Each trial contained at least ten letter Ls. In *target* trials (50% of trials), a letter T was also present at a random grid location and orientation (see Figure 49a). In *non-target* trials (the remaining 50%), an additional L replaced the T (see Figure 49b). Trial order was pseudorandomized in blocks of ten such that five

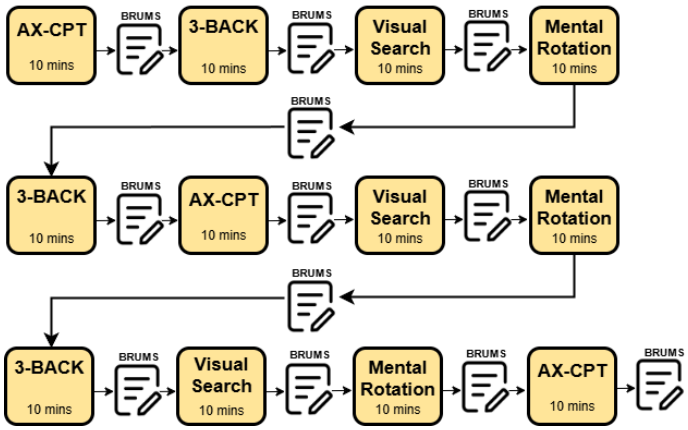


Figure 46: ***Task Battery for inducing mental fatigue***

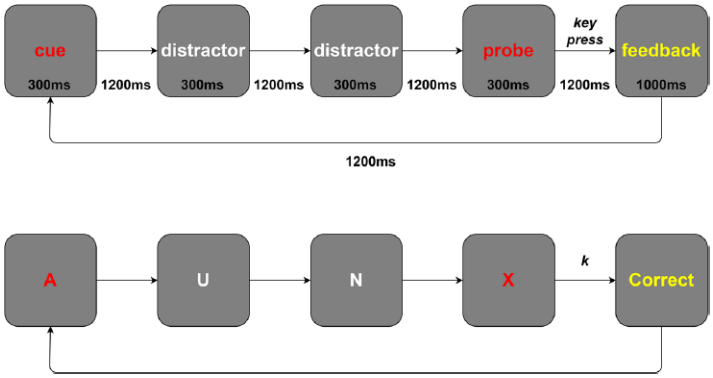


Figure 47: ***AX-CPT, example target trial with correct response***

were target trials and five were non-target trials. Participants were instructed to press $\mathtt{k}$ when a T was present and $\mathtt{d}$ otherwise. There was no time limit for responses. After each response, feedback was displayed for $1000\,\mathrm{ms}$, followed by a $1200\,\mathrm{ms}$ interval, as in the AX-CPT and n-back tasks. Letter rendering parameters (font, size, and luminance) matched those used in the n-back task.

- **Mental Rotation Task**: In mental rotation tasks, participants view two items and determine whether they match; typically, one item is rotated relative to the other, requiring the participant to visualize and "mentally rotate" the item to decide if the pair is the same [51]. We used the validated stimulus set of Ganis and Kievit [18]. In their stimuli, the left-hand shape is always presented in a canonical orientation, while the right-hand shape is rotated by $0°$, $50°$, $100°$, or $150°$ about a vertical axis (with 25% of the set at each rotation). Half of the items depict matching shapes (see Figure 50a), and half depict non-matching shapes (see Figure 50b). Non-matching shapes are pseudo-mirror images constructed with the same number of cubes and the same arm configuration as their matching counterparts.

  We used the same subset of 96 stimuli as in the original validation study [18]. Based on that subset, a mean response time of approximately $3000\,\mathrm{ms}$ (the largest mean reported in [18]) implies an average trial duration of $5.2\,\mathrm{s}$, allowing each participant to complete at

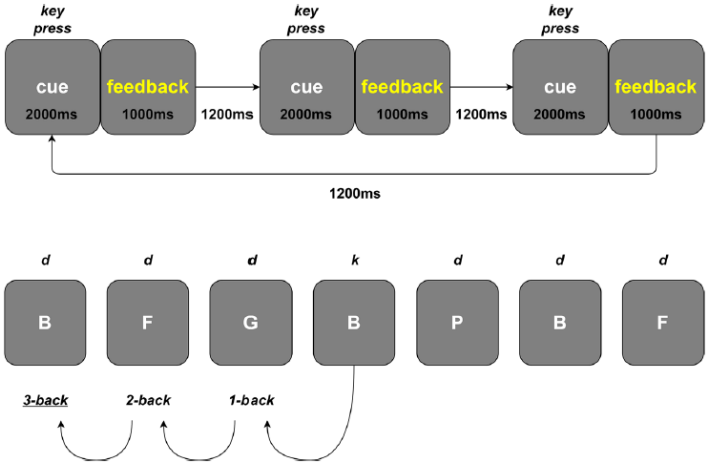


Figure 48: ***N-Back test & correct responses***

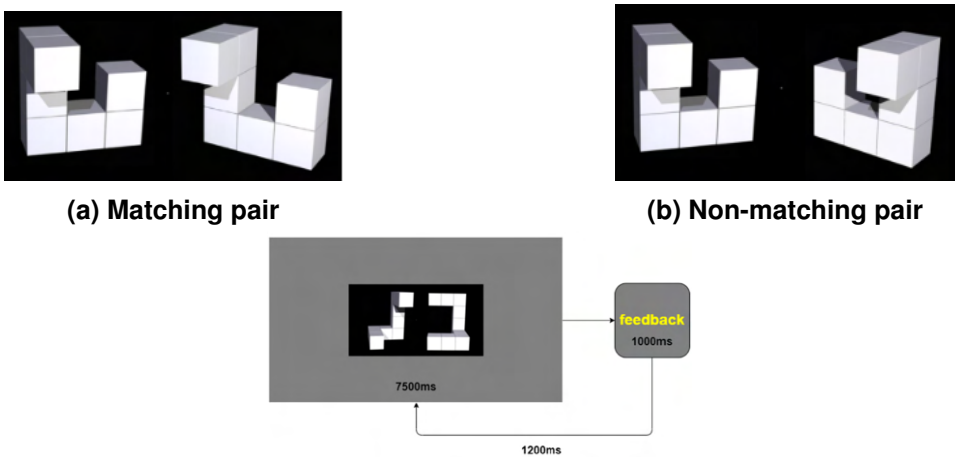
(a) T present, rotated 180° **(b) T NOT present**

Figure 49: ***Visual Search test***

least 115 trials. Stimuli were presented in pseudorandom order with the constraint that all 96 items were shown at least once before any item could repeat, ensuring that each participant saw every stimulus at least once within 115 trials. Stimuli appeared centrally at 50% of the screen width and 25% of the screen height. Each stimulus remained on screen for $7500\,\mathrm{ms}$; participants responded with `k` if the shapes matched and `d` if they did not. Following each response, written feedback was displayed for $1000\,\mathrm{ms}$, followed by a $1200\,\mathrm{ms}$ inter-trial interval (see Figure 50c).

5.1.8 Data Extraction, Data Merging & Visual Pattern

For each participant, eye videos from the tasks were processed by the same real-time pipeline to yield raw, per-frame measurements. A $\sim$2 h recording (approximately $270{,}000$ frames) produced one row per frame, each linked to its exact timestamp taken from the video metadata. The pipeline emitted eye-related byproducts at frame level, including `timeStamp`, `frameNumber`, `pupilSize`, `pupilCntrX`, `pupilCntrY`, `blinkProbability`, `angularDistance`, `angularVelocity`, `angularAcceleration` and `label`. Using the exported start and end timestamps for each test, we isolated the relevant segments and discarded data outside or between tests. To accelerate time-based lookup and slicing, we indexed frame timestamps with a `cKDTree`. For each test segment, frame-level variables were then processed with the sliding-window method (Section 5.1.4) to compute the corresponding higher-level features. Each windowed feature vector was tagged with the test name (indicating the originating task)



(a) Matching pair **(b) Non-matching pair**

(c) Process and feedback

Figure 50: *Mental Rotation Test*

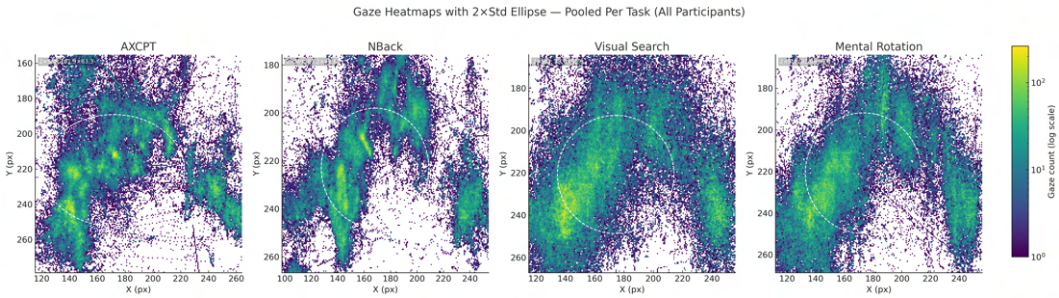


Figure 51: Gaze heatmaps for all participants for each task. A dashed ellipse (2×Std) centered at the mean indicates dispersion

and grouped into a separate data batch per 10-minute test. Finally, every row in the batch was labeled with the participant's self-reported mental-fatigue score.

Raw x,y gaze coordinates from all participants were analyzed, restricted to one minute after onset through one minute before offset for each test block. Trimmed gaze samples were then pooled across participants for each test (AXCPT, NBack, Visual Search, Mental Rotation). Spatial distributions were visualized with log-scaled 2D histogram heatmaps, zoomed to the 5–95% quantile range of X and Y. To quantify dispersion, we computed the mean and standard deviation of gaze positions per task, and overlaid ellipses corresponding to $2\times$Std centered at the mean. These metrics are summarized in the Table 10. The pooled heatmaps (Fig. 51) highlight task-specific dispersion patterns. Focused tests (AXCPT, NBack) exhibited compact high-density regions and smaller ellipses, indicating constrained gaze. In contrast, visual search and mental rotation tasks showed broader distributions and larger vertical dispersion, consistent with scanning and visuospatial manipulation. Pooling all tests would obscure these task-specific oculomotor signatures. Thus, AXCPT and NBack were combined as the Focus group. The Exploratory group was split into its two constituent trial types, reflecting both their distinct cognitive demands and the nature of the experimental tasks.

Task	Mean X	Mean Y	Std X	Std Y	N Points	Σ time (s)
AXCPT	171.8	220.8	50.9	31.6	3441	5063.4
NBack	168.2	222.8	40.6	24.9	3361	3176.2
Visual Search	174.3	220.8	39.8	27.9	3581	3490.7
Mental Rotation	173.3	220.8	40.9	28.9	3511	3688.0

Table 10: Pooled gaze dispersion statistics by task. Ellipse width/height and dispersion time are aggregated

5.1.9 TICC Clustering

The aforementioned measures resulted in a large amount of time series data. Determining four levels of mental fatigue from time series data obtained is challenging. Moreover, manual labeling of a large amount of mental fatigue time series data is labor-intensive and tedious. To this end, a highly efficient subsequence clustering method, the **Toeplitz Inverse Covariance-Based Clustering method (TICC)** method [28] (see Figure 52) was used to identify and classify multiple levels of operator mental fatigue and automatically label relevant time series data.

TICC is a model-based multivariate time series subsequence clustering method for discovering repeated patterns in temporal data [28]. TICC defines each cluster as a dependency network showing the relationships between different sensors in a short subsequence, which is able to find accurate and interpretable structure in the data. Considering that the data related to mental fatigue (e.g., eye movement metrics, task performance metrics and subjective assessment) obtained in this experiment is a multi-dimensional time series and mental fatigue is gradually accumulated over time, TICC is suitable for the problem of multi-level classification of mental fatigue in this study.

As defined by TICC, we first consider a time series of T sequential observations,

$$\mathbf{X}_{\text{orig}} = \begin{bmatrix} | & | & & | \\ \mathbf{x}_1 & \mathbf{x}_2 & \cdots & \mathbf{x}_T \\ | & | & & | \end{bmatrix}, \qquad \mathbf{x}_t \in \mathbb{R}^n, \tag{2}$$

where $\mathbf{x}_t$ is the t-th multivariate observation. In brief, the goal is to cluster these T observations into K clusters. TICC focuses on clustering a short subsequence of size $w \ll T$ that ends at t. The observations $\mathbf{x}_{t-w+1}, \ldots, \mathbf{x}_t$ are constructed into an nw-dimensional vector $\mathbf{X}_t$. As a result, a new sequence from $\mathbf{X}_1$ to $\mathbf{X}_T$ is generated, which is a helpful medium for providing proper context for each of the T observations. Therefore, the TICC method does not directly cluster the observations, but clusters these subsequences $\mathbf{X}_1, \ldots, \mathbf{X}_T$.

TICC defines each cluster as a Markov Random Field (MRF) to describe the interrelationships between various observations in a representative subsequence of the cluster [28]. Specifically,

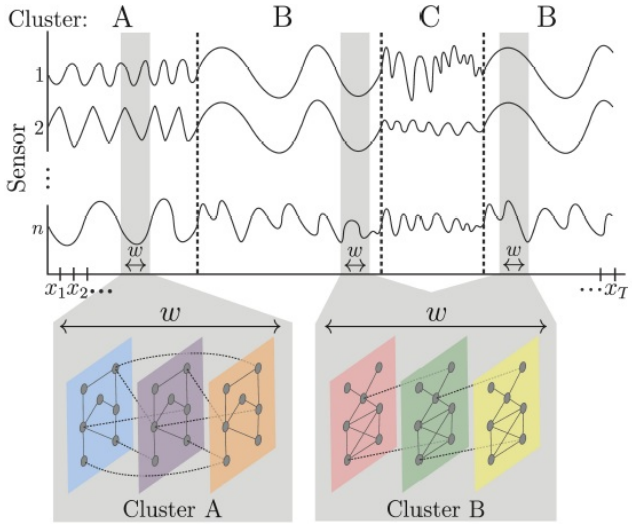


Figure 52: ***TICC segments a time series into a sequence of latent* states *(clusters), e.g.,*** A, B, C***. Each cluster is characterized by a correlation network—modeled as a Markov Random Field (MRF)—defined over a short window of size*** w***. This MRF encodes the (time-invariant) partial-correlation structure for any window that falls within a segment assigned to that cluster. In other words, windows belonging to the same cluster share the same sparse inverse-covariance (graph) pattern. The TICC algorithm jointly learns (i) the cluster-specific MRFs and (ii) the segmentation of the time series into clusters, balancing data likelihood, sparsity, and temporal consistency.***

a Gaussian inverse covariance $\Theta_i \in \mathbb{R}^{nw \times nw}$ is used as the mathematical expression of the cluster's MRF in the form of block Toeplitz:

$$\Theta_i = \begin{bmatrix} A^{(0)} & A^{(1)} & \cdots & A^{(w-1)} \\ A^{(1)^\top} & A^{(0)} & \cdots & A^{(w-2)} \\ \vdots & \vdots & \ddots & \vdots \\ A^{(w-1)^\top} & A^{(w-2)^\top} & \cdots & A^{(0)} \end{bmatrix}, \tag{3}$$

where $A^{(0)}, A^{(1)}, \ldots, A^{(w-1)} \in \mathbb{R}^{n \times n}$ describe the intra-time and inter-time partial interdependencies, so that $A^{(0)}$ defines the interrelationship between concurrent values of sensor i and sensor j (e.g., the hazard detection performance recording system and the eye tracker).

The overall goal of TICC is to solve the K inverse covariances $\Theta = \{\Theta_1, \ldots, \Theta_K\}$ and get the corresponding point assignment sets $P = \{P_1, \ldots, P_K\}$ ($P_i \subset \{1, 2, \ldots, T\}$), which is an optimization problem and can be expressed as follows:

$$\underset{\Theta \in \tau,\, P}{\arg\min} \sum_{i=1}^{K} \|\lambda \odot \Theta_i\|_1 \; + \sum_{\mathbf{X}_t \in P_i} \left(-\,\ell(\mathbf{X}_t, \Theta_i) \; + \; \beta\, \mathbb{I}[\mathbf{X}_t, \mathbf{X}_{t-1} \notin P_i]\right), \tag{4}$$

where τ is the set of symmetric block Toeplitz $nw \times nw$ matrices, $\|\lambda \odot \Theta_i\|_1$ is an ℓ_1-norm penalty of the Hadamard product to incentivize a sparse inverse covariance, where $\lambda \in \mathbb{R}^{nw \times nw}$ is a regularization parameter. $\ell(\mathbf{X}_t, \Theta_i)$ is the log-likelihood that $\mathbf{X}_t$ came from cluster i. The

penalty parameter β is used to enforce temporal consistency, and $\mathbb{I}[\cdot]$ is an indicator function judging whether adjacent points are assigned to the same cluster.

TICC solves the optimization problem expressed as Eq. (4) by two main steps: Assigning Points to Clusters and Toeplitz Graphical Lasso. A variation of the expectation–maximization (EM) algorithm is used to alternate between assigning points to clusters and then updating the cluster parameters.

Step 1. Assigning Points to Clusters. All observation points are assigned to clusters by determining C and solving the combinatorial optimization problem for $P = \{P_1, \ldots, P_K\}$, which can be expressed as follows:

$$\min_{C} \sum_{i=1}^{K} \sum_{\mathbf{X}_t \in P_i} \left(-\ell(\mathbf{X}_t, \Theta_i) \; + \; \beta\, \mathbb{I}[\mathbf{X}_t, \mathbf{X}_{t-1} \notin P_i] \right). \tag{5}$$

This step assigns each T subsequences to one of the K clusters to collectively maximize the log likelihood and the time consistency, and trade off between the two objectives adjusted by the regularization parameter β. TICC uses a dynamic programming algorithm to assign each $\mathbf{X}_t$ into a cluster.

Step 2. Updating Cluster Parameters (Toeplitz Graphical Lasso). Updating the cluster parameters $\Theta_1, \ldots, \Theta_K$ by solving the optimization problem expressed as Eq. (4) while holding the given P constant. Each Θ_i is solved in parallel, which can be expressed as:

$$\min_{\Theta_i} \; -\log\det\Theta_i \; + \; \mathrm{tr}\big(S(P_i)\,\Theta_i\big) \; + \; \frac{1}{|P_i|}\, \|\lambda \odot \Theta_i\|_1\,, \quad \text{subject to } \Theta_i \in \tau, \tag{6}$$

where $|P_i|$ is the number of points assigned to cluster i, $S(P_i)$ is the empirical covariance of these points. This is a variation on the graphical lasso problem. Alternating direction method of multipliers (ADMM) is used to solve a separate Toeplitz graphical lasso for each cluster at every iteration of the TICC algorithm to solve the overall optimization problem expressed in Eq. (4). A message passing algorithm is used to iteratively converge to the global optimal solution. The detailed TICC solution process can be referred to in [28].

5.1.10 Data clustering

As mentioned in the section subsubsection 5.1.9, clustering was performed with TICC, which partitions time series into groups with shared temporal structure. We set K=4 to mirror four mental-fatigue states and selected per-dataset parameters (Section 5.1.8) via BIC minimization [50]. The resulting eye movement clusters exhibit distinct temporal patterns. Next, we interpret these clusters in relation to mental fatigue.

For each participant-dataset pair, we computed means and z-scores for seven eye movement metrics using literature-based fatigue directions [10]. The seven metrics that we retained are the peak saccade velocity ↓, mean saccade velocity ↓, Blink count ↑, Mean pupil diameter ↓, mean fixation duration ↓, mean saccade amplitude ↓ and mean saccade duration ↑ while excluding saccade count and fixation count due to the lack of strong evidence for stable effects[10]. For each cluster c in each dataset, we then computed the **NaiveScore**$_c$ $= \sum_{j=1}^{k} s_j\, z_{c,j}$, where s_j is the sign as specified in the previous paragraph, k is the number of metrics and $z_{c,j}$ is the z-score for metric j of the cluster c . Clusters were relabeled 1–4 within

each dataset by sorting naive fatigue scores in ascending order (1 = lowest naive score, 4 = highest naive score). We then combined the relabeled datasets into a single unified dataset.

We trained a **Linear Ridge Regression model** on the unified dataset, using the seven eye metrics as predictors and the relabeled cluster index as the target. After the training, we keep four coefficient which correspond to the four eye metrics as the most reliable ones for mental fatigue classification indicators: saccade amplitude, saccade peak velocity[8], blink count[8] and mean pupil diameter[7]. We drop mean saccade velocity and saccade duration because saccade kinematics follow the main sequence, a tight amplitude–velocity–duration coupling, so these add little beyond peak velocity and amplitude [29]. We also drop fixation coefficient because its direction varies strongly with task demands[20]. We L_1-normalize the four coefficients, i.e., we rescale them so that $\sum_{i=1}^{4} |w_i| = 1$ while preserving their signs. The resulting signed coefficients are referred as weight (Table:11). We use linear ridge regression (L2) so the learned coefficients map directly to our additive fatigue score. L2 controls coefficient magnitudes (avoids extremes) and preserves non-zero coefficients (unlike LASSO), yielding stable, interpretable weights. Continuing, we follow the same procedure as Paragraph 2 of Subsection 5.5.10, but now we apply the four weights in the naive score equation by setting $k = 4$, substituting s_j with the corresponding weight w_j and replacing $z_{c,j}$ with the z-score associated with the same eye metric as the weight w_j. The resulting relabeled, unified dataset will be used to train our machine learning models.

Metric	Absolute Weight Value	Direction
Peak saccade velocity	0.2366	$\downarrow$
Mean saccade amplitude	0.3720	$\downarrow$
Blink count	0.1299	$\uparrow$
Mean pupil diameter	0.2616	$\downarrow$

Table 11: Oculomotor Weight Table

5.1.11 Machine Learning Training

Based on the comparative results in Table 12, the KNN classifier was selected as the core model for integration into the real-time pipeline. Although XGBoost achieved the highest AUC, KNN provided the best overall balance across accuracy, precision, recall, and F1-score, which makes it more suitable for deployment. For this reason, the performance of KNN was further examined using the confusion matrix and the overall ROC curve (Fig. 54). The confusion matrix shows that almost all samples were classified correctly, with only a very small number of misclassifications across the four fatigue states. The ROC curve further confirms the robustness of the model, indicating excellent discriminative ability.

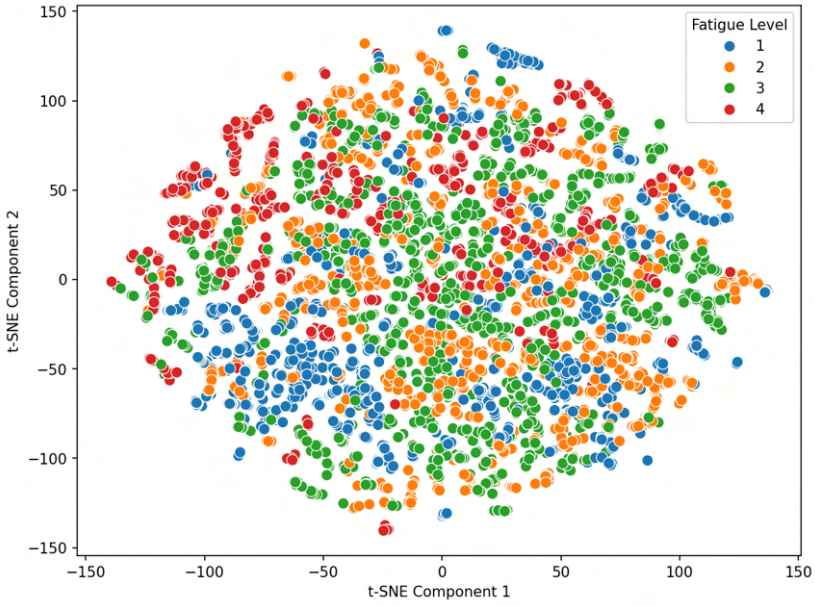


Figure 53: ***t-SNE projection of feature space showing the four ranked fatigue states (1 = lowest, 4 = highest)***

5.2 Augmented Reality for Wildfire Operations

5.2.1 Introduction

Wildfires are among the most challenging natural disasters, characterized by rapid spread, unpredictable behavior, and vast areas of impact. Effective emergency response in such scenarios requires heightened situational awareness, rapid decision-making, and seamless team coordination. Firefighting, in particular, is an inherently dangerous profession, often conducted in conditions of low visibility due to smoke, extreme temperatures, and spatial disorientation caused by debris or unfamiliar environments. These factors severely test the cognitive and physical limits of responders, increasing the risk of error and compromising

Model	Accuracy	Precision	Recall	F1-score	AUC
Baseline	0.3469	–	–	–	–
KNN	**0.9900**	**0.9900**	**0.9900**	**0.9900**	0.9974
XGBoost	0.9894	0.9894	0.9894	0.9891	**0.9987**
SVM	0.9861	0.9861	0.9861	0.9861	0.9976

Table 12: Performance comparison table of ML models; best in bold

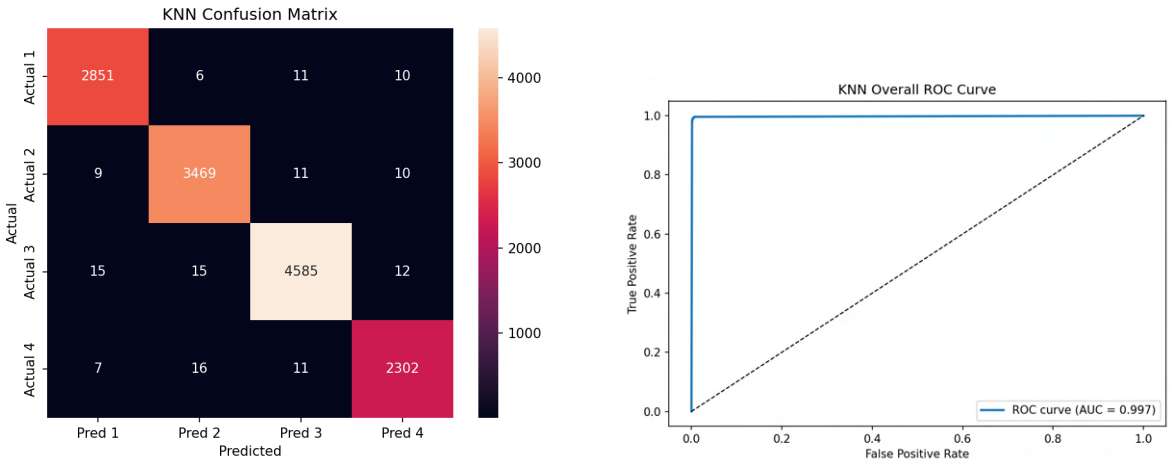


Figure 54: ***a) confusion matrix and b) ROC AUC curve***

safety. Maintaining situational awareness under such stress is therefore critical for both operational success and firefighter survival.

To address these challenges, this work presents the design and implementation of an AR system developed for the Microsoft HoloLens 2, aimed at supporting firefighting team leaders during real-time wildfire emergency operations. Leveraging recent advances in AR technology, the system provides hands free access to vital information, such as building schematics, real-time team locations, and detected hazards, directly within the user's field of view. By integrating digital overlays with the responder's natural perception of the environment, the proposed system seeks to enhance awareness, improve coordination, and reduce cognitive load in the high-stress, dynamic context of wildfire response.

5.2.2 Server and Data Communication

In this system it is crucial to maintain a continuous connection and data flow between the Kafka data broker and the HoloLens 2 devices. The server as described and in the subsubsection 5.1.2 acts as a bridge ensuring that data is transmitted in real time and that all devices are synchronized. This is particularly important in emergency situations where timely information can make a significant difference in decision-making and resource allocation.

Key responsibilities of the server include:

- **Real-Time GPS Data Transfer:** Receiving GPS data from a mobile app and relaying it to the Hololens 2 device, ensuring that the location of each team member is accurately represented in the AR environment.
- **Kafka Integration:** Subscribing to relevant Kafka topics (e.g., firespread predictions, hazard alerts) via KafkaJS and relaying those messages to connected devices.
- **WebSocket Communication:** Establishing and maintaining WebSocket connections with the mobile app and HoloLens 2 devices, allowing for low-latency, bidirectional communication.

The server is implemented in Node.js, chosen for its event-driven, non-blocking I/O model that easily scales to handle multiple simultaneous WebSocket connections and Kafka consumers. Its core modules and workflow are as follows:

- **Express.js Host:** Provides HTTP routing for health checks and initial client handshakes, and upgrades selected endpoints to WebSocket.
- **WebSocket Manager:** Maintains a pool of client sockets (mobile and HoloLens 2), each identified by a unique device ID. Incoming GPS data from the mobile app is received, optionally transformed, and then forwarded to the corresponding HoloLens 2 socket.
- **KafkaJS Consumer:** Connects to one or more Kafka topics, consumes real-time event streams (e.g., flood model outputs, hazard notifications), and pushes those messages after minimal processing, to all subscribed HoloLens 2 clients via WebSocket.
- **Data Format and Parsing:** All messages are encoded as JSON objects for lightweight, language-agnostic transmission. GPS updates include latitude/longitude and timestamp fields; Kafka messages contain a topic identifier, payload data (e.g., radiative power, hazard type), and server-assigned metadata.

A high-level sequence of interactions is:

1. **Mobile App** $\rightarrow$ **Server:** The app opens a WebSocket and streams JSON-encoded GPS positions at regular intervals.
2. **Server** $\rightarrow$ **HoloLens 2:** Upon receipt, the server looks up the target device's socket and relays the GPS JSON payload immediately.
3. **Kafka** $\rightarrow$ **Server:** The KafkaJS consumer listens for new messages on configured topics, and upon message arrival, wraps them in a minimal JSON envelope.
4. **Server** $\rightarrow$ **HoloLens 2:** The server broadcasts or directs each Kafka-sourced alert to the appropriate HoloLens 2 clients.

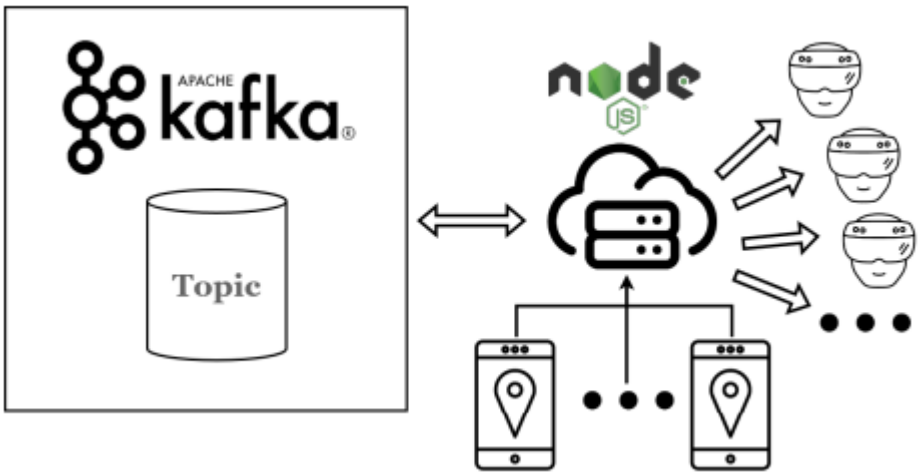


Figure 55: Server Communication Architecture

5.2.3 UI Panels and Layout

The user interface (UI) of the system is designed to be intuitive and responsive, allowing users to access critical information and controls with minimal cognitive load. The layout is structured around five main panels: the hand-following menu, the map panel, the weather information panel, and two system setup panels. Each panel serves a specific purpose and is designed to be easily accessible in the mixed-reality environment. The panels are arranged to provide a clear hierarchy of information, with the most important data and controls always available to the user.

The panels were designed using Unity's main building components, except for the hand-following menu, which was implemented using an MRTK3 prefab. The panels are visually distinct, with clear labels and icons indicating their functions. The layout is optimized for use in a mixed-reality environment, considering the user's field of view and the need for quick access to information.

Additionally, a custom script is attached to each panel to follow the user's field of view. The script uses Unity's Vector3.Lerp function to smoothly interpolate the panel's position toward a target location relative to the main camera, maintaining a consistent offset. This allows the panels to follow the user's gaze and remain in a fixed position within the mixed-reality environment. The script also includes functionality to hide and show the panels, with a toggle function that records the panel's current position relative to the camera and moves it off-screen when hidden. When the panel is shown again, it calculates a new position in front of the camera based on the camera's horizontal rotation (yaw) and the stored distance, ensuring the panel reappears appropriately oriented toward the user before resuming its following behavior.
Hand-Following Menu

This menu is a redesigned menu based on the MRTK3 `HandConstraintPalmUp` prefab, which allows users to access various system functions and settings. The menu is designed to follow the user's hand movements, allowing the user to quickly access the functions they need without interrupting their workflow. The menu includes buttons to control all the main system features, that are listed below:

- **Map Visibility.** Shows or hides the map panel, allowing users to focus on other information when needed.
- **Weather Panel.** Open the weather monitoring panel that current and future weather data.
- **Team Members Locations & POIs,** Displays the locations of team members and the POIs on the map and in the AR environment.
- **Fire Spread Simulations.** Shows or hides the fire spread simulation visualization on the map, allowing users to visualize potential fire spread patterns based on current conditions.
- **Active Fires.** Displays the locations of active fires on the map and on the AR environment, providing users with real-time information about fire activity in the area.
- **Destination Selection.** Allows users to select a destination on the map, which can be

used for navigation and routing purposes.

Figure 56: Hand Following Menu Panel

Main Map Panel

The main map panel is a central component of the system, by displaying the user's and team members' locations, the POIs, and the active fire locations with 3D markers on the map, along with a route to the selected destination.

The map panel is designed to be interactive, allowing users to Unlock and adjust the map's position and scale, and place it in a convenient location in the AR environment, based on their preferences or the current situation needs. This is achieved by using the MRTK3 `ManipulationHandler` component, which enables users to manipulate game objects in the AR environment using hand gestures. Along with a custom script that recalculates the map's position and scale relative to the user when it is locked, ensuring that it follows the user's movements and remains in the selected new position on the AR environment.

Weather Monitoring Panel

The weather monitoring panel is designed to display current and forecasted weather data, including temperature, wind speed, and wind direction. It also includes a toggle to show or hide 3D wind direction vectors on AR environment, and a slider to choose the forecasted time, allowing users to visualize how the weather conditions are expected to change over time. The panel is designed to be easy to read and understand, with clear labels and visual indicators for each data point.

System Set-Up Panels

The system setup panels are designed to allow users to configure the system settings, including the GPS location setup and device orientation calibration. These panels are designed to be user-friendly and intuitive, with clear instructions and visual cues to guide users through the system setup process. After the setup process is completed, the panels

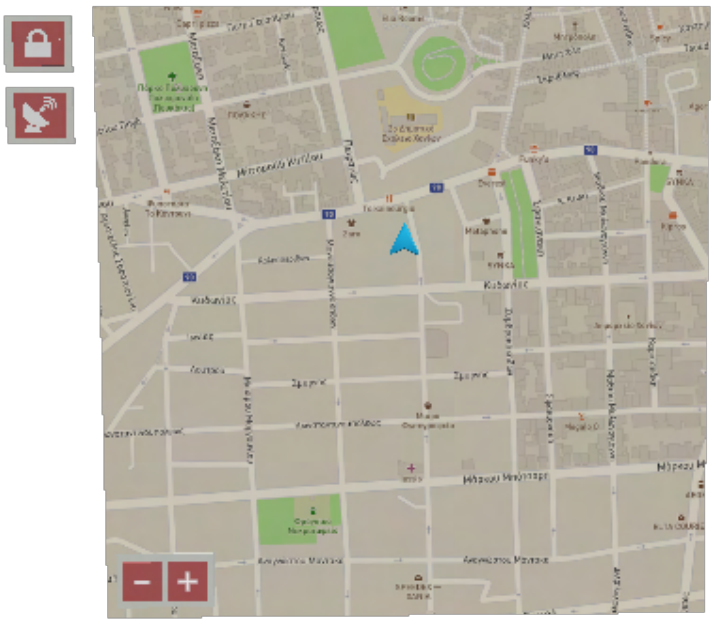


Figure 57: Main Map Panel

are hidden, and the system is ready for use. The Calibration panel can be accessed at any time to recalibrate the system if needed.

Interaction Methods

The system supports multiple interaction methods to accommodate different user preferences and scenarios provided by the MRTK3. These methods include:

- **Manual Interaction.** Users can interact with the system using hand gestures, such as tapping or dragging, to select and manipulate elements on the screen. This method is suitable for users who prefer direct manipulation of the interface and provides a tactile experience.
- **Distance Interaction.** Users can interact with the system from a distance using the hand ray pointer, which allows them to select and manipulate elements without needing to be in close proximity, with a ray casted from the user's hand. This method is useful for users who may be wearing gloves or have limited mobility, as it allows them to interact with the system without needing to reach out the target object.
- **Gaze Interaction.** The system supports gaze-based interaction, allowing users to select and interact with elements by looking at them and then confirming the selection with a hand pinch gesture. This method is particularly useful for users who may have difficulty using hand gestures or manual controls.

All the system panels and elements are designed to work with all the interaction methods, allowing users to choose the method that best suits their needs and preferences. The system

Figure 58: Weather Monitoring Panel

also includes visual feedback to indicate when an element is selected or activated, providing users with clear confirmation of their actions.

5.2.4 Location Setup and Calibration Process

The location setup and calibration process is a critical step in ensuring the accuracy and reliability of the system's AR features. This process involves configuring the system to accurately represent the user's real-world location and orientation, to display the AR objects and information correctly in the user's field of view.

The Hololens2 device does not provide a built-in method for setting the GPS location and orientation of the device, so the system does not have a way to determine the user's location and orientation relative to the real world. This can lead to inaccuracies in the AR experience, such as misaligned objects or incorrect positioning of information based on GPS data. To address this issue, the system includes a location setup and calibration process that allows users to correctly configure the system's GPS location and orientation.

GPS Location Setup

The phone application and the HoloLens2 device are paired using a unique pairing ID, which is provided to the user during the setup process on the HoloLens2 device. The user enters this ID into the phone application, establishing a connection between the two devices. Once paired, the user can choose to start sharing their GPS location with the HoloLens2 device. Then the phone application continiusly sends the location of the user to the server using a WebSocket connection. The server then sends the GPS to the HoloLens2 device, allowing the system to accurately determine the user's location in real time.

To solve the problem of setting the GPS location, a custom phone application was developed.

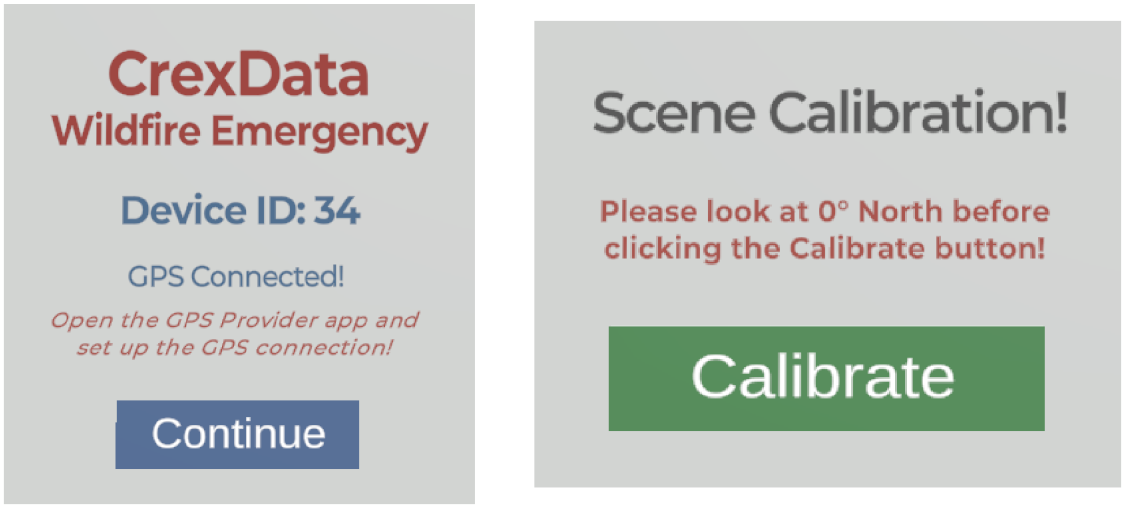


Figure 59: System Setup Panels

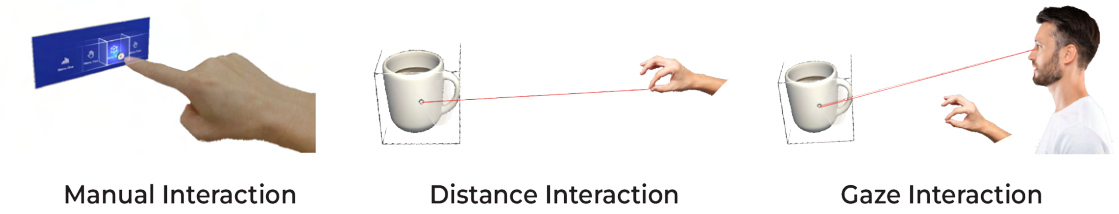


Figure 60: Different Interaction Methods

This application reads the GPS location of the user's phone, as a pair of latitude and longitude coordinates, and sends it to the server. The server then sends the GPS location to the HoloLens2 device, allowing the system to accurately determine the user's location in real time. The phone application and the HoloLens 2 device are paired using a unique pairing ID, which is provided to the user during the setup process on the HoloLens 2 device. The user enters this ID into the phone application, establishing a connection between the two devices. Once paired, the user can choose to share their GPS location with the HoloLens 2 device. Then the phone application continuously sends the location of the user to the server using a WebSocket connection. The server then sends the GPS to the HoloLens 2 device, allowing the system to accurately determine the user's location in real time.

Device Orientation Calibration

The system incorporates a calibration process deployed to align the AR environment with the real-world north. During this procedure, the user is guided through an on-screen panel displayed on the HoloLens 2, which instructs them to face the real-world north as indicated by a compass integrated into the companion mobile application. The compass provides continuous directional feedback, enabling the user to accurately align their position with the physical environment before completing the calibration.

After the user aligns their position with the real-world north, they are prompted to initiate

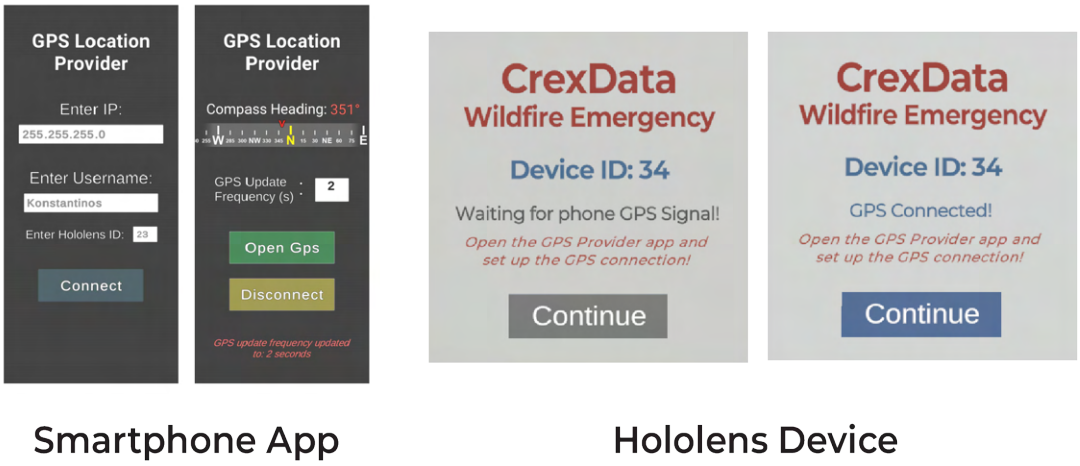


Figure 61: GPS Location Setup Process

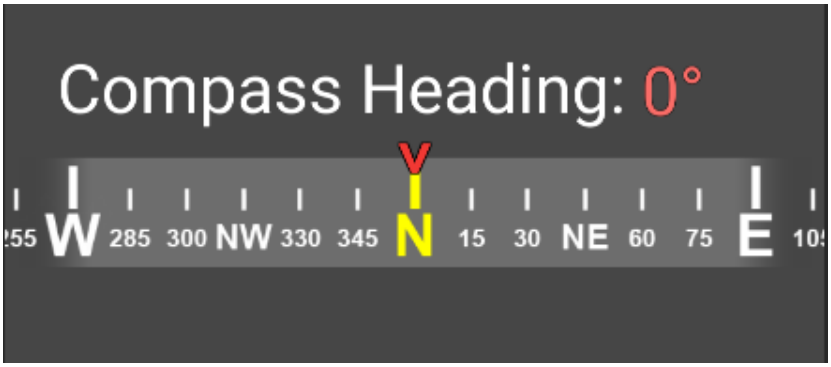


Figure 62: Compass in the Phone Application

the calibration process through the AR interface. During this step, the system determines the angular difference between the real-world and virtual north orientations. This calculated offset is then applied globally within the AR environment, ensuring that all virtual elements are correctly oriented with respect to the real-world reference. As a result, the visual alignment between digital and physical coordinates remains consistent, providing an accurate and stable augmented reality experience. Once the calibration process is complete, the system is ready to use, and the AR objects and information are displayed correctly in the user's field of view. The system also includes a recalibration button that allows the user to recalibrate the system at any time if needed

5.2.5 3D Map in AR Envrironment

This section describes the implementation of the 3D map using Mapbox, including the integration process, features, and functionalities. The Mapbox SDK was used to create a 3D map that displays the user's location, team members' locations, points of interest (POIs), and active fire locations. The map also includes a route calculation feature that allows users to navigate to a selected destination. The Mapbox SDK for Unity is a powerful tool that allows

developers to create interactive maps and location-based applications using the Mapbox platform. The SDK provides a set of APIs and components that enable developers to easily integrate Mapbox maps into their Unity projects, allowing for the creation of custom maps, geolocation features, and real-time data visualization.

To integrate the Mapbox SDK into the Unity project, the following steps were followed:

- Download the Mapbox SDK for Unity from the Mapbox website.
- Import the SDK into the Unity project using the Unity Package Manager.
- Create a Mapbox account and obtain an access token to use the Mapbox services.
- Configure the Mapbox settings in Unity, setting the access token and map style.
- Create a new map object in the Unity scene and configure its properties, such as location, zoom level, and map style.

The Mapbox features implemented in the system are a 3D map display based on the user's GPS location and a route calculation feature that allows users to navigate to a selected destination.

Display the Map

To display the map in the AR environment, the user needs to select the map from the hand-following menu. Then the system calls the Mapbox API to generate the map tiles based on the user's GPS location and the selected zoom level, then the map is displayed in the AR environment, along with the user markerpointing to the user's GPS location.

Figure 63: Open the Map Panel

Additional Map Features

- **Map Repositioning**: The map repositioning feature allows users to move the map in the AR environment by using hand gestures. To use this feature, there is a toggle lock/unlock button on the map panel. When the map is unlocked, the user can move

the map by dragging it with their hand and place it the desired location in the AR environment based on the current situation or the user's preferences. When the map is locked again, a function is called to recalculate the map's offset position relative to the user's position, ensuring that the map follows the user and remains in the selected position in the AR environment.

- **Zoom In/Out**: Additionally, the map includes a zoom in/out feature that allows users to adjust the map's scale to view a larger area or to zoom in and view more details around their location. This feature is implemented using a functions that changes the zoom variable in map script provided by Mapbox SDK and recalls the Mapbox API to update the map tiles based on the new zoom level. Also to keep the map marker in the correct position on the map after the scale change, a function is implemented that recalculates all the spawned markers' positions based on the new scale. A problem that was encountered during the implementation of this feature was that the map generated by Mapbox SDK does not have a constant size when different zoom levels are selected. This means that the 3D map object size changes slightly when the zoom level is changed, which is causing overplaping problems between the map and the UI elements. To solve this problem, we used the MRTK3's Magic Windows prefab which is

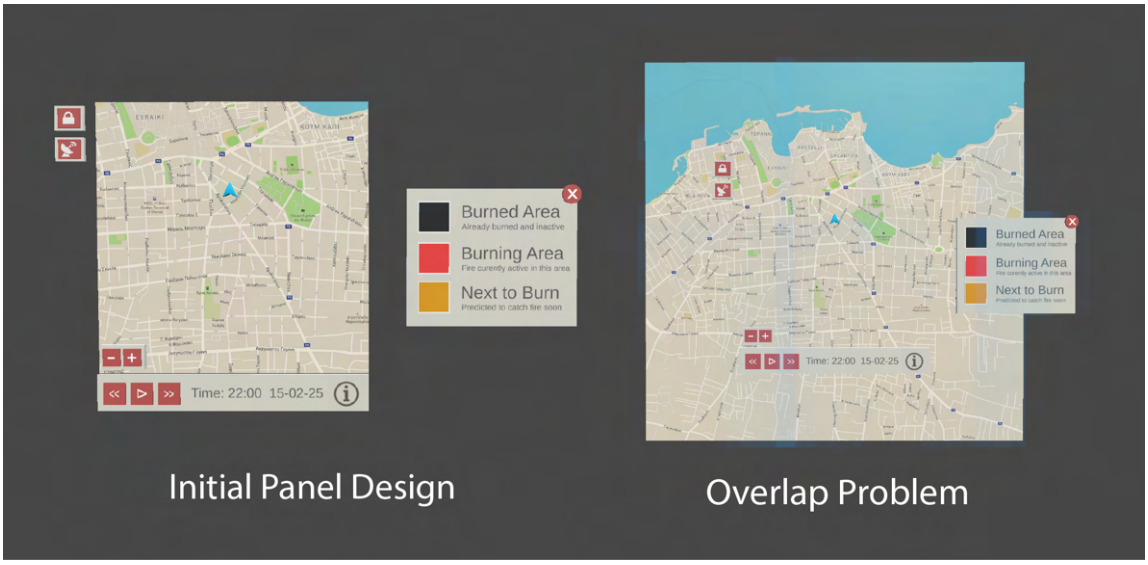


Figure 64: Map Overlapping Problem

included in the MRTK3 package. The Magic Windows prefab consists of a square mesh with a transparent material that acts as a mask, and then there is another material that is applied to the objects that has to be masked.

To apply this technique to our problem, the square mesh is placed in front of the map with the desired size, and then added the second material to the map tiles and all the map markers. This way the system only renders elements that are inside the square mask, and keeps everything outiside the choosen area invisible for the user, maintaining a constant map size, preventing overlapping problems with the UI elements and providing a better experience for the user.

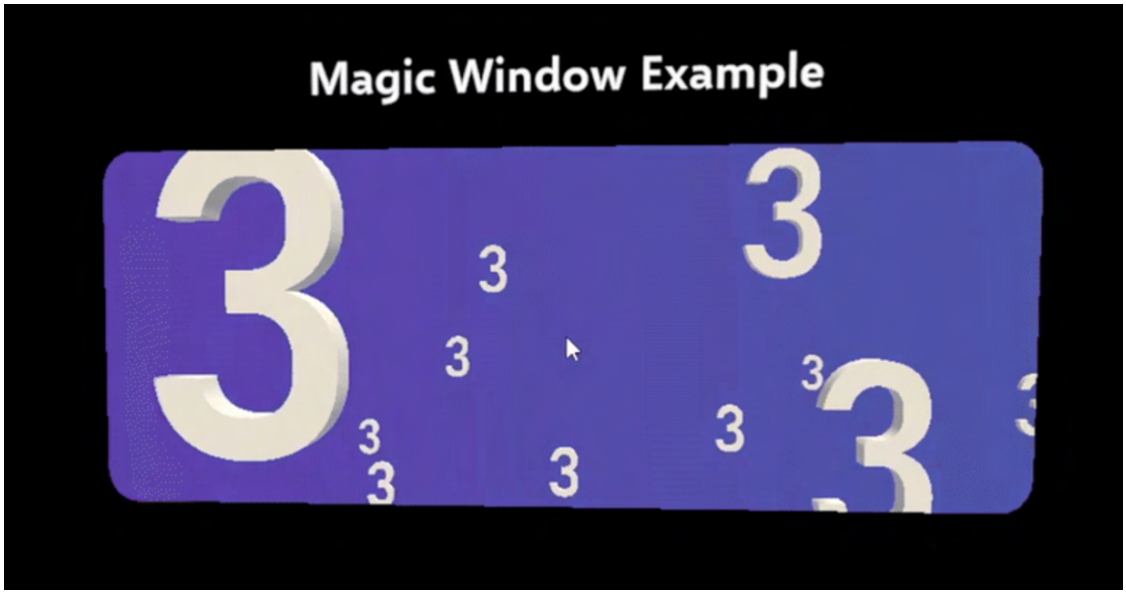


Figure 65: Magic Window Example

- **Street View and Satellite Mode**: The map also includes a street view and satellite mode feature that allows users to switch between different map styles. This feature is implemented by a toggle button on the map panel, and a function that changes the selected style variable in the map script. The map styles are defined in the Mapbox settings, and the function recalls the Mapbox API to update the map tiles based on the selected style. This feature allows users to view the map in different perspectives, providing more context and information about their surroundings.

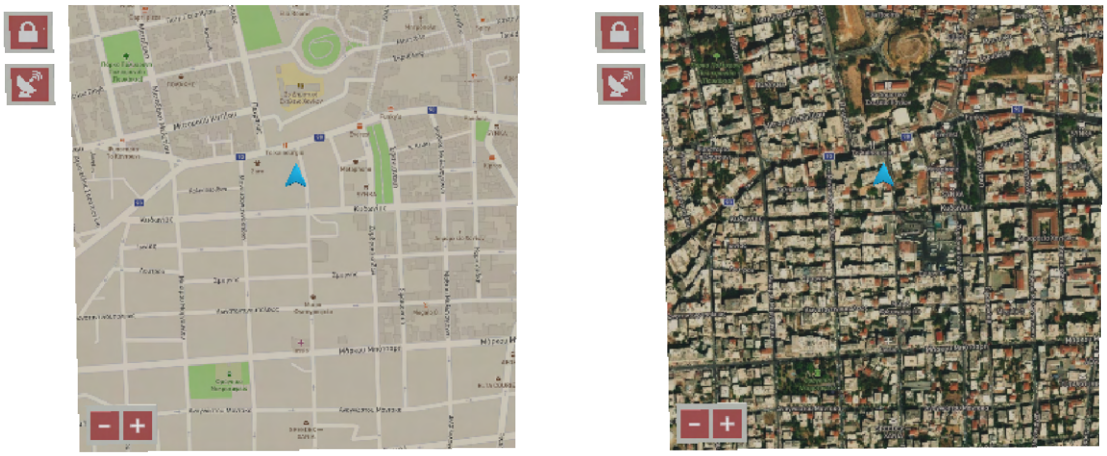


Figure 66: Street View and Satellite Mode

- **Spawn Markers on the Map**: The system places all markers on the map—such as the user, team members, points of interest, and active fire locations—by using their GPS coordinates and some basic information about how each marker should look and be

organized. For each marker, it knows which map to use, what visual icon to show, where it should appear based on its location, how big it should be, which group it belongs to (for example, all team markers together), and what name it should have. It then converts the GPS position into the correct spot on the map, adjusts the marker's size, aligns it with the map's orientation, and groups and labels it so everything is clearly displayed and easy to manage.

5.2.6 Navigation and Routing

This section outlines how navigation and routing are implemented in the AR system. Using the Mapbox Directions API, the system calculates optimal routes from the user's current position to selected destinations such as POIs, team members, or chosen points on the map—based on real-time routing information.

Route Calculation and Visualization

To calculate the route, the system used a function that takes the user's current GPS latitude and longitude coordinates and the destination point coordinates as input parameters. The function then calls the Mapbox Directions API providing it with the starting and destination coordinates, and the mode of transportation (driving, walking, cycling, etc.), for our system we used the walking mode. Then the API gives a `respose` that cointains a list of GPS coordinates representing all the checkpoints the user needs to follow between the starting point and the destination. The system then uses these coordinates to create a line renderer object that connects all the checkpoints, displaying the route on the map. After the route is calculated, the system procceds to visualize the route on the maplong with an AR Guidance marker that shows the direction to follow.

Figure 67: Line Renderer Setup Function

3D Map Display

To display the route on the map, the system uses the Unity's Line Renderer component, which creates a line connecting all the given points in the unity world space. In our case to correctly display the route on the map, the system first converts the GPS coordinates of the route checkpoints to Unity world space using the `GeoToWorldPosition` function provided that was explained in the previous section.

Figure 68: Traslating GPS Coordinates to Unity World Space

The system then calls a new function called `LineRendererSetup` that takes the new route coordinates and creates a new game object with the Line Renderer component. The function also sets the material, width, and other properties of the line renderer to customize its appearance and placing it correctly on the map, to match the system's design and provide a clear visual representation of the route.

AR Route Guidance

To provide users with real-time guidance along the route, the system displays an easily visible AR marker in the form of a 3D arrow, designed in Blender. This arrow is shown at a fixed distance in front of the user, slightly above the ground, and moves with them in the AR environment, giving a clear and continuous indication of the direction they need to follow toward the next checkpoint in the real world. To ensure that the arrow correctly points to this checkpoint, the system first converts the GPS coordinates of both the user and the next checkpoint into the AR scene's coordinate system using standard geospatial calculations based on the WGS84 ellipsoid model and the Haversine formula, which express latitude and longitude differences as distances in meters. The resulting position is used to place an invisible target marker at the next checkpoint inside the AR scene. The system then determines the direction from the user to this target and orients the 3D arrow accordingly, so that it always faces the next checkpoint while staying anchored at a fixed distance in front of the user as they move.

5.2.7 Team Members' Locations Visualization

This section describes the implementation of the team members' locations visualization feature in the system. This feature is designed to provide users with real-time information about the locations of their team members, enhancing situational awareness and coordination during firefighting operations. **Phone - Server - HoloLens Communication**

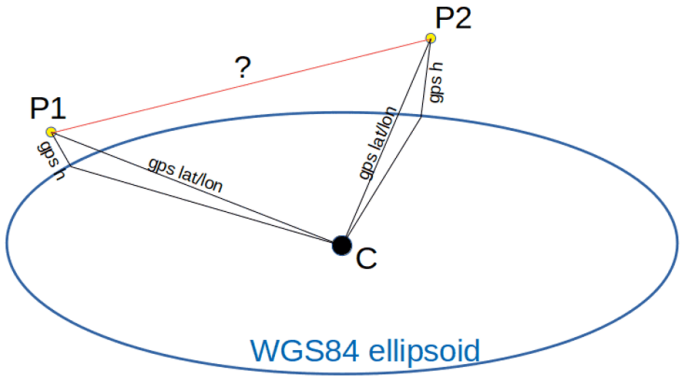


Figure 69: WSG84 Ellipsoid Model

To provide the GPS data to the server, each team member must have an android phone device with the GPS location provider application installed, the same application that was used to set the GPS location of the user. The application then continuously reads the GPS location of the user's phone and sends it to the server using a WebSocket connection, along with the user's name. After the server receives GPS data from team members, it immediately sends this information in JSON format to the team operator's HoloLens device, tagged with the `"teamMember"` message type so it can be correctly identified. The system then reads the incoming message and extracts key details such as the user's name, ID, and GPS coordinates (latitude and longitude). Using this information, it checks whether the team member is already registered in its internal list: if not, it creates a new firefighter entry and adds it to the list; if the team member is already present, it simply updates that person's stored GPS coordinates with the latest data.

Display Team Members' Locations

After the sysyem stores the GPS data of the team members, it can display their locations both in the AR environment and on the map. To display team members on the map, the system follows the same process used for placing other markers. For each team member in the list, it takes the user's GPS coordinates and the appropriate marker type, then creates a visual marker on the map and positions it according to the user's real-world location. In this way, every team member appears on the map at the correct position, updated in real time.

Display Team Members' Locations

To display the team members on the map, the system follows the same process used for the other markers. For each team member in the list, it uses the user's GPS coordinates together with the appropriate marker representation to create a marker on the map and place it at the corresponding position. This ensures that all team members are visualized on the map at locations that match their real-world positions.

To display the team members in the AR environment, the systemfirst translates the GPS coordinates of each team member to Unity world space using the same technique that was

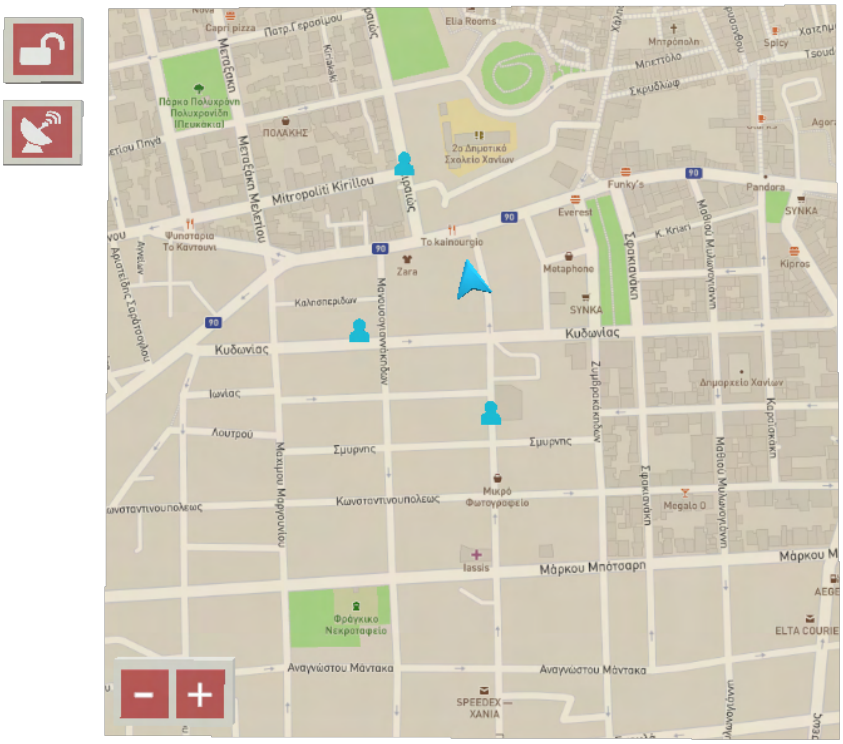


Figure 70: Team Members Display on the Map

used to place the checkpoints in the scene for the routing feature. And the it spawns a 3D human figure prefab at the translated coordinates. The human figure helps the user to easily identify the positions of the team members in the AR environment, and it is easily visible and distinguishable from other AR objects.

To navigate to a team member, the user can click on any team member's marker on the map, which will open a pop-up menu with the team member's name and a button to navigate to their location. When the user clicks the `Show Route` button, which calls the `CalculateRoute` function from the RoutePlanner script, passing the selected team member's GPS coordinates as the destination point. The system then calculates the route to the selected team member and displays it on the map, along with the AR guidance marker pointing to their location. This feature allows user to quickly and easily navigate to their team members locations, enhancing coordination and communication during firefighting operations.

5.2.8 Points of Interest (POIs) Visualization

The POIs are designed to be easily identifiable and distinguishable from other AR objects in the scene. Currently, the system is designed to be extensible, allowing for the addition of new POI types in the future, but currently it includes the following types of POIs:

- **Water sources.** Indicating the location of water sources in the area.
- **Power Lines.** Pointing to the locations of power lines in the area, which can be

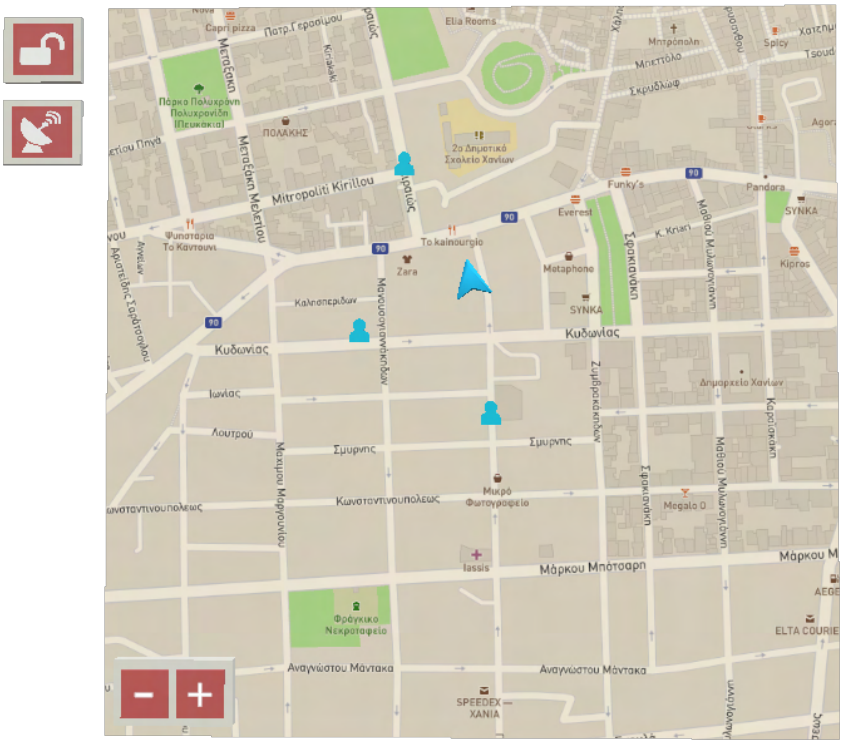


Figure 71: Team Members Display on the Map

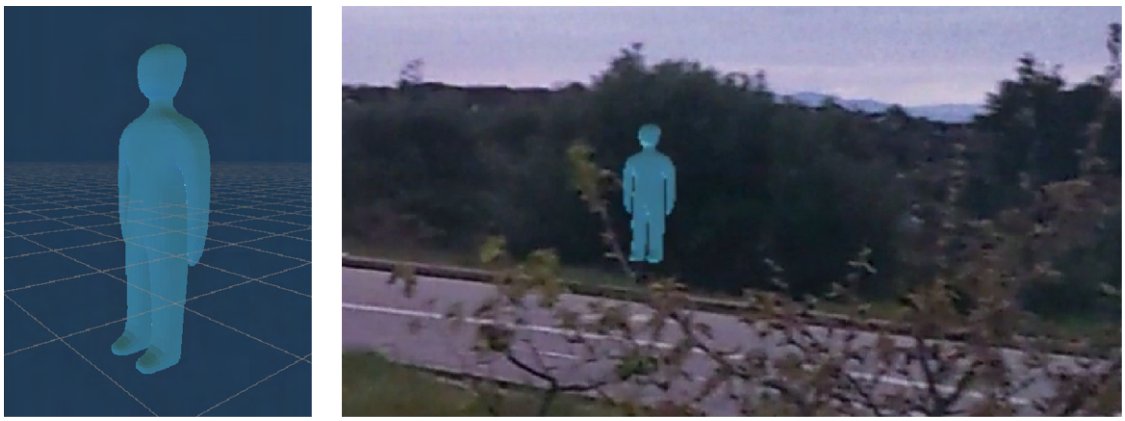


Figure 72: 3D Human Figure Prefab

hazardous during firefighting operations.

- **Danger Zones.** Represented by a danger symbol, indicating spots that are hazardous or unsafe for firefighting operations.
- **Settlement Areas.** Indicating the locations of settlements or populated areas close to the fire zone.

Each POI type is represented by a unique vector image, designed to be easily recognizable

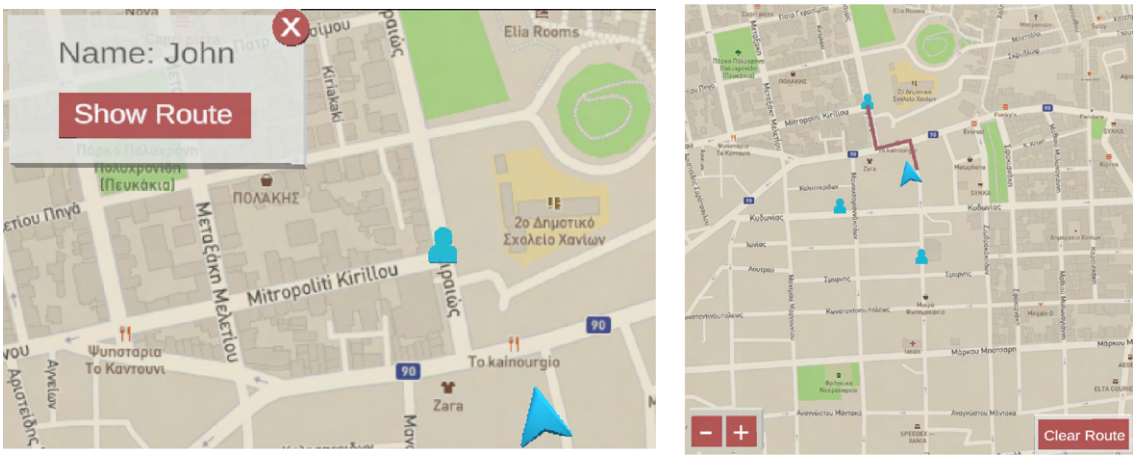


Figure 73: Navigating to a Team Member

and distinguishable from other POIs.

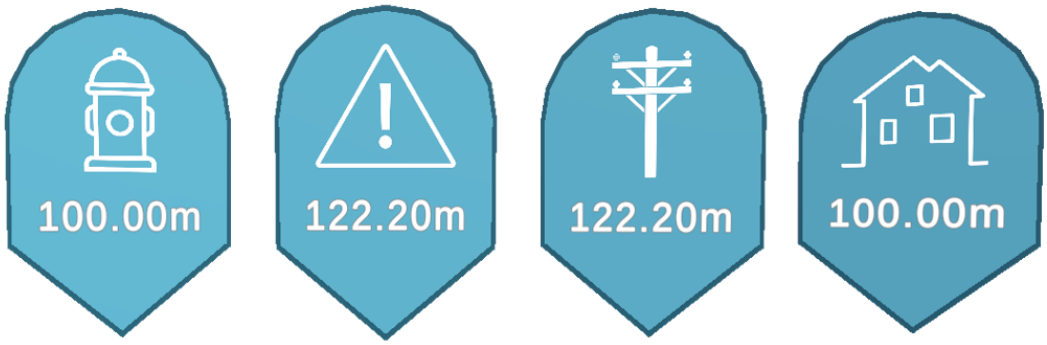


Figure 74: Types of POIs

The system receives Point of Interest (POI) data from the server in JSON format, tagged with the `"spawnPOI"` message type to distinguish it from other incoming messages, such as team member locations. Upon arrival at the HoloLens 2 device, the message is processed and the POI's name, type, and GPS coordinates are extracted. Using this information, the system creates a new POI entry and adds it to the internal list of POIs, so that it can later be visualized both in the AR environment and on the map.

Each Point of Interest is then displayed on the map following the same procedure used for the other markers. Based on the POI's type, an appropriate visual representation and scale are selected, and a corresponding marker is created at the position derived from its GPS coordinates. In this way, all POIs are clearly shown on the map at their correct real-world locations, consistent with the representation of team members and other spatial information.

After the POIs are displayed on the map, the user can select any POI marker as a navigation

target. Once a POI is selected, the system interprets its GPS coordinates as the destination, calculates an optimal route from the user's current position, and displays this route on the map. In this way, the user is provided with a clear path to critical locations or resources. At the same time, the AR guidance marker is updated to point toward the selected POI, allowing the user to navigate easily to the chosen location in the real world.

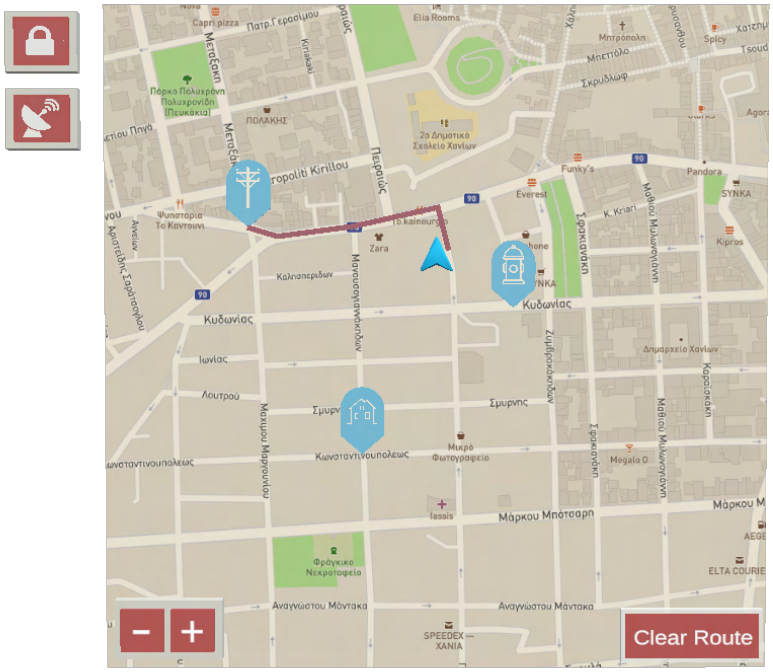


Figure 75: POI display on the map.

The system also displays POIs directly in the AR environment, providing users with real-time spatial awareness of important locations around them. To achieve this, it applies the same geospatial conversion technique used for team members, translating each POI's GPS coordinates into positions in the Unity world space. At these positions, corresponding 3D POI objects are created and organized within the scene hierarchy, ensuring that the AR view remains clear, consistent, and easy to manage.

Figure 76: POI Display in the AR Environment

To provide users with additional information about the POIs, each POI prefab includes a distance indicator that shows the distance from the user to the POI. The distance indicator is a simple text label that displays the distance in meters, and it is updated every few seconds to reflect the user's current position. The distance is calculated by subtracting the user's position from the POI position in the Unity world space.

Visibility Issues and Solutions

When displaying the POIs in the AR environment, the system encountered visibility issues with the POIs that were far away from the user. The POIs were not clearly visible in the AR environment, making it difficult for the user to identify their type and their distance. To solve this problem, the system includes a custom script attached to each POI prefab on the scene, which based on the distance from the user that was calculated in the previous section, adjust the prefab's scale to be larger when the user is far away from the POI, and smaller when the user is close to the POI. This way, the system ensures that all POIs are clearly visible in the AR environment, regardless of their distance from the user.

Each POI prefab also includes a script that rotates the prefab to always face the user, ensuring that the POIs are always visible from every position in the AR scene. This is achieved by using the `LookAt` function in Unity, which rotates the prefab to face a target object in the unity

world. In this case, the target object is the user's camera, which resemples the Hololens 2 position in the AR environment.

5.3 Fire Monitoring

This section describes the implementation of the fire monitoring feature in the system. The fire monitoring feature is designed to provide users with real-time information about active fires in the area, helping them to make informed decisions during firefighting operations. The system is designed to display the active fire locations in both the AR environment and on the map, providing users with real-time information about the fire's location and behavior.

Active Fire Locations

The system receives the active fire locations data from the NASA FIRMS API, which provides information about active fires detected by satellite, three hours after the fire is detected. The Data is provided to the system in JSON format, each fire location is represented by a latitude and longitude coordinates, along with additional information such as the fire's confidence level and brightness. The system then parses the JSON file to extract the relevant information and store it in a list of fire data objects, which are then proccessed and displayed in the AR environment and on the map.

Example of the JSON response from the NASA FIRMS API:


```
{
    "latitude": 38.37386,
    "longitude": 24.06201,
    "acq_date": "2025-05-12",
    "acq_time": "0005",
    "confidence": "n",
    "brightness": 300.65,
    "satellite": "N21",
    "track": 0.38,
    "instrument": "VIIRS",
    "bright_t31": 284.73,
    "frp": 0.97,
    "version": "2.0NRT",
    "scan": 0.42,
    "daynight": "N"
}
```


Figure 77: NASA FIRMS API JSON Response

The JSON response contains a list of fire data objects, each containing the following informa-

tion:

- **latitude & longitude.** The GPS coordinates of the fire location, indicating its position in the world.
- **brightness.** The brightness value of the fire, indicating its intensity.
- **confidence.** The confidence level of the fire detection, indicating the reliability of the data.
- **acq_date, acq_time & daynight.** The date and time of the fire detection, along with the day or night indicator, indicating whether the data was collected during the day or night.
- **satellite, instrument & version.** The satellite and instrument used to detect the fire, along with the version of the data.
- **bright_t31 & frp.** Additional data related to the fire detection, such as the brightness temperature and fire radiative power.
- **scan.** The scan angle of the satellite when the fire was detected.

Then used a python script to parse the JSON response and erase all the unnecessary data, and focus only on a certain area based on the user's location. The script then saves the filtered data in a new JSON file, which is then used by the Unity system to display the active fire locations.

Display Fire Locations on Map

To display the active fire locations on the map, the system first processes a JSON file containing fire data and converts it into an internal list of fire objects. When the user chooses to visualize fire locations, the system goes through this list and, for each active fire, creates a corresponding marker on the map. Using the same general mechanism employed for other markers, each fire marker is positioned according to its GPS coordinates, ensuring that all active fires are accurately represented on the map at their real-world locations.

3D Representation in AR

To display the active fire locations in the AR environment, the system applies the same technique used for team members and POIs. The GPS coordinates of each fire are converted into positions in the Unity world space, and a corresponding 3D fire object is then placed at each of these locations.

The fire object is a dedicated 3D model designed to represent active fires in the AR environment, similar in concept to the POI markers, and it also includes a distance indicator showing how far the user is from the fire location. This distance is computed using the same approach as for POIs and is updated periodically to reflect the user's current position, providing users with an up-to-date sense of proximity to each active fire.

Fire Spread Visualization and SPARK

Figure 78: Active Fire Locations Display on the Map

Figure 79: Active Fire Locations Display in the AR Environment

The fire spread visualization feature is designed to provide users with real-time information about the potential spread of the fire in the area. The system uses the output data provided by the SPARK fire spread model to calculate the potential spread of the fire based on the current weather conditions and the fire's location. The SPARK fire spread model provides a set of parameters that describe the potential spread of the fire, including the fire's location, arival time, the fire's intensity and flame height. These data are aranged as a grid of cells, each cell representing a specific 30m x 30m area in the world and containing the fire data for that area the time of the fire's arrival.

These data are provided in a CSV file format, that contains the following columns:

- **X.** The X coordinate of the cell center in Web Mercator (EPSG:3857) projection.
- **Y.** The Y coordinate of the cell center in Web Mercator (EPSG:3857) projection.
- **Arrival Time.** The time of the fire's arrival in the cell, in seconds since the start of the simulation.
- **Flame Height.** The height of the flames in the cell, in meters.
- **Fire Intensity.** The intensity of the fire in the cell, in kilowatts per meter (kW/m).

The system reads the fire spread information from a CSV file and converts it into a theoretical grid, where each cell corresponds to one entry of the fire spread data. For every cell in this grid, the system stores its position (X and Y coordinates) together with the associated fire information. These cells are then instantiated in the scene as visual elements, each rendered with a color that reflects its current state based on the fire's arrival time. In this way, the grid provides an intuitive, color-coded representation of how the fire is expected to propagate across the area.

There are three different colors used to represent the fire spread data:

- **Red.** The cells where the fire is at the moment.
- **Yellow.** Indicates that the fire will arrive in the cell in the next 5 minutes.
- **Black.** The burned area, indicating that the fire has already passed through the cell.

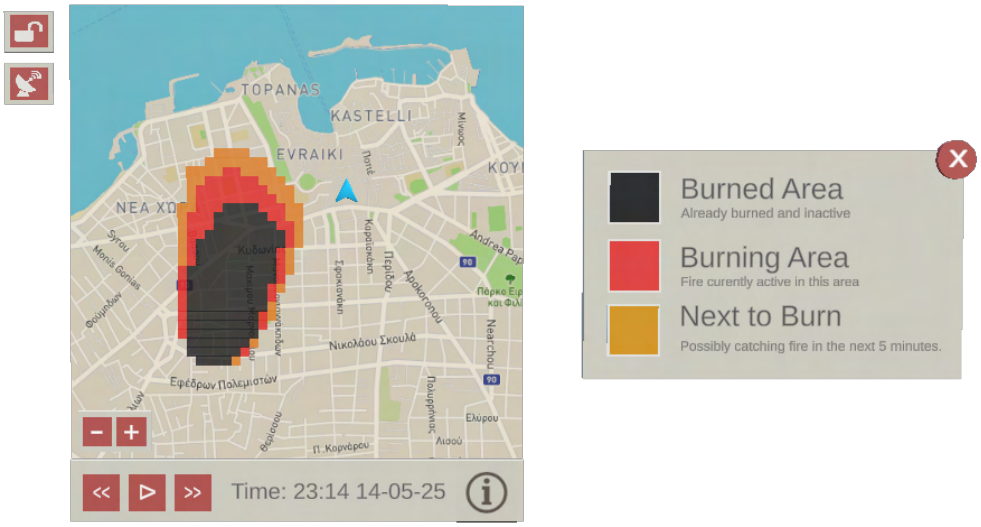


Figure 80: Fire Spread visualization on the Map

Controlling the Fire Spread Visualization

The system includes a simple UI panel that allows the user to control the fire spread visualization. The panel includes two back and forward buttons that allow the user to navigate through

the fire spread data, with a time interval of 1 minute, and a play/pause button that allows the user to start and stop the simulation.

When the user clicks the forward button, the system advances the current time step by one unit and updates the fire spread visualization accordingly. It revisits each cell in the grid and adjusts its color based on the fire's arrival time at that location, so that the overall display reflects the current state of the fire's progression across the area.

When the user presses the back button, the visualization steps one time unit backward, allowing the user to review earlier stages of the fire's evolution. The play/pause control offers a similar form of interaction by automatically advancing the fire spread visualization over time, updating the display at regular intervals until the user pauses the simulation. To indicate the temporal context of the currently displayed fire state, the interface includes a text label showing the corresponding date and time, which is refreshed whenever the visualization changes.

Alongside the control buttons, an information panel explains the meaning of each color used to represent different fire states. This panel is designed to be clear and intuitive, enabling users to quickly interpret the fire spread, navigate through different time steps, and better understand how the fire may evolve across the area. In this way, the interface supports more informed decision-making during firefighting operations.

5.4 Weather Data Monitoring

This section describes the implementation of the weather data monitoring feature in the system. The access to real-time weather data is crucial for firefighting operations, as it provides users with important information about the current and forecasted weather conditions in the area, helping them to make more accurate predictions of the fire behavior and progress in the area. The main weather panel is based on a panel prefab of the MRTK3 package, and was modified to fit the system's design, and to display the weather data in a clear and organized way. The weather panel includes the following information:

- **Current Temperature.** Displayed in Celsius degrees.
- **Wind Speed.** Displayed in meters per second (m/s).
- **Wind Direction.** Displayed in degrees, indicating the direction from which the wind is blowing.

Along with an uncertainty value for each one of the parameters, indicating the level of uncertainty in the data.

Data Source

The system retrieves weather information from the OpenMeteo API, which provides real-time data and hourly forecasts for the next 24 hours for any location worldwide. To obtain this information, the system sends a request to the API using the user's current latitude and longitude. Once the response is received, the JSON data is processed and the relevant weather details such as temperature, wind speed, and humidity—are extracted for use within the application.

Figure 81: Weather Data Panel

The API request URL used in the system is:
`https://api.open-meteo.com/v1/forecast?latitude={0}&longitude={1}`
`&current_weather=true&hourly=temperature_2m,windspeed_10m,winddirection_10m`
`&forecast_days=1`

After receiving the JSON response from the Open-Meteo API, the system processes it to extract the current temperature, wind speed, and wind direction, and stores these values in an internal list of weather data objects for later presentation to the user.

The JSON response from the Open-Meteo API is structured as follows:

The JSON response has two parts, the `current_weather_units` object that contains the current weather data, and the `hourly_units` object that contains the forecasted weather data for the next 24 hours. Each weather data object contais the reqested data types listed below:

- **temperature_2m.** The temperature at 2 meters above ground level, in Celsius degrees.
- **windspeed_10m.** The wind speed at 10 meters above ground level, in meters per second (m/s).
- **winddirection_10m.** The wind direction at 10 meters above ground level, in degrees.

Uncertainty Visualization

The system includes a slider that allows the user to view the forecasted weather data for the next 24 hours. The user can use the slider with a simple grab gesture to select a specific hour, and the system will retrieve the corresponding weather data from the list of weather


{"latitude":35.5,"longitude":24.0,"generationtime_ms":0.054001808166503906,"utc_offset_seconds":0,"timezone":"GMT","timezone_abbreviation":"GMT","elevation":12.0,

"current_weather_units":{"time":"iso8601","interval":"seconds","temperature":"°C","windspeed":"km/h","winddirection":"°","is_day":"","weathercode":"wmo code"},
"current_weather":{"time":"2025-05-11T02:15","interval":900,"temperature":17.1,"windspeed":5.0,"winddirection":111,"is_day":0,"weathercode":0},

"hourly_units":{"time":"iso8601","temperature_2m":"°C","windspeed_10m":"km/h","winddirection_10m":"°"},
"hourly":{"time":["2025-05-11T00:00","2025-05-11T01:00","2025-05-11T02:00","2025-05-11T03:00","2025-05-11T04:00","2025-05-11T05:00","2025-05-11T06:00","2025-05-11T07:00","2025-05-11T08:00","2025-05-11T09:00","2025-05-11T10:00","2025-05-11T11:00","2025-05-11T12:00","2025-05-11T13:00","2025-05-11T14:00","2025-05-11T15:00","2025-05-11T16:00","2025-05-11T17:00","2025-05-11T18:00","2025-05-11T19:00","2025-05-11T20:00","2025-05-11T21:00","2025-05-11T22:00","2025-05-11T23:00"],
"temperature_2m":[17.6,17.4,17.2,16.8,16.8,18.0,19.1,19.9,20.3,20.9,21.2,21.3,21.1,21.1,21.0,21.0,20.2,19.2,18.2,17.8,17.4,17.2,16.9,16.9],
"windspeed_10m":[4.4,4.8,5.0,4.5,4.2,4.7,10.5,13.0,15.3,15.9,15.6,15.8,14.3,12.1,10.7,8.1,7.4,4.1,4.6,1.1,2.3,3.4,4.4,2.9],
"winddirection_10m":[99,103,111,119,121,67,49,46,41,39,29,24,18,17,20,32,23,15,315,288,219,238,215,210]}}

Figure 82: Open Meteo API JSON Response

data objects and display it in the weather panel. The slider is also based on the MRTK3 slider prefab, and it is designed to be easy to use and intuitive for the user. The slider is set to a range of 0 to 23, representing the 24 hours of the forecasted data, and it includes a label that displays the selected hour and date.

Figure 83: Weather forecast slider

Wind Direction Visualization in AR

Along with the weather data panel, the system also includes a wind direction visualization feature that displays the current wind direction in the AR environment. The wind direction is represented by some 3D arrows that are spawned in the AR environment, moving in the direction of the wind.

To support this feature, the system includes a wind direction visualization module that is controlled through the Wind Direction Display toggle on the weather panel. When the user activates this option, the system generates a grid of arrows in the AR environment, using a predefined grid size and spacing. Each arrow is instantiated as a 3D object and arranged in a regular pattern, forming a field of indicators that can later be oriented to show the current wind direction across the area.

Whenever the weather data is updated, the system also refreshes the orientation of all wind

arrows in the AR environment so that they match the latest wind direction. The wind direction, expressed in degrees, is used to rotate each arrow so that it correctly points in the direction the wind is blowing.

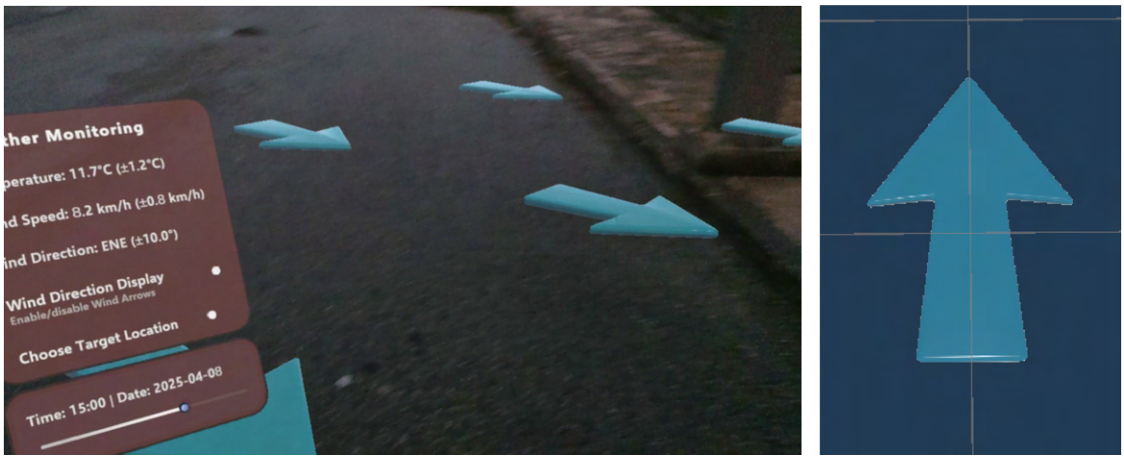


Figure 84: 3D wind arrows display

To convey a sense of motion, each arrow includes a simple behaviour that makes it move periodically in the direction of the wind. The arrow advances forward along its pointing direction for a fixed distance and then returns to its original position, creating a looping animation that visually reinforces the wind flow. Both the speed of this movement and the distance traveled can be adjusted in the Unity inspector, allowing the animation to be easily tuned to different visualization needs.

Weather Data on Target Location

Another feature implemented in the system is the ability to display current and forecasted weather data for any selected location on the map, a requirement that emerged during the initial trials. To achieve this, the system spawns a movable pointer on the map, which the user can drag to the location of interest. The pointer's position relative to the map is then converted into GPS coordinates using the geospatial conversion tools provided by the Mapbox SDK, and these coordinates are used to request updated data from the Open-Meteo API. Once the API response is received, the JSON data is processed in the same way as for the user's own location, and the corresponding weather information is presented in the weather panel.

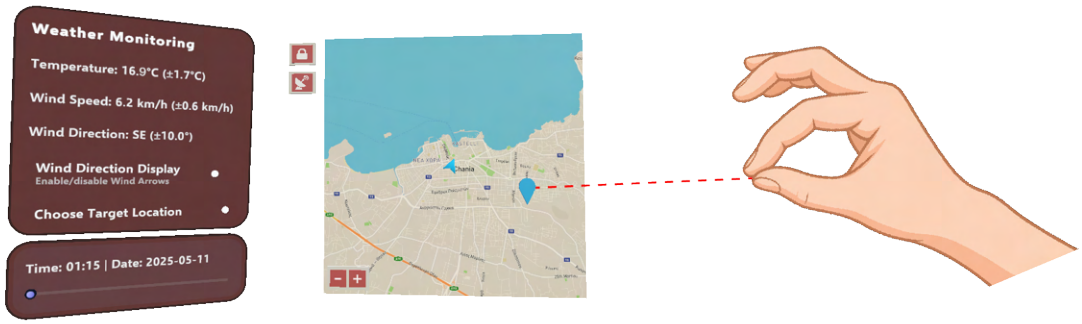


Figure 85: Weather data visualization for a selected target location.

5.5 Wildfire Evaluation

Test in Lab

The initial trials were conducted locally in our lab, involing two participants, one firefighting Officer and one urban planning professor. The trials were designed to test all the implemented features of the system, and to gather feedback about the usability and effectiveness of the system in real-world scenarios. The participants were asked to perform a series of tasks using the system and to provide feedback about their experience. The participants provided valuable feedback about the system, focusing on the usability and effectiveness of the implemented features. They mentioned that all the implemented features would be useful for firefighting operations, and they found the system easy to understand and use. They also provided suggestions for improvements and additional features that could enhance the system's usability and effectiveness. Some of the suggestions have already been implemented in the system after the trials, while others are still under consideration for future work.

Implemented Suggestions:

- The participants had difficulty using the menu buttons, as some of them were too far from the user, making it difficult to select them with the hand gestures. So they suggested to place the menu buttons closer to the user for easier access. The system was modified to allow allow the user to directly select the menu buttons with a simple hand gesture, and also enabled the gaze selection feature, which allows the user to select a UI element by looking at it, making it easier to interact with the menu buttons.
- They also requested the ability to display weather information for a selected location on the map, rather than just the user's current location. This feature was implemented in the system after the trials, by lettng the user to move a pointer prefab on the map to the location of interest, to display the weather data for that location.
- The participants expressed a preference for a satellite map style, that would provide a more realistic view of the environment and help them more when using some of the system's features. This feature was also added to the system, by adding a toggle button on the map panel that allows the user to switch between street view and satellite view on the map.

Other Suggestions:

- They suggested adding more information about the route, such as if there is a bridge or a tunnel on the route, or the conditions of the path, to help them make more informed decisions about the route to take.
- The participants requested a local wind direction feature, which would provide information about the wind changes effected by the fire in the area.
- They also suggested providing more frequent predictions for the weather data, to help them make more accurate predictions about the fire behavior and progress in the area.

The initial trials provided valuable feedback about the system, and helped to identify areas for improvement and additional features that could enhance the system's usability and effectiveness. The feedback received from the participants was taken into consideration when

implementing the system, and many of the suggestions were implemented in the system after the trials. The trials also helped to validate the effectiveness of the implemented features, and to assess the usability of the system in real-world scenarios.

The results of the simulation models are visualized in head-worn AR for firefighters. For testing with Finnish stakeholders, the developed AR system was subjected to a set of test cases [21] and integrated into a realistic test scenario [22]. The tests with the AR system were carried out at an Emergency Service Academy in Finland as part of an annual stakeholder meeting in Kuopio. In an exhibition-like setup, 15 Finnish firefighters at different command levels with varied levels of experience, tested the AR system. Both quantitative and qualitative feedback was collected during the evaluation. The stakeholders were each given an individual introduction to AR and the developed system. Participants who had no previous experience with AR were offered individual guidance and were guided through the system functionalities before executing the actual test cases. As part of the testing, features such as the visualization of fire, interactive maps, POIs and the visualization of team members were used by stakeholders. In order to obtain quantitative feedback, firefighters were provided with questions in a questionnaire and asked to rate the questions using a 5-point Likert scale based on the System Usability Scale [14]. The users rated the visualization of relevant POIs (1) with a slightly above-average rating of 3.93 out of 5.00 points. Trust in the system for the visualization of simulation results (2) of a forest fire simulation model with AR was rated with 3.60 out of 5.00 points, illustrating basic technological acceptance. This cautious evaluation is due to the general acceptance of simulation models in the response phase, as it became clear in the qualitative feedback. This means that the users were satisfied with the visualization of the results, but since simulation models are difficult to use for the response phase due to their long computing time, this question was cautiously rated. The support provided by a map for better orientation at the scene (3) was rated particularly positively, achieving the best result with 4.20 out of 5.00 points. Qualitative feedback included comments such as "[. . .] useful for building fires if the relevant information can be called up directly via the AR glasses" or "The usability was good!". The intuitiveness of the system and the fast learning curve that the user experiences when using the system for the first time were also emphasized. Requests for additional actions included AR visualization training, obtaining official approvals, conducting further practical tests and provision of detailed information on use and reliability. Overall, the results showed user acceptance of the visualization functions. Firefighters provided valuable feedback on further improvements to the system.

5.5.1 Weather Emergency Use Case: Plans for the next 18 months

For the next 18 months of CREX DATA, these are the technical plans for AR system development for the Weather Emergency Use Case:

- Incorporate data and optimize functionality based on the evaluation testing of AR prototype for flood management, while in Dortmund and Insbruck, in June 2024
- Implement additional functionality for gaze-based interaction
- Experiment with real-time streaming of data, connecting to the forecasting architecture
- Implement real-time physiological data monitoring of rescuers while on the field.

6 Conceptual Modeling for Explainable AI and Uncertainty Communication

Summarizing the content of D5.1, this section starts from discussing the role of conceptual models (CMs) in CREXDATA for enhancing explainability and uncertainty communication in AI systems. In our initial formalization, conceptual models (CMs) serve two main purposes: (1) providing explanations for models and their predictions using domain-specific concepts at an appropriate level of abstraction, and (2) defining the types and semantics of possible uncertainties, as well as effective ways to communicate them to analysts.

A CM represents the concepts and relationships of a given domain, i.e., its semantics. Conceptual models are useful because data-driven ML models typically lack semantic richness. Linking ML model components, outputs, or explanations to a CM can make them more meaningful and interpretable for users.

To achieve this goal, two components are needed:

1. **A suitable conceptual model** containing the concepts and relationships relevant to a given ML model. CMs are often represented as entity–relationship diagrams.
2. **A semantic mapper**, i.e., a structure that links CM elements to the ML model. For instance, features must be matched to conceptual attributes, and decision rules must be aligned with domain relationships.

A simple example combining a CM with an ML model is provided in [37] and illustrated in Figure 86. The authors introduce the Model Embedding Method (MEM), which maps concept attributes in the CM to dataset features. Feature contributions, computed via SHAP values, are aggregated into concept-level contributions using the associations defined in the CM. These concept contributions form directed local scores showing how concepts influence each other.

Although such formalized diagrams are precise, they may not be ideal for domain experts or end users, who often find them difficult to interpret. In many cases, natural-language descriptions and simple schematic representations are more effective.

Generative AI (e.g., LLMs) may help translate formal CMs into human-readable descriptions. However, the domain CM itself still requires expert or literature-derived definitions. Therefore, rather than a fully formal CM, a set of human-understandable definitions of domain concepts and relationships—paired with a mapping to data features—is sufficient.

Acquiring concept definitions is a known challenge, often referred to as the *knowledge acquisition bottleneck* [16]. The interactive workflow shown in Figure 22 can help mitigate this challenge. In this workflow, a domain expert creates labelled examples for training and testing, organized into groups representing distinct patterns. Beyond labels, the expert can provide explanatory annotations for patterns and features. These annotations naturally form a human-readable conceptual model that can support explainability and uncertainty communication.

Figure 87 illustrates a CM fragment for a model recognizing vessel movement behaviours. It includes:

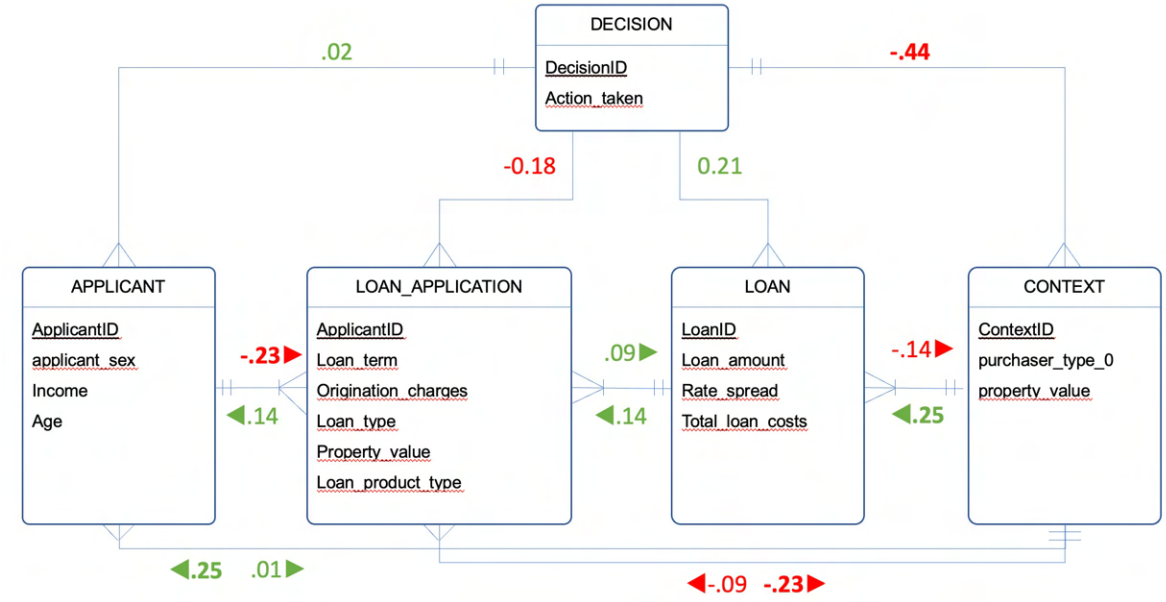


Figure 86: An example of using a conceptual model to represent feature contributions to a model decision. Source: [37].

- human-oriented definitions of behaviour types,
- examples and characteristic descriptions,
- relevant attributes used to distinguish behaviours,
- data-derived features used by the ML model.

Attributes connect pattern characteristics to model features, forming the semantic mapping between conceptual and data levels.

A CM enables translation of feature-based explanations into domain-language statements. For example, instead of:

```
min_speed > 0.1 and Q3_speed < 7.2,
```

an explanation may state that the vessel exhibits *"slow and steady speed"*, consistent with trawling.

Using Conceptual Models to Communicate Uncertainty Domain CMs also support uncertainty communication. For probabilistic predictions (e.g., "trawling with 60% probability"), an explanatory module can analyze alternative high-scoring classes, identify contributing features, map them to conceptual characteristics, and generate human-readable uncertainty statements such as: "It may be directed movement due to relatively high speed" or "It may be port-related due to proximity to a port."

Further uses of a CM for uncertainty communication include:

- clarifying the scope, assumptions, and intended conditions of use of an ML model;

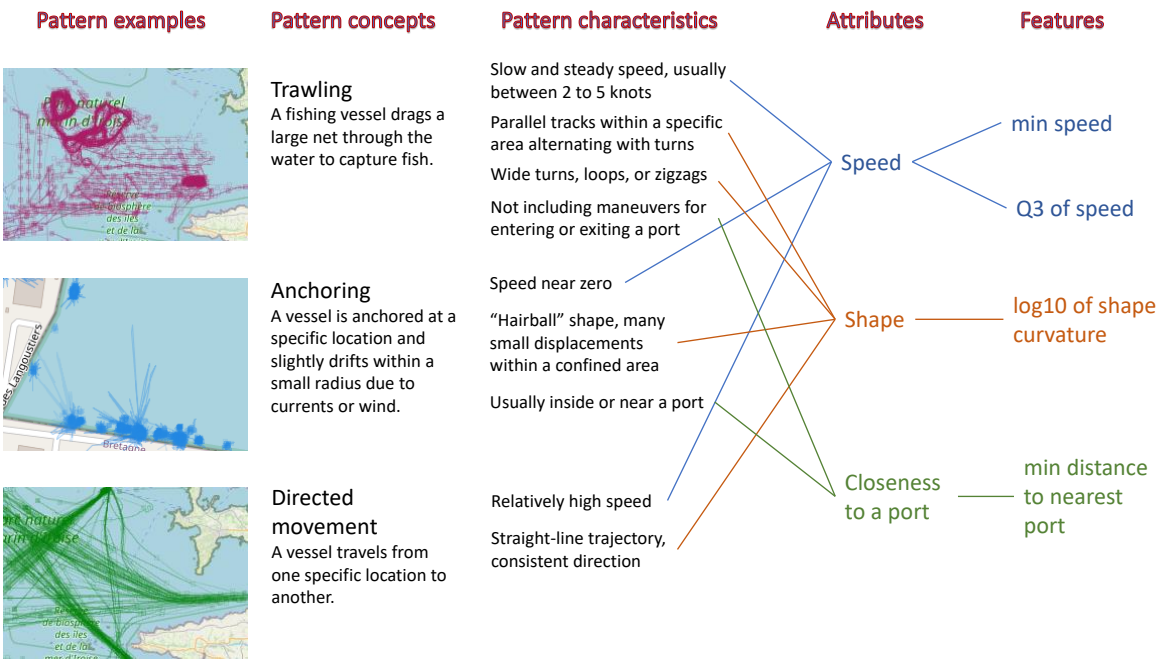


Figure 87: An illustrative fragment of a conceptual model for recognizing vessel movement patterns.

- highlighting gaps, biases, or distribution characteristics of training data;
- identifying well- and poorly-covered regions of the feature space, where extrapolation may occur;
- describing external factors (e.g., environmental or economic conditions) that influence prediction reliability.

6.1 Incorporating Domain Knowledge into XAI: Findings and Conceptual Implications

Our workpackage aimed to enhance explainable AI (XAI) methods by embedding domain expert knowledge into the explanation process. Our work in Task 5.1 focused on instance-based XAI techniques, which generate local explanations by constructing synthetic instances in the neighbourhood of a target instance and training a surrogate decision tree that approximates the behaviour of the black-box model locally. Then rule-based explanations are derived from this local surrogate.

In principle, such rule-based local explanations should capture the essential logic behind the model's prediction. However, in our application domains - classification of vessel movement patterns and prediction of future COVID-19 incidence levels - the rules extracted by LORE were often not meaningful or logically sound and did not provide causal insights. Even when the extracted rules were technically correct in describing the model's decision boundary, they failed to communicate information that domain experts could recognise as valid reasoning.

This observation led us to hypothesise that explanation quality might improve if synthetic

neighbours were generated in a domain-aware manner. We therefore introduced domain constraints into the instance generation step, ensuring that locally sampled data adhered to feasible temporal dynamics, behavioural patterns, or epidemiological relationships. While this domain-driven sampling indeed produced **more realistic synthetic instances**, it did **not** lead to better or more meaningful explanatory rules. The underlying limitation was not the data itself, but the explanatory algorithms: instance-based surrogate models, even when built on high-quality local samples, do not automatically capture the causal or semantic structure of the domain.

These findings point to a broader conclusion: classical instance-based explanation methods remain fundamentally constrained by **their purely data-driven logic**. They can, possibly, perform adequately in domains with low feature complexity or static decision boundaries (e.g., credit approval), yet they struggle in domains that involve temporal structure, interdependent variables, or semantically rich concepts. Our negative result is therefore informative: **embedding domain knowledge into synthetic data generation is necessary but insufficient for achieving human-aligned explanations**. To bridge the gap, explanation methods must evolve to incorporate domain structure, semantics, and causal relationships more explicitly.

At the same time, our work reinforces the importance of domain knowledge in XAI. Explanations that ignore expert knowledge will likely remain superficial or misleading. However, acquiring expert knowledge in explicit, machine-readable form is a long-standing bottleneck, dating back to the challenges of building early knowledge-based systems. This is where our experiments with large language models (LLMs) provide an important insight. We found that LLMs can act as **intelligent intermediaries**, supporting experts in externalising their tacit knowledge and translating it into conceptual structures, constraints, and domain relationships suitable for downstream computational use.

Our investigations revealed that LLMs are capable of **acquiring, representing, and operationalising domain knowledge** through iterative interaction with human experts. This capability positions LLMs not only as tools for generating more realistic synthetic data, but also as potential agents for producing or refining explanations that align with expert reasoning. A promising research direction is the integration of LLM capabilities with algorithmic XAI outputs. Rather than replacing existing methods such as LORE or SHAP, LLMs can be used to interpret, validate, and enrich their outputs by grounding them in previously elicited domain knowledge. For example, an LLM could take a rule or feature attribution produced by an XAI algorithm and augment it with causal context, conceptual relationships, or domain constraints learned earlier. Such hybrid reasoning, i.e., combining statistical signals from the model with structured domain semantics, has the potential to convert low-level, pattern-based outputs into explanations that are coherent, causally informed, and cognitively aligned with expert understanding.

In summary, LLMs offer a promising path toward bridging the gap between statistical patterns discovered by machine learning models and the structured, semantically rich knowledge used by human decision-makers. Their capacity to externalise expert knowledge and integrate it with algorithmic XAI outputs opens new possibilities for conceptual modelling in XAI.

6.1.1 General Conceptual Model for XAI Integrating Domain Knowledge and Causal Reasoning

The conceptual model for explainable AI (XAI) in our approach is meant to support human-understandable, causally coherent explanations of predictive model behaviour. It combines outputs from data-driven XAI methods with structured domain knowledge and reasoning capabilities provided by LLMs.

Core Entities

Predictive Model: The black-box or complex model whose predictions are to be explained. This model may operate on temporal, spatial, or multivariate data.

Instance: A specific data point or episode for which a prediction is generated. Instances serve as the target for explanation.

Prediction: The output of the predictive model for a given instance, e.g., class label, risk score, or next-step prediction.

Data-driven Explanation: Local explanations produced by methods such as LORE, SHAP, or PDP. These include feature attributions, decision rules, or surrogate models that capture the model's behaviour in the neighbourhood of an instance.

Domain Knowledge: Conceptual, causal, or structural knowledge about the application domain, expressed in formal or semi-formal representations (e.g., rules, hierarchies, causal graphs). This knowledge reflects relationships between variables, temporal dependencies, and constraints that are meaningful to experts.

Causal Reasoning Layer: Mechanisms for integrating domain knowledge with data-driven explanations to assess, validate, or enrich the explanation. This layer can identify whether features or rules reflect plausible causal relations, temporal effects, or interdependencies, and can adjust explanations accordingly.

Relations and Processes

- **Instance** $\rightarrow$ **Prediction:** The model maps an instance to a predicted output.
- **Instance** $\rightarrow$ **Data-driven Explanation:** XAI methods extract local rules, feature importances, or surrogate models for a specific instance.
- **Data-driven Explanation** $\rightarrow$ **Domain Knowledge Integration:** Explanations are augmented or validated using domain knowledge to ensure semantic coherence and causal plausibility.
- **Domain Knowledge** $\rightarrow$ **Explanation Refinement:** LLMs or reasoning engines leverage prior knowledge to restructure explanations, resolve inconsistencies, or highlight causal chains.
- **Explanation** $\rightarrow$ **Human Interpretation:** The final output combines model-derived insights and domain reasoning into a cognitively aligned, interpretable justification.

Supporting Components

- **Temporal Context Encoding:** Explicit representation of past events, lags, and sequential dependencies relevant for the prediction.
- **Causal Constraints:** Rules or priors specifying which variables can influence others and over what time horizons.
- **Uncertainty Representation:** Indicators of prediction confidence, explanation reliability, and potential conflicts between model logic and domain knowledge.
- **LLM-enabled Reasoning:** Iterative interaction with a language model to externalize expert knowledge, generate machine-readable structures, and refine explanations for human alignment.

Principles The conceptual model ensures that explanations are:

- **Causally coherent:** Capturing plausible cause-effect relationships between features and outcomes.
- **Domain-aligned:** Respecting expert knowledge and structural constraints of the application domain.
- **Human-interpretable:** Providing justifications that are understandable and actionable by domain experts.
- **Feedback-aware:** Able to incorporate observations from users to improve future explanations.

This model underlies the design of hybrid XAI systems where LLMs act as intelligent intermediaries between purely data-driven explanations and structured domain knowledge, resulting in explanations that are not only accurate with respect to the model but also meaningful and causally aligned with expert understanding.

In the following, we illustrate how LLMs can support the construction of conceptual models tailored to specific application domains. We present examples of conceptual models developed for our two use cases, vessel movement classification and COVID-19 incidence prediction, demonstrating how LLM-assisted modelling can capture domain entities, relationships, temporal dependencies, and causal assumptions that are essential for meaningful and human-aligned explanations.

6.1.2 Conceptual Model of Vessel Movement Patterns

The application domain concerns the analysis of fishing vessel trajectories with the goal of identifying characteristic movement behaviours. Vessel trajectories consist of sequential geospatial positions recorded over time, each associated with instantaneous movement attributes such as speed and heading. These trajectories exhibit recognisable patterns that correspond to distinct operational behaviours of fishing vessels. The conceptual model underlying this application is schematically represented in Fig. 88.

Behavioural classes. Domain experts distinguish four principal classes of vessel movement behaviour:

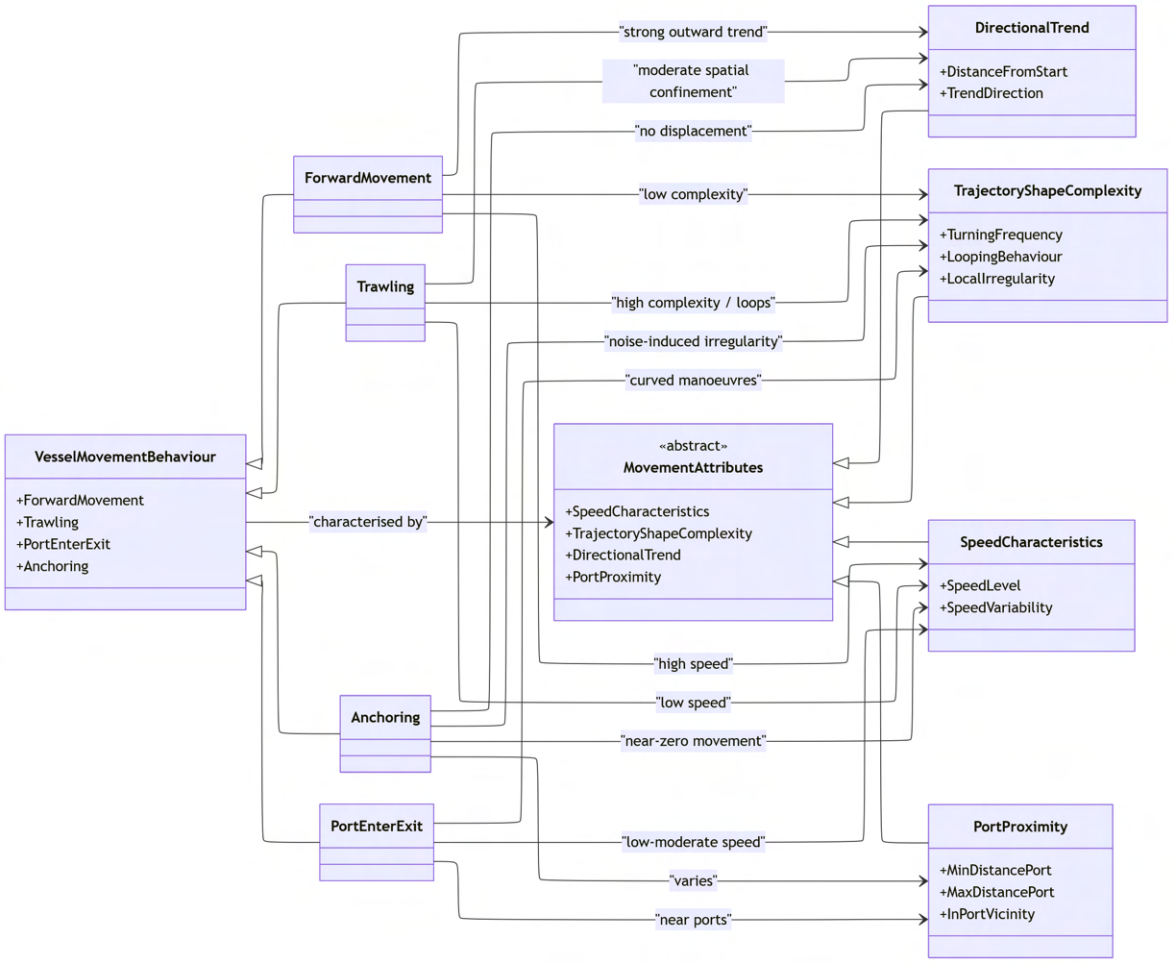


Figure 88: Representation of the concept model for the maritime use case created with the help of an LLM (GPT 5)

- **Forward movement:** Regular displacement at moderate or high speed along a relatively straight path. This behaviour reflects transit between areas and is characterised by stable vessel motion without frequent directional changes.
- **Trawling:** Fishing activity performed at low speed and accompanied by repeated turning manoeuvres or looping trajectories. Trawling commonly manifests as wide arcs, tight loops, or alternating straight movement and sharp turns, reflecting the operation of trawling equipment.
- **Port enter/exit:** Movements occurring in the vicinity of ports, where vessels reduce speed and perform curved or manoeuvring trajectories when approaching or leaving port areas. These movements share some characteristics with trawling (low speed and curvature) but occur spatially close to known port locations.
- **Anchoring:** Situations in which the vessel remains effectively stationary, with only small fluctuations in recorded positions. These fluctuations may appear as erratic micro-movements due to positioning noise, giving rise to highly irregular local path shapes without actual displacement.

Domain-relevant movement attributes. The behavioural classes are characterised by interpretable movement attributes grounded in maritime domain knowledge:

- **Speed level and variability** distinguish active movement (forward or trawling) from anchoring, with trawling specifically associated with sustained low speed.
- **Trajectory shape complexity** captures directional variability, turning frequency, and looping behaviour. Trawling patterns exhibit higher shape complexity than forward movement, whereas anchoring results in erratic but locally bounded shape fluctuation.
- **Distance from the starting position** reflects the directional persistence of movement. Forward movement typically shows a strong outward trend, whereas trawling and anchoring remain more spatially confined.
- **Proximity to ports** is essential for distinguishing operational fishing activity from port manoeuvres, as port entry and exit are characterised by low-speed curved movements occurring near port locations.

Conceptual structure. These domain concepts jointly form a conceptual model in which vessel movement behaviour is determined by the interplay of:

1. *Speed characteristics* (absolute level and distribution),
2. *Global path geometry* (overall curvature, directional trend),
3. *Local path irregularity* (short-scale deviations from a trend),
4. *Contextual spatial factors* (distance to ports).

The classes of interest emerge as distinct combinations of these domain attributes. Forward movement is characterised by high speed and low shape complexity; trawling by low speed and elevated shape complexity; port enter/exit by low speed and curvature combined with short distance to ports; and anchoring by near-zero displacement accompanied by noisy

fluctuations. These behavioural signatures constitute the domain knowledge foundation for interpreting model outputs in the vessel-movement application.

6.1.3 Conceptual Model of Uncertainties in the Vessel Movement Application

In the vessel-movement domain, uncertainty arises from how movement is observed, represented, and interpreted. The conceptual model below (schematically represented in Fig. 89) identifies the main *sources*, *implications*, and *mitigation strategies* relevant to classifying vessel behaviours such as trawling, forward movement, anchoring, and port enter/exit.

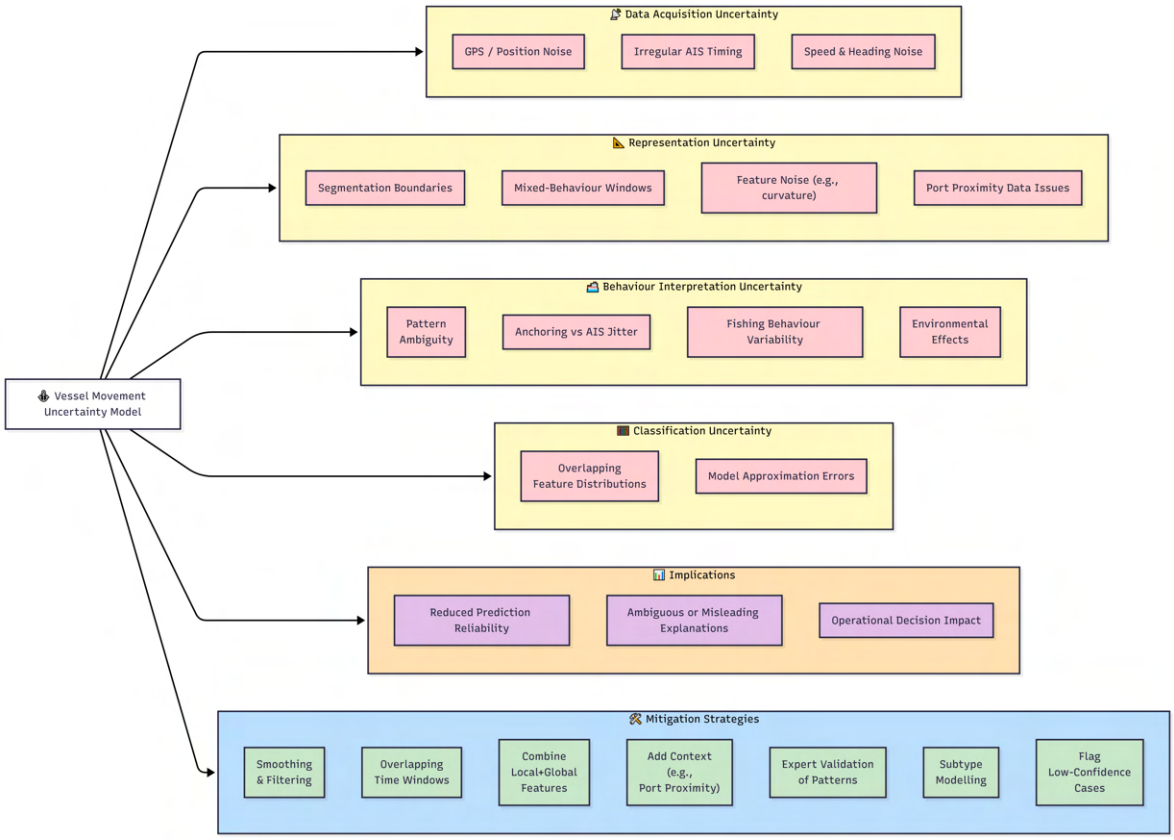


Figure 89: Representation of the uncertainty concepts for the maritime use case created with the help of an LLM (GPT 5)

Sources of Uncertainty

- Data Acquisition Uncertainty
- Representation Uncertainty
- Behavioural Interpretation Uncertainty
- Classification Uncertainty

Data Acquisition Uncertainty AIS (Automatic Identification System) data contain several forms of measurement noise:

- **Positional noise:** GPS inaccuracies cause jitter in recorded coordinates.
- **Irregular timing:** AIS broadcasts occur at uneven intervals, affecting speed and trajectory estimates.
- **Speed and heading noise:** Short-term fluctuations or circular-angle artefacts complicate behaviour interpretation.

Representation Uncertainty

Segmentation into Time Windows

- **Boundary uncertainty:** Behaviour may be split across window borders.
- **Mixed-behaviour windows:** A window can contain more than one movement type.

Feature Construction Uncertainty High-level features introduce abstraction-related uncertainty:

- **Curvature estimation:** Sensitive to positional noise.
- **Proxy features:** Simplified indicators of direction or path geometry may misrepresent true behaviour.
- **Port proximity:** Depends on the completeness of external datasets.

Behavioural Interpretation Uncertainty

Pattern Ambiguity

- **Trawling - port manoeuvre overlap:** Both involve low speed and curved paths.
- **Anchoring vs. noise:** AIS jitter may mimic slow movement.

Behavioural Heterogeneity

- **Variable fishing styles,** from wide curves to tight loops.
- **Environmental effects** (weather, sea state) influence observed motion.

Classification Uncertainty

- **Overlapping feature distributions:** Blurred class boundaries.
- **Model approximation:** Learning algorithms may not capture all behaviour nuances.

Implications These uncertainties affect:

- the reliability of predicted movement classes,
- the interpretability of model explanations,
- downstream monitoring or decision-making that relies on behaviour classification.

Mitigation Strategies

Data-Level

- Smoothing of noisy signals,
- use of overlapping windows to reduce segmentation artefacts.

Feature-Level

- Combining global and local trajectory indicators,
- incorporating contextual features such as port proximity.

Behaviour-Level

- Consulting domain experts to validate typical and atypical patterns,
- modelling multiple subtypes of fishing behaviour.

Model-Level

- Comparing predictions with domain rules,
- highlighting low-confidence or ambiguous classifications.

Role in Explainability Explicit modelling of uncertainty helps distinguish genuine behavioural signals from artefacts, preventing XAI methods from attributing importance to noise or representation errors. This supports more trustworthy interpretations of model behaviour and guides integration of domain expertise into the explanation process.

6.1.4 Conceptual model for explainability in pandemic use case

This conceptual model (Fig. 90) defines the entities, relationships, and temporal dependencies necessary for generating human-aligned, causally meaningful explanations of predictive models for COVID-19 incidence.

Problem statement

A multi-region, weekly time-series problem: predict near-term COVID-19 incidence using recent disease and mobility patterns, excluding early outbreak weeks with atypical dynamics. The description of the data and model is provided in Section 3.1.6.

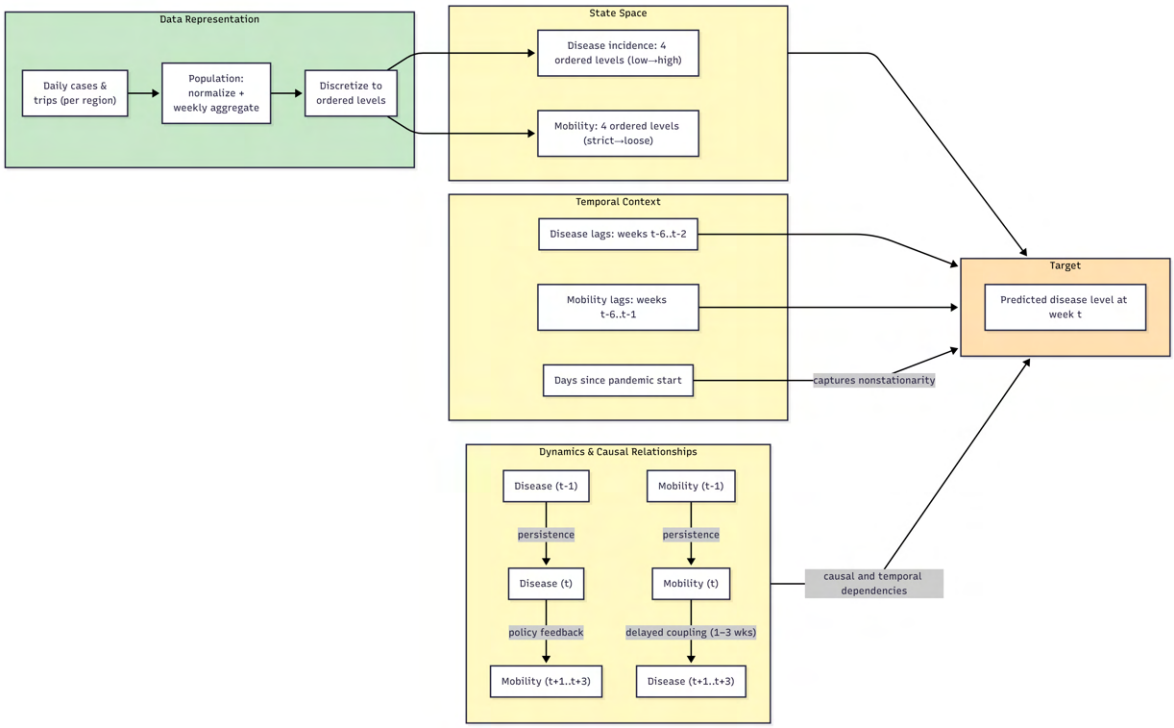


Figure 90: Schematic representation of the conceptual model for the pandemic use case generated with the help of an LLM (GPT 5).

State Space

- Disease incidence: four ordered levels (low to high).
- Mobility: four ordered levels (strict to loose).
- Time context: days since pandemic start (captures longer-term drift).

Data Representation

- Daily cases and trips are population-normalized, aggregated to weeks, then discretized into ordered levels.
- Each training instance is a region-week with a target disease level and a compact temporal context.

Temporal Context

- Inputs: disease levels from 6 to 2 weeks prior; mobility levels from 6 to 1 weeks prior; days since start.
- Target: disease level for the current week.

Observed Features:

The input variables used by the predictive model, including:

- Disease levels in preceding weeks (-6 to -2)
- Mobility levels in preceding weeks (-6 to -1)
- Days since pandemic onset

Qualitative Dynamics

- Persistence: both disease and mobility tend to remain at their current levels week to week.
- Policy feedback: higher disease leads to tighter mobility restrictions shortly after.
- Delayed coupling: tighter mobility reduces disease after a delay (about 1 to 3 weeks); looser mobility raises disease with similar delay.
- Nonstationarity: gradual changes over time are represented by the time marker.

Causal Relationships

- **Disease** $\rightarrow$ **Mobility:** Rising incidence generally triggers reductions in mobility due to interventions.
- **Mobility** $\rightarrow$ **Disease:** Changes in mobility influence subsequent disease levels with a time lag of 1–3 weeks.
- **Temporal Feedback Loops:** The disease–mobility relationship is bidirectional and delayed.

6.1.5 Conceptual model of uncertainties in COVID-19–mobility dynamics

Basic pathway

- When people move and mix more, they have more contacts.
- More contacts can raise transmission.
- New infections occur but are seen later as reported cases, hospitalizations, or deaths because of incubation, testing, and reporting delays.

Main sources of uncertainty

- Random chance: outbreaks can surge or fade due to superspreading and luck.
- Limited knowledge:
  - How strongly different types of mobility (work, retail, transit) change contacts.
  - How long the delays are from infection to reported outcomes, which vary by place and time.
  - Nonlinear effects (thresholds, saturation) where small changes may have little effect until a tipping point.
  - Missing factors like masking, ventilation, and crowding that also shape transmission.

- Data and measurement:
  - Mobility data may not represent all groups equally (device ownership, trip purpose).
  - Reported cases depend on testing access, criteria, and backlogs; definitions can change.
  - The size of the immune or at-risk population changes with vaccination and prior infection.
- Changes over time (nonstationarity):
  - New variants alter transmissibility and immune escape.
  - People adapt behavior; policy fatigue reduces compliance.
  - Seasonality and environment affect how easily the virus spreads.
- Differences across places and contexts (heterogeneity):
  - Urban vs. rural settings, socioeconomic factors, and transport systems.
  - Indoor vs. outdoor activities; essential vs. discretionary trips.
  - Contact networks in households, workplaces, and communities.

Feedbacks and confounders

- Rising cases lead to behavior change and policies that reduce mobility; improving trends relax them.
- Other time-varying influences (vaccination rollout, healthcare pressure, testing intensity, weather) affect both mobility and outcomes.

What we actually observe Infections are not seen directly. Indicators are delayed, incomplete, and noisy, and their quality changes over time and place.

Implications for analysis.

- Model mobility-to-contact links with uncertainty bounds; allow time-varying parameters.
- Represent delays as distributions, not fixed lags; test robustness across lag ranges.
- Account for feedback using concurrent policy/risk variables or designs that reduce reverse causality.
- Stratify by activity type and region to capture heterogeneity; monitor for regime shifts (variants, seasons).
- Allow relationships and delays to change over time.
- Test results across lag ranges and parameter choices.
- Treat findings as associations unless supported by causal evidence.

A schematic representation of the uncertainty conceptual model for the pandemic use case is shown in Fig. 91.

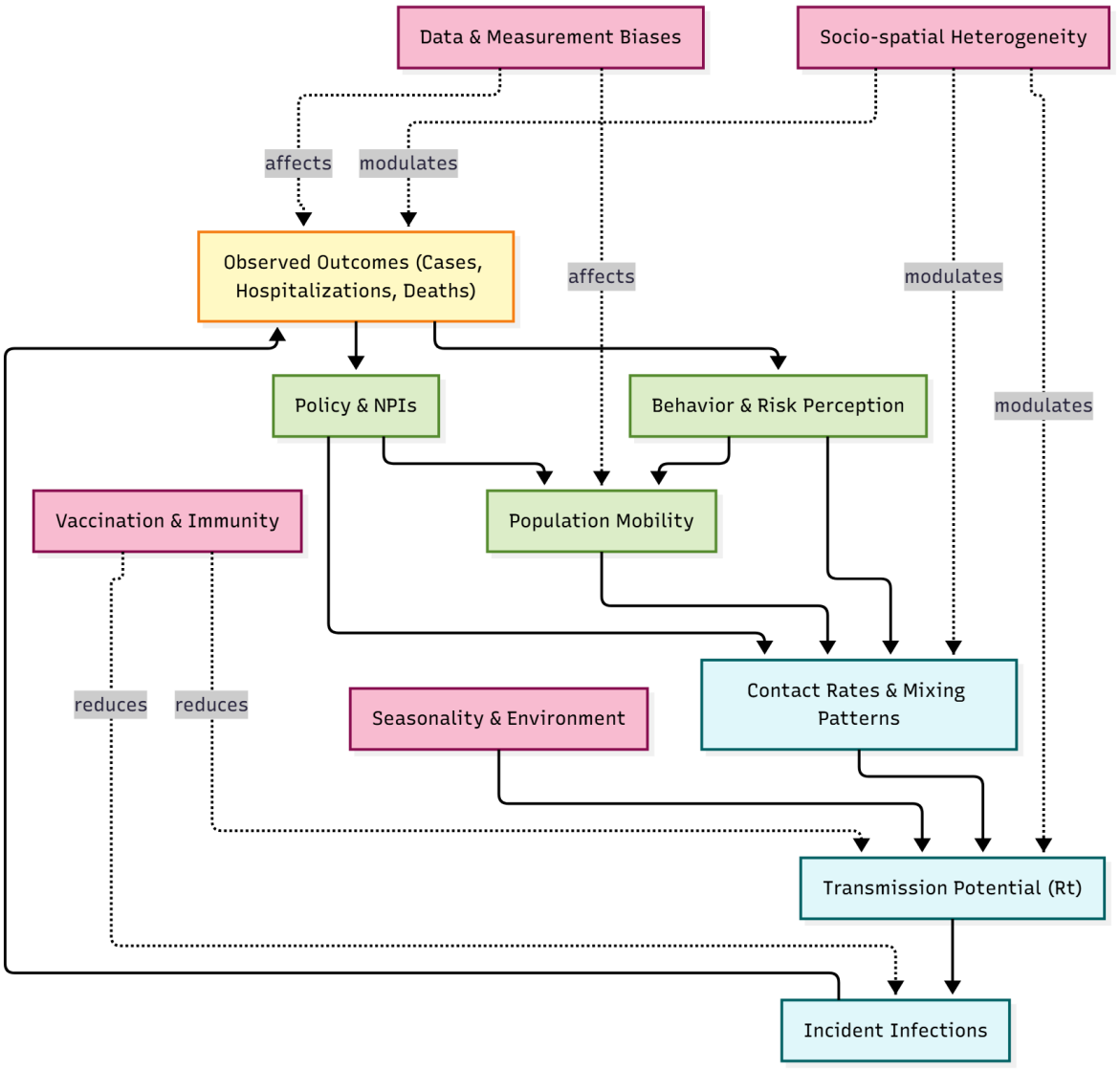


Figure 91: Diagrammatic representation of the conceptual model of the uncertainties in the pandemic use case.

6.2 Conceptual model for visualizing uncertainty in the Emergency Case

In alignment with both characteristics of CREXDATA technologies and the domain of weather-induced emergency management, the conceptual considerations regarding uncertainty are transferred to the demonstrator system of the Emergency Case (see deliverable D2.3, including findings of evaluations). The objective was to cover the specific user requirements in this use case:

- create awareness for uncertainty
- make users aware of different kinds of uncertainty
- label information pieces, including results from XAI, with uncertainty indicators
- support users in reasoning and decision making
- open a space of possible reactions for the user
- involve domain-specific experience
- enable uncertainty labeling in real time
- provide interactive visualization
- specify thresholds
- enable interactive filtering of data
- include domain-specific concepts and abstraction levels

For the Emergency Case, the **Uncertainty Management Metamodel** (UMM) was introduced which makes it possible to translate core aspects of uncertainty into a tangible form and act accordingly. This metamodel consists of a classification model for uncertainty, a set of recommendations for action for different types of uncertainty, and approaches for visualizing uncertainty. It is based on uncertainty-related literature and, on the other hand, domain-specific visualization concepts like tactical symbols. Fire officers mark relevant items like fire engines, trapped people or danger zones by symbols on a situational map. These symbols are not standard in its narrow sense, but at least informally agreed within a country and similar across European countries. This includes to add a question mark besides such a symbol to state that this piece of information is only an assumption – which in fact means a semantic labelling that this information is “uncertain”. Based on different levels of understanding, core characteristics of uncertainty are identified, which in turn are used to define requirements for the metamodel.

In an illustrative example, a firefighter must decide where to set up flood barriers (see Figure 92, cf. technologies for parameter optimization in D4.2). Available information includes, for example, weather data, publicly posted photos from citizens affected by the flooding, flood maps from 20 years ago, and AI-based flood simulation results (see use of Floodwaive surrogate model in D2.3). All of these factors are subject to uncertainty. The weather data may be outdated or distorted by errors in the sensor technology. The posted photos may also be outdated or even manipulated or generated by AI. The flood maps are also outdated. New

roads or other changes in the cityscape are not included. The AI-based flood simulations cannot be 100% verified, as the AI's procedures themselves are not completely transparent.

Figure 92: Decisions under uncertainty in the fire department.

To support such use cases, the Emergency Case created and applied a conceptual metamodel. Additionally, the uncertainty concepts are assigned a) relevant forms of visualization and b) recommendations for action, which can be presented to a user as additional/meta information about the information artifact, thus enabling explicit/targeted handling of these uncertainties. The conceptual model was entitled MORCE, including the factors:

- **M**easurability of Uncertainty: quantifiable, not quantifiable
- **O**bservability of Uncertainty: observable, not observable
- **R**eapperability of Uncertainty: singular issue, potentially repetitive issue
- **C**ategory of Uncertainty: output, input, model, time, AI, constraints, human, system
- **E**ffect of Uncertainty: technical integrity, human safety, effort, perception of risk

Results from data processing are characterized by attributes across all of these concepts. Some can be assigned automatically (like temporal indicators that are based on a comparison of data creation and data usage, cf. research in Complex Event Forecasting incl. the temporal dimension) or can be assigned manually, as is currently done using tactical symbols. They can also be assisted by Visual Analytics techniques and XAI methods, like described in Section 6.1. Uncertainty is represented by each and every combination of the five MORCE factors, resulting in 256 possible combinations. For a very generic view close to the tactical symbol representation, these factor combinations are categorized in four categories:

- Low uncertainty (all factors are on low level)
- One question mark: Medium uncertainty (some factors are on medium level, like timeliness is not guaranteed)
- Two question marks: High uncertainty (some factors are on medium to high uncertainty, that need to be considered explicitly in the next situational report)
- Three question marks: Very high uncertainty (that, from an end user point of view, requires immediate action – which is an established category of information in command rooms)

Colors to represent the uncertainty were tested in a workshop with firefighters at different command levels from FDDO, but were not accepted, as they are intensely used in the domain with very specific semantics. For instance, red color is clearly associated with "danger",

which is not necessarily true for uncertain information. However, all end users agreed on the relevance of categorization for quickly recognizing and evaluating uncertainty information in operations. The test was done using a mockup layer on top of the integrated ARGOS platform. Test items were X postings (see Figure 93) and fatigue assessments by XAI based on biometric data.

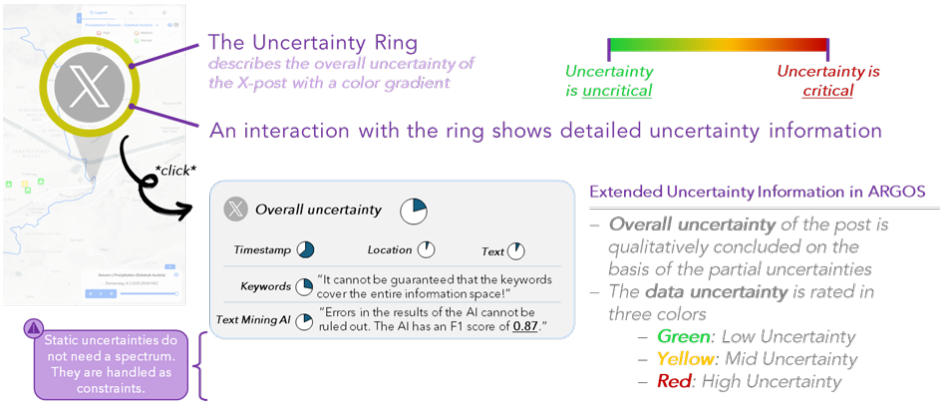


Figure 93: Mockup tested with end-users, utilizing color schemes and "uncertainty rings" around (tactical) symbols (example: classified postings recognized by the text mining technology).

The implementation is done within the Demonstrator System (see deliverable D2.3), i.e., both types of visualization. Figure 94 presents an overview of how uncertainty metadata is accessed via the map view in the ARGOS geo-information system. The three question marks express very high uncertainty, and a mouse-over menu presents an MORCE-based overall rating for the different types of information that were used as input for information of interest. Figure 95 adds a corresponding concept for visualization in the Augmented Reality application.

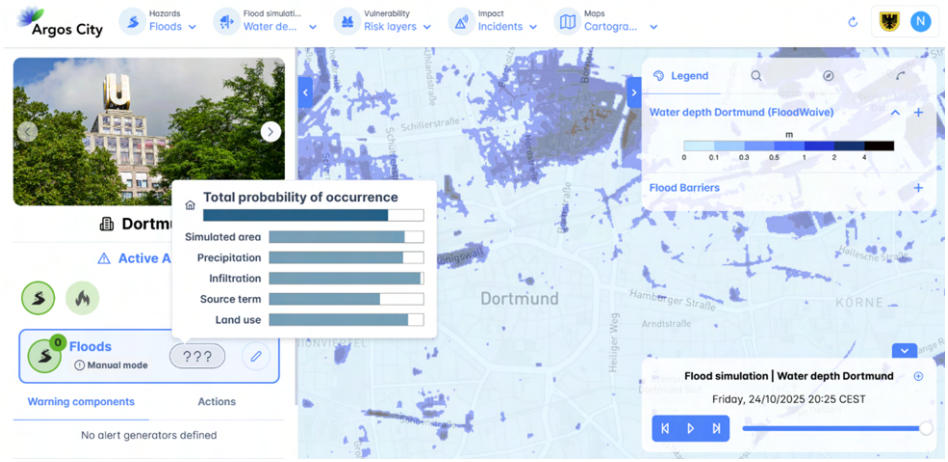


Figure 94: Visualization of uncertainty in the ARGOS geo-information system.

a) 

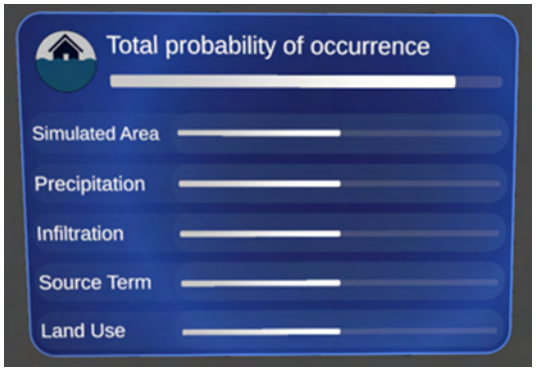
b) 

Figure 95: Decisions under uncertainty in the fire department: a) highest level of abstraction, b) visualization regarding sources of uncertainty comparable to ARGOS view.

6.3 Lessons Learned and Open Challenges

Across the maritime and pandemic use cases, our work on conceptual modelling demonstrated that explicit domain models are essential for connecting data-driven predictions with the logic of real-world processes. Several lessons emerged.

First, conceptual models provided the structural backbone needed to interpret the behaviour of prediction models. In the maritime case, categorising uncertainties related to sensing, environment, and vessel behaviour enables a clearer understanding of when and why model outputs become unreliable. In the COVID–19 use case, modelling the causal interplay between incidence levels, mobility patterns, policy reactions, and temporal delays allows explanations to reflect epidemiological and behavioural realities rather than mere statistical associations. These models can help translate features into semantically meaningful concepts and make it possible to interpret predictions in terms of underlying mechanisms.

Second, conceptual clarity is necessary but insufficient for high-quality XAI. Even when domain constraints were incorporated into synthetic neighbourhood generation or rule-based local explanations, purely data-driven XAI methods failed to capture causal dependencies, latent feedback loops, or temporal asymmetries that domain experts consider fundamental. This indicates that future XAI must integrate conceptual knowledge and causal reasoning more deeply, rather than rely exclusively on post-hoc pattern extraction from feature values.

Third, large language models showed clear potential as mediators between conceptual models and prediction-based explanations. In our experiments, LLMs were able to assist in formalising expert knowledge. We expect that they can also align explanation templates with domain logic and detect inconsistencies or missing causal links. This suggests a future in which LLMs serve as reasoning engines that contextualise predictions, verify the plausibility of explanations, and support analysts in constructing uncertainty-aware narratives. However, achieving this reliably requires mechanisms for ensuring factual grounding, reproducibility, and traceability of LLM-generated reasoning.

Open Challenges and Research Directions.

Despite these achievements, several challenges remain open:

- **Operationalising conceptual models in real workflows.** Current prediction pipelines do not yet incorporate conceptual models as first-class components. Further work is needed to integrate these models into data preprocessing, model training, evaluation, and deployment so that predictions can be automatically contextualised with domain semantics and known uncertainty sources.
- **LLM-guided causal reasoning over predictions.** While LLMs can articulate causal structures, reliably applying them to interpret model outputs requires controlled reasoning frameworks. Research is needed on hybrid systems where LLMs operate under explicit constraints derived from domain conceptual models, ensuring that explanations remain aligned with validated domain logic.
- **Uncertainty-aware explanation generation.** Explanations should not only describe why a model produced a certain prediction but also indicate when the prediction may be unreliable, which uncertainty category is responsible, and how this affects possible decisions. Formal methods for linking uncertainty concepts to explanation components are needed.
- **Decision support grounded in causal and conceptual reasoning.** End-users require recommendations or scenario analyses that incorporate both model predictions and known uncertainty types. Future research must focus on workflows where prediction outputs, conceptual models, and LLM-generated causal reasoning are jointly used to support choices such as anomaly investigation (maritime) or policy timing evaluation (pandemics).
- **Validation of LLM-generated explanations.** A major challenge is assessing the accuracy, stability, and trustworthiness of explanations produced or enriched by LLMs. Techniques for verifying reasoning chains against conceptual models and observational data are necessary to avoid hallucination or plausible-but-wrong narratives.

Figure 96 illustrates the envisioned workflow that integrates model predictions, conceptual domain knowledge, LLM-based causal reasoning, and explicit uncertainty quantification into a unified explanation and decision-support process. The workflow illustrates how raw model predictions are contextualised through domain-knowledge structures and enhanced by LLM reasoning, then combined with uncertainty estimates to produce explanations that are both logically grounded and uncertainty-aware. These enriched explanations feed into a decision-support workflow, where scenario analysis and justification generation enable informed decision making. A feedback loop allows experts to refine conceptual models and reasoning processes, ensuring continuous improvement.

Overall, the project showed that conceptual modelling is foundational for human-aligned explainability and uncertainty-aware decision support. However, realising its full potential requires new research on integrating conceptual models, data-driven predictions, and LLM reasoning into coherent, verifiable, and practically deployable analytical workflows.

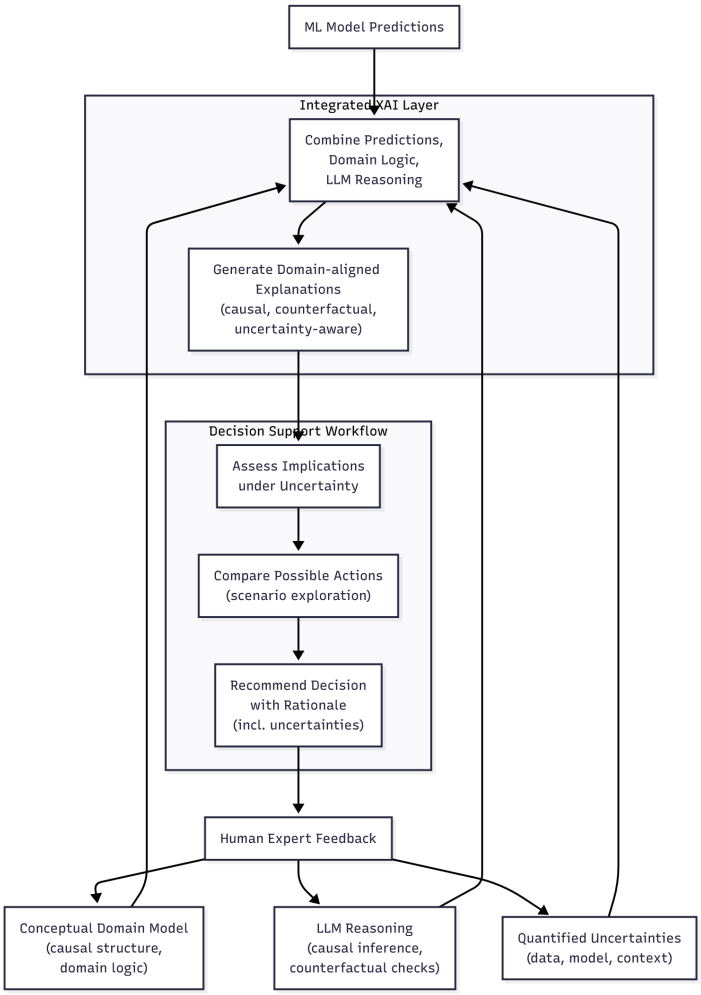


Figure 96: Workflow integrating machine-learning predictions, conceptual domain knowledge, LLM-enabled causal reasoning, and uncertainty quantification into a unified XAI process.

7 Conclusions

This deliverable has presented the research activities carried out in Workpackage 5 during the second period of the CREXDATA project. The main objective of WP5 is to develop methodologies to bridge the gap between complex AI models and end-users, though the combination of Explainable AI, Visual Analytics, and Augmented Reality techniques. At the end of the whole project, we provided a set of tools, libraries, prototypes, and methodologies that strictly integrate these research areas to enhance the interpretability of complex situations, giving support to the decision-maker in augmenting their understanding of the data and models, and to consciously manage the uncertainty that is inherent in situated scenarios.

The results of the WP were strictly integrated with the other WPs of the project:

- We exploited the input and requirements from **WP2** to develop and refine the methods and tools, ensuring an alignment with the needs of the use cases.
- We collaborated with **WP3** to integrate the developed methods and tools into the CREXDATA platform, ensuring seamless interoperability and usability.
- We worked closely with **WP4** to align model training and explanation techniques, to ensure a coherent approach to model interpretability.

8 Acronyms and Abbreviations

- AAE - Adversarial Autoencoder
- ABELE - Adversarial Black-box Explanation Learning Engine
- AI - Artificial Intelligence
- AR - Augmented Reality
- CM - Conceptual Model
- CNN - Convolutional Neural Network
- COVID-19 - Coronavirus Disease 2019
- EU - European Union
- ISIC - International Skin Imaging Collaboration
- ML - Machine Learning
- PCA - Principal Component Analysis
- PGAAE - Progressive Growing Adversarial Autoencoder
- ResNet - Residual Neural Network
- SHAP - SHapley Additive exPlanations
- t-SNE - t-distributed Stochastic Neighbor Embedding
- VA - Visual Analytics
- XAI - Explainable Artificial Intelligence

References

[1] Natalia Andrienko and Gennady Andrienko. Exploring relationships between events in context. In Mennatallah El-Assady and Hans-Jörg Schulz, editors, *EuroVis Workshop on Visual Analytics (EuroVA)*. The Eurographics Association, 2024.

[2] Natalia Andrienko and Gennady Andrienko. Exploring Relationships between Events in Context . In Mennatallah El-Assady and Hans-Jörg Schulz, editors, *EuroVis Workshop on Visual Analytics (EuroVA)*. The Eurographics Association, 2024.

[3] Natalia Andrienko, Gennady Andrienko, Alexander Artikis, Periklis Mantenoglou, and Salvatore Rinzivillo. Human-in-the-loop: Visual analytics for building models recognising behavioural patterns in time series. *IEEE Computer Graphics and Applications*, 2024.

[4] Natalia Andrienko, Gennady Andrienko, Alexander Artikis, Periklis Mantenoglou, and Salvatore Rinzivillo. Human-in-the-loop: Visual analytics for building models recognising behavioural patterns in time series. *IEEE Computer Graphics and Applications*, pages 1–15, 2024.

[5] Natalia Andrienko, Gennady Andrienko, and Salvatore Rinzivillo. Leveraging spatial abstraction in traffic analysis and forecasting with visual analytics. *Information Systems*, 57:172–194, 2016.

[6] Natalia Andrienko, Gennady Andrienko, and Gota Shirato. Episodes and topics in multivariate temporal data. 42(6):e14926, 2023.

[7] Lewis Arnold, Soniya Aryal, Brandon Hong, Mahiethan Nitharsan, Anaya Shah, Waasiq Ahmed, Zakariya Lilani, Wanzi Su, and Davide Piaggio. A systematic literature review of eye-tracking and machine learning methods for improving productivity and reading abilities. *Applied Sciences*, 15(6), 2025.

[8] Valentina Bachurina and Marie Arsalidou. Multiple levels of mental attentional demand modulate peak saccade velocity and blink rate. *Heliyon*, 8(1):e08826, 2022.

[9] Hamada S Badr, Hongru Du, Maximilian Marshall, Ensheng Dong, Marietta M Squire, and Lauren M Gardner. Association between mobility patterns and covid-19 transmission in the usa: a mathematical modelling study. *The Lancet Infectious Diseases*, 20(11):1247–1254, 2020.

[10] T. Bafna and J. P. Hansen. Mental fatigue measurement using eye metrics: A systematic literature review. *Psychophysiology*, 58(6):e13828, June 2021. PMID: 33825234.

[11] Jacques Bertin. *Sémiologie Graphique: Les Diagrammes, les Réseaux, les Cartes*. Mouton, 1967.

[12] Jacques Bertin. *Semiology of Graphics: Diagrams, Networks, Maps*. University of Wisconsin Press, 1983.

[13] Francesco Bodria, Fosca Giannotti, Riccardo Guidotti, Francesca Naretto, Dino Pedreschi, and Salvatore Rinzivillo. Benchmarking and survey of explanation methods for black box models. *Data Min. Knowl. Discov.*, 37(5):1719–1778, 2023.

[14] John Brooke et al. Sus-a quick and dirty usability scale. *Usability evaluation in industry*, 189(194):4–7, 1996.

[15] William S. Cleveland and Robert McGill. Graphical perception: Theory, experimentation, and application to the development of graphical methods. *Journal of the American Statistical Association*, 79(387):531–554, 1984.

[16] J. Cullen and A. Bryman. The knowledge acquisition bottleneck: Time for reassessment? *Expert Systems*, 5(3):216–225, 1988.

[17] Guozhu Dong and Huan Liu. *Feature engineering for machine learning and data analytics*. CRC press, 2018.

[18] Giorgio Ganis and Rogier Andrew Kievit. A new set of three-dimensional shapes for investigating mental rotation processes: Validation data and stimulus set. *Journal of Open Psychology Data*, Mar 2015.

[19] Peter Gatalsky, Natalia Andrienko, and Gennady Andrienko. Interactive analysis of event data using space-time cube. In *Proceedings of IV' 04*, pages 145–152, 2004.

[20] H. Gorin, J. Patel, Q. Qiu, A. Merians, S. Adamovich, and G. Fluet. A review of the use of gaze and pupil metrics to assess mental workload in gamified and simulated sensorimotor tasks. *Sensors*, 24(6), March 2024.

[21] Iris Graessler, Marcel Ebel, and Jens Pottebaum. Model-based planning of test cases and test scenarios to support engineering of cyber-physical systems. In *2024 IEEE International Symposium on Systems Engineering (ISSE)*, pages 1–8. IEEE, 2024.

[22] Iris Graessler, Marcel Ebel, and Jens Pottebaum. Method for systematic generation of test scenarios for early customer involvement. In *in CIRP DESIGN: 35th CIRP Design Conference, Patras, Greece*. CIRP, 2025.

[23] C. W. J. Granger. Investigating causal relations by econometric models and cross-spectral methods. *Econometrica*, 37(3):424–438, 1969.

[24] Arthur Gretton, Karsten M. Borgwardt, Malte J. Rasch, Bernhard Schölkopf, and Alexander Smola. A kernel two-sample test. *Journal of Machine Learning Research*, 13(25):723–773, 2012.

[25] Riccardo Guidotti, Anna Monreale, Fosca Giannotti, Dino Pedreschi, Salvatore Ruggieri, and Franco Turini. Factual and counterfactual explanations for black box decision making. *IEEE Intell. Syst.*, 34(6):14–23, 2019.

[26] Riccardo Guidotti, Anna Monreale, Salvatore Ruggieri, Franco Turini, Fosca Giannotti, and Dino Pedreschi. A survey of methods for explaining black box models. *ACM Compututins Surveys*, 51(5), August 2018.

[27] Torsten Hägerstrand. What about people in regional science. *Transport Sociology: Social aspects of transport planning*, pages 143–158, 1970.

[28] David Hallac, Sagar Vare, Stephen Boyd, and Jure Leskovec. Toeplitz inverse covariance-based clustering of multivariate time series data. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '17, page 215–223, New York, NY, USA, 2017. Association for Computing Machinery.

[29] Christopher M. Harris and Daniel M. Wolpert. The main sequence of saccades optimizes speed-accuracy trade-off. *Biological Cybernetics*, 95(1):21–29, July 2006. Epub 2006-03-23.

[30] Eleanor Hassan, · Jones, and · Buckingham. A novel protocol to induce mental fatigue. *Behavior Research Methods*, 56, 08 2023.

[31] Todd S. Horowitz and Jeremy M. Wolfe. Visual search has no memory. *Nature*, 394(6693):575–577, 1998.

[32] Manuel Viejo Isabel Valdés. Madrid will close its borders for 10 days in early December — english.elpais.com. https://english.elpais.com/spanish_news/2020-11-20/madrid-will-close-its-borders-for-10-days-in-early-december.html. [Accessed 25-02-2025].

[33] M. G. Kendall. A new measure of rank correlation. *Biometrika*, 30(1-2):81–93, 06 1938.

[34] Maurice G. Kendall. *The Advanced Theory of Statistics, Vol. 2: Inference and Relationship*. Charles Griffin & Company, 1945.

[35] W. K. Kirchner. Age differences in short-term retention of rapidly changing information. *Journal of Experimental Psychology*, 55(4):352–358, 1958.

[36] Menno-Jan Kraak. The space-time cube revisited from a geovisualization perspective. *Proceedings of the 21st International Cartographic Conference*, pages 1988–1996, August 2003.

[37] Wolfgang Maass, Arturo Castellanos, Monica C. Tremblay, Roman Lukyanenko, and Veda C. Storey. Ai explainability: Embedding conceptual models. In Niels Bjørn-Andersen, Roman Beck, Stacie Petter, Tina Blegind Jensen, Tilo Böhmann, Kai Lung Hui, and Viswanath Venkatesh, editors, *Proceedings of the 43rd International Conference on Information Systems, ICIS 2022, Digitization for the Next Generation, Copenhagen, Denmark, December 9-14, 2022*. Association for Information Systems, 2022.

[38] Samuele Marcora, Walter Staiano, and Victoria Manning. Mental fatigue impairs physical performance in humans. *Journal of applied physiology (Bethesda, Md. : 1985)*, 106:857–64, 02 2009.

[39] Raffaele Mazziotti, Fabio Carrara, Aurelia Viglione, Lupori Leonardo, Lo Verde Luca, Benedetto Alessandro, Ricci Giulia, Sagona Giulia, Amato Giuseppe, and Pizzorusso Tommaso. Human and Mouse Eyes for Pupil Semantic Segmentation, February 2021.

[40] Raffaele Mazziotti et al. MEYE: Web App for Translational and Real-Time Pupillometry. *eNeuro*, 8(5):ENEURO.0122–21.2021, September 2021.

[41] Tamara Munzner. *Visualization analysis and design*. CRC press, 2014.

[42] Lei Peng, Ziyue Lin, Natalia Andrienko, Gennady Andrienko, and Siming Chen. Contextualized visual analytics for multivariate events. *Visual Informatics*, 9(2):100234, 2025.

[43] Miguel Ponce-de Leon, Javier Del Valle, José María Fernandez, Marc Bernardo, Davide Cirillo, Jon Sanchez-Valle, Matthew Smith, Salvador Capella-Gutierrez, Tania Gullón, and Alfonso Valencia. Covid-19 flow-maps an open geographic information system on covid-19 and human mobility for spain. *Scientific Data*, 8(1):310, 2021.

[44] Miguel Ponce-de Leon, Javier del Valle, José María Fernández, Marc Bernardo, Davide Cirillo, Jon Sanchez-Valle, Matthew Smith, Salvador Capella-Gutierrez, Tania Gullón, and Alfonso Valencia. COVID19 Flow-Maps daily cases reports, 2021.

[45] Miguel Ponce-de Leon, Javier del Valle, José María Fernández, Marc Bernardo, Davide Cirillo, Jon Sanchez-Valle, Matthew Smith, Salvador Capella-Gutierrez, Tania Gullón, and Alfonso Valencia. COVID19 Flow-Maps daily-mobility for Spain, 2021.

[46] Miguel Ponce-de Leon, Javier del Valle, José María Fernández, Marc Bernardo, Davide Cirillo, Jon Sanchez-Valle, Matthew Smith, Salvador Capella-Gutierrez, Tania Gullón, and Alfonso Valencia. COVID19 Flow-Maps population data, 2021.

[47] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. *arXiv preprint arXiv:1505.04597*, 2015.

[48] Fabio Michele Russo, Carlo Metta, Anna Monreale, Salvatore Rinzivillo, and Fabio Pinelli. Explainable ai in time-sensitive scenarios: Prefetched offline explanation model. In Dino Pedreschi, Anna Monreale, Riccardo Guidotti, Roberto Pellungrini, and Francesca Naretto, editors, *Discovery Science*, pages 167–182, Cham, 2025. Springer Nature Switzerland.

[49] J.W. Sammon. A nonlinear mapping for data structure analysis. *IEEE Transactions on Computers*, C-18(5):401–409, 1969.

[50] Gideon Schwarz. Estimating the dimension of a model. *The Annals of Statistics*, 6(2):461–464, 1978.

[51] Roger N Shepard and Jacqueline Metzler. Mental rotation of three-dimensional objects. *Science*, 171(3972):701–703, 1971.

[52] David J. Sheskin. *Handbook of Parametric and Nonparametric Statistical Procedures*. Chapman and Hall/CRC Press, 6th edition, 2020.

[53] Sidney Siegel and N. John Castellan. *Nonparametric Statistics for the Behavioral Sciences*. McGraw-Hill, 2nd edition, 1988.

[54] Samuel Stuart, Brook Galna, Sue Lord, Lynn Rochester, and Alan Godfrey. Quantifying saccades while walking: Validity of a novel velocity-based algorithm for mobile eye tracking. In *2014 36th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pages 5739–5742, 2014.

[55] Gábor J. Székely and Maria L. Rizzo. Energy statistics: A class of statistics based on distances. *Journal of Statistical Planning and Inference*, 143(8):1249–1272, 2013.

[56] P.C Terry, A.M Lane, and G.J Fogarty. Construct validity of the profile of mood states — adolescents for use with adults. *Psychology of Sport and Exercise*, 4(2):125–139, 2003.

[57] Peter C. Terry, Andrew M. Lane, Helen J. Lane, and Lee Keohane. Development and validation of a mood measure for adolescents. *Journal of Sports Sciences*, 17(11):861–872, 1999. PMID: 10585166.

[58] Colin Ware. *Information visualization: perception for design*. Morgan Kaufmann, 2019.

[59] Wang Xi, Tao Pei, Qiyong Liu, Ci Song, Yaxi Liu, Xiao Chen, Jia Ma, and Zhixin Zhang. Quantifying the time-lag effects of human mobility on the covid-19 transmission: a multi-city study in china. *IEEE Access*, 8:216752–216761, 2020.