Ca' Foscari
University
of Venice

Ph.D. Degree
in Computer Science

Cycle XXXVIII

Final Thesis

Modelling and Analysis of Mobility Data in Spatiotemporal Databases

Supervisors
Professor Salvatore Orlando
Professor Alessandra Raffaetà
Professor Marta Simeoni

Graduand
Giulia Rovinelli

Matric. number: 867381

Academic Year 2025/2026

Acknowledgments

First and foremost, I would like to thank my PhD supervisors: Professor Salvatore Orlando, Professor Alessandra Raffaetà and Professor Marta Simeoni. Their guidance, support, and encouragement have played a crucial role throughout these years. I am grateful for their trust, availability, scientific insight, and for the constant support I have received during this journey. I am profoundly grateful for the time and effort you have dedicated to mentoring me, and for everything you continue to teach me every single day. Thank you.

I would like to thank Professor Davide Rocchesso for his support, his advice, and the work we carried out together. His expertise and thoughtful feedback have enriched this work, and I am grateful for the many discussions we shared.

My deepest thanks go to Professor Esteban Zimányi, who welcomed me to Brussels for my research stay abroad. His guidance, inspiring ideas, and constant encouragement have played a central role throughout my PhD. His presence, enthusiasm for research, and generous advice have made a significant contribution to both the direction of this thesis and my growth as a researcher.

I am also grateful to Professor Mahmoud Sakr for the many valuable suggestions and the long discussions we shared during my time in Brussels.

Thank you to Professors Karine Zeitouni and Yannis Theodoridis for your careful and thoughtful review of this thesis. I am deeply grateful for your invaluable feedback, which has helped me improve this work.

I would also like to thank the research group of Professors Zimányi and Sakr for their helpful suggestions and the support they offered during the *Friday meetings*, as well as for warmly welcoming me into the group.

My thanks also go to all the colleagues and researchers I have met along this journey. Each of you has taught me something invaluable, and I am truly grateful for the support, ideas, and encouragement you have shared with me over these years.

I would also like to thank my friends for always supporting me. Your care and encouragement have been truly important throughout these years.

Finally, my deepest gratitude goes to my family. Thank you for your constant support, your patience, and the love you have given me throughout these years. Your presence has been a constant source of strength throughout this journey, and I love you more than words can express.

Giulia

Abstract

Mobility is a fundamental property of both human society and the natural world. Understanding how people, animals, and other moving entities interact with their surroundings — whether through migration, transportation, or daily movement — is essential to interpret behaviours, optimise systems, and reveal the dynamics that shape our environment. From marine species and atmospheric phenomena to vessels navigating oceans and aircraft crossing continents, mobility reflects the continuous interplay between human activity, natural processes, and the forces that govern the Earth's systems.

Within the framework of the *Interconnected Nord-Est Innovation Ecosystem* (iNEST), mobility plays a key role in fostering innovation across scientific, technological, and cultural domains. The project aims to develop advanced solutions that enhance well-being, promote sustainable development, and stimulate economic and cultural growth through interconnected regional ecosystems. This thesis contributes to that vision by addressing mobility as a multidimensional phenomenon — encompassing human, environmental, and technological dynamics — and by developing data-driven approaches to understand and model such interactions in both maritime and urban contexts.

In the first part of the thesis, we analyse vessel trajectories using Automatic Identification System (AIS) data, enriched with semantic information. Building upon these trajectories, we propose a model for the spatiotemporal characterisation of underwater noise generated by vessels. The model integrates a description of underwater sound propagation with trajectory-based information to infer how noise spreads across the area of interest. All these analyses have been implemented within MobilityDB, an open source moving object database designed to support the efficient representation and analysis of spatiotemporal data. Moreover, we optimise the underwater noise model through table partitioning and parallelisation techniques. These enhancements significantly improved computational performance while preserving the accuracy of the noise model. The resulting framework provides a scalable and efficient solution for large-scale underwater noise analysis.

During the study of underwater sound propagation, it became evident that modelling acoustic fields — as well as other natural or anthropogenic phenomena

such as hurricanes, oil spills, or wildfires — requires a data model capable of representing *deformable moving regions*. This observation motivated the second part of the thesis, in which we propose a new spatiotemporal data type that represents a deformable moving region with a circular geometry, whose position and radius vary continuously over time. In addition to the new data type, called *temporal circular buffer*, we developed a suite of functions and operators to efficiently query and analyse this representation.

The third part of the thesis extends the study of mobility to the urban domain, through the analysis of ticket validation data from the public transport network of Venice. The uncontrolled growth of tourism has led to a condition of *overtourism*, with increasing pressure on the city's infrastructure and residents. Access to large-scale ticket validation datasets is therefore crucial to understand mobility patterns, support evidence-based policy design, and promote sustainable forms of tourism. In this thesis, we analyse the movements of different user categories to investigate how they move through the city. Understanding these dynamics is essential to ensure passenger safety, optimise water bus allocation, and mitigate the impacts of overtourism in a fragile urban ecosystem such as Venice.

Overall, this thesis advances the study and analysis of mobility through scalable spatiotemporal models and the application of diverse analysis methods to the maritime and urban domains, while extending the capabilities of spatiotemporal databases.

Keywords. Mobility • Spatiotemporal Database • Semantic Trajectories • Underwater Noise Model • Moving Regions

Sommario

La mobilità è una proprietà fondamentale sia della società umana sia del mondo naturale. Comprendere come persone, animali e altre entità in movimento interagiscono con l'ambiente — che si tratti di migrazione, trasporto o spostamenti quotidiani — è essenziale per interpretare i comportamenti, ottimizzare i sistemi e comprendere le dinamiche che modellano il nostro ambiente. Dalle specie marine e fenomeni atmosferici alle imbarcazioni che attraversano gli oceani e aeromobili che sorvolano i continenti, la mobilità riflette l'interazione continua tra attività umane e processi naturali che governano i sistemi della Terra.

Nel contesto dell'*Interconnected Nord-Est Innovation Ecosystem* (iNEST), la mobilità riveste un ruolo centrale nel promuovere l'innovazione nei domini scientifico, tecnologico e culturale. Il progetto mira a sviluppare soluzioni che migliorino il benessere, favoriscano uno sviluppo sostenibile e stimolino la crescita economica e culturale attraverso ecosistemi regionali interconnessi. Questa tesi contribuisce a tale visione affrontando il tema della mobilità come fenomeno multidimensionale — che abbraccia aspetti umani, ambientali e tecnologici — sviluppando approcci basati sui dati per comprendere e modellare tali interazioni nei contesti marittimo e urbano.

Nella prima parte della tesi vengono analizzate le traiettorie delle imbarcazioni utilizzando i dati del Sistema di Identificazione Automatica (AIS), arricchiti con informazioni semantiche. Sulla base di tali traiettorie, proponiamo un modello per la caratterizzazione spazio-temporale del rumore sottomarino generato dalle navi. Le analisi sono state implementate all'interno di MobilityDB, un database open source per oggetti in movimento che consente una rappresentazione e un'analisi efficienti dei dati spazio-temporali. Inoltre, il modello di rumore sottomarino è stato ottimizzato attraverso tecniche di partizionamento delle tabelle e parallelizzazione. Le ottimizzazioni introdotte hanno migliorato significativamente le prestazioni computazionali, preservando l'accuratezza del modello di propagazione del suono. Il framework risultante fornisce una soluzione scalabile ed efficiente per l'analisi di grandi moli di dati acustici.

Durante lo studio della propagazione del suono sottomarino, è emerso che la modellazione dei campi acustici — come di altri fenomeni naturali, quali uragani,

sversamenti di petrolio o incendi — richiede un modello di dati capace di rappresentare le *deformable moving region*. Questa osservazione ha motivato la seconda parte della tesi, in cui proponiamo un nuovo tipo di dato spazio-temporale in grado di rappresentare tali regioni con geometria circolare, la cui posizione e raggio variano nel tempo. Oltre al nuovo tipo di dato, denominato *temporal circular buffer*, abbiamo sviluppato un insieme di funzioni e operatori per interrogare e analizzare in modo efficiente tale rappresentazione.

La terza parte della tesi estende lo studio della mobilità al dominio urbano, attraverso l'analisi dei dati di validazione dei biglietti del trasporto pubblico di Venezia. La crescita del turismo ha portato a una condizione di *overtourism* con una pressione crescente sulle infrastrutture cittadine e sui residenti. L'accesso a dataset di convalida dei biglietti è pertanto cruciale per comprendere i modelli di mobilità e promuovere forme di turismo sostenibile. In questa tesi, analizziamo i movimenti di diverse categorie di utenti per investigare come essi si spostano all'interno della città. Comprendere tali dinamiche è fondamentale per garantire la sicurezza dei passeggeri, ottimizzare l'allocazione dei vaporetti e mitigare gli effetti del turismo in un ecosistema urbano fragile come quello veneziano.

Questa tesi avanza lo studio e l'analisi della mobilità attraverso modelli spazio-temporali scalabili e l'applicazione di diversi metodi di analisi ai domini marittimo e urbano, ampliando al contempo le capacità dei database spazio-temporali.

Contents

List of Figures

List of Tables

Acronyms

ADT Abstract Data Type 29

BRIN Block Range Index 24

DBMS Database Management System 23

GIS Geographical Information System 24

GiST Generalised Search Tree 24

MOD Moving Object Database 11

POIs Points of Interest 15

SP-GiST Space-Partioned Generalised Search Tree 24

Chapter 1

Introduction

Mobility is a fundamental characteristic of both human society and the natural world. The movement of people, animals, and physical phenomena shapes landscapes, drives ecological processes, and influences how communities develop and evolve. Different kinds of mobilities — from human travel and vessel navigation to wildlife migration and the propagation of environmental phenomena – reveal the continuous interplay between dynamic entities and the spaces they inhabit. Understanding these movements is crucial for interpreting behaviours, optimising infrastructures, mitigating environmental impacts, and designing sustainable and resilient systems. With the intensification of global mobility and the rapid expansion of available data, movement modelling and analysis have become major research challenges across multiple domains.

Within this broader scenario, the *Interconnected Nord-Est Innovation Ecosystem* (iNEST) provides a strategic framework for advancing knowledge and technological innovation. The initiative promotes the development of solutions that enhance well-being, foster sustainable development, and stimulate cultural and economic growth through the integration of regional ecosystems. This thesis contributes to that vision by addressing mobility as a multidimensional phenomenon — encompassing human, environmental, and technological dynamics – and by developing data-driven approaches to understand and model such interactions in both maritime and urban contexts.

The theoretical framework underlying the thesis is that of semantic trajectories, which enrich raw sequences of spatiotemporal points by integrating contextual and semantic information, including environmental/physical features, and the activities performed by the object during the movement. This semantic enrichment enables a more expressive representation of mobility data and supports advanced analysis of movement patterns and behavioural dynamics.

The first research direction addresses maritime mobility. We analyse vessel

trajectories derived from Automatic Identification System (AIS) data, enriched with semantic information such as the speed of the boat and the type of activity performed by each vessel. This semantic characterisation allows us to better understand how ships move within the area of interest and how they exploit the maritime space. In addition, we identify individual trips for each vessel and assign an anomaly category to each of them, distinguishing, for example, vessels that remain within port boundaries, trips exceeding twenty-four hours, or cases in which AIS transmissions appear to have been intentionally disabled. This initial analysis and semantic enrichment highlighted the potential for constructing an underwater noise model grounded in vessel movement. Existing approaches to underwater noise estimation typically rely either on direct acoustic measurements [27, 34, 40, 100], numerical simulations [69, 74, 75], or methods based on raw AIS data [36, 87]. The model developed in this thesis exploits semantic enriched vessel trajectories, integrating underwater sound propagation principles with detailed behavioural and navigational information to infer how noise spreads across the marine environment. By explicitly linking acoustic emissions to vessel positions, speeds, and activities, the model provides a scalable and data-driven representation of the acoustic footprint of maritime traffic. All stages of the analysis were implemented within MobilityDB [152, 153], an open-source moving object database built on top of PostgreSQL [52] and PostGIS [51], designed for the efficient storage, indexing, and querying of spatiotemporal data. To further enhance computational performance, we introduce partitioning and parallelisation strategies that exploit both native PostgreSQL capabilities and the distributed architecture of Citus [24]. These optimisations significantly enhance the performance of large-scale noise modelling, thereby enabling complex analyses over extensive maritime datasets.

During the development of the underwater noise model, it became evident that modelling acoustic fields — as well as other natural or anthropogenic phenomena such as hurricanes, oil spills, or wildfires — requires a data model capable of representing *moving regions*. In general, Moving Object Databases (MODs) [57] provide mechanisms for storing and querying large volumes of spatiotemporal data describing objects whose position or extent evolves over time. Within this framework, *moving reals* represent the temporal evolution of numerical attributes (for instance, speed), *moving points* capture the movements of entities such as cars, aircraft, or vessels, and *moving regions* describe spatial objects whose geometry changes over time, including forest fire perimeters, glacier extents, or pollution plumes. While substantial research has focused on moving points, the modelling of moving regions has received comparatively less attention. This is mainly due to the intrinsic complexity of representing deformable geometries and to the limited availability of real-world datasets. Nevertheless, the ability to model such phenomena is essential for applications in environmental monitoring, risk assessment,

and spatiotemporal simulation.

Moving regions can be broadly categorised into two classes: deforming and non-deforming. Non-deforming moving regions, also referred to as *rigid temporal geometries*, are geometries that may translate and rotate over time while preserving their shape throughout the movement. Earlier work on rigid moving regions, such as [62], introduced a data model for storing fixed-shape moving regions and implemented it in SECONDO [53], an extensible research-oriented database. However, SECONDO is primarily intended as a research prototype and is therefore not suited for operational or industry use. More recently, Schoemans et al. [124] proposed a new data model for rigid temporal geometries and integrated it into MobilityDB, thereby providing a practical and efficient solution for handling fixed-shape moving regions within a production-grade environment.

On the other hand, deforming moving regions are geometries whose shape evolves freely over time. Such representations are particularly important when modelling natural phenomena with continuously changing boundaries. Previous research has explored the reconstruction of deforming moving regions from temporal snapshots [61, 134], as well as the development of data models designed to represent these regions efficiently [46, 63]. Despite these contributions, the support for deformable geometries in existing moving object database systems remains limited. This thesis extends the capabilities of moving object databases by introducing a new data model for deforming moving regions with circular geometry, whose position and radius vary continuously over time. Alongside the definition of the abstract data type, referred to as the *temporal circular buffer*, we introduce a comprehensive set of functions and operators enabling its efficient manipulation, querying, and analysis. In addition, we implement the proposed temporal circular buffer within MobilityDB, together with the complete set of functions and operators required to support this new data type. The practical relevance of the data type is illustrated through a case study in which temporal circular buffers are employed to represent and analyse the impact of underwater noise in the maritime domain.

Beyond the maritime domain, the thesis extends the study of mobility to an urban context through the analysis of large-scale ticket validation data from the public transport network of Venice. Although validation datasets are increasingly available for public transport systems in many cities [44, 67, 138], the dataset examined in this thesis is particularly valuable, as it captures detailed mobility traces in a city whose unique morphology, reliance on water-based transport, and intense tourist pressure, make it fundamentally different from conventional urban environments. Moreover, in recent years, Venice has faced a growing condition of *overtourism*, placing significant strain on its infrastructure and residents, and amplifying the need for data-driven tools to understand and manage mobility.

In this thesis, the ticket validation dataset is used to analyse mobility through the lens of user categories and temporal validation patterns, allowing the identification of *who* moves through the city, *when* these movements occur, and *where* they take place within the urban network. Understanding these dynamics is crucial for ensuring passenger safety, optimising the allocation of water buses, and mitigating the broader impacts of tourism on the fragile urban ecosystem. By providing a detailed depiction of how different groups interact with the transport system, this analysis offers valuable insights to support the management of mobility in a complex and highly sensitive urban environment.

1.1 Main Contributions

The main contributions of this thesis are three-fold. First, the work focuses on the study of trajectories, their semantic enrichment, and the construction of an underwater noise model based on vessels trajectories.

- Starting from AIS data for all vessel types, we reconstruct and enrich their trajectories with semantic attributes describing navigational behaviour and vessel activity, identifying individual trips and classifying behavioural anomalies. This analysis is carried out within MobilityDB, the spatiotemporal database used to store, manage, and query the enriched trajectories [11, 114].
- We propose two spatiotemporal models for underwater noise. The first is a new model specifically designed for fishing vessels, which estimates the noise levels based on engine power. The second model applies to all other vessel categories and derives underwater noise emissions from vessel length and type (e.g., recreational, cargo, cruise). Combined with semantic trajectories, these models enable the estimation of the spatial and temporal distribution of vessel-generated noise [115, 116].
- We implement the full modelling pipeline in MobilityDB and conduct an extensive study on database partitioning strategies to improve the performance of large-scale underwater noise simulations. In PostgreSQL, we evaluate range, hash, and list partitioning, as well as space-tiling techniques, assessing their impact on both storage organisation and query execution. Furthermore, we leverage the distributed architecture provided by Citus, deploying the model on a four-node cluster to exploit inter-node parallelism. This combined approach enables the execution of large-scale noise models with significantly enhanced computational performance [113, 117], [155].
- We conduct a series of analyses to assess underwater noise across multiple frequency bands (63, 125, 400, and 4,000 Hz), enabling a detailed characteri-

sation of how different frequency components propagate through the marine environment [116].

- We demonstrate the analytical potential of the proposed models through a broad set of analyses, including, among others, the construction of bivariate maps that reveal underwater noise propagation patterns, the assessment of the environmental impact associated with different vessel categories, the analysis of seasonal variations in noise pollution, the investigation of the acoustic effects of COVID-19, and the development of interactive visualisations designed to support policymakers in interpreting and managing marine noise [116].

Second, the thesis contributes to spatiotemporal data modelling by introducing a new representation for deforming moving regions [152].

- We propose a new data type, called *temporal circular buffer*, which captures deforming moving regions with circular geometry whose position and radius vary continuously over time. The abstract data type is formally defined together with the full set of functions and operators required to support its semantics.

- We implement the proposed temporal circular buffer within MobilityDB, together with the complete set of functions and operators required to support this new data type. This implementation enables the efficient querying, manipulation, and analysis of circular moving regions.

- We demonstrate the applicability of the proposed model through a case study in which temporal circular buffers are employed to represent spatially evolving phenomena in the maritime domain.

Third, the thesis extends the study of mobility to the urban domain by analysing a large and uniquely valuable ticket validation dataset from the public transport network of Venice. This dataset, collected within the framework of the iNEST project, captures mobility in a city whose exceptional morphology and reliance on waterborne transport — where water buses constitute the primary means of movement — make it fundamentally different from conventional urban environments [153,154].

- We analyse a large and heterogeneous ticket validation dataset from the urban public transport network, covering multiple user categories, a variety of ticket types, three distinct seasons, major events such as Carnival, Easter, and the Venice Film Festival, and key holiday periods. This dataset provides a detailed representation of both ordinary and exceptional mobility dynamics in Venice.

- We conduct an extensive exploratory analysis of urban mobility patterns, examining seasonal variations, daily mobility during major events, and hourly validation distributions across different user groups.
- We reconstruct individual travel sequences from ticket validations and analyse sparse user trajectories, identifying dwell times, morning departure and afternoon return patterns, and the most frequently used stops for residents, students, workers, and retirees, while also characterising multi-day mobility patterns for tourists.
- We apply unsupervised clustering techniques to group stops according to two complementary perspectives: their predominant user categories, and their temporal validation profiles. This approach reveals meaningful spatial and behavioural structures within the transport network.
- We compare clustering outcomes across multiple time periods to assess the stability and variability of stop usage patterns, providing insights into how mobility demand evolves in relation to seasonal changes and event-driven dynamics.

1.2 Structure of the Thesis

This thesis is divided in three parts. The first part introduces the theoretical foundations of semantic trajectories and outlines how raw spatiotemporal data can be enriched with contextual information to support mobility analysis. These concepts are then applied to the maritime domain through the reconstruction and semantic enhancement of AIS vessel trajectories, which form the basis for modelling underwater noise generated by vessels and for its implementation within MobilityDB.

- Chapter 2 introduces the concept of semantic trajectories and outlines the foundations of spatiotemporal mobility analysis. It also presents moving object databases, with particular attention to MobilityDB.
- Chapter 3 provides a detailed description of the datasets employed in the first part of this thesis and outlines the reconstruction and semantic enrichment of AIS trajectories.
- Chapter 4 introduces our first underwater noise propagation model for fishing vessels, leveraging their semantic trajectories to characterise vessel-generated underwater noise.

- Chapter 5 extends the underwater noise model presented in the previous chapter to all vessel categories. Moreover, the chapter also discusses the implementation of the noise model in MobilityDB, including performance optimisation through table partitioning and parallelisation.

The second part delves into the representation and analysis of deformable moving regions, motivated by the need to model phenomena whose shape and extent vary continuously over time. It introduces the *temporal circular buffer*, a new spatiotemporal data type designed to capture such dynamics, and details the suite of analytical functions and topological operators developed to support its use within a database environment.

- Chapter 6 defines the abstract data type of the temporal circular buffer, formalises its semantics, and describes the algorithms developed for distance computation, turning point detection, and spatial relationships.

- Chapter 7 examines the implementation of the data type and its associated operators in MobilityDB, showcasing a real-case example that illustrates its applicability in practical spatiotemporal analyses.

The third part of this thesis, Chapter 8, presents the analysis of urban mobility in Venice, focusing on the movements of passengers within the public transport network. Finally, Chapter 9 concludes the thesis and outlines possible directions for future work.

1.3 List of Publications

- **[152]** Giulia Rovinelli, Esteban Zimányi, Marta Simeoni, and Alessandra Raffaetà. "*Temporal Circular Buffer: A Spatiotemporal Data Type for Circular Moving Regions*" International Journal of Geographical Information Science. *In preparation*

- **[153]** Giulia Rovinelli, Gianluca Cuzzolin, Matteo Grazioso, Claudio Lucchese, Salvatore Orlando, and Alessandra Raffaetà. "*Analysis of the Mobility in Venice during Carnival, Easter and the Venice Film Festival*" Journal of Intelligent Transportation Systems: Technology, Planning, and Operation. *In preparation*

- **[154]** Giulia Rovinelli, Gianluca Cuzzolin, Matteo Grazioso, Claudio Lucchese, Salvatore Orlando, and Alessandra Raffaetà. "*Public Transport Mobility in Venice*" Scientific Data. *Ready to be submitted*

- **[155]** Giulia Rovinelli, Esteban Zimányi, Marta Simeoni, Davide Rocchesso, and Alessandra Raffaetà. "*A scalable AIS-based model for vessel-generated underwater noise*" GeoInformatica. *First Round of Revision*

- Giulia Rovinelli. "*A Spatiotemporal Framework for Underwater Noise Modelling*" Proceedings of the 19th International Symposium on Spatial and Temporal Data, pp. 132–136, 2025, https://doi.org/10.1145/3748777.3748803

- Giulia Rovinelli, Davide Rocchesso, Marta Simeoni, Esteban Zimányi, and Alessandra Raffaetà. "*Spatiotemporal characterisation of underwater noise through semantic trajectories*" GeoInformatica, vol. 29, pp. 845–876 (ISSN 1573-7624), 2025, https://doi.org/10.1007/s10707-025-00547-x

- Giulia Rovinelli, Esteban Zimányi, Marta Simeoni, Davide Rocchesso, and Alessandra Raffaetà. "*A Scalable Model for Vessel-Generated Underwater Noise: Enhancing Efficiency through Parallelisation*" 7th International Workshop on Big Mobility Data Analytics (BMDA 2025), CEUR-WS, vol. 3946, Workshops of the EDBT/ICDT Joint Conference, EDBT/ICDT-WS 2025, 2025 (ISSN 1613-0073), https://ceur-ws.org/Vol-3946/BMDA-5.pdf

- Giulia Rovinelli, Davide Rocchesso, Marta Simeoni, and Alessandra Raffaetà. "*Using semantic trajectories for spatiotemporal characterisation of underwater noise*" 6th International Workshop on Big Mobility Data Analytics (BMDA 2024), CEUR-WS, vol. 3651, Workshops of the EDBT/ICDT Joint Conference, EDBT/ICDT-WS 2024, 2024 (ISSN 1613-0073), https://ceur-ws.org/Vol-3651/BMDA-5.pdf

- Bruno Brandoli, Alessandra Raffaetà, Marta Simeoni, Pedram Adibi, Fateha Khanam Bappee, Fabio Pranovi, Giulia Rovinelli, Elisabetta Russo, Claudio Silvestri, Amilcar Soares, and Stan Matwin. "*From multiple aspect trajectories to predictive analysis: a case study on fishing vessels in the Northern Adriatic sea*" GeoInformatica, vol. 26, pp. 551-579 (ISSN 1573-7624), 2022, https://doi.org/10.1007/s10707-022-00463-4

- Giulia Rovinelli, Stan Matwin, Fabio Pranovi, Elisabetta Russo, Claudio Silvestri, Marta Simeoni, and Alessandra Raffaetà. "*Multiple aspect trajectories: A case study on fishing vessels in the northern adriatic sea*" 4th International Workshop on Big Mobility Data Analytics (BMDA 2021), CEUR-WS, vol. 2841, Workshops of the EDBT/ICDT Joint Conference, EDBT/ICDT-WS 2021, 2021 (ISSN 1613-0073), https://ceur-ws.org/Vol-2841/BMDA_12.pdf

Part I

Mobility Data

Chapter 2

Background

Mobility can be defined as the continuous change in the position of entities within *space* and *time* — the two fundamental dimensions through which everything exists and evolves [98]. It is an intrinsic property of our world: people commute between places, animals migrate, and atmospheric systems move across regions. In computer science, these phenomena are represented through *mobility data* [109], which describe the movement of entities whose position varies over time. Such data are typically available as sequences of location and timestamp pairs, where the location may represent either a point — defined by latitude and longitude — or an area capturing the movement of extended entities such as animal herds or groups. Mobility data science has emerged as an interdisciplinary field that applies scientific methods, algorithms, and computational systems to extract and interpret knowledge from large-scale, heterogeneous, and often noisy mobility datasets [120]. Its aim is to uncover meaningful patterns, behaviours, and interactions across a broad range of domains, including transportation, environmental monitoring, urban planning, and ecology.

The widespread adoption of positioning devices, sensors, and Internet of Things (IoT) technologies has led to the continuous collection of massive amounts of mobility data, capturing the temporal evolution of moving entities with increasing spatial and temporal resolution. This growing availability of large-scale data has opened new research directions and enabled the development of advanced systems for managing and analysing movement information.

In order to understand how such data can be modelled and analysed, we begin by introducing the most basic representation of movement: the *moving object* [37, 55, 57], which captures the spatiotemporal evolution of entities over time. A moving object is an entity whose spatial position — and possibly other attributes — evolves continuously over time. Examples include vehicles, vessels, aircraft, or animals equipped with tracking devices. The behaviour of moving objects is typi-

cally reconstructed from discrete observations collected at successive time instants. Large volumes of such data are produced by positioning systems such as GPS and are now central to numerous application domains, including autonomous navigation, social computing, contact tracing, intelligent transportation, and maritime monitoring. Consequently, there is a growing demand for software systems capable of efficiently managing and analysing large-scale spatiotemporal data. These systems must support all stages of the data science cycle, including acquisition, storage, cleaning, transformation, analysis, visualisation, and privacy preservation.

Next, we delve into the concept of trajectories [37, 103, 107], which provide a structured representation of how moving objects evolve through space and time. In addition to their geometric description, semantic trajectories [79, 95] enrich positional data with contextual information that characterises the purpose, activity, or meaning of movement. By linking spatial and temporal components with semantic dimensions — such as who is moving, where, when, how, and why — semantic trajectories allow a more comprehensive understanding of movement behaviour and facilitate advanced analyses such as pattern recognition, clustering, and reasoning.

Building upon these notions, the need to store, query, and analyse trajectories — together with other forms of moving object data — led to the development of Moving Object Database (MOD) [57, 99]. The next section introduces the concepts required to present these systems that extend traditional spatial databases by incorporating time as a fundamental dimension, allowing the modelling of continuously evolving entities. The theoretical foundations of this field were established by Güting [37, 55, 57], who proposed a formal framework for spatiotemporal data modelling based on the concept of spatiotemporal data types.

This chapter is organised as follows. Section 2.1 introduces the concept of moving objects and illustrates how their evolution can be represented through trajectories. It then examines the notion of trajectory in more detail and discusses its semantic enrichment. Section 2.2 describes spatial databases, while Section 2.3 presents moving object databases, which support the representation of entities whose position evolves over time. Finally, Section 2.4 provides a detailed overview of MobilityDB, the spatiotemporal database used throughout this thesis.

2.1 Mobility Data Analysis

A *moving object* [37, 55, 57] is an entity whose spatial position changes over time. In many applications, only the time-varying location of the object is of interest, and it is therefore modelled as a *moving point* [46]. A moving point is represented as a sequence of ordered pairs $(\text{location}, \text{timestamp})$, where each element describes the position of the object at a specific moment in time. The location component is

typically expressed in geographic coordinates (latitude and longitude), while the timestamp defines the temporal dimension of the observation. Typical examples include vehicles, vessels, aircraft, satellites, or animals equipped with positioning devices. These objects are characterised by continuous movement over time, while their positions can only be recorded at discrete moments due to the limitations of acquisition, storage, and processing technologies [80]. Starting from this discrete set of spatiotemporal points, it becomes essential to reconstruct the continuous movement of the object, thereby obtaining a trajectory.

2.1.1 Raw Trajectories

According to Parent et al. [95], a raw trajectory is defined as follows.

Definition 2.1. A *raw trajectory* is a time-ordered sequence of geographical points,

$$T = (\mathit{trajID}, \mathit{moID}, \{p_1, p_2, \ldots, p_n\})$$

where *trajID* and *moID* are the identifiers of the trajectory and the moving object, respectively. Each point p_i is a tuple $\langle x_i, y_i, t_i \rangle$, where x_i and y_i are the geographical coordinates, and t_i is the timestamp at which the location is recorded. The number of points n represents the length of the trajectory.

Figure 2.1 illustrates an example of a trajectory. The red solid line represents the movement of a point object through space and time, where the temporal dimension is combined with the spatial dimensions to form a three-dimensional coordinate system. The dashed line shows the projection of the movement onto the x–y plane, representing the object's spatial path over time [104].

It should be noted that a movement track contains only the positions recorded at specific sampling instants, while the actual movement is continuous. When analysing trajectories, we must reconstruct the continuous movement of an object from a finite set of discrete observations. For this reason, the estimation of intermediate positions becomes necessary. This reconstruction is achieved through *interpolation*, a process that determines how a value evolves between consecutive time instants. Interpolation is fundamental for describing the continuous position of a moving object, since location measurements provided by tracking devices are discrete in time. Several interpolation methods exist, but the most common are *piecewise-constant* and *linear* interpolation.

Definition 2.2. *Piecewise-constant interpolation*, also known as *zero-order hold* or *stepwise interpolation*, keeps the last observed value unchanged until the next observation. Specifically, for an observation at time t_i with value y_i, the interpolated value $f(t)$ for any time $t \in [t_i, t_{i+1})$ is y_i:

$$f(t) = y_i \quad \text{for } t \in [t_i, t_{i+1}) \tag{2.1}$$

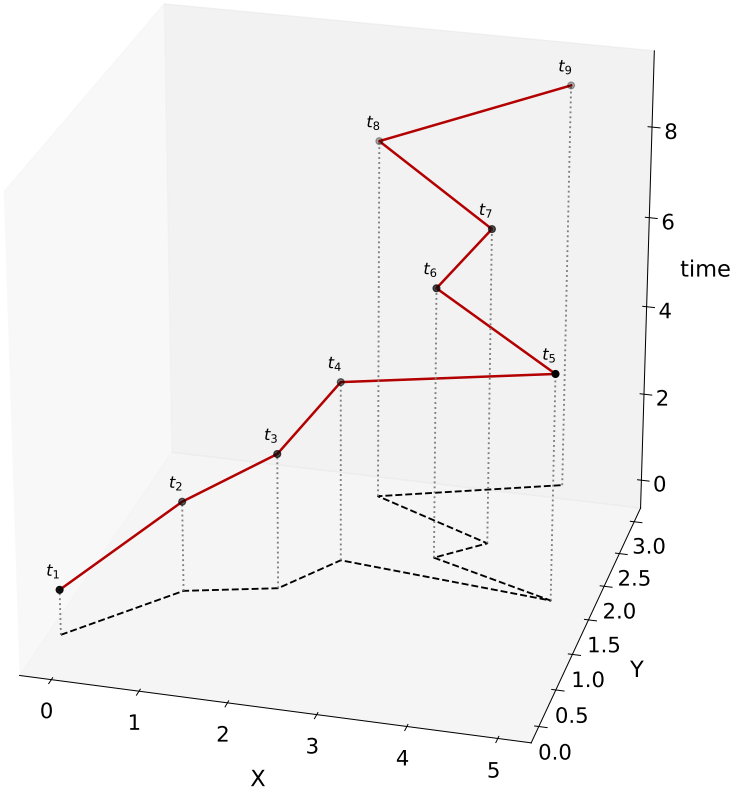


Figure 2.1: Trajectory of a moving point.

This produces a stepwise function and is often appropriate for categorical attributes or when no assumption about in-between change can be justified.

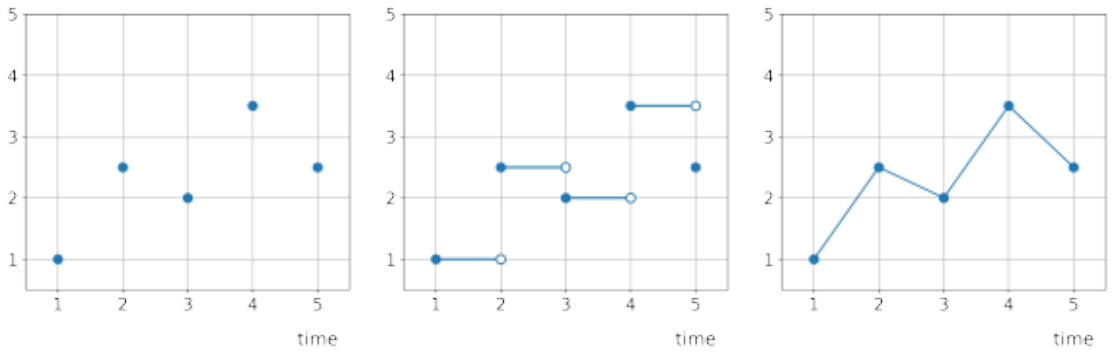
Definition 2.3. *Linear interpolation* assumes that the value evolves linearly between two consecutive observations. Given two points observed at times t_i and t_{i+1} with values y_i and y_{i+1}, respectively, the interpolated value $f(t)$ for any time $t \in [t_i, t_{i+1}]$ is computed as:

$$f(t) = y_i + \frac{y_{i+1} - y_i}{t_{i+1} - t_i}(t - t_i) \tag{2.2}$$

This corresponds to the straight line connecting the two points.

Figure 2.2 shows how (a) discrete temporal positions can be used to obtain either (b) a stepwise interpolation or (c) a continuous one.

In many cases, movement tracks may contain unusually long temporal gaps between two consecutive recorded positions — longer than what would be expected



(a) Discrete points. (b) Stepwise. (c) Continuous.

Figure 2.2: From (a) discrete temporal positions to (b) stepwise and (c) continuous interpolation modes.

from the nominal sampling rate. Such gaps indicate that information about the object's movement is missing over a certain period of time. When this absence of data occurs unintentionally, for instance due to sensor malfunction or loss of signal, it is referred to as a *hole* in the trajectory. Conversely, when the interruption is intentional — for example, when a user deliberately disables the tracking device — it is known as a *semantic gap*. Distinguishing between these two situations is essential, as they have different implications for data analysis. Holes can often be mitigated or corrected through interpolation techniques which estimate the missing positions based on neighbouring data points. Semantic gaps, on the other hand, should remain explicitly marked and must not be interpolated, since the missing data reflect a deliberate discontinuity in the movement [109, 137].

Moreover, many applications do not require analysing an entire trajectory but instead focus on specific subsequences that are meaningful for the task at hand. Identifying such consecutive and behaviourally coherent parts is known as *trajectory segmentation*. This process is crucial because the segmentation method directly affects the resulting features used to characterise movement: a well-designed segmentation procedure yields more informative and representative descriptors of the object's behaviour [38].

Definition 2.4. *Trajectory segmentation* [95] is the process of identifying meaningful parts of a trajectory, called *episodes*. After segmentation, the trajectory is represented as a set of episodes,

$$T_{seg} = (trajID, moID, \{p_1, p_2, \ldots, p_n\}, \{e_1, e_2, \ldots, e_m\})$$

where m denotes the number of identified episodes. Each episode e_i is defined as a tuple, $\langle start_i, end_i \rangle$, where $start_i$ represents the spatiotemporal beginning of e_i and end_i its spatiotemporal termination.

One of the most widely adopted approaches to trajectory segmentation is the identification of *stop* and *move* episodes [130]. In this framework, a move represents a continuous change in position, corresponding to the travelling phase of an object, while a stop denotes a period during which the object remains within a limited spatial region for a certain duration.

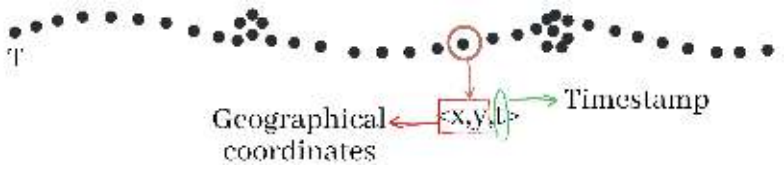
Definition 2.5. A *stop* [130] is a part of a trajectory such that the moving object has explicitly defined this part of the trajectory to represent a stop, or the travelling object does not move in space for a non-empty interval of time. All stops are temporally disjoint, i.e. the temporal extents of two stops are always disjoint.

Definition 2.6. A *move* [130] is a part of a trajectory, such that it is delimited by two stops (or by the beginning of the trajectory and the first stop, or by the last stop and the end of the trajectory), and the time and spatial range between the start and the end of the move is non-empty.

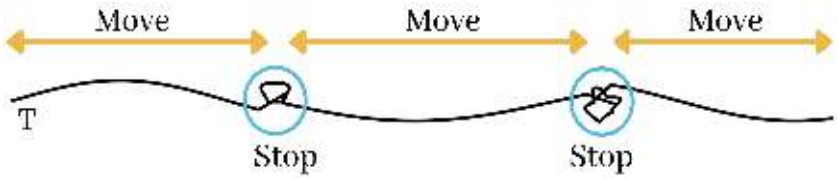
This stop–move model provides a behavioural abstraction of movement. For example, in an urban context, stops may correspond to workplaces, public transport stations, or tourist attractions, while moves represent the trips connecting these locations. By distinguishing between stationary and dynamic phases, this model supports higher-level mobility analyses such as activity recognition, pattern discovery, and mobility profiling.

Before going into a detailed analysis of what semantic trajectories are and how they extend raw trajectories, we first illustrate an example application scenario that informally describes the movement of a tourist visiting the city of Rome. Tourism represents one of the most significant sources of revenue for many countries, and cities with a strong cultural heritage often experience intense and continuous flows of visitors. Analysing tourists' movements can therefore provide valuable insights for urban planning, resource management, and sustainable tourism strategies. Consider, for instance, a tourist visiting Rome. A typical tourist moves across the city for visiting a variety of places (e.g., monuments, museums, parks, and attractions) while using many services (restaurants, accommodation, shops, and public transport). All these tourists places and services are referred to as Points of Interest (POIs). The sequence of spatiotemporal positions recorded by a tracking device during the tourist's visit constitutes what we previously defined as a raw trajectory (see Figure 2.3a). This discrete sequence can be interpolated to reconstruct the tourist's continuous movement between consecutive observations. Interpolation thus allows us to approximate the path followed by the tourist through space and time.

Let us now suppose that the tourist first visits a POI and then moves towards another one. In this case, the trajectory can be segmented into five episodes: two



(a) A raw trajectory.



(b) A segmented trajectory.

Figure 2.3: Examples of the different types of trajectories: (a) a raw trajectory, and (b) a segmented trajectory.

stop episodes at the POIs and three move episodes corresponding to the displacements between them. As this example illustrates (see Figure 2.3b), trajectory segmentation produces a more abstract representation of movement than the original raw trajectory. While the continuous representation captures a detailed sequence of spatiotemporal positions, the segmented representation provides a higher-level description of the same movement as a sequence of meaningful episodes, each defined by a tuple consisting of a time interval and an annotation value [95, 109].

2.1.2 Semantic Trajectories

The concept of semantic trajectories emerged in the early 2000s, when researchers first introduced the idea of describing movement not only through spatial and temporal data, but also through its associated meaning and context. As discussed previously, the stop–move paradigm [130] represents one of the earliest approaches to the semantic enrichment of trajectories. Building on this notion, Spaccapietra and Parent [129] introduced the concept of *annotation* to describe any additional information that enhances the knowledge associated with a trajectory, or with one of its parts, and that is stored together with the trajectory data. Annotations may originate from direct observation (for example, the detected activity of a moving entity such as stopping, waiting, or turning), from sensor measurements (e.g., speed, heading, or acceleration), or may be inferred through reasoning processes (for instance, deducing a traveller's transportation mode from movement features and contextual data). An annotation value can take several forms: it may be a

simple attribute (for example, the label *"resting"* indicating the behaviour of a marine animal), a reference to an external entity in an application database (such as the identifier *"tram line 5"* in a public transport table), or even a more complex structure combining multiple attributes and relational links. For instance, in a maritime context, a complex annotation for a fishing vessel could be *"trawling, species"*, where *species* refers to the object representing the target species caught during that activity.

Definition 2.7. A *semantic trajectory* [95] is a raw or segmented trajectory enriched with one or more annotations. It is defined as a tuple

$$ST = (trajID, moID, \{p_1, p_2, \ldots, p_n\}, \{e_1, e_2, \ldots, e_m\})$$

where each episode $e_i = \langle start_i, end_i, V_i, A_i \rangle$ represents an annotated segment of the trajectory. V_i is the value of the annotation of the segmentation criterion that defines an episode (e.g., stop or move), while A_i is the (possibly empty) set of annotations associated with the episode.

There are many different ways to annotate a trajectory, depending on the nature of the data and the intended analysis. For instance, some approaches exploit the spatial context by considering the proximity of trajectory points to relevant points of interest [147]. In this setting, a trajectory can be divided into segments influenced by specific POIs, where each position is associated with the closest POI within a defined spatial range. Building on this idea, Spinsanti et al. [131] propose a semantic enrichment process in which stops are annotated with the most likely POIs visited by an individual. Their method involves two main stages: first, relevant POIs are collected and filtered based on spatial and contextual criteria; second, probable activities are inferred for each POI and subsequently used to label the stops of the trajectory, thus providing a meaningful interpretation of the traveller's behaviour.

Another common approach relies on the identification of the transportation mode used by the moving object, which is crucial for urban mobility studies and transportation planning. The segmentation in this case is based on characteristic motion patterns that distinguish different travel modes, such as typical speed ranges, continuity of motion, and route constraints. For example, slow and irregular movements can be associated with walking, while faster and more continuous motion may correspond to vehicular transport. Public transport modes like buses or trams are often identified through periodic stops or alignment with predefined routes. Several studies proposed automated methods for this task, employing statistical or machine learning models [73, 150]. Speed, acceleration, and direction change rate are often used as input features to classify trajectory segments into travel modes such as walking, cycling, driving, or using public transport. In more

recent approaches, these methods are further enhanced by incorporating contextual information — such as road network topology or public transport schedules — to improve segmentation accuracy and ensure consistency with real-world mobility patterns.

Another important contribution to semantic trajectory modelling is SeMiTri (Semantic Middleware for Trajectories), proposed by Yan et al. [148]. SeMiTri enriches trajectories semantically by integrating their geometric properties with contextual information derived from geographic and application data. The framework is characterised by three annotation layers: (*i*) the *Semantic Region Annotation Layer*, which associates trajectories with geographical regions through the identification of Regions of Interest (ROIs); (*ii*) the *Semantic Line Annotation Layer*, which labels trajectory segments using semantic lines, where a classifier is employed to infer the transportation mode of the moving object; and (*iii*) the *Semantic Point Annotation Layer*, where stops are annotated according to the type of the POI associated with them.

Beyond spatial enrichment with POIs and transportation mode, trajectories can also be annotated with the *activity* performed by the moving object, offering a behavioural dimension to movement analysis. Such activity-based annotations capture what the object is doing at specific times or locations, complementing spatial information with semantic context. The literature is rich in studies addressing the enrichment of trajectories with activity annotations. For instance, Liao et al. [73] annotate stop episodes with the activities performed by the moving object (e.g., *"AtHome"*, *"Shopping"*) and associate move episodes with the corresponding transportation modes. In ecological research, many studies rely on manual annotation of activity episodes directly observed by scientists in the field, as shown by Cagnacci et al. [14]. In the urban domain, Gong et al. [50] propose a probabilistic framework for inferring trip purposes and uncovering travel patterns from taxi trajectory data. Their approach estimates the probability of visiting specific points of interest using Bayesian inference that integrates both spatial and temporal constraints, combined with Monte Carlo simulations to account for variability in individual mobility behaviour. Similarly, Cheng et al. [17] introduce an unsupervised method for semantic place annotation of trajectories based on prior spatial, temporal, and duration probabilities. Together, these contributions demonstrate the variety of methodologies — from manual field observation to probabilistic inference and machine learning — developed to associate meaningful behavioural semantics with raw mobility data.

Another, more general semantic trajectory conceptual data model is the one proposed by Bogorny et al. [10], called CONSTAnT (Conceptual Data Model for Semantic Trajectories of Moving Objects). It provides a domain-independent conceptual framework for defining and enriching semantic trajectories. The model

identifies key components — such as semantic subtrajectories, semantic points, geographical places, events, goals, environmental context, and behaviour — that together support a comprehensive and flexible representation of movement data across different application domains.

Although some of the works mentioned so far proposed innovative approaches, all of them are limited to enriching trajectories with a small number of semantic dimensions, often predefined in terms of number and type. To address these limitations, Mello et al. [79] introduced the MASTER model, which provides a new conceptual view of trajectories known as the *Multiple-Aspect Trajectory* (MAT). In this model, a trajectory is treated as a complex object enriched with multiple heterogeneous data dimensions — or *aspects* — that provide contextual information about the movement.

Definition 2.8. A *semantic aspect* $a_i = (desc, A)$ [79] is a real-world attribute that is relevant for trajectory analysis, where *desc* is the aspect description, and $A = \{a_i, a_2, \ldots, a_k\}$ is a set of attribute. Semantic aspects and attributes could be heterogeneous and different in type. All sets of semantic aspects can be empty.

Different types of aspects can be distinguished:

- *Volatile aspects* are associated with the individual points of a trajectory, as they vary dynamically during movement. Examples include instantaneous speed, acoustic intensity, and the activity performed by the moving object (e.g., the vessel's operational status, such as fishing or cruising).
- *Long-term aspects* remain constant throughout an entire trajectory and are therefore linked to it as a whole. Typical examples include the duration or the overall length of the trip.
- *Permanent aspects* characterise the entire life of a moving object and are thus associated with the object itself. For instance, a vessel's flag state or a person's place of birth can be considered permanent aspects.

Based on these foundations, a multiple aspect trajectory is defined as follows.

Definition 2.9. A *Multiple-Aspect Trajectory (MAT)* [79] is defined as a tuple

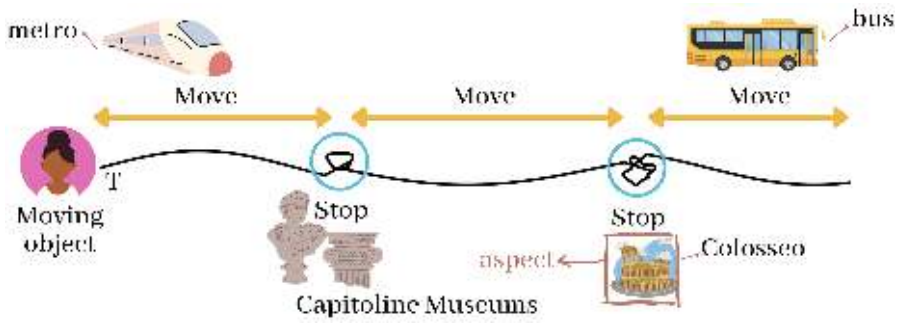
$$\text{MAT} = (trajID, mo, P, E, A_{traj})$$

where *trajID* is the identifier of the trajectory; $mo = (moID, A_{mo})$ represents the moving object with identifier *moID*; $P = \{p_1, p_2, \ldots, p_n\}$ is a sequence of timestamped geographical points, where each point $p_i = \langle x_i, y_i, t_i, A_{point_i} \rangle$, and $\langle x_i, y_i \rangle$ are the geographical coordinates and t_i is the timestamp. $E = \{e_1, e_2, \ldots, e_m\}$ is a sequence of episodes of a given moving object *mo*, where each episode $e_i =$

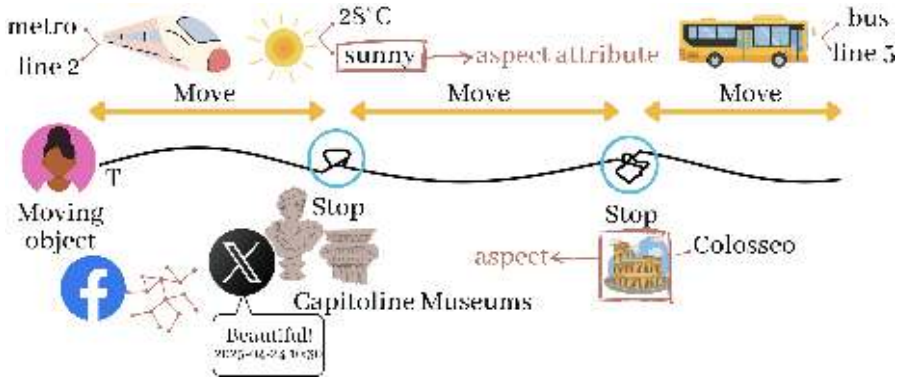
$\langle start_i, end_i, A_{episode_i} \rangle$, and $start_i$ and end_i are the timestamped geographical points corresponding to the beginning and end of the episode; A_{traj}, $A_{episode}$, A_{point}, and A_{mo} are sets of semantic aspects, with $A = \{a_1, a_2, \ldots, a_k\}$.

It is important to note that semantic enrichment may affect only specific portions of a trajectory, while remaining absent in others. This variability considerably increases the complexity involved in constructing and representing multiple-aspect trajectories compared with semantic trajectories.

To further illustrate this concept, let us revisit the example of a tourist visiting the city of Rome. In the previous discussion, we saw how the raw trajectory — composed solely of spatial coordinates and timestamps — could be segmented into *stops* and *moves*. The tourist, starting from the hotel, visits the Capitoline Museums and later moves towards the Colosseum. In this case, the journey from the hotel to the museum corresponds to a *move*, followed by a *stop* representing the museum visit; similarly, the trip from the museum to the Colosseum forms another move, while the visit to the monument constitutes the subsequent stop.



(a) A semantic trajectory.



(b) A multiple-aspect trajectory.

Figure 2.4: Examples of the different types of trajectories: (a) a semantic trajectory, and (b) a multiple-aspect trajectory.

By associating these segments with contextual information such as the visited POIs and the means of transportation used — for example, taking the metro from the hotel to the museum, walking from the museum to the Colosseum, and returning by bus — we obtain a *semantic trajectory*, as we can see in Figure 2.4a.

Following the MASTER model, this trajectory can be further enriched by integrating additional dimensions of information. For instance, beyond knowing that the tourist travelled by metro, we can include the specific line taken (e.g., Line 2), record environmental aspects such as weather conditions (e.g., a sunny day with an average temperature of 28°C), and even incorporate user-generated content, such as a social media post tagged at the Capitoline Museums with the comment *"Beautiful"*. The resulting trajectory captures multiple interrelated dimensions of the same movement — spatial, temporal, environmental, and behavioural — thus representing a comprehensive example of a MAT (see Figure 2.4b).

2.2 Spatial Databases

After defining the concept of semantic trajectories, it is important to address the problem of modelling and representing trajectories within database systems. Before discussing moving object databases, we first focus on spatial databases, which form the foundation for spatiotemporal data management. Spatial databases can be broadly categorised into two main types: relational (SQL-based) and NoSQL systems. Traditional relational database management systems (RDBMSs) with spatial extensions have been established and refined over several decades. These systems are stable, mature, and efficient, and have been successfully applied across a wide range of enterprise-level domains.

Traditional RDBMSs are widely used for their efficiency in data management and query processing. Consequently, research on spatial and spatiotemporal database systems has largely evolved through the extension of existing RDBMS architectures. Examples include PostgreSQL/PostGIS [51], Oracle Spatial [94], IBM DB2 Spatial Extender [1], Microsoft SQL Server [41], MySQL Spatial [86], and SpatiaLite [48], which represent some of the most widely adopted spatial RDBMSs in both academia and industry [4]. These systems are well-established, stable, and equipped with efficient SQL query engines. They support common spatial data formats such as Well-Known Text (WKT) and Well-Known Binary (WKB), as well as geometry objects in compliance with the Open Geospatial Consortium (OGC) Simple Feature Access specification [91, 92].

The OGC consists of two implementation standards. Part 1 (Common Architecture) [91] defines the abstract geometry model and provides the formal structures and operations required to represent and manipulate geographic features. It constitutes a profile of the spatial schema defined in ISO 19107, which specifies

the abstract spatial schema for geographic information and establishes the conceptual foundations adopted by most spatial database systems. In the OGC model, a geometry is formally interpreted as a subset of Euclidean space embedded in a coordinate reference system. Each geometry is characterised by three mutually disjoint point sets: the interior, the boundary, and the exterior. This topological structure provides the basis for the formal definition of spatial relationships between geometries. The specification defines a hierarchy of geometry types, including zero-dimensional (`Point`), one-dimensional (`LineString` or `Curve`), and two-dimensional (`Polygon` or `Surface`) primitives, as well as geometry collections. Spatial predicates such as `intersects`, `contains`, and `within` are evaluated according to the Dimensionally Extended Nine-Intersection Model (DE-9IM) [32], which characterises spatial relationships through the pairwise intersections of the interior, boundary, and exterior of two geometries.

Part 2 (SQL Option) [92] extends this framework by defining the SQL schema and geometry types necessary for the storage, retrieval, querying, and updating of spatial feature collections within relational database management systems. It specifies a set of SQL geometry types and associated functions, thereby enabling the integration of spatial data within standard SQL environments. These include geometry types such as `Point`, `LineString`, and `Polygon`, as well as spatial functions for geometry construction, spatial predicates, and spatial analysis (e.g., `ST_Intersects`, `ST_Contains`, and `ST_Distance`). The standard also defines metadata structures describing the geometry columns stored in database tables and supports the use of spatial indexing mechanisms to accelerate spatial queries. The standard is widely adopted by major GIS and database platforms, including ESRI products, Oracle Spatial, and PostgreSQL/PostGIS.

Moreover, most spatial RDBMSs rely on R-tree–based indexing for spatial data access, with the exception of SQL Server and IBM DB2, which employ grid-based indexing strategies. Among them, only PostgreSQL/PostGIS and Oracle Spatial natively support the storage and processing of spatial raster data. Furthermore, advanced systems such as PostgreSQL/PostGIS, Oracle Spatial, and SQL Server provide a comprehensive set of spatial relationship and analysis functions defined in the OGC and ISO SQL/MM (Part 3) standards, thus enabling a wide range of spatial operations — including joins, range queries, and topological predicates. However, traditional spatial RDBMSs face several limitations when dealing with modern, large-scale, and heterogeneous spatial data. Their performance can degrade due to I/O bottlenecks, limited parallelism, and restricted horizontal scalability. Moreover, modelling complex and multidimensional datasets within the rigid schema of relational systems remains challenging.

In recent years, these systems have evolved significantly, with continuous integration of new functionalities aimed at meeting the demands of big spatial data.

For instance, PostgreSQL/PostGIS now supports the GeoJSON format for web-based spatial applications. Given its robustness, extensibility, and open-source nature, PostgreSQL/PostGIS stands out as one of the most widely adopted spatial RDBMSs.

In parallel with the evolution of spatial extensions for traditional relational databases, a new class of data management systems, known as NoSQL (Not-Only-SQL) databases, has emerged to address the limitations of relational models in handling large-scale, heterogeneous data. NoSQL systems were designed to overcome the lack of parallelism, I/O bottlenecks, and limited horizontal scalability of traditional RDBMSs. They provide flexible schema definitions that make them suitable for managing semi-structured and unstructured data originating from diverse sources and formats. NoSQL databases can be broadly categorised into four main types, according to their underlying data model: (*i*) key–value stores (e.g., Redis [108], Oracle NoSQL [93]); (*ii*) column-family databases (e.g., Cassandra [133]); (*iii*) document-oriented databases (e.g., MongoDB [83], Couchbase [22]); and (*iv*) graph databases (e.g., Neo4j [88]). These systems are typically fault-tolerant, horizontally scalable, and capable of supporting high update rates.

Among the various spatial RDBMSs discussed above, PostgreSQL and its spatial extension PostGIS are of particular relevance to this thesis. Their open-source nature, standards compliance, and extensibility make them highly suitable for spatial and spatiotemporal data management and analysis. An overview of the architecture and spatial capabilities of PostgreSQL and PostGIS is provided below.

PostgreSQL. PostgreSQL [52] is an open-source, object-relational Database Management System (DBMS) derived from the original Postgres project. It was developed at the Computer Science Department of the University of California, Berkeley, in 1996. PostgreSQL extends the traditional relational model by supporting advanced data types, user-defined functions, and custom operators, providing a flexible framework for managing complex data. It is fully ACID-compliant and conforms closely to the SQL standard while offering additional functionalities such as table inheritance, procedural languages, and robust transaction management. One of its most distinctive features is its extensibility: users can define new data types, or index methods, making it particularly suitable for domain-specific applications. Due to its reliability and scalability, PostgreSQL has become a reference platform for both research and production environments, as well as the foundation for several database extensions, including spatial and temporal data management systems.

PostGIS. PostGIS [51], released in 2001, is an open-source spatial extension of the PostgreSQL DBMS that adds support for a wide range of geographic ob-

jects. It enables the storage, processing, and analysis of Geographical Information System (GIS) data directly within the database by defining spatial data types, functions, and operators. Through this extension, GIS objects such as `Point`, `LineString`, and `Polygon` can be stored and manipulated, as well as composite geometries including `MultiPoint`, `MultiLineString`, `MultiPolygon`, and `GeometryCollection`. PostGIS extends the SQL language with a rich set of spatial functions for geometric and geographic computations (such as `ST_Distance`, `ST_Area`, `ST_Buffer`), and functions for evaluating spatial relationships. It supports efficient spatial indexing through Generalised Search Tree (GiST) [65], Space-Partioned Generalised Search Tree (SP-GiST) [5], and Block Range Index (BRIN) index structures, which significantly optimise spatial queries. The extension also includes raster data support and ensures compliance with the OGC Simple Feature Access for SQL (SFA-SQL) [92], and ISO SQL/MM Spatial [132] standards, thus guaranteeing interoperability with other GIS tools and software.

2.3 Moving Object Databases

Although spatial databases enable the representation and analysis of static spatial phenomena, many real-world applications require modelling entities that change their position over time. This need has led to the development of database systems capable of managing both spatial and temporal dimensions. Since the mid-1990s, extensive research has focused on *Moving Object Databases* (MODs), also referred to as *spatiotemporal databases*. These systems are designed to model, store, and query objects whose spatial properties evolve continuously over time, providing the computational foundation for managing dynamic spatial phenomena such as vehicle movements, environmental changes, or mobile sensor data. This requires substantial extensions to both the data model and the query language of traditional DBMSs. Moreover, the implementation of the database system itself must be enhanced at multiple levels, for instance, by introducing new data structures for representing moving objects, efficient algorithms for query processing, indexing and join techniques specifically tailored to spatiotemporal data, extensions to the query optimiser, and user interface components that enable the visualisation and animation of moving objects [109].

Research on moving objects has led to two distinct perspectives on the management and querying of databases involving entities that move through space and time. The first focuses on queries concerning the current positions of moving objects, and on their predicted temporal evolution in the (near) future. This approach, often referred to as *tracking*, aims to support real-time applications that monitor and predict object movements. For example, in an air traffic control application where each database object represents an aircraft and its position, a

query such as "retrieve all aircraft that will be within 30 miles of the airport in the next 10 minutes" can be directly evaluated using this model. To model moving object data in a way suitable for such queries, the *Moving Objects Spatio-Temporal* (MOST) model and the *Future Temporal Logic* (FTL) query language were proposed [107, 144, 145, 146]. Although this model provides mechanisms to represent moving objects and evaluate predictive queries, it only captures the current and near-future predicted positions of moving entities, typically represented as motion vectors. As a result, one of its main limitations concerns the frequency at which these vectors must be updated to minimise prediction errors. Furthermore, the model does not maintain complete movement histories, nor does it offer a comprehensive set of spatiotemporal data types and operations.

The second approach, in contrast, focuses on representing the complete histories of moving objects. In this model, the database stores and manipulates information describing an object's movement — particularly its position over time — thus enabling the reconstruction of its motion through space and the computation of its trajectory [31]. For this reason, this category of MODs is often referred to as *trajectory databases* [37, 46, 55]. The idea is to represent the temporal evolution of spatial entities through specific data types based on two fundamental abstractions: the *moving point* and the *moving region*. These abstractions describe objects whose position changes over time, and — in the case of moving regions — whose spatial extent also varies with time [55]. Time-dependent geometries are represented by dedicated data types equipped with associated operations, thus extending the abstract data type concept within the DBMS data model and query language. To achieve this goal, several key design principles must be satisfied:

- *Orthogonality* in the type system design, ensuring that constructors can be applied uniformly across types.
- *Genericity* and *consistency* of operations, meaning that operations should be polymorphic and semantically coherent.
- *Closure* and *consistency* between structure and operations of non-temporal and corresponding temporal types.
- *Abstract level of modelling*. Using this approach, both the structural definitions of entities and the semantics of operations can be specified at different levels of abstraction. The main advantage of the abstract level lies in its simplicity, as it avoids the need to express semantics in terms of finite representations. For example, the trajectory of a moving object can be described either as a curve or as a polygonal line in two-dimensional space.
- *Continuous functions*. Since the positional data of moving objects are typically obtained from modern localisation systems in discrete form, it becomes

necessary to employ continuous functions to express movement and trajectories in a continuous manner over time.

- *Lifting.* This mechanism is essential to ensure consistency between non-temporal and temporal data types. It allows functions and operations originally defined for static spatial objects to be *lifted* — that is, systematically extended — to their temporal counterparts, enabling coherent manipulation of time-dependent geometries.

Regarding moving object databases, several research prototypes have been developed over the years, including SECONDO [54], Hermes [99], UlTraMan [26], HadoopTrajectory [6], TrajSpark [149], and TrajMesa [72]. Among them, MobilityDB [152] represents an industrial-strength moving object database system. Tables 2.1 and 2.2, as presented by Alam et al. [4], provide a concise summary of the main characteristics of Hadoop-, Spark-, and NoSQL-based spatiotemporal systems. Below, we highlight the main features of the listed prototypes.

SECONDO is a DBMS prototype that has been developed since the mid-1990s at the University of Hagen. It is an open-source,[1] freely available system that runs on multiple platforms, including Windows, Linux, and macOS. Rather than relying on a fixed data model, SECONDO provides an extensible architecture that supports the implementation of new models. The system is organised into three main components that can operate either jointly or independently [53]: (*i*) the *kernel*, which supports query processing through a collection of implemented algebras, each defining specific type constructors and operators; (*ii*) the *optimiser*, which provides the core functionalities of a SQL-like query language; and (*iii*) the *graphical user interface*, which is designed to be extendable with new visualisation tools for additional data types and includes advanced viewers for spatial and spatiotemporal objects.

SECONDO supports a wide range of data types and operators for modelling and analysing spatiotemporal phenomena. In particular, it provides native support for moving points, as well as trajectories, moving lines, and moving regions. In this context, moving regions are defined as spatial objects that may translate or rotate over time, while preserving their shape — that is, they do not support topological deformation. The system also offers a comprehensive set of operators for spatiotemporal queries [119]. Over time, SECONDO has evolved into Parallel SECONDO [56] and Distributed SECONDO [89], designed to enhance scalability and performance in distributed environments. Parallel SECONDO extends the system to multi-core and cluster architectures by partitioning both spatial and temporal dimensions across nodes, using grid partitioning and R-tree indexing for efficient trajectory queries. Distributed SECONDO, instead, integrates with

[1] `https://secondo-database.github.io/`

Table 2.1: Hadoop-based and Spark-based Spatiotemporal Systems [4].

	System Type	Data Types	Partitioning	Indexing	Query Language	Supported Queries
Parallel SECONDO [56] (2015)	Spatiotemporal (Trajectory)	Point, LineString Region, Instant Period, Periods Interval	3D Grid	B-Tree R-Tree	SQL-Like Executable	Range, Join
HadoopTrajectory [6] (2019)	Spatiotemporal (Trajectory)	Point, Region, Instant, Interval, Periods, TrajSegment, Trajectory	N/A	Grid, R-Tree (3D Extension) e.g., 3DR-Tree	N/A	Range, kNN, Join
TrajSpark [149] (2017)	Spatiotemporal (Trajectory)	Point, Timestamp	Quad-Tree, KD-Tree	Local: Hash, Global: Multi-Level, L1: Temporal, L2: Spatial, L3: B^+-Tree	N/A	Single Object, Range, kNN
UlTraMan [26] (2018)	Spatiotemporal (Trajectory)	Point, Timestamp	STR	Two-Level (Global, Local) R-Tree	N/A	ID, Range, kNN

NoSQL backend (such as Cassandra) and employs a *main memory R-tree* (MMR-tree) for efficient in-memory indexing of spatial data in distributed environments.

Moreover, SECONDO includes a built-in benchmark for evaluating moving object database systems, known as BerlinMOD [31]. BerlinMOD provides a standardised framework for generating synthetic movement data and assessing system performance under controlled experimental conditions. It defines a set of benchmark queries covering various aspects of spatiotemporal query processing — such as range, nearest neighbour, and join operations — thus enabling consistent and reproducible performance comparisons among different MOD implementations.

Another system dealing with moving objects is Hermes [96, 97, 99]. Hermes was developed at the University of Piraeus as an extension of traditional object-relational DBMSs, aiming to provide native support for the storage, indexing, and querying of spatiotemporal data. Unlike SECONDO, which was designed as a research prototype with its own extensible algebraic framework, Hermes builds directly upon Oracle and PostGIS, and is therefore accessible through SQL. However, it does not exploit their native type systems. For instance, to define the moving point type, it relies on a custom-defined (x, y) pair rather than on the built-in point types of PostGIS or Oracle. As a result, the functions provided by the underlying DBMS cannot be reused. In addition to moving point, Hermes introduces abstract data types such as trajectory, together with indexing mechanisms like the TB-tree [103], to efficiently handle queries involving movement and time. The TB-tree supports both standard spatiotemporal operations and more complex ones, such as k-nearest neighbour (kNN) and trajectory similarity queries.

Beyond SECONDO and Hermes, several more recent systems have been pro-

Table 2.2: NoSQL-based Spatio-temporal Systems [4].

	System Type	Underlying NoSQL System	Data Types	Partitioning	Indexing	Query Language	Supported Queries
Distributed SECONDO [89] (2015)	Spatiotemporal (Trajectory)	Cassandra	Point, LineString Region, Instant Period, Periods Interval	3D Grid	R-Tree (MMR-tree)	SQL-Like Executable	Join
TrajMesa [72] (2020)	Spatiotemporal (Trajectory)	GeoMesa	Point, Timestamp	N/A	XZT and XZ2$^+$	SQL-Like	ID-Temporal Range, kNN, Similarity

posed to address large-scale trajectory data management and analytics, primarily focusing on distributed and cloud environments. UlTraMan [26] is a unified framework built on top of Apache Spark, designed to support distributed storage, parallel processing, and advanced analytics over massive trajectory datasets. It provides abstractions for trajectory representation and implements a variety of operations, including map-matching, trajectory similarity, and clustering, by leveraging Spark's in-memory computation model. Moreover, it adopts the STR [71] partitioning strategy to create balanced partitions of trajectory points. TrajSpark [149] also builds upon Apache Spark and adopts a multi-level indexing strategy that combines temporal and spatial partitioning with B$^+$-tree indexing. It supports single-object, range, and kNN queries efficiently, demonstrating improved scalability for trajectory analytics.

HadoopTrajectory [6] extends the Hadoop ecosystem to enable efficient processing of spatiotemporal data through spatial partitioning and MapReduce-based parallelism. It enhances the Hadoop framework with spatiotemporal data types and a 3D R-tree indexing infrastructure to improve the efficiency of large-scale trajectory analytics. Finally, TrajMesa [72] is a distributed NoSQL storage engine designed specifically for big trajectory data based on GeoMesa. It incorporates a module for trajectory preprocessing containing functions for noise filtering, segmentation, stay point detection, map matching, and other statistical analysis. Moreover, TrajMesa employs optimised spatio-temporal indexing mechanisms from GeoMesa to support efficient SQL-like trajectory queries.

For a more comprehensive discussion and an extended review of existing database solutions, the reader is referred to [4].

2.4 MobilityDB

MobilityDB [152, 153] has emerged as a robust moving object database system built on top of PostgreSQL and PostGIS, providing native support for temporal and spatiotemporal types, as well as a rich set of operations for querying and

analysing moving objects. It extends their type system with Abstract Data Type (ADT) for representing moving objects, while fully integrating with the existing indexing, optimisation, and query processing infrastructure. It is compliant with the OGC standards on moving features, and supports a wide range of spatiotemporal operations through standard SQL. Its type system is based on the concept of type constructors making the system inherently extensible and well suited for additional future implementations.

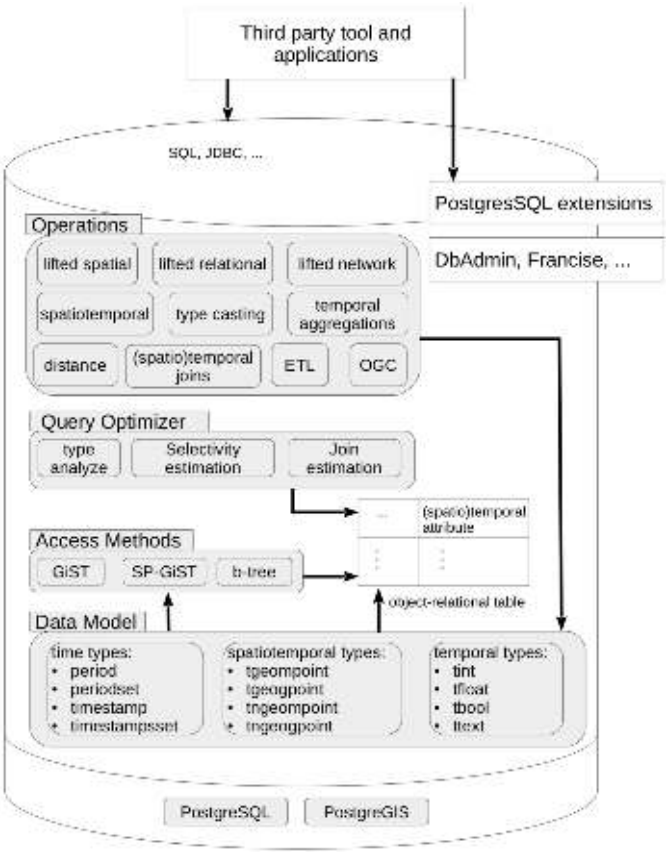


Figure 2.5: MobilityDB architecture [7].

As illustrated in Figure 2.5, MobilityDB leverages the core functionalities of both PostgreSQL and PostGIS. The same design principle applies to the implementation of operations, with the goal of maximising compatibility between MobilityDB and its underlying platform. In addition to the primitive types provided by PostgreSQL and PostGIS, MobilityDB introduces specific temporal, spatiotemporal, and temporalised spatial types. For these temporal types, three access methods based on indexing are available: GiST, SP-GiST, and B-tree. The first two are implemented to efficiently support spatiotemporal, spatial only, and temporal only

queries, whereas the B-tree index is also extended to support equality searches. The query optimiser of MobilityDB collects statistics for temporal attributes, to use them in estimating the selectivity of the different query predicates, and selecting the optimal execution plan. The type analyser is the function that collects the statistics.

Beyond indexing and query optimisation, the internal representation of temporal data in MobilityDB also accounts for how values evolve over time. Since input data are typically discrete (e.g., GPS observations), continuous temporal types rely on interpolation functions between consecutive instants. MobilityDB supports two interpolation modes: *step* and *linear*. In step interpolation the temporal value remains constant between timestamps. In linear interpolation the value evolves in a continuous linear manner.

2.4.1 Operations on Temporal Types

A temporal type in MobilityDB is defined by combining a *base type* with a *time type*. The base types are, for instance, `bool`, `int`, `float`, `text`, `geometry(point)`, and `geography(point)`, where the latter denotes geometries/geographies restricted to 2D or 3D points. Recall that in PostGIS, the `geometry` and `geography` types store spatial values using, respectively, planar and spherical coordinate systems. The time type, on the other hand, can be *timestamp*, *timestamp set*, *time span*, or *span set*. In particular, a *timestamp* (`timestamptz`) represents a single instant in time, whereas a *timestamp set* (`tstzset`) is a finite collection of discrete instants. A *time span* (`tstzspan`) denotes a continuous interval between two instants, and a *span set* (`tstzspanset`) is a collection of such intervals. The following examples illustrate these temporal types.

```
SELECT timestamptz '2001-01-01';
--  2001-01-01 00:00:00+01
SELECT tstzset '{2001-01-01, 2001-01-03, 2001-01-05}';
-- {"2001-01-01 00:00:00+01", "2001-01-03 00:00:00+01", "2001-01-05
  00:00:00+01"}
SELECT tstzspan '[2001-01-01, 2001-01-02]';
-- [2001-01-01 00:00:00+01, 2001-01-02 00:00:00+01]
SELECT tstzspanset '{[2001-01-01, 2001-01-03], [2001-01-05,
 2001-01-09)}';
-- {[2001-01-01 00:00:00+01, 2001-01-03 00:00:00+01], [2001-01-05
  00:00:00+01, 2001-01-09 00:00:00+01)}
```

MobilityDB defines more than 2,300 query operations on temporal types. These operations are *polymorphic*, in the sense that their arguments may be of different types and the return type is determined by the types of the arguments. When an operation involves more than one temporal value as an argument, the result

is defined only over the intersection of their temporal extents. If the temporal periods of the arguments are disjoint, the result is null.

A major class of temporal operations is obtained by *lifting* the operations on non-temporal types, to also accept temporal types. This concept has been proposed by Güting et al. [55]. The lifting transformation can be illustrated as:

$$(LIFT(op)(\alpha, \beta))(t) = op((\alpha(t), \beta(t))$$

Here op denotes a static operation, $LIFT(op)$ its lifted version, and α, β are temporal arguments. Since lifted operators take temporal inputs, their output is likewise a temporal type. The notation $\alpha(t)$ denotes the temporal functions of α, i.e., the value of α at time t. For example, arithmetic binary operators (`+`, `-`, `*`, `/`) can be lifted whenever one or both arguments are temporal numbers, yielding a temporal number as a result. The implementation of lifting in the sequence model requires two operations: (i) synchronisation; and (ii) computation of the function's turning point [120].

To explain synchronisation, consider the lifted temporal sum of two integers. Recall that, since these are temporal integers, their interpolation can only be stepwise.

```
SELECT tint '{[1@2001-01-01, 2@2001-01-03, 1@2001-01-06,
  1@2001-01-07)}' + tint '{[3@2001-01-02, 3@2001-01-04],
  [3@2001-01-06, 3@2001-01-08)}';
-- {[4@2001-01-02, 5@2001-01-03, 5@2001-01-04], [4@2001-01-06, 4@2001
  -01-07)}
```

As illustrated in Figure 2.6, the result of the operation is defined only over the time intervals where both operands are defined. Between instants 1 and 2 and between instants 4 and 6, only the left operand is defined; therefore, no result exists in these intervals.

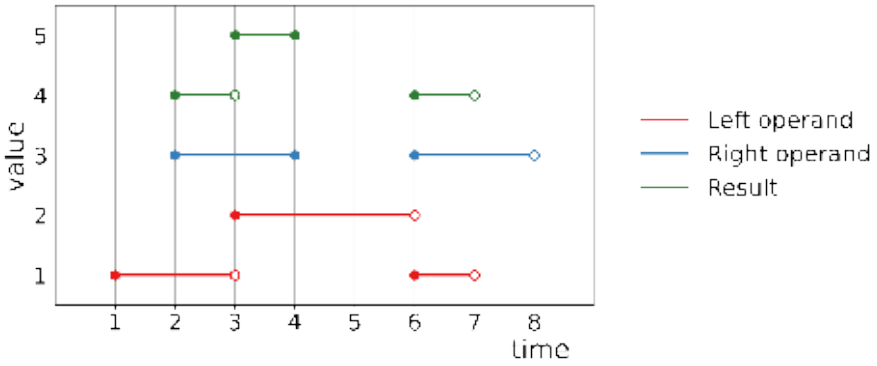


Figure 2.6: Addition of two temporal integers.

The second key ingredient in lifting operations is the computation of turning points. This is required when the result is not a linear function. Consider two temporal float values (`tfloat`) with linear interpolation. When multiplying them, the resulting product is a quadratic function of time, which must be approximated by a linear one. This approximation should preserve the local maxima and minima of the quadratic function, i.e., its turning points. If these turning points are not preserved, the lifted result is erroneous. Figure 2.7 illustrates this with the following query.

```
SELECT tfloat '{[1@2001-01-01, 3@2001-01-07]}' *
  tfloat '{[3@2001-01-01, 1@2001-01-07]}';
```

Both operands are defined over the same time span (from January 1 2001 to January 7 2001). If one ignored the turning point, the result would be evaluated only at the endpoints — yielding a constant value of 3 over the whole interval (the purple line in Figure 2.7):

```
-- {[3@2001-01-01, 3@2001-01-07]}
```

This is incorrect because on January 4 2001 both `tfloat` values equal 2, and therefore $2 \times 2 = 4$. Preserving the turning point leads to the following result (the green line in Figure 2.7):

```
--  {[3@2001-01-01, 4@2001-01-04, 3@2001-01-07]}
```

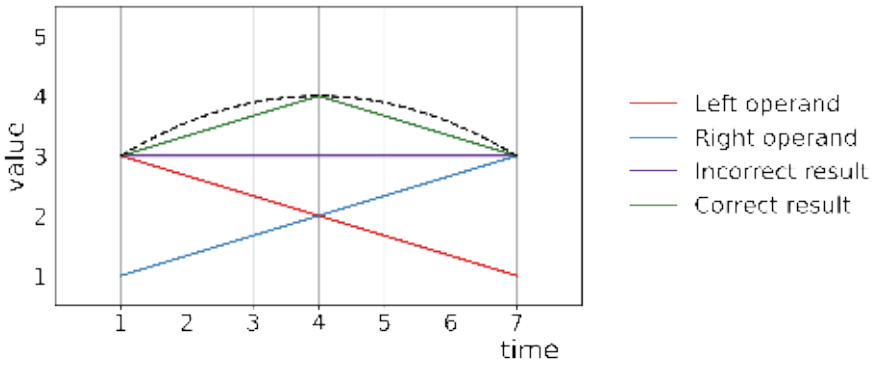


Figure 2.7: Multiplication of two temporal floats showing the actual result (dashed line), the approximation when computing the turning point (green line), and the result without computing the turning point (purple line) [120].

A wide range of temporal functions in MobilityDB rely on the lifting concept. For instance, the temporal distance function (implemented through the operator `<->`) computes the distance between two moving objects over time, returning a

temporal float that represents the distance at each instant. Similarly, the operator `|=|` returns the closest distance ever between two moving objects, i.e., the minimum value of the temporal distance function.

MobilityDB also supports several temporal predicates, such as `tIntersects` and `tContains`, which return a temporal boolean type (`tbool`). These predicates describe the time intervals during which a temporal object satisfies a given spatial or temporal relationship — for example, when one moving object intersects or contains another. In addition, MobilityDB defines *ever* and *always* comparison operators, such as `eIntersects` and `aContains`. These functions evaluate whether a temporal geometry and a static geometry (or another temporal geometry) ever or always satisfy a given spatial predicate — for example, whether two objects ever intersect or always remain within each other's interior.

Another class of operations comprises those that are meaningful only for temporal types. An example is the function `trajectory`, which takes a `tgeompoint` as input and returns a `geometry(LineString)` representing the overall path of the moving object. This function is particularly useful for studying the trajectories of moving entities, enabling spatial analysis on their complete paths — for instance, to compute intersections, overlaps, or spatial extents of movement.

Other functions of this kind include `length(tgeompoint)`, which returns the total distance travelled by a moving object as a `float`, and aggregation functions such as `min/max(tfloat)`, which return the minimum and maximum values reached by a temporal float over its lifetime. For a comprehensive list of temporal operators and predicates implemented in MobilityDB, the reader is referred to the MobilityDB documentation [151].

After introducing the main temporal functions and the lifting mechanism, it is also important to clarify the structure of the type system on which MobilityDB is built. Formally, the type system defines four type constructors that correspond to the four time types: *INSTANT*, *INSTANTs*, *SEQUENCE*, and *SEQUENCEs*. The notation D_S is used to denote the domain of some type S. We assume that the null value is not included in D_S since null values are not allowed in any part of the representation of temporal types. Let $\mathcal{B}$ be the set of base types, i.e., $\mathcal{B} = \{$`bool`, `int`, `float`, `text`, `geometry(point)`, $\ldots\}$, and given a base type $S \in \mathcal{B}$, e.g., `geometry(point)`, the *INSTANT* type constructor constructs a temporal type using S and *timestamptz* as follows [152]:

$$D_{\text{INSTANT}(S)} = D_{metavalue(S)} \times D_{timestamptz}$$

Here $metavalue(S)$ corresponds to the base type, either saved as-is or as a delta type, that is $metavalue(S)$ could be D_S or another transformation.

To represent the continuous evolution of temporal types, we define the *SEQUENCE* type constructor. Given a base type S, the domain of *SEQUENCE(S)*

is defined as follows [152]:

$$
\begin{aligned}
D_{\text{SEQUENCE}(S)} = \{(I, li, ui, interpolation) \mid \\
&(i)\ I = [(v_1, t_1), \ldots, (v_n, t_n)] \text{ is a non-empty list of} \\
&\quad INSTANT(S) \\
&(ii)\ li, ui \in \{true, false\} \\
&(iii)\ interpolation \in \{step, linear\} \\
&(iv)\ \text{consecutive pieces of the interpolation function are} \\
&\quad \text{not collinear}\}
\end{aligned}
$$

It represents the evolution of a value over a given time period or span, which may be open or closed at either endpoint. This evolution can be stepwise or continuous, depending on the chosen interpolation method.

Finally, the *INSTANTs* and *SEQUENCEs* constructors define sets of *INSTANT* and *SEQUENCE* types respectively, each with specific constraints on valid values. In practice, the *INSTANTs* and *SEQUENCE* constructors are merged into a single *SEQUENCE* constructor (implemented as an SQL function). Figure 2.8 illustrates the types that are constructed by the four type constructors, where `v@t` denotes a value `v` of the base type occurring at the time instant `t`.

To define a new temporal type in this type system, it is thus sufficient to define the domain of its metavalue $D_{metavalue(S)}$. The type constructors then automatically generate the corresponding temporal types. In MobilityDB, this design significantly reduces the development effort, as new temporal types can be introduced with minimal overhead, allowing developers to focus primarily on implementing data management and query functions for these types.

2.4.2 Distributed MobilityDB

While MobilityDB extends PostgreSQL and PostGIS to manage and query moving objects, its scalability is inherently limited by the underlying single-node

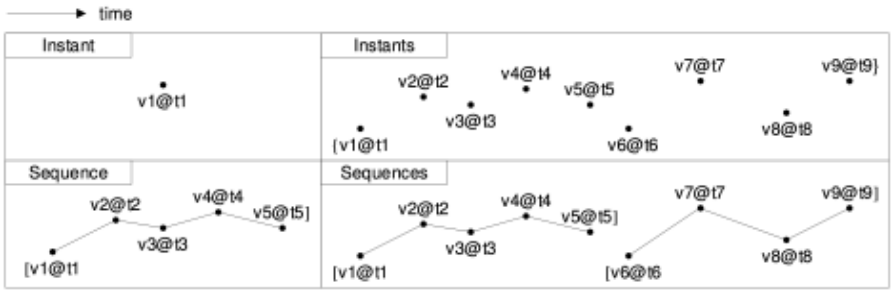


Figure 2.8: Sequence representation of moving objects [7].

architecture of PostgreSQL. To overcome this limitation and enable distributed spatiotemporal data processing, MobilityDB can be integrated with Citus [24], a PostgreSQL extension that provides horizontal scalability and distributed query execution. This section first introduces the main features of Citus and its distribution model, and then describes how these concepts are leveraged within distributed MobilityDB to support large-scale moving object data management.

Citus

Citus [19] is an extension of PostgreSQL designed to ease horizontal scaling, making it suitable for handling large datasets across multiple machines. It distributes both data and queries across a cluster, allowing users to leverage the power of a distributed system while maintaining compatibility with existing PostgreSQL tools. By using sharding and replication Citus scales PostgreSQL across several servers. *Sharding* is a method employed in distributed systems to divide data horizontally across multiple servers or nodes. It involves splitting a large dataset into smaller, more manageable pieces known as *shards*. Each shard holds a portion of the data, and, collectively, they represent the entire dataset. As shown in Figure 2.9, the architecture consists of a coordinator node and several worker nodes (three in this example). Data are stored in shards, which correspond to regular PostgreSQL tables and can therefore benefit from existing features such as indexes, constraints, and other relational mechanisms. Shards can be co-located within shard groups, so that joins and foreign-key relationships involving the shard key can be executed locally without cross-node communication. This distributed design enables Citus to exploit the aggregate memory, bandwidth, storage, and CPU resources of all participating nodes, making it possible to fit larger datasets into memory through horizontal scaling.

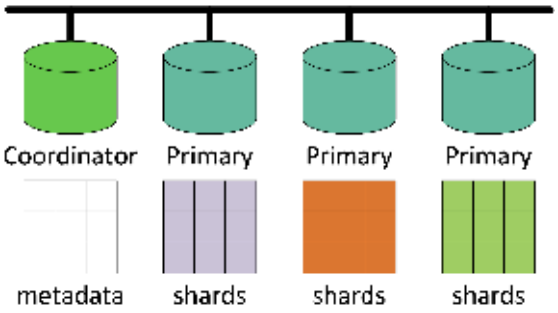


Figure 2.9: Transparent-sharding architecture [120].

Large tables can be partitioned across worker nodes by specifying a sharding key when the table is created. Tuples are routed by the coordinator to the appropriate workers, where the data are physically stored. The coordinator node

maintains lightweight metadata describing the sharding structure, enabling it to distribute queries efficiently, while the actual data storage and most query execution occur on the worker nodes. Such tables are referred to as *distributed tables*. Conversely, *reference tables* that are replicated across servers, are needed for (local) joins and foreign keys with distributed tables on any attribute. Furthermore, shards may themselves be replicated across workers to enhance fault tolerance and preserve data availability in the event of node failures.

All queries issued to the distributed cluster are executed through the coordinator, which generates a distributed query plan. The original user query is partitioned into smaller subqueries that can be executed independently on the corresponding shards. The coordinator then schedules and monitors the execution on the worker nodes, reassigning tasks to replica nodes in the event of a failure. Finally, the partial results are gathered and aggregated by the coordinator before being returned to the client.

Extending MobilityDB with Citus

The management and analysis of large-scale trajectory datasets — such as vessel movements, vehicular traces, or mobile sensor data — require database systems that can efficiently handle high data volumes and intensive query workloads. While MobilityDB provides comprehensive support for spatiotemporal data types and queries, its architecture is limited by the single-node nature of PostgreSQL, which constrains both storage capacity and computational scalability.

To overcome these limitations, Bakli et al. [7, 8] investigated the use of Citus to deploy MobilityDB in a distributed environment. In the proposed architecture, Citus is responsible for the physical distribution of data, partitioning temporal and spatiotemporal tables into shards stored across different worker nodes. MobilityDB, running on each node, maintains its full query semantics and type system, ensuring that temporal and spatial operations can still be executed efficiently. Figure 2.10 illustrates the overall architecture of MobilityDB deployed over Citus, highlighting how the two systems interact to enable distributed spatiotemporal data processing.

An important aspect of the integration between MobilityDB and Citus lies in how the system handles custom spatiotemporal data types and query execution. During the sharding phase, Citus treats MobilityDB's custom data types — such as `tgeompoint` and `tgeogpoint` — as binary objects. This design ensures native compatibility without requiring Citus to understand the internal semantics of these types. All queries are sent to the coordinator node, whose distributed planner decomposes each query into smaller fragments that are executed in parallel on the worker nodes. Once execution is completed, the coordinator aggregates the partial results and returns the final output to the user.

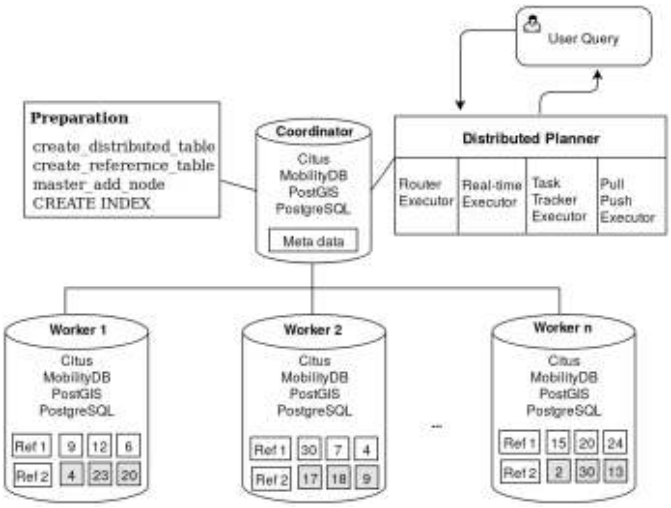


Figure 2.10: Distributed MobilityDB cluster architecture [7].

Citus's ability to distribute MobilityDB queries depends on the nature of the query itself, as its planner has no explicit knowledge of spatiotemporal semantics. Several categories of operations are natively supported by this integration. These include queries that contain a filter on the shard key (routable queries), spatiotemporal joins involving replicated reference tables, local access to GiST and SP-GiST indexes created on each shard, and functions that operate on individual temporal attributes. Such operations can be pushed down to the worker nodes, which execute them locally and in parallel, thereby enabling efficient distributed computation without requiring additional coordination logic at the Citus level.

Despite these advantages, two important classes of operations remain unsupported in the current integration: non–co-located spatiotemporal joins and temporal aggregations. The first category includes joins that compare data across shards, such as self-joins on trajectories to identify objects that have approached each other in space and time. Since Citus supports only equality-based joins, these operations cannot be efficiently distributed, as this would require costly cross-node data movement. Similarly, temporal aggregation functions provided by MobilityDB cannot be executed in parallel.

Experimental evaluations conducted on two clusters of four and twenty-eight nodes using the BerlinMOD benchmark demonstrated substantial performance improvements for supported query classes. Out of seventeen benchmark queries, thirteen were successfully distributed, while the remaining four involved complex non–co-located joins. Execution times were significantly reduced when moving from single-node to multi-node configurations, showing that the integration effectively alleviates I/O bottlenecks and scales with dataset size. Overall, these re-

sults confirm the potential of combining MobilityDB with Citus to achieve scalable spatiotemporal data management, while also highlighting the need for enhanced distributed query-planning mechanisms to support more complex spatiotemporal operations.

Chapter 3

Reconstruction and Semantic Enrichment of AIS Data

The study of mobility patterns relies on the ability to capture, model, and interpret the movement of entities across space and time. Mobility data are increasingly available from a variety of sources — such as satellite navigation systems, mobile devices, sensor networks, or vessel tracking technologies — providing large volumes of spatiotemporal information that describe how people, vehicles, and objects move within their environment. These data are typically collected in raw form, as discrete sequences of timestamped positions, and therefore lack the semantic information necessary to understand the meaning and purpose of the observed movements. Raw trajectories offer only a geometric and temporal depiction of motion. They describe *where* and *when* an entity has moved, but not *why* it moved, *what* it was doing, or *under which* circumstances. To derive meaningful knowledge from such data, it is therefore essential to move from the geometric to the semantic level, enriching positional information with contextual attributes that reveal the underlying behaviour and interactions with the surrounding space.

This transformation is achieved through two complementary processes: trajectory reconstruction and semantic enrichment. The first involves cleaning, filtering, and interpolating raw spatiotemporal samples to reconstruct the continuity of motion and to cope with the irregularities and noise inherent in data collection. The second associates the reconstructed trajectories with semantic information, such as the type of moving object, its activity or operational state, and its spatial or environmental context. Together, these processes allow for the transition from simple traces to semantic trajectories, enabling more expressive analyses of mobility. Such an approach can be applied across domains: to maritime navigation through AIS data, to urban transport via GPS traces, or to human mobility captured by smartphones and sensors.

In this chapter, particular attention is given to the maritime domain, where

the reconstruction and enrichment of vessel trajectories provide a means to better understand navigation patterns and human activities at sea. The Automatic Identification System (AIS) data provides an extensive source of spatiotemporal information from which semantic trajectories can be derived. While the underlying concepts and methodological framework are general and applicable to various forms of mobility data, the specific implementations and functions developed in this work are tailored to the characteristics of the AIS dataset and the maritime context.

The Northern and Central Adriatic Sea is one of the most intensively exploited areas of the Mediterranean, characterised by diverse maritime activities such as fishing, transport, tourism, and research, and resulting in increasing pressure on marine ecosystems and species. Understanding and representing the main factors driving such intense vessel activity is of paramount importance both for ecologists and for local policymakers. Indeed, this knowledge can support the development of effective management strategies aimed at ensuring the sustainable use of marine space and resources, while preserving the ecological balance and overall health of the marine ecosystem.

In this setting, starting from AIS data describing vessel movements, we reconstruct their trajectories and enrich them with semantic information. Our work relies on multiple complementary data sources, including AIS positional data, vessel-related information (e.g., type, length), harbours area geometries, environmental features, and a dataset describing the measurements of underwater noise. These heterogeneous sources are integrated to enhance the reconstructed trajectories with various semantic attributes, such as the type of activity performed by each vessel and its speed. We also provide an implementation of our spatiotemporal database using MobilityDB [152]. On the one hand, the implementation in MobilityDB allows us to perform various analyses on the dataset and to assess the effectiveness of the conceptual framework proposed. On the other hand, it highlights the potential of the system for the management and analysis of trajectories enriched with semantic information.

This chapter is structured as follows. Section 3.1 presents related works on the integration of vessel movement data with contextual and semantic information in the maritime domain. Section 3.2 describes the datasets used for the reconstruction and enrichment of the trajectories. Finally, Section 3.3 details the implementation of the proposed methodology in MobilityDB.

3.1 Related Work

Handling the fusion of ship movements with contextual and semantic information in the maritime domain is a recognised challenge [20, 102]. Several strategies

were proposed to properly deal with the fusion of heterogeneous ocean data. For example, the works in [28, 128] show a platform in the maritime vessel traffic domain for discovering real-time traffic alerts by querying and reasoning across numerous streams (e.g., AIS, weather, ice, etc.). The authors use semantic web technologies to integrate heterogeneous data sources. In [13], the authors propose a model for the integration and analysis of data for vessel movement in a real-time maritime situation awareness system, also using semantic web techniques and tools. They introduce an ontology to model the maritime domain and to provide a common view on the different data sources. In particular, they define a movement ontology in which each position of a trajectory is modelled as being a move or a stop and they enrich trajectories also with information coming from the linked open data cloud, such as GeoNames[1] or DBPedia[2] or OpenStreetMap.[3] The queries are posed by using SPARQL, and the ontology-based data access system *Ontop* [15] and its extension *Ontop-spatial* [9] are employed to map the relational data to the ontology and to translate queries to SQL queries. This approach has been proved worthwhile to detect routine traffic of vessels and abnormal vessel behaviour.

In [141] a *Semantic Model of Ship Behaviour* (SMSB) is proposed to represent and reason on the meaning of the behaviours. Their steps include building a semantic network based on maritime traffic rules and detection methods to identify basic ship behaviours in various maritime scenes (e.g., dock, anchorage, traffic lane, etc.), and a *Dynamic Bayesian Network* (DBN) to reason about potential ship behaviours. Their results show that basic behaviours and potential behaviours in all typical scenes of any harbour can be obtained accurately and expressed conveniently using SMSB. In [122] a framework named SPARTAN is presented. SPARTAN allows for real-time semantic integration of big mobility data with other data sources, aiming at providing enriched trajectories which are exploited by higher-level analysis tasks. The design and implementation of SPARTAN use well-known big data technologies (Apache Flink and Kafka), and their experimental evaluation shows the efficiency and scalability of the framework using maritime and aviation data.

All aforementioned works use specific semantic models that are focused on the data integration component for detecting anomalies [13, 128], discovering spatio-temporal links between entities [122], or finding particular behaviours at harbours [141]. Other more general semantic models are stops and moves [95], CONSTANT [10] and MASTER [79]. It is important to highlight that our modelling represents an instance of the MASTER framework. Accordingly, we followed

[1] `http://geonames.org`
[2] `http://dbpedia.org`
[3] `http://openstreetmap.org`

its design recommendations and developed a concrete implementation tailored to a specific objective: integrating AIS and environmental variables to represent and analyse vessel activities in the Northern and Central Adriatic basin.

3.2 Data Sources

In this section, we present the data sources used in this study. Specifically, six datasets are employed: AIS data, information on vessel characteristics, harbours areas, a spatiotemporal grid of the Northern and Central Adriatic Sea, environmental data, and hydrophone data from the SOUNDSCAPE project [40].

3.2.1 Automatic Identification System (AIS) Data

The Automatic Identification System (AIS) was originally developed to enhance navigation safety by preventing vessel collisions. The International Maritime Organisation[4] (IMO) requires AIS transmission for ships over 300 gross tons, all passenger vessels, and boats longer than 15 metres. Specifically, AIS data include several key components: the vessel's name; the Maritime Mobile Service Identity (MMSI), which uniquely identifies the vessel; its International Maritime Organization (IMO) number; the vessel's call sign; its position, given by latitude and longitude coordinates (in decimal degrees); the timestamp of the transmission (in UTC); speed over ground (SOG); course over ground (COG); heading; length overall (LOA); and the vessel type (e.g., cargo, sailing boat, fishing vessel). In some cases, AIS data may also provide details such as the starting port, destination, and estimated time of arrival.

In this thesis, we rely on two different AIS datasets. The first dataset contains AIS records for fishing vessels only, operating in the Northern Adriatic Sea during 2020, and is used in Chapter 4. The second dataset includes AIS data for all vessel categories operating in the Northern and Central Adriatic Sea during the same year, and is employed for the analysis presented in Chapter 5. The use of two different datasets is motivated by the fact that the Italian Coast Guard provided the data for all vessel categories only at a later stage. The two datasets, that are not public available, are described in detail below.

AIS Data for Fishing Vessels

In Chapter 4 we work on a dataset provided by the Italian Coast Guard, consisting of terrestrial AIS data for trawl fishing vessels operating in the Northern Adriatic Sea from January 2020 to December 2020. It is worth noting that, the year 2020

[4] `https://www.imo.org/`

Period	No. Vessels	No. AIS records
Year 2020	714	92,916,965
April 2020	548	5,392,677
June 2020	642	9,841,079

Table 3.1: No. of vessels and AIS records for the year 2020, April, and June 2020.

represents a period of restricted shipping activity due to the COVID-19 pandemic outbreak. In Italy, a lockdown was imposed from March 9 to May 18 2020, during which many activities, including restaurants, were either closed or had to limit their operations. A second lockdown period began on November 6 2020, when Italy adopted a new set of restrictions that remained in effect throughout December 2020.

For the AIS data of fishing vessels, we focus on April and June 2020, to investigate the differences in environmental pollution during a period when all activities were significantly reduced, compared to a period of regular maritime operations. Table 3.1 reports the number of vessels and AIS data records for the year 2020, as well as for April and June 2020.

AIS Data for All Vessel Types

In Chapter 5 we use a dataset provided by the Italian Coast Guard, consisting of terrestrial AIS data for all types of vessels operating in the Northern and Central Adriatic Sea from January 2020 to December 2020. We applied a data cleaning procedure to remove duplicated, incomplete, and out of range data. Specifically, duplicated AIS entries (identical position and timestamp) were eliminated, as they clearly represent transmission errors. As the estimation of underwater noise requires both the length overall (LOA) and vessel type, AIS records transmitted by vessels lacking either attribute were excluded. Only AIS records located within the Northern and Central Adriatic Sea were retained; erroneous terrestrial points, as well as those occurring within rivers or other inland waterways, were removed. For vessel types other than tankers, container ships, cruise and passenger ships, AIS points within ports with recorded speeds below 0.5 knots were removed, as they likely correspond to stationary conditions after berthing, when vessels are moored but still transmitting AIS signals. Conversely, for tankers, container ships, cruise and passenger vessels (i.e., those with AIS ship type identifiers between 60 and 89), such points were retained because these ships generally do not switch off their engines after being berthed; more precisely, their main propulsion systems are inactive, but auxiliary engines remain operational to supply electrical power and other essential onboard services. After data cleaning, approximately 32.5% of

	No. Vessels		No. AIS records	
Period	**Raw**	**Cleaned**	**Raw**	**Cleaned**
January	2,139	1,893	25,707,105	20,403,601
February	2,297	2,034	25,045,757	19,607,111
March	2,190	1,941	21,437,011	15,387,298
April	1,999	1,703	21,322,047	13,161,003
May	2,460	2,171	24,745,515	15,619,650
June	3,272	3,024	31,838,085	20,803,598
July	3,718	3,537	34,500,114	21,788,265
August	3,567	3,365	20,709,501	13,168,429
September	3,423	3,188	19,785,086	12,451,660
October	3,036	2,730	18,436,188	12,014,857
November	2,547	2,213	17,970,259	12,071,466
December	2,325	1,933	16,834,648	11,327,564
Year 2020	7,356	7,122	278,331,316	187,804,502

Table 3.2: No. of vessels and AIS records (raw and cleaned) for the year 2020.

the records for the year 2020 were removed. Table 3.2 summarises the number of vessels and AIS records (both raw and cleaned) for the year 2020.

Table 3.3 provides a detailed overview of the vessel types present in the dataset, including the number of unique vessels, the total number of AIS records, and the average length overall (LOA) for each category. Recreational vessels represent the largest share of the fleet, with about 2,700 vessels (38.2% of the total), followed by

Type of vessel	**AIS Ship type ID**	**No. Vessels**	**No. AIS records**	**Mean LOA**
Fishing	30	754	83,930,247	21.64
Tug	31, 32, 52	126	12,434,108	36.13
Naval	35	36	80,903	40.85
Recreational	36, 37	2,721	13,902,561	23.38
Government/Research	51, 53, 55	71	430,002	22.69
Cruise	60-69 (LOA $>$ 100 m)	62	3,562,093	188.08
Passenger	60-69 (LOA $\leq$ 100 m)	222	9,989,440	52.34
Cargo	70-79	1,916	36,940,168	136.88
Tanker	80-89	887	14,132,712	160.54
Dredger	33	11	546,814	39.84
Other	all other type IDs	316	11,855,454	40.48

Table 3.3: Vessel types in the cleaned dataset with the number of vessels, AIS records, and mean LOA (in metres) for 2020.

cargo with 1,916 units (26.9%), and tankers with 887 (12.5%). In terms of AIS data volume, however, fishing vessels overwhelmingly dominate the dataset, accounting for nearly 84 million AIS records (44.7% of all transmissions) — well above any other category. Cargo vessels follow with nearly 37 million records (19.7%), while recreational vessels contribute nearly 14 million (7.4%). This contrast suggests significant differences in operational patterns and AIS reporting intensity among vessel types.

Finally, Figure 3.1 shows the distribution of AIS records by vessel type across the four seasons of 2020 (winter: January–March; spring: April–June; summer: July–September; and autumn: October–December). Fishing vessels consistently dominate AIS transmissions across all seasons, confirming their intensive and regular activity at sea. Nonetheless, their AIS volume gradually decreases from winter to autumn. This trend could be attributed to COVID-related restrictions and seasonal variability in fishing activity. Cargo, tanker, and tug vessels also display a similar downward trend, reflecting a general decline in maritime activity towards the end of the year. Conversely, passenger and cruise vessels display clear seasonal variability, with a pronounced increase during the summer months, corresponding to the peak of passenger and tourism-related traffic. Recreational vessels show an even more evident seasonal pattern, with intense activity in summer being about three times higher than in spring and more than four times higher than in autumn, followed by a further decline in winter. The inset highlights minor vessel categories — such as dredgers, government/research vessels, and naval units — that, despite contributing only a small portion of the total AIS data, still show noticeable sea-

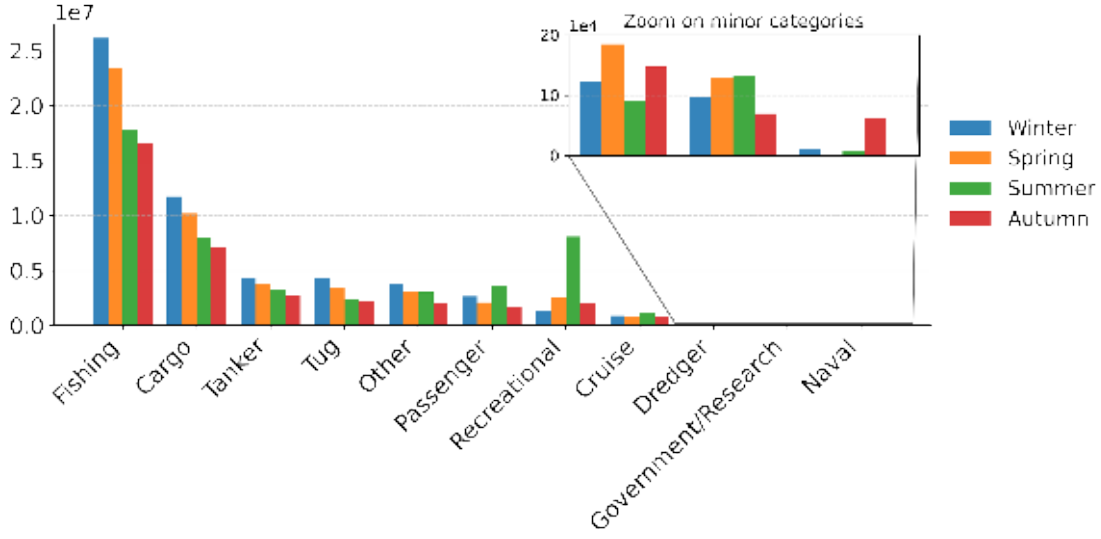


Figure 3.1: AIS data per vessel category across the four seasons of 2020.

sonal variability due to their occasional or mission-oriented activities.

3.2.2 Vessel Information

In addition to AIS data, knowing the characteristics of the vessels is crucial. Specifically, for the fishing vessels, the Italian Coast Guard provided us with a dataset containing several attributes, including the MMSI of the vessel, the vessel's name, its country, the name of the port where the vessel is registered, the type of gear used, and the engine power (in horsepower). The information on the gear type is particularly relevant, as it enables to determine when a vessel is engaged in fishing activities. More precisely, if the average speed of the fishing vessel is in the range of the fishing speed of the gear the boat is equipped with, the boat is assumed to be in a fishing phase, otherwise it is assumed to be in a navigation phase. The considered gears and their minimum and maximum speed during fishing activities are reported in Table 3.4.

ID	**Gear description**	**Min speed**	**Max speed**
PS	Purse seines	0	1.5
GNS	Set gillnets	0	1
LLS	Set longlines	0	2
DRB	Dredges	0	2
SOTB	Small bottom otter trawl	2	4.5
LOTB	Large bottom otter trawl	2	4.5
PTM	Pelagic pair trawl	2	5.5
RAP	Rapido	4	7

Table 3.4: Gears and their minimum and maximum fishing speed (in knots).

Moreover, both the vessel category and the length overall (LOA) are key parameters for underwater noise assessment, as they are used for the estimation of the source level generated by boats. These two types of information were obtained from AIS data whenever available; when either of them was missing, they were retrieved from *MarineTraffic*[5] and *VesselFinder*[6] based on the vessel's MMSI.

3.2.3 Harbour Areas

An additional dataset provides information on the spatial extent and characteristics of harbour areas, which are essential to determine when vessels return to

[5] `https://www.marinetraffic.com/en`

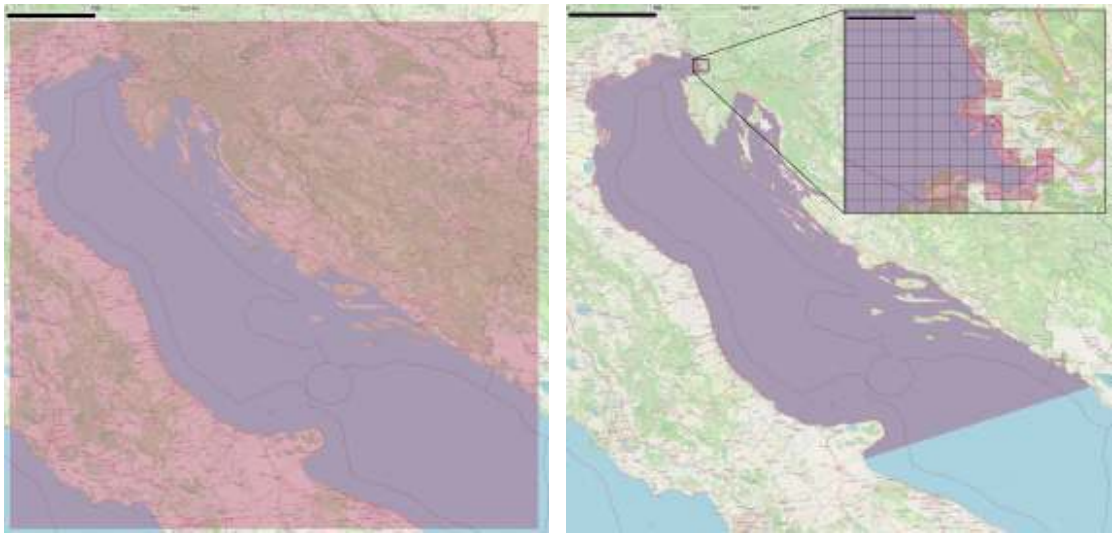
[6] `https://www.vesselfinder.com/en`

port and subsequently depart for a new trip. The dataset includes the geometries of the harbours, together with their names, the corresponding country, and the administrative region they belong to.

Within the Northern and Central Adriatic Sea, a total of 178 harbours were identified. This area encompasses four countries, with 99 harbours located in Croatia, 72 in Italy, 4 in Montenegro, and 3 in Slovenia.

3.2.4 Grid of the Northern and Central Adriatic Sea

In order to model the spatial distribution of vessel-generated underwater noise, we partition the Northern and Central Adriatic sea (specifically the GSA 17 area) into a regular grid composed of square spatial cells (1 km $\times$ 1 km). The cell size was defined by environmental experts to ensure an adequate representation of sound propagation while maintaining a balance between computational efficiency and spatial accuracy.



(a) Initial squared grid including both land and sea areas.

(b) Subset of the grid covering the area of interest (GSA 17).

Figure 3.2: Regular grid of 1 km $\times$ 1 km.

The study area was delimited by the geographic boundaries of the Adriatic sea, within which a regular square grid was automatically generated to ensure uniform spatial coverage (Figure 3.2a). Subsequently, to isolate only the marine areas, we obtained the geometries of the Italian mainland from the *"Istituto Nazionale di Statistica"* (ISTAT),[7] while those of neighbouring countries intersecting the grid

[7] `https://www.istat.it/it/archivio/222527`

(e.g., Croatia and Slovenia) were derived from the *Natural Earth* dataset.[8] These geometries were employed to refine the grid boundaries and to distinguish between terrestrial and marine areas. As a result, the final grid, consisting of 101,113 cells, accurately follows the coastline and provides a detailed representation of the maritime regions. Figure 3.2b illustrates the resulting 1 km $\times$ 1 km spatial grid covering the GSA 17 area.

3.2.5 Environmental Data

To calculate the propagation of underwater sound, it is essential to consider the environmental factors that play a crucial role in sound absorption. Therefore, we consider daily measurements of sea surface temperature (in degrees Celsius), daily measurements of sea salinity (in PSU, Practical Salinity Units), seawater potential of hydrogen (pH), and sea depth (in metres). The temperature dataset includes 7,651 daily measurements, while the salinity and pH datasets comprise 7,866 and 22,440 daily measurements, respectively. Temperature, salinity, and pH data are sourced from *Copernicus*,[9] whereas sea depth information, consisting of 10,387 records, comes from the *European Marine Observation and Data Network* (EMODnet).[10]

Since the environmental measurements are available only at discrete locations, an *Inverse Distance Weighting* (IDW) interpolation is applied to generate continuous surfaces for each environmental variable across the Adriatic Sea grid. IDW is a deterministic spatial interpolation technique that estimates values at unsampled locations as a weighted average of nearby observations, with weights decreasing as distance increases. The resulting interpolated values are then assigned to each grid cell to ensure a complete environmental characterisation of the study area. These environmental features are used to calculate the coefficient of absorption in seawater (denoted as α). The values of α are computed on a daily basis and used to enrich the spatiotemporal grid, thus providing a temporally varying environmental characterisation that captures daily variations in the absorption coefficient.

3.2.6 Hydrophone Data

To determine the ambient noise levels of the Northern and Central Adriatic Sea and fine tune the propagation model, we leverage direct acoustic measurements from the Interreg Project SOUNDSCAPE [40, 101]. The SOUNDSCAPE dataset is composed of 20 and 60 seconds averaged sound pressure levels (SPLs, dB ref 1μPa),

[8] `https://www.naturalearthdata.com/`

[9] `https://www.copernicus.eu/en`

[10] `https://emodnet.ec.europa.eu/en`

Figure 3.3: SOUNDSCAPE hydrophones in the Northern Adriatic Sea.

recorded across a wide frequency range divided into third-octave bands. SPL is the level of the root-mean-square sound pressure expressed in decibel, relative to a reference value of 1μPa [34]. These data were collected at nine monitoring stations from March 2020 to June 2021, encompassing both COVID-19 lockdown periods: the first from March to May 2020 and the second from November 2020 to March 2021.

Figure 3.3 illustrates the positions and names of the nine monitoring stations set up by the project SOUNDSCAPE. In this work, we use the 60-second interval dataset and focus on the 63 Hz and 125 Hz frequencies, which are considered standard by the *European Marine Strategy Framework Directive* (MSFD), as well as on 400 Hz and 4,000 Hz. We select the low-frequency bands (63 Hz and 125 Hz) because sound at these frequencies propagates over longer distances due to its lower absorption coefficient, which substantially increases the computational cost of underwater noise modelling compared with higher frequencies. We additionally include the higher frequencies (400 Hz and 4,000 Hz), as environmental factors become more influential at higher ranges and this part of the acoustic spectrum is particularly relevant for dolphin communication.

Table 3.5 presents, for each hydrophone, its location (latitude and longitude) and the number of records in the dataset of 60-second averaged SPLs (one-third octave, base 10), covering the period from March to December 2020. The acoustic measurements collected at the nine hydrophone stations are used to derive a

Monitoring station	Longitude	Latitude	Mar–Dec 2020
MS1 - Venice (IT)	12°30.883′	45°19.383′	387,432
MS2 - Rimini (IT)	12°42.656′	44°10.254′	340,928
MS3 - Ancona (IT)	13°40.932′	43°31.954′	343,479
MS4 - Trieste (IT)	13°33.917′	45°37.095′	318,533
MS5 - Susak Lošinj (HR)	14°17.293′	44°29.545′	418,330
MS6 - Lošinj (HR)	14°34.510′	44°32.747′	387,488
MS7 - Žirje (HR)	15°36.020′	43°37.788′	339,706
MS8 - Split (HR)	16°25.336′	43°29.895′	358,889
MS9 - Ivana D (HR)	13°15.720′	44°46.953′	217,928

Table 3.5: Location and number of records for each hydrophone in the 60-second averaged SPL dataset from the SOUNDSCAPE project (March–December 2020).

continuous representation of ambient noise levels across the Northern and Central Adriatic Sea through an IDW interpolation. The interpolated data are then aggregated to compute the monthly ambient noise, which is subsequently assigned to each cell of the spatiotemporal grid.

3.3 From AIS Data to Semantic Trajectories

Starting from the AIS data of the vessels, our goal is to reconstruct their trajectories and subsequently enrich them with semantic information. This process is carried out using MobilityDB [152]. After the cleaning of the raw data, described in Section 3.2, the AIS records are characterised by the longitude, latitude, timestamp of the transmission, the vessel length overall (LOA), and an identifier describing the vessel type.

The first volatile aspect considered in our modelling is the vessel speed, which is associated with each AIS transmission. In addition, for every AIS record we determine whether the vessel is located within a port area or outside it. These two attributes form the basis for identifying the operational activity of the vessel at each time instant. Building on these attributes, we assign an activity to each AIS transmission, describing what the vessel is doing at a given time. The possible values of the activity attribute are summarised in Table 3.6.

The *in port*, *exiting from port*, and *entering to port* conditions are inferred from the boolean attribute indicating whether the AIS record is located within a port area. Let us assume that we want to assign an activity to a point a, while the subsequent AIS record corresponds to point b. If both a and b are located inside a port, the activity of a is set to 0 (in port), and the process moves on to the next

ID	Activity Description
0	In port
1	Exiting from port
2	Entering to port
3	Fishing
4	Navigating

Table 3.6: Identifiers and descriptions of vessel activities.

point. If a is inside a port and b is outside, the activity of a is set to 1 (exiting from port), since the vessel is moving from the port to open waters. Conversely, if a is outside and b is inside the port, the activity of a is set to 2 (entering to port). Once outside the port area, the activity depends on the vessel type. If the vessel belongs to the fishing category, the activity is determined by its average speed: if the value falls within the typical fishing speed range associated with the gear the boat is equipped with (see Table 3.4), the vessel is considered to be in a fishing phase (3); otherwise, it is assumed to be in a navigation phase (4). For all other vessel types, the assigned activity is 4 (navigation).

Once the activity has been assigned to each AIS point, we can organise the data into distinct trajectories representing the trips followed by the vessels. Two criteria are applied to identify the beginning of a new trip. A new trip starts when:

- the vessel is inside a port area and no transmission is received for more than 9 minutes;
- an AIS record is located outside a port area, the previous record is inside a port area, and the time gap between the two transmissions exceeds 20 minutes.

The first condition represents a situation in which the vessel has completed a trip, switched off its AIS transponder, remained docked for a certain period, and then started a new trip. The second condition accounts for cases in which the vessel leaves the port and resumes transmission only once it is already in open waters (the 20-minute threshold corresponds to the minimum time required for a vessel to leave the port area).

Algorithm 1 describes the procedure adopted to assign each AIS record to its corresponding trip, that is, to split the movement of each vessel into trips. For the entire year 2020, a total of 318,730 trips are identified. Note that sec denotes the time interval (in seconds) between the current position and the previous one; sec_{in} is the minimum interval (in seconds) to start a new trip when the vessel is inside a port (set to 9 minutes); and sec_{out} is the minimum interval (in seconds)

Algorithm 1 Given sec_{in}, sec_{out}, $\mathbb{V}$ the set of vessels, and $\mathbb{P}$ the set of AIS data points ordered by the time of transmission, the algorithm updates the *tripId* of each AIS record.

$c = 1$
for each $v \in \mathbb{V}$ **do**
  $\mathbb{P}_v = \{p \in \mathbb{P} \mid p.mmsi = v\}$ `// ` $\mathbb{P}_v$ `is ordered by the time of transmission`
  $p_1 =$ first point in $\mathbb{P}_v$
  $p_1.tripId = c$
  for each subsequent point $p_i \in \mathbb{P}_v$, with $i = 2, \ldots, |\mathbb{P}_v|$ **do**
    if ($p_i.inPort$ and $p_i.sec > sec_{in}$) or
     ($p_{i-1}.inPort$ and $\neg p_i.inPort$ and $p_i.sec > sec_{out}$) **then**
      $c = c + 1$
    end if
    $p_i.tripId = c$
  end for
  $c = c + 1$
end for

to start a new trip when the current AIS record is located outside a port while the previous record is inside a port area (set to 20 minutes).

After the detection of the trips, we construct the trip table, named `vessel_trip`. Each row of this table corresponds to a single trip associated with a specific vessel and contains both static and temporal attributes describing its spatiotemporal behaviour. The structure of the table is presented below.

```
CREATE TABLE vessel_trip (
  trip_id integer PRIMARY KEY,
  mmsi integer NOT NULL,
  vessel_name character varying,
  anomaly integer,
  duration interval,
  speed tfloat,
  activity tint,
  trip tgeompoint,
  traj geometry
);
```

The field `trip_id` uniquely identifies each trip, while `mmsi` stores the vessel's Maritime Mobile Service Identity (MMSI), which uniquely identifies the vessel. The attribute `vessel_name` contains the name of the vessel, when available. The column `speed` (`tfloat`) records the temporal evolution of the vessel speed; `activity` (`tint`) stores the temporal evolution of the operational activities described in Table 3.6; and `trip` (`tgeompoint`) captures the full spatiotemporal

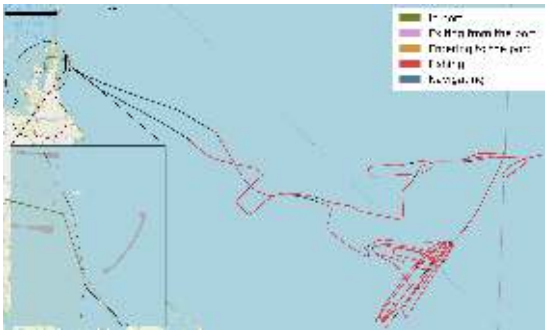
trajectory of the vessel during the trip. The attribute `duration` corresponds to the total duration of the trip, and the column `traj` represents the static geometry of the trajectory, obtained as the spatial projection of the temporal trajectory.

Figure 3.4 shows an example of a fishing vessel trajectory. Starting from its raw AIS data (Figure 3.4a), the trajectory is reconstructed (Figure 3.4b) and subsequently enriched with the activity attribute. In Figure 3.4c, the different colours describe the vessel's activities along the trajectory, allowing the user to immediately identify where the vessel is fishing and to observe the overall movement behaviour. Specifically, green sections correspond to parts of the trajectory when the vessel is in port, pink indicates when it is exiting the port, and yellow when entering it. Blue segments represent navigation phases, while red segments correspond to fishing activity. The figure highlights several circular movements within the fishing segments, which domain experts have confirmed to be typical of this type of fishing behaviour.

Finally, the attribute `anomaly` highlights irregular or unexpected behaviours in the vessel trajectories. As discussed in Chapter 2, it is crucial to distinguish between *holes* (missing data due to transmission issues) and *semantic gaps* (inten-

(a) AIS data.

(b) Trajectory reconstruction.



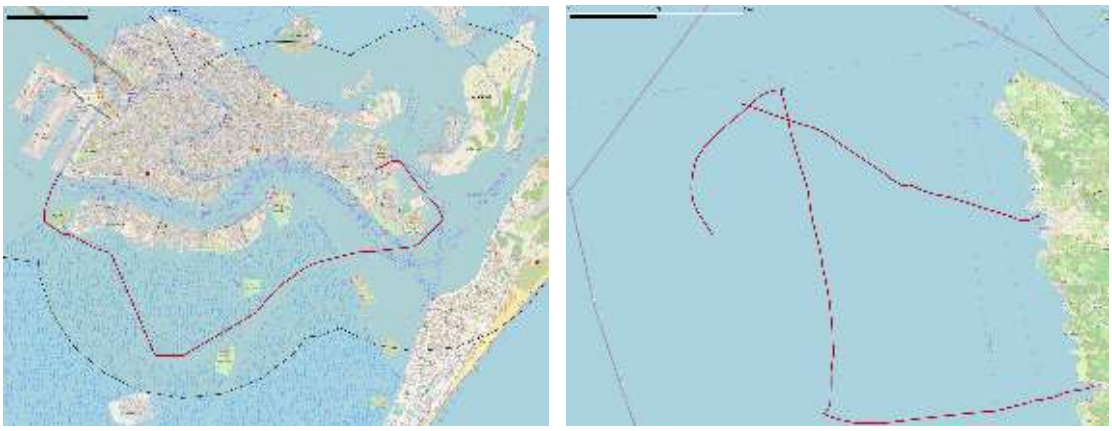
(c) Trajectory enrichment with the activity.

Figure 3.4: Example of trajectory reconstruction and enrichment for a fishing vessel.

tional interruptions). In this case, environmental experts identified a time interval longer than 30 minutes as a semantic gap, that is, a voluntary interruption of the AIS signal which should therefore not be reconstructed. Possible anomalies are defined as follows: the time interval between two consecutive AIS data is longer than 30 minutes outside the port, suggesting a semantic gap (anomaly is set to 1); a vessel remains inside a port area for the entire trip (anomaly is set to 2); the duration of the trip exceeds 24 hours (anomaly is set to 3). If none of the above cases occurs, the trip is considered as normal and *anomaly* is set to 0.

Figure 3.5 shows two examples of trajectory anomalies. In Figure 3.5a, we can observe the trajectory of a firefighting vessel, which remains entirely within the port area of Venice (anomaly 2). Figure 3.5b, on the other hand, presents the trajectory of a Croatian fishing vessel that departs from the port of Umag at 08:57 a.m. on September 29 and returns to the port of Novigrad at 10:14 a.m. on the following day. There is a gap between 11:53 a.m. and 8:41 p.m. on September 29 (anomaly 1). Given both the duration of this gap (approximately nine hours), and its location — close to the boundary between Italian and international waters (the purple line in the figure) — it is reasonable to assume that this represents a voluntary deactivation of the AIS transponder to avoid being tracked while operating outside Croatian territorial waters (i.e., in Italian waters).

From an implementation perspective, all the steps described above are carried out directly in PostgreSQL and MobilityDB. To compute the vessel speed, the AIS records of each vessel are first ordered by the timestamp of the transmission, and the PostgreSQL window function `LEAD` is applied. This function provides access to the value of a field in the subsequent row within the same partition, in this case the



(a) Trajectory of a firefighting vessel remaining inside a port area (anomaly 2).

(b) Trajectory of a fishing vessel with a semantic gap (anomaly 1).

Figure 3.5: Examples of anomalies detected in vessel trajectories.

next spatiotemporal position of the same vessel. The distance between consecutive positions is then divided by the corresponding time difference to obtain the speed, which is finally multiplied by the conversion factor 1.94 to express the result in knots.

The boolean attribute indicating whether each AIS transmission occurred within a port area is obtained by performing a spatial intersection between the geometry of each AIS point and the polygons representing port boundaries. Moreover, all temporal attributes (`tfloat`, `tint`, and `tgeompoint`) in the `vessel_trip` table are built using the constructors and functions provided by MobilityDB, each one based on the appropriate base type. For instance, the column representing the speed is created as follows:

```
tfloatSeqSetGaps(
  array_agg(tfloat(speed, date_time)
    ORDER BY date_time),'30 minutes'::interval)
```

First, the function `tfloat` is used to construct the temporal float in MobilityDB, where each value of longitude is paired with its corresponding timestamp and ordered by the transmission time. Then, the function `tfloatSeqSetGaps` reconstructs the temporal sequence of AIS transmissions while accounting for possible gaps. Specifically, it separates consecutive sequences when there is an absence of AIS data for more than 30 minutes.

The attribute `duration` is computed using the `duration` function of MobilityDB, which returns the temporal extent of the spatiotemporal object provided as input. The column `traj`, on the other hand, is obtained using the `trajectory` function of MobilityDB, which takes a `tgeompoint` as input (in our case, the `trip` column) and returns a value of type `geometry`.

The detection of semantic gaps (anomaly 1) leverages the `numSequences` function of MobilityDB. This function takes a temporal object as input (in our case, the `trip` column) and returns the number of distinct temporal sequences it contains. A value greater than one indicates that the trajectory is composed of multiple sequences separated by time gaps longer than 30 minutes, and therefore an anomaly of type 1 is assigned. For anomaly 2, a temporal boolean (`tbool`) is first constructed to represent whether each point is located inside a port area (`true`) or outside (`false`). The operator `%=` (always equal) of MobilityDB is then applied to check whether the `tbool` remains `true` for the entire trip, meaning that the vessel never leaves the port area. Finally, anomaly 3 is detected by comparing the `duration` of the trip with the 24-hour threshold.

Chapter 4

Underwater Noise Modelling & Analyses

Marine ecosystems encompass a vast diversity of habitats and species, whose preservation is vital to maintaining global biodiversity and ensuring ocean health. However, increasing anthropogenic pressures — including overexploitation, pollution, and habitat degradation — are compromising their resilience and functionality. Among the various pressures, underwater noise pollution — mainly due to vessels activities — has been shown to cause both short- and long-term effects on marine species. Vessel-generated noise, primarily originating from propeller cavitation, engine and machinery vibrations, and hydrodynamic flow around the hull, can travel great distances underwater, disrupting the natural soundscapes that marine organisms depend on for navigation, communication, feeding, and reproduction. Documented impacts include disruption of acoustic communication, behavioural alterations, increased stranding and mortality rates [29, 126, 142]. Unlike chemical or particulate contaminants, acoustic pollution does not accumulate in the marine environment; however, its spatial and temporal persistence is maintained through the continuous activity of vessels fleets. Consequently, the characterisation of underwater noise in space and time is essential for monitoring aquatic ecosystem health, evaluating potential risks, and supplying critical insights to ecologists and policy makers. Such assessments facilitate the formulation of effective management strategies aimed at sustaining productive and resilient marine ecosystems.

Measuring underwater noise is inherently complex and computationally demanding. Deploying hydrophones requires specialised expertise, proper calibration, and significant resources, while subsequent data processing also demands substantial computational capacity, particularly when covering broad spatial or temporal scales. Furthermore, estimating underwater noise in areas where empirical data collection is infeasible, or extending the covered area beyond the mon-

itoring capabilities of existing hydrophones, remains a critical requirement. To address this limitation, many approaches in the literature have employed acoustical models derived from numerical simulations to predict sound pressure levels within the designated regions of interest, see e.g. [49, 69, 74]. These models, however, rely on extensive environmental and vessel-specific input data and involve intensive calculations. Hence, developing sound propagation models that achieve a balance between accuracy and computational efficiency is crucial. Such models must provide reliable predictions while minimising resource use, enabling broader and more scalable applications.

In this work, we follow a complementary approach that avoids using numerical simulation and fully relies on AIS data to calculate the underwater noise generated by vessels in space and time. Similar approaches in the literature are e.g., Erbe et al. [36] and Neenan et al. [87], where the authors' main goal is to readily provide noise maps of the area of interest. However, rather than developing an ad-hoc analysis for a specific case study, our aim is to propose a conceptual framework for underwater noise characterisation that can be instantiated on any sea area and can be used as a quick and effective means to monitor the noise pollution.

The proposed framework is based on semantic trajectories [79, 95]. As described in Section 3.3, starting from AIS data, we reconstruct the vessel trajectories and deploy them in a spatiotemporal database. These trajectories are enhanced with semantic information, such as the characteristics of the vessels and the activities conducted along their paths, which are then used to infer how the noise spreads in the area of interest. Building upon this framework, we introduce two models for estimating underwater noise levels.

- The first model, presented in this chapter, focuses specifically on fishing vessels operating in the Northern Adriatic Sea, one of the most exploited basins of the Mediterranean Sea. To estimate the emitted source level (SL), we rely on the engine horsepower information available for all fishing vessels in our dataset. However, a direct mapping between horsepower and acoustic emissions is not available. To derive such a relationship, we identify a reference vessel and use it to derive the relationship between engine power and acoustic emissions (see Section 4.2.2). A further distinctive feature of this model is that underwater noise is computed at four frequency bands — 63, 125, 400, and 4,000 Hz — covering both low-frequency components, which dominate long-range propagation, and higher frequencies relevant for species sensitive to mid- and high-frequency sound.

- The second model, described in Chapter 5, extends the analysis to the full set of vessel categories operating across the Northern and Central Adriatic Sea (the GSA 17 area). In this case, since engine-power information is not

available for all other vessel categories, the source level is estimated using the model introduced by MacGillivray and de Jong [75], which calculates underwater noise emissions as a function of the vessel's LOA and its type. In our work, we refine this model and we employ it to estimate the underwater noise generated by all vessels operating in the study area. We further improve the computational efficiency of the original formulation. In particular, we restrict the analysis to the 63 Hz frequency band, which is the most computationally demanding in terms of sound propagation, and we enhance the model's performance by optimising the implementation and incorporating parallelisation techniques.

The dataset used in this chapter is the one described in Section 3.2.1 which comprises all AIS data of the Italian and Croatian fishing vessels for the year 2020. In order to determine the acoustic characteristics of the vessels' engines and fine tune the propagation model, we take advantage of the direct acoustic measurements produced by the Interreg project SOUNDSCAPE [40] that carried out an acoustic monitoring in the Northern Adriatic Sea from March 2020 to June 2021. In particular, we consider the frequency of 63 Hz, 125 Hz, 400 Hz and 4,000 Hz. The frequencies of 63 Hz and 125 Hz are the ones established as standard frequencies by the *European Marine Strategy Framework Directive* (MSFD), while 400 Hz and 4,000 Hz allow us to show the increasing significance of environmental factors at higher ranges.

This chapter is organised as follows. Section 4.1 presents the literature on underwater noise characterisation. Section 4.2 introduces the theoretical principles governing underwater sound propagation and describes how source sound levels, propagation loss, and ambient noise are computed for the development of the model. Section 4.3 describes how the sound propagation model is implemented within MobilityDB [153]. As outlined in Section 3.2, the study area is partitioned into a regular grid whose cells are enriched with environmental features. In our model, instead of relying on physical listening points (hydrophones), we treat the centroid of each cell as a *virtual* listening point at which the received sound level is computed. Finally, Section 4.4 presents the results of our analyses, illustrating the impact of fishing activities on underwater noise and demonstrating the flexibility and expressiveness of the proposed modelling framework.

4.1 Related Work

Underwater noise arising from human activities is known to have various adverse effects on aquatic life. These can range from acute effects such as permanent or temporary hearing impairment to chronic effects such as developmental deficiencies and physiological stress [126, 142]. As a general study, we mention the

work by Cruz et al. [23] that summarised the status of European waters regarding continuous underwater radiated noise from shipping. The goal was to provide recommendations on possible future activities. The work focused on four main topics: characteristics and quantification of noise sources from various ship types, impacts on marine fauna, existing policies, including guidelines, decisions, resolutions and regulations and mitigation measures for the abatement of ship noise and noise-related impact.

The specific topic of interest in this paper, i.e., underwater noise generated by (fishing) vessels, has been explored by various authors in the literature. Below, we review some relevant contributions classified into three categories: (*i*) works relying on direct underwater noise measurements, (*ii*) approaches based on acoustic models obtained by numerical simulation, and (*iii*) proposals — like ours — that use only AIS data along with a sound propagation model.

4.1.1 Approaches based on Direct Measurements

Some works in the literature address the issue of underwater noise data acquisition, transmission and storage. The most interesting one in our perspective is the dataset produced by the SOUNDSCAPE project [40], which has been presented in Section 3.2.6. The dataset is available on Zenodo [101] and the whole process of data acquisition, storing and post-processing is described by Petrizzo et al. in [100]. Furthermore, Picciulin et al. [106] perform some analyses and describe the spatial and temporal variations of the sound pressure levels recorded by the SOUNDSCAPE project during the monitoring period.

Other approaches propose methods for acoustic real-time measurements and monitoring, such as Diviacco et al. [27] and Moran et al. [84]. Differently, Farcas et al. [42] carried out multi-site measurements to validate a large-scale shipping noise map constructed using a generic shipping noise model. Finally, the work by Picciulin et al. [105] shows two acoustic surveys conducted at 40 listening points distributed along the three inlets that connect the Venice lagoon to the sea, in order to characterise the local noise levels and evaluate the fish spatial distribution by means of its sounds.

4.1.2 Approaches based on Numerical Simulation

Most of the approaches to underwater noise characterisation are based on acoustic models obtained by numerical simulation to predict the sound levels in the area of interest. They simulate sound propagation taking into consideration reflection, diffraction and absorption phenomena and require precise information in input and a relevant computational effort (in terms of time and resources) to produce the acoustic maps of the area of interest. Such models usually account

for many environmental variables (e.g., sea temperature, wind, waves, salinity), precise bathymetry information and sound speed profiles, possibly at different sea depths. They use AIS data to derive the vessels' trajectories and include their emitted sound into the model. The results obtained are often validated through real acoustic data measured on specific sites.

As a first approach in this category, we mention the work by MacGillivray et al. [74]. The authors present an acoustic model to predict anthropogenic sound levels at the Great Barrier Reef Marine Park in Australia. The model uses AIS data and wind speed data to simulate the time-dependent noise field in the area of interest and for the frequencies 64 kHz and 375 kHz. The model uses three months of AIS data and assign acoustic source levels to each vessel according to predefined values already calculated for the various vessels categories. The model includes environmental parameters such as ocean temperature and salinity as well as bathymetry information. The obtained results were compared against real data collected by an acoustic recorder placed on a specific site in the same period.

A similar approach is used by Larayedh et al. [69] to investigate the shipping noise in the Red Sea for the frequency band 40-100 Hz. The acoustic model, based on simulation, includes two months of AIS data belonging to specific categories of shipping vessels (tankers, bulkers, container ships, open hatch vessels and vehicle carriers), whose sound levels were derived from the study of MacGillivray and de Jong [75]. Bathymetry data, sea temperature and salinity are included, as well as spatial and temporal sound speed profiles specifically calculated for the Red Sea. The authors emphasise the computational effort needed to calculate the resulting maps of predicted noise levels, which required the use of a supercomputer. Results include the maps of predicted spectral noise level (averaged over the considered frequency band) for a specific day at different depths, and maps considering the two months under exam.

Along the same line, Ghezzo et al. present in [49] a numerical reconstruction of the sound field in the Northern Adriatic Sea basin for the year 2020. Acoustic modelling was performed by the Quonops© [123] underwater noise prediction system, which is based on simulation. *Natural Sound Maps* were produced by taking into account the sound propagation properties of the local environment (hourly wind and waves data, daily mean sea temperature and salinity, bathymetry data). Additionally, a combination of natural sources and AIS-based marine traffic were used to produce the *Baseline Sound Maps*. The AIS dataset included all types of vessels and the source level noise produced by the various categories of vessels were derived from the model proposed by MacGillivray and de Jong [75]. The trawling activity of the fishing vessels was taken into account to consider an increased noise level when the trawler is in use. Calibration of the AIS-based model was performed using the SOUNDSCAPE [40] measured data. The authors present

annual Baseline sound maps and excess 20 dB sound maps for the frequencies 63 Hz, 125 Hz and 250 Hz. The study included specific analyses on protected areas of the Northern Adriatic Sea.

The work by Ghezzo et al. [49] is highly relevant to our study, as it examines the Northern Adriatic Sea in 2020, uses the same source of ground-truth data (the SOUNDSCAPE time-series measurements), and considers some of the same frequencies (63 Hz and 125 Hz), while adopting a different methodological approach (ours falls within the category of *"Approaches Based on AIS Data"*, described in the next subsection). Unfortunately, the results of the two studies cannot be directly compared. This is because, in this chapter, our analysis is based exclusively on AIS data from fishing vessels, whereas Ghezzo et al. incorporate AIS information from all vessel categories. This difference in fleet coverage leads to marked differences in both the magnitude and spatial patterns of the estimated noise emissions, thus preventing a direct comparison. Moreover, it is important to note that our primary goal is different from [49]: we aim to provide a flexible tool for performing, with reduced computational effort, a number of various analyses, not limited to the generation of noise maps. This is made possible by the creation of a spatiotemporal database which stores semantically enriched vessel trajectories.

4.1.3 Approaches based on AIS Data

Other proposals in the literature avoid the use of numerical simulation and give a more simplified description of the sound propagation effects. They are based on AIS data and they usually adopt a sound propagation model to calculate the transmission loss in the area of interest. The goal is to readily provide acoustic maps that can be used by policymakers for a first quantitative spatiotemporal underwater noise evaluation. They are computationally efficient, even for large datasets.

As a first approach in this setting, we mention the one by Erbe et al. [36] that uses the ship transits derived from AIS data and calculated by the Canadian Guard Coast. The authors apply a sound propagation model to derive a cumulative large-scale noise map of the area of interest for the frequency band 10-2000 Hz. The authors include a comparison with field measurements. Similarly, Neenan et al. [87] develop a vessel noise modelling method using AIS data and online data on estimated source levels of individual ships. The authors divide the area of interest into a spatial grid of 1 km $\times$ 1 km and consider a single frequency of 80 Hz. They calculate the propagation loss in a 5 km ray around each vessel's position taking into account also sediment type and bathymetry. The goal is to produce heatmaps of average received sound levels over monthly periods.

The approach followed in this paper is similar to that in [36, 87] since we also use AIS data and a model of sound propagation to derive a spatiotemporal

characterisation of the underwater noise of the area of interest. However, the main goal of the works [36] and [87] is to produce cumulative noise maps. Instead, in our approach, cumulative noise maps are only one of the possible outputs. The two key features that distinguish our approach are the use of semantic trajectories and their deployment into a spatiotemporal database. Semantic trajectories enable the association of semantic information with trajectories, like the activities performed by a vessel during the trip. The spatiotemporal database allows for answering any query about underwater noise at different temporal and spatial granularities, with the possibility of considering any set of vessels. For example, a user can visualise the noise produced by a single vessel along its trajectory, accounting for increased noise while fishing, generate noise maps for a specific area and time period, or create time-lapse videos illustrating underwater noise dynamics.

A further proposal in this setting that demonstrates the potential of using AIS data for underwater noise estimation is Jallkanen et al. in [66]. This work uses worldwide terrestrial and satellite AIS data covering the years 2014-2020 and the STEAM (Ship traffic Abatement Model) tool by the Finnish Meteorological Institute [68] to estimate vessel noise source levels. The STEAM model predicts instantaneous vessel noise levels that were then cumulated over time to produce a noise source energy map. The instantaneous noise levels in each vessel position (and in a sphere around the source) were reported in a spatiotemporal grid cell, whose resolution is 0.1^{o} (WGS84 coordinate system). The gridded data were produced for the frequencies 63 Hz, 125 Hz and 2,000 Hz. Results included: (*i*) a quantification of the global underwater noise emissions from shipping in the period 2014–2020, showing that at the current rate the emissions are doubling every 11.5 years; (*ii*) a study of the emissions in different areas, revealing a large variability of shipping noise in different regions; (*iii*) a study on the emissions during the COVID-19 pandemic, demonstrating a drastic reduction of emission levels globally; (*iv*) and a study of the emissions of the various vessels categories, indicating that container ships are the largest contributor to shipping noise emissions.

4.2 Underwater Noise Model

In this section we describe a model for underwater sound propagation that is used in Section 4.3 to provide a spatiotemporal characterisation of underwater noise in the Northern Adriatic Sea. We present the theoretical laws governing underwater sound propagation, and we also describe how we obtain the estimation for source sound levels, propagation loss and ambient noise, necessary for the definition of the model. We recall that the estimation is based on the 60-second averaged SPL dataset released by the project SOUNDSCAPE and we focus on the frequencies of 63 Hz, 125 Hz, 400 Hz and 4,000 Hz.

4.2.1 Sound Propagation Model

The basic objective of noise modelling is to assess how much noise a particular activity will generate in the surrounding area [43]: the aim is to model the received noise level (RL) at a given point (or points), based on the sound source level (SL) of the noise source, and the amount of sound energy which is lost as the sound wave propagates from the source to the receiver (transmission loss or propagation loss, TL). The relation between these quantities is encapsulated in the classic sonar equation [135]:

$$RL = SL - TL \tag{4.1}$$

This straightforward expression is fundamental to modelling underwater noise, and its simplicity belies considerable complexity in the task of computing the transmission loss in order to estimate the received noise.

Sound propagation is profoundly affected by some factors such as the conditions of the surface and bottom boundaries of the sea as well as by the variation of sound speed within the ocean volume [39]. Air has a density 800 times lower than the density of water, therefore a sound that propagates inside the water has a higher propagation speed, equal to about 1500 m/s, against about 340 m/s of air. So, with a sampling period of 60 seconds it makes sense to neglect propagation time within the area affected by a sound source (area of influence) and, within the sampling interval, consider the noise level distribution as stationary. When a boat switches the engine on, we consider the noise as instantaneously propagated in the area of influence within the sampling period, without actually propagating the wavefront in space-time.

Sound propagation speed is also influenced by various chemical-physical factors such as temperature, salinity and pressure [112], varying both during the day and with the seasons in the superficial part [111], and with depth. In shallow water, that is the predominant context in Northern Adriatic Sea, sound wave reflections off the seabed strongly affect propagation, and bathymetry plays an important role in determining propagation loss [3]. The computation of the transmission loss considering all these parameters is not a simple task and for this reason various models have been introduced. Before discussing the transmission loss, however, we focus on how to evaluate the source level, that is, in our case, the noise generated by the fishing vessels.

4.2.2 Source Level Estimation

The principal sources of underwater noise are machinery, propellers, and cavitation. Our AIS dataset and the vessel information dataset contain details about fishing boats, including length overall (LOA), engine horsepower, and fishing gear.

However, these datasets do not include direct measurements of the sound pressure levels of the fishing vessels. So, we need to infer such values considering the general literature about underwater noise and the measurements provided by the SOUNDSCAPE project.

A first issue is how to evaluate the increase of noise when a trawler is in action. Many studies focus on assessing the noise of vessels when they are free-running (i.e., when they are not performing fishing activities). However, trawling vessels typically generate higher levels of radiated noise compared to free-running vessels operating under the same machinery settings [25]. While published data on the radiated noise from operating trawling vessels are limited, some studies have reported increases in radiated noise ranging from 5 dB to 15 dB during trawling activities [82]. Specifically, in [25] it is noted that the effect of trawling is minimal below 100 Hz and increases with frequency. Accordingly, we assign an increase of 5 dB at 63 Hz when the vessel is trawling, 10 dB at 125 Hz, and 15 dB at higher frequencies, specifically at 400 Hz and $4,000$ Hz. This approach aligns with the findings in [82] and reflects the change in source levels during trawling as discussed in [49].

To recover the sound pressure level of a specific fishing vessel, we consider the measurements of the hydrophone MS9 located at $13°15.720E$ $44°46.953N$, in the middle of Adriatic Sea (see Table 3.5), with 42 m depth, terrigenous sandy seafloor, taken on March 31, 2021 between 5:40 pm and 5:55 pm. Here, there is a unique fishing vessel crossing nearby the hydrophone and taken as the reference boat. Thus, the recorded noise is associated to the trip 1001 of the reference boat (length=27.45 m, engine power=835 Hp), while trawling at about 3.9 kn (knots) between 500 m and 60 m from the hydrophone. A linear regression was performed between the measured values of SPL, expressed in decibel, and logarithmic distance, thus allowing to assign a vessel with an 835 Hp engine, when not trawling, an estimated source level (at the conventional distance of 1 m from the point source) that varies with frequency as reported in Table 4.1.

	63 Hz	125 Hz	400 Hz	4,000 Hz
SPL (1m)	136 dB	133 dB	126 dB	123 dB

Table 4.1: SPL for the reference boat at different frequencies.

In order to associate the source levels with all the other vessels, we need to relate the sound pressure level to the engine horsepower, the latter being available in our dataset. If we assume that a constant fraction of engine power gets converted into acoustic power (i.e., acoustic power scales linearly with horsepower), then 3 dB are added per doubling in engine power. We adopt such a linear progression on

logarithmic scale of engine power and the resulting value is denoted with SL_0. For example, for engines between 100 Hp and 835 Hp, considering a frequency of 63 Hz, we obtain a range between 123 dB and 136 dB.

Differences in source level may result from variations in speed. Specifically, as noted in [18], the intrinsic factor of speed can influence the broadband source level of ships according to the following relation:

$$SL = \begin{cases} SL_0 & \text{if } v \leq v_0 \\ SL_0 + 15.39\ dB \times \log_{10} \dfrac{v}{v_0} & \text{if } v > v_0 \end{cases} \tag{4.2}$$

where $v_0 = 3.9$ kn corresponds to the speed of the reference boat and v is the actual speed of the vessel. For simplicity, we assume that our model source boat radiates uniformly in all directions, although the acoustic signature of actual boats in navigation is louder from the side-aspect and stern-aspect than from the bow-aspect.

4.2.3 Transmission Loss

In the ideal scenario, where surface and seabed reflections as well as absorption losses are neglected, and propagation speed is uniform, simple spherical spreading governs transmission loss [35]:

$$TL = 20 \times \log_{10}(r) \tag{4.3}$$

where r is the distance from the source to the receiver.

As observed in more realistic scenarios, sound will initially exhibit spherical spreading at short distances, where boundary effects are negligible, followed by cylindrical spreading at long ranges. In between, there is a transition region where neither spherical nor cylindrical spreading accurately describes the sound propagation. This situation can be approximated by assuming a sudden shift from spherical to cylindrical spreading at a *transition range* r_{trans}. While some recommend using the water depth as a rough estimate for this transition range, it is important to use this approach with caution. As highlighted in [35], the optimal transition range r_{trans} varies depending on seabed characteristics. Simulation by parabolic equation modelling [90] (confirmed by normal-mode modelling) shows that for a flat bottom at 50 m depth (soft seabed, $\rho = 1500$ kg/m^3, $c = 1700$ m/s), there is about 48 dB attenuation at 400 m, corresponding to over 8 doublings with spherical attenuation. Between 2 and 4 km the attenuation is about 4.5 dB, which is compatible with mode stripping attenuation (between spherical and cylindrical): 15 $\log_{10}(r)$. Propagation modelling and experimental measurements in [127] show decreased attenuation in shallow water. For soft seabeds, propagation loss

is greater than for hard seabeds, but attenuation is lower at 15 m than at 30 m depth. Up to 100 m, at 30 m depth, with soft seabed, sound pressure decrease is almost identical to spherical spreading. At 15 m depth, with soft seabed, sound pressure decrease is spherical up to about 60 m, then the decrease is much lower. The same study reports a 4 dB increase in sound level when speed changes from 6 to 11 kn, confirming the velocity-dependent component of 15.39 dB $\times \log_{10}(v/v_0)$ from Equation 4.2. In [3] Ainslie proposes a modification to cylindrical spreading for long ranges, once the reflection losses due to multiple bottom reflections begin to accumulate. Following [3], we adjust the propagation loss to implement the combination between spherical propagation and mode stripping as follows:

$$TL = \begin{cases} 20\ \log_{10}(r) & \text{if } r \leq r_{trans} \\ 15\ \log_{10}(r) + 5\ \log_{10}(r_{trans}) & \text{if } r > r_{trans} \end{cases} \tag{4.4}$$

The 15 $\log_{10}(r)$ dependence on range is known as mode stripping because it results from the gradual erosion of steep ray paths (high-order modes) after multiple bottom reflections. To determine r_{trans}, we refer to the trajectory of trip 1001 of the reference boat. At 63 Hz the transition is expected to occur at around 400 m, approximately 10 times the water depth. This indicates that r_{trans} is parametrically dependent on depth through a multiplicative factor of 10. For the middle frequencies (125 Hz and 400 Hz) we set the commutation between the two propagation regimes at 4 times the depth based on our simulations. For the highest frequency (4,000 Hz) we restrict such commutation range even further, to twice the depth.

With simple geometric spreading, the role of absorption in propagation is not accounted for. Environmental absorption features may affect the transmission loss, especially for large distances and high frequencies. In more realistic models, one needs to consider all the environmental aspects that influence the sound propagation underwater, by adding a term proportional to distance from the source [35]:

$$TL_{tot} = TL + \alpha \times r \tag{4.5}$$

In the literature there are several models for predicting the absorption of sound (α) in sea water which retain the essential dependence on temperature, pressure, salinity, acidity and other environmental features. In the Francois and Garrison model [47] the general equation for the absorption of sound in sea water, at a given frequency f, is given as the sum of contributions from boric acid, magnesium sulfate, and pure water. At a frequency below 100 Hz only the first contribution is relevant, as α is approximately of the order of 10^{-6} dB/m [35]. Specifically, at frequencies of 63 Hz and 125 Hz, α is on the order of 10^{-6} dB/m, while at 400 Hz it increases to the order of 10^{-5} dB/m, and at 4,000 Hz it reaches the order of 10^{-4} dB/m.

4.2.4 Ambient Noise

The received noise level (RL) at a given point is computed starting from Equation 4.1. However, the formula does not consider the ambient (or background) noise, which is present in the marine environment. The received noise level RL exceeding ambient noise is

$$RL = SL - TL_{tot} - AN \tag{4.6}$$

where SL is the sound source level, TL_{tot} the transmission loss and AN the ambient noise.

We use the SOUNDSCAPE measurements also to estimate the ambient noise. In particular, we employed the exceedance level L_{90} [136], which indicates the sound level that is exceeded 90% of the time and is equal to the 10^{th} percentile statistic. As mentioned in [106], L_{90} can be referred to the common natural acoustic conditions. This sound level is very different at the nine stations along the year and at the various frequencies (see Figure 3 in [106]). Hence, for each frequency and for each month of the year, we compute the *ambient noise map* in the following way. We use the grid that partitions the Northern Adriatic Sea into square spatial cells of 1 km $\times$ 1 km, as described in Section 3.2.4. Given a certain frequency and a month, to assign an ambient noise value to each cell of our grid, we started from the cells where hydrophones are located: we associated the exceedance level L_{90} of the hydrophone at the corresponding cell. Then we applied an Inverse Distance Weighting (IDW) interpolation using QGIS,[1] an Open Source GIS that supports viewing, editing, and analysis of geospatial data. It is worth recalling that, in this type of interpolation, the sample points are weighted so that the influence of each point decreases with distance from the unknown point being estimated. This approach allows us to assign an ambient noise value that varies for each grid cell, capturing the differences in underwater noise across various regions of the Northern Adriatic Sea. For instance, in June 2020, at 63 Hz the most silent area is Ancona (Italy) with an ambient noise of 60.78 dB and the loudest zone is Žirje (Croatia) with a value of 82.62 dB. Instead, at 4,000 Hz Ancona has an ambient noise of 84.65 dB and Žirje reaches a value of 92.80 dB.

4.2.5 Aggregating Received Noise Levels

At a measurement point that is equally distant from two equally-powerful sound sources, the two contributions would add up in magnitude and phase. However, distinct and independent sources, such as two boats, can be treated as incoherent sources. Even in a narrow frequency band, there will be a random phase difference

[1] `https://qgis.org/en/site/`

between the two sources. Therefore, the noise in a 1/3 octave band around 63 Hz (or in any other band) gets increased by 3 dB if there are two equal contributions, by 6 dB if there are four equal contribution, etc. [34]. More precisely, what is added are the intensities, after inversion of the logarithmic function that defines the decibel. Generally, if we have n sources reaching a cell with n different values of RL, the total noise level is:

$$RL_{total} = 10 \times \log_{10}(10^{RL_1/10} + \ldots + 10^{RL_n/10}) \tag{4.7}$$

4.3 Implementation using MobilityDB

In this section we develop a framework for the spatiotemporal characterisation of underwater noise. As already mentioned, we partition the Northern Adriatic Sea into a regular grid composed of square spatial cells (1 km $\times$ 1 km) and we estimate the noise generated by the fishing vessels in any cell at regular time intervals (every 60 seconds). We retain the 60-second interval temporal resolution to be synchronised with the SOUNDSCAPE project's measurements. The overall process is illustrated in Figure 4.1. As described in Chapter 3, starting from terrestrial AIS data we reconstructed the trajectories and enriched them with semantic information such as vessel activity and speed. Since in this chapter our objective is to model the underwater sound propagation generated by the vessels, it is essential to associate each vessel with the noise it produces. For this reason, we introduce an additional permanent aspect representing the sound level generated by the vessel's engine, denoted as SL_0, which is derived from the vessel's engine horsepower.

We then incorporate environmental factors and hydrophone data to enrich the grid of the Northern Adriatic Sea with environmental variables such as sea salinity, sea surface temperature, sea depth, pH, and ambient noise. Based on vessels' semantic trajectories and the enriched grid, we develop our sound propagation model, which is described in Section 4.3.1. Finally, we generate underwater noise maps, which are crucial for ecologists and policymakers to study underwater noise pollution and ensure a productive and healthy marine ecosystem. Most of this process (including trajectory reconstruction and enrichment, the development of the spatiotemporal grid, and sound propagation modelling) is implemented using MobilityDB. Further implementation details and the advantages of using MobilityDB are discussed in Section 4.3.2.

4.3.1 Construction of the Noise Maps

In this section we describe the high-level procedure for assigning a noise level at a certain frequency f induced by the fishing vessels to the cells of a regular grid,

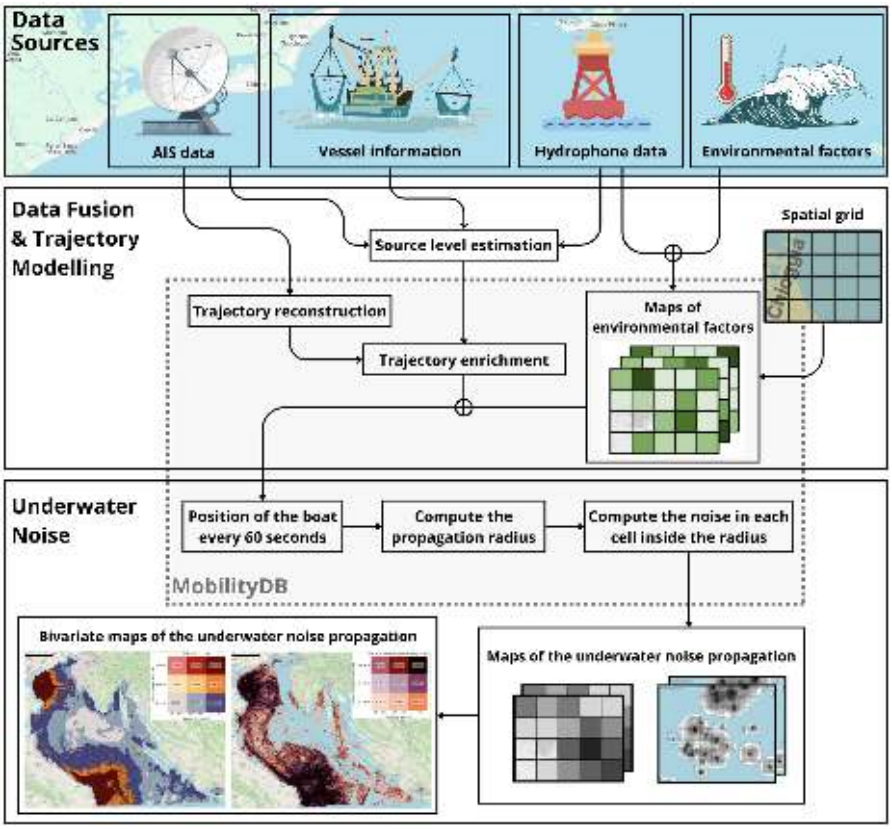


Figure 4.1: Overview of the process consisting of three main steps: data sources, reconstruction and enrichment of trajectories, and underwater noise propagation.

partitioning the Northern Adriatic Sea, every 60 seconds. We consider a set of spatiotemporal cells $\mathbb{G} = \mathbb{S} \times \mathbb{T}$ where $\mathbb{S}$ is a regular grid consisting of 1 km $\times$ 1 km spatial cells, and $\mathbb{T}$ is a set of time instants, such that t_0 is a fixed time instant and $t_{i+1} = t_i + 60s$. Hence, each spatiotemporal cell $c \in \mathbb{G}$ consists of two components, (g, t), representing the spatial cell g at time instant t. For each frequency $f \in \{63, 125, 400, 4000\}$ we write $\mathbb{G}_f$ to denote the set of spatiotemporal cells annotated with pieces of information which possibly depend on the chosen frequency. The annotations are: (i) *ctd* contains the coordinates of the centroid of g; (ii) *depth* stores the depth of the sea in c; (iii) α stores the absorption of sound as defined in Section 4.2.3; (iv) *an* records the ambient noise in c estimated as reported in Section 4.2.4; (v) *rl* records the total noise received in c, i.e., by the centroid of g at time instant t. It is worth noticing that only the latter three annotations depend on the frequency f.

Let $\mathcal{TR}$ be the set of the trajectories of the fishing vessels, $\mathbb{G}_f$ be the spatiotemporal grid with $f \in \{63, 125, 400, 4000\}$. Algorithm 2 computes the total

Algorithm 2 Given $\mathcal{TR}$, $\mathbb{T}$, $\mathbb{G}_f$ and $f \in \{63, 125, 400, 4000\}$, the algorithm computes the total received noise level for each $c \in \mathbb{G}_f$ at frequency f.

Let mp: $map\langle cell, float\rangle$
for each $tr \in \mathcal{TR}$ **do**
  for each $t \in \mathbb{T}$ **do**
    p $= (tr(t), t)$
    $c_p =$ unique cell $c \in G_f$ such that $p \in c$
    $v_0 = 3.9$
    if $p.speed > v_0$ **then**
      $SL = tr.mmsi.SL_0 + 15.39 \cdot \log_{10} \frac{p.speed}{v_0} + inc_f \cdot p.fishing$
    else
      $SL = tr.mmsi.SL_0 + inc_f \cdot p.fishing$
    end if
    $r_{trans} = c_p.depth \cdot mult_f$
    $r = 10^{(SL - 5 \cdot \log_{10}(r_{trans}) - c_p.an)/15}$
    for each $c = (g, t) \in \mathbb{G}_f$. d$(c.ctd, p{\downarrow}1) < r$ **do**
      $dist =$ d$(c.ctd, p{\downarrow}1)$
      if $dist \leq r_{trans}$ **then**
        $RL = SL - 20 \cdot \log_{10}(dist) - c.\alpha \cdot dist - c.an$
      else
        $RL = SL - 15 \cdot \log_{10}(dist) - 5 \cdot \log_{10}(r_{trans}) - c.\alpha \cdot dist - c.an$
      end if
      $mp[c] = mp[c] + 10^{RL/10}$
    end for
  end for
end for
for each $c \in \mathbb{G}_f$ **do**
  $c.rl = 10 \cdot \log_{10}(mp[c])$
end for

received noise level for every cell $c \in \mathbb{G}_f$ at the frequency f. We use $p{\downarrow}1$ to denote the projection on the first component of the spatiotemporal point p, i.e., its coordinates, and d$(z_1,\ z_2)$ for the Euclidean distance between two spatial points z_1 and z_2.

The noise is estimated every 60 seconds at the selected frequency, i.e., in the time instants belonging to $\mathbb{T}$ at frequency f. The centroids of the grid cells are considered as virtual listening points (we have 43,386 of these points), and consequently the noise *received* at a centroid point models the noise in all the points of the cell at a certain time instant and at frequency f.

In order to build the noise map at frequency f, we get the positions of all the fishing vessels at the same time instants, i.e., every 60 seconds. For each point

(Line 4), we determine the cell it belongs to (Line 5) and we calculate the noise generated by the fishing vessel (Lines 7–11) obtained by adding to the sound level associated with the horsepower of the boat ($mmsi.SL_0$), a contribution related to the actual speed of the vessel in p (see Equation 4.2), and the noise due to the fishing activity if it occurs in p. Notice that the latter, inc_f, can range from 5 dB up to 15 dB depending on the frequency, as discussed in Section 4.2.2. In Line 13, the sound propagation radius r (expressed in metres), i.e., the distance at which the noise generated by the fishing vessel gets drowned into ambient noise, is computed. This is obtained by using Equation 4.6 and setting the received noise (RL) to 0:

$$0 = SL - TL_{tot} - AN$$

In computing the radius we ignore the coefficient of absorption α in Equation 4.5 for TL_{tot}, allowing for a simplification of calculation and getting an overestimation of r, hence the approximation is safe. With some simple mathematical steps we get $r = 10^{(SL-5\cdot\log_{10}(r_{trans})-c_p.an)/15}$ where r_{trans} is the transition range when spherical propagation shifts to a lower attenuation. Its value depends on the sea depth in c_p (Line 12) and it varies according to frequency f, $mult_f$, ranging from 2 up to 10 times the sea depth, as explained in Section 4.2.3. Then, we propagate the noise in the cells that are within the radius r (Lines 14–21) according to Equation 4.4 and we compute the relative received noise level by Equation 4.6. Finally, by using Equation 4.7, we combine all the received sound levels to obtain the total noise level at frequency f to be associated with the cell (Line 26).

Concerning the complexity, let $n = |\mathcal{TR}|$, $m = |\mathbb{T}|$, $k = |\mathbb{G}|$, a be the area of a grid cell and r the largest radius arising in Line 13. Then the complexity is $O(n \cdot m \cdot r^2/a + k)$. The factor r^2/a is motivated by the fact that in Line 14 we consider the cells in a neighbourhood of radius r. Note that r depends on the source level, which is bounded by the maximum engine power and the maximum speed of the monitored fishing vessels. In our case, the radius r for each frequency is less than 71,254 m for 63 Hz, 111,399 m for 125 Hz, 31,802 m for 400 Hz, and 5,478 m for 4,000 Hz.

In order to process all this data and build our model, we used a machine that features 32 Intel(R) Xeon(R) CPU E5-4610 v2 processors running at 2.30 GHz, offering multithread performance. It is equipped with 256 GB of DDR4 ECC RAM and it utilises a 500 GB RAID 5 storage configuration. On this machine we deployed PostgreSQL 16.6, PostGIS 3.5, and MobilityDB 1.3. Regarding the execution time required to build the sound propagation model from AIS data, it is important to distinguish between the reconstruction and enrichment of vessel trajectories and the propagation of underwater noise. We provide the performance for June 2020, as it is one of the months with the highest number of AIS records. Specifically, the reconstruction and enrichment of fishing vessel trajectories from

AIS data for this month take only 46 minutes. The execution time of underwater noise propagation depends on the frequency used, as it varies across frequencies due to differences in ambient noise, vessel SL, the increase in SL during fishing activities, and the absorption coefficient α. For the entire month of June, the execution time required to propagate the underwater noise, starting from the semantically enriched trajectories, is approximately 44 hours for 63 Hz and 125 Hz frequencies, 10 hours for 400 Hz, and 1 hour for 4,000 Hz.

In this chapter we focus primarily on assessing the feasibility of the model and demonstrating its potential through various analyses, rather than optimising its efficiency. A more in-depth evaluation of computational performance, including code optimisation and execution time improvements, is addressed in Chapter 5.

4.3.2 Implementation Details

To construct and store the set of trajectories and to implement the underwater noise model, we used MobilityDB [152]. The motivations behind our choice of MobilityDB for implementing Algorithm 2 are as follows. The size of the datasets implies that the processing must be made in a database context. For this reason, we decided to use PostgreSQL and its spatial extension PostGIS. Although PostGIS enables the representation of trajectories, in which the timestamps are encoded in the M dimension, the temporal support provided is rather limited. MobilityDB extends PostgreSQL and PostGIS with temporal capabilities and provides a rich API for manipulating temporal types. This enables to express processing pipelines such as the one depicted in Algorithm 2 using PL/pgSQL, PostgreSQL procedural language, which adds control structures to the SQL language to perform complex computations.

MobilityDB is implemented in C, as it is PostgreSQL and PostGIS. PostgreSQL takes care of the store and management of the tables as well as bringing the data from disk storage to main memory, where the processing is performed. As the query results are represented as tables, the user may decide to store them back to the database for further processing.

Regarding performance, MobilityDB has efficient data structures that are optimised for temporal data management. Over these structures, a rich temporal algebra and specialised spatiotemporal indices are defined. In addition, the MobilityDB data model enables a *lossless* data compression by removing redundant measures when the value does not change or when it can be derived from other values with linear interpolation. This process, called *normalisation*, enables high compression rates on real-world data, which also enhances the performance.

Finally, expressing spatiotemporal processing in a high-level language such as SQL has multiple advantages. These include, from the user perspective, a better abstraction level for formulating complex spatiotemporal pipelines such as those

implemented in Algorithm 2, and from the implementation perspective, that the database management system can perform efficient query processing by estimating the best execution plan for the SQL queries.

The structure of the table storing the vessel trips was already presented in Section 3.3. Since we now aim to model underwater noise, two additional columns need to be added: the source level estimation of the vessel (SL), computed at each instant (as it depends on the vessel's speed and activity), and the propagation radius, which varies according to the source level, the absorption coefficient, and the ambient noise of the cell where the vessel is located. Both attributes are of type `tfloat`.

To improve the spatial operations on trajectories, like `ST_Intersects`, we add a spatial index on the attribute `traj` of the table `vessel_trip`:

```
CREATE INDEX Vessel_Trip_Geom_Idx ON vessel_trip USING gist(traj);
```

Indexing speeds up searching by organising the data into a search tree which can be quickly traversed to find a particular record. The spatial index structure used is R-Tree.

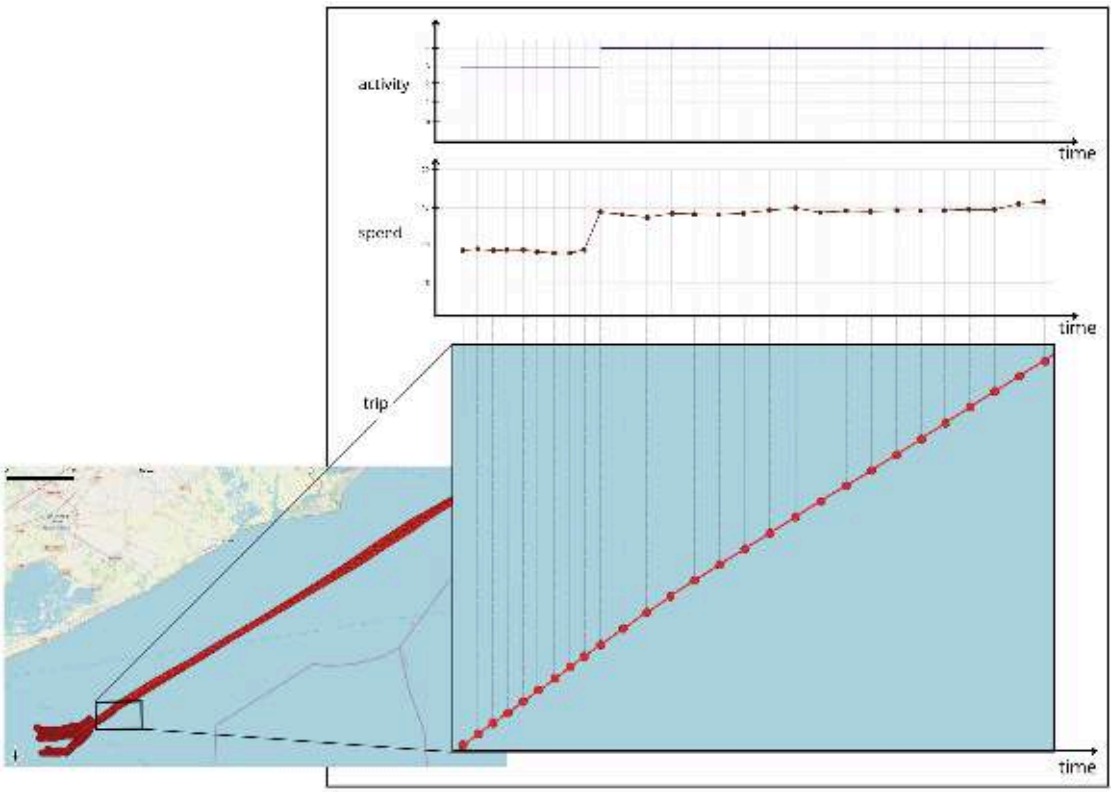


Figure 4.2: Representation of the three temporal types `activity`, `speed` and `trip`.

Once reconstructed the trajectories from the AIS data, we need to get the values of all the trajectories at the same time instants, every 60 seconds. MobilityDB offers an efficient function, `tsample()`, to sample a temporal value according to period buckets. We apply such a function to get speed, activity and position every minute, i.e., `tsample(speed,'1 min')`, `tsample(activity,'1 min')`, `tsample(trip,'1 min')`, and these values are inserted into the table `vessel_trip` for the attributes `speed`, `activity` and `trip`, respectively. In this way the three temporal values are built on the same set of minutes, which is $\mathbb{T}$. Figure 4.2 displays an example of the values for the attributes `activity`, `speed` and `trip`. The vertical lines represent the time instants, the three attributes are synchronised and the sample period is one minute. `Trip` represents the movement of the fishing vessel, which consists of a sequence of points whose distance between consecutive points becomes larger because the speed increases. In fact the temporal value `speed` shows a variation from 10 kn up to 15 kn. This change of speed causes a shift in activity as well: the attribute `activity` varies from 3 denoting *fishing* to value 4 indicating *navigation*.

In order to compute the total received noise level for each cell of our grid at a given frequency f, we proceed as specified by Algorithm 2 and illustrated in Figure 4.3. For each spatiotemporal point p belonging to `trip` we compute the propagation radius r. By using the function `ST_Buffer` we build, around the spatial coordinates of p, a buffer b with radius r (Step 2 in Figure 4.3). Then, we select all the cells whose centroids are inside b through the predicate `ST_Intersects` (Step 3 in Figure 4.3) and we compute the distance between the point p and these centroids (Step 4 in Figure 4.3). We use this distance to estimate the transmission loss which allows us to determine the received noise level in the selected cells. By

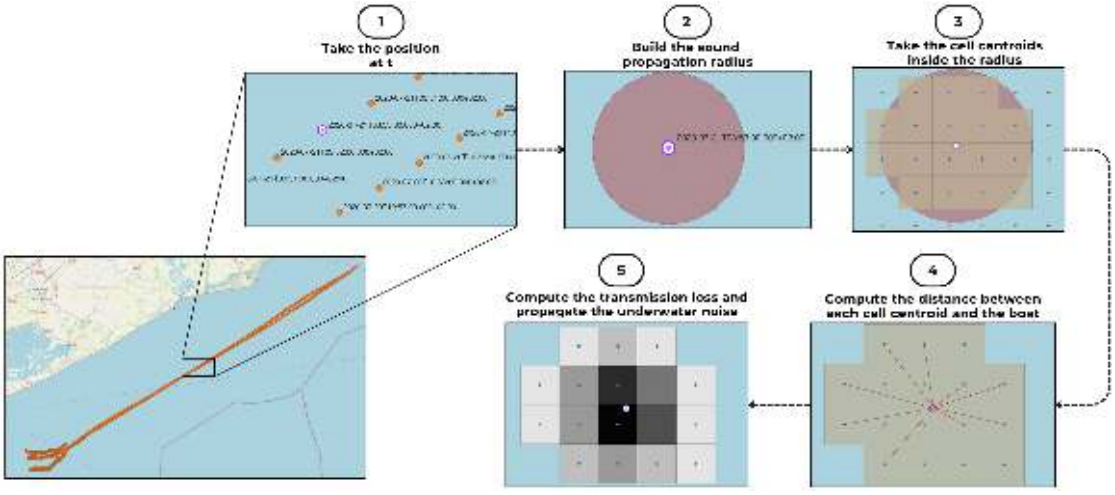


Figure 4.3: Main steps in the calculation of the noise maps.

grouping by cell id and time, we combine all the contributions of the points of the different trajectories through Equation 4.7 (Step 5 in Figure 4.3).

4.4 Analyses and Results

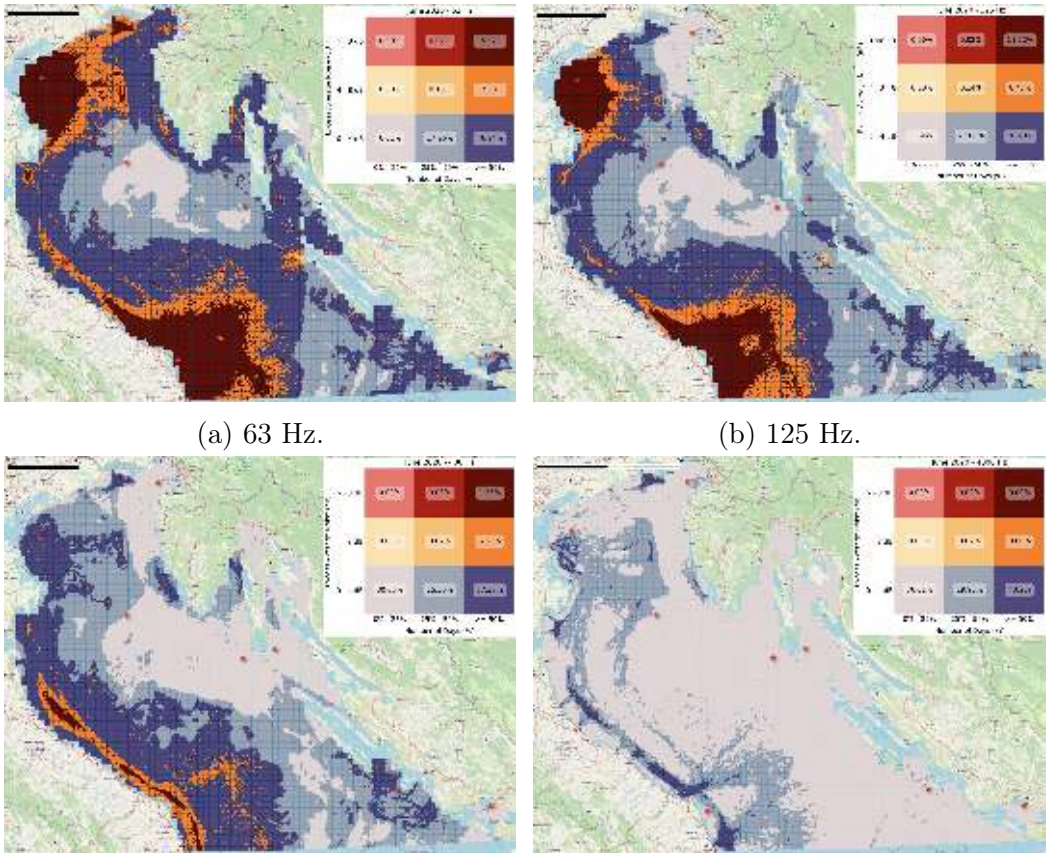
In this section, we illustrate the results of some experiments, which aim at estimating the impact of fishing activities on underwater noise while also demonstrating the expressiveness and flexibility of our approach.

4.4.1 Underwater Noise Maps at Different Frequencies

The first experiment consists of assessing the impact of the fishing activity on the underwater noise level considered at different frequencies. Our framework allows us to perform this analysis according to various time granularities, e.g., yearly, seasonally, monthly or daily. We chose to focus on June 2020 because it is one of the months with the highest fishing activity in that year. For this period, the AIS dataset consists of 9,841,079 records, belonging to 642 fishing vessels (see Table 3.1).

As explained in Section 3.2.1, we consider the frequencies of 63 Hz, 125 Hz, 400 Hz, and 4,000 Hz. For each frequency, we created a bivariate map for June, illustrating two variables at once. The first variable represents the average underwater noise that exceeds the ambient noise, while the second indicates the percentage of days in the period of analysis (1 month) in which a cell is active. For a cell c, a day d is called *active* if the received sound level on d in c exceeds the ambient noise. Bivariate maps are crucial for understanding underwater pollution, as the damage on living species depends both on noise level and noise persistence. Elevated noise levels over a short duration do not affect marine life as significantly as noise at prolonged intervals, which can have more severe consequences for aquatic ecosystems.

In Figure 4.4, we can observe the bivariate maps for June for the frequencies of 63 Hz, 125 Hz, 400 Hz, and 4,000 Hz. This figure clearly demonstrates that the impact of fishing vessels varies significantly depending on the frequency analysed. These variations stem from several frequency-dependent factors: the initial SPL (SL_0), the absorption coefficient (α), the increased noise generated by the vessel during fishing activities, and the ambient noise, which rises with higher frequencies. For example, at 63 Hz, ambient noise starts at 60.78 dB, whereas the lowest value at 4,000 Hz is 81.07 dB. Figure 4.4d shows that at 4,000 Hz, fishing vessels have a negligible impact on the underwater environment: all the cells are characterised by excess noise levels below 4 dB and only 4% of the study area exhibits persistence greater than 50%, i.e., two weeks (dark blue cells). In the majority of the cells,



(a) 63 Hz. (b) 125 Hz.

(c) 400 Hz. (d) 4,000 Hz.

Figure 4.4: Underwater noise bivariate maps for June 2020. The red stars are the hydrophones of SOUNDSCAPE.

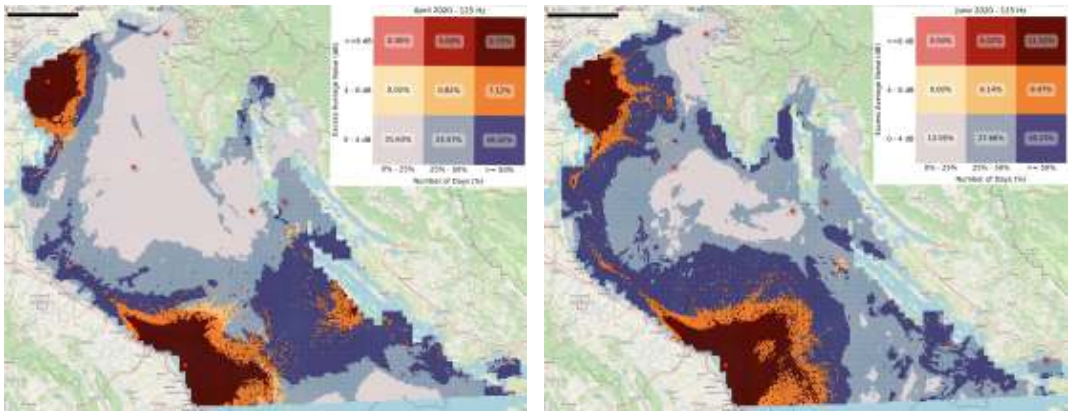
76%, the excess noise has a low persistence, below one week (light pink cells). Figure 4.4c displays that at 400 Hz still 93% of the basin presents excess noise level below 4 dB but now there is an area along the Italian coast, including several harbours, like Rimini and Ancona, extending southward to Civitanova Marche (which marks the southernmost boundary of the map) where the excess noise level is between 4 dB and 8 dB for a persistent period (more than 50% of the days - dark orange cells) reaching the peak over 8 dB in a very limited zone (only 1.35%). Besides, the area with low excess noise level and low persistence, depicted in light pink, is less than half (around 31%) of that obtained for the frequency 4,000 Hz.

Figure 4.4a and Figure 4.4b illustrate that at 63 Hz and 125 Hz, fishing vessels have a significant impact on the ecosystem. In fact, more than 20% of the basin, in areas of the Northern Adriatic Sea that are most frequently used for fishing, such as in front of Venice and near Ancona and along the Italian coast, the excess noise

levels are over 4 dB with large persistence (more than two weeks — dark orange and dark red cells). At 63 Hz there is the highest percentage of cells (almost 16%) with the highest excess noise levels (greater than 8 dB) for a long time (over two weeks — dark red cells) and the lowest percentage (6.6%) of cells with excess noise levels under 4 dB for less than 1 week (light pink cells).

4.4.2 Effect of the COVID-19 Pandemic

In order to investigate the effect of the COVID-19 pandemic outbreak on underwater noise due to fishing activities in the Northern Adriatic Sea, we focus on April and June 2020. While April 2020 is a month during the lockdown, June 2020 represents the post-lockdown period, during which fishing activities gradually returned to pre-pandemic levels, reflecting the easing of restrictions and a progressive resumption of usual activities. Looking at Table 3.1, we observe that the number of AIS data in April is limited: only 5,392,677 records, slightly more than half of the AIS data in June, despite April typically being a period of intense fishing activity. To compare the during-, and post-lockdown periods, we generated two bivariate maps representing the average underwater noise at 125 Hz that exceeds the ambient noise, along with the persistence of this noise (i.e., the percentage of active days for each cell w.r.t. the total days of the month). Figure 4.5a illustrates the excess noise map for April, during the lockdown. There is a large portion of the basin, i.e., 82%, with an excess noise level below 4 dB and for 31% of the study area there is also a low persistence (below one week — light pink cells), thus resulting in a very silent sea. The noisiest areas, with excess noise level over 8 dB for a persistent period (more than half a month — dark red cells), are located in



(a) April. (b) June.

Figure 4.5: During, and post-Covid underwater noise at 125 Hz for the months of April, and June 2020.

the Venice zone, including Chioggia, and from Ancona to the southernmost part of the map. There is also a zone in front of the Croatian coast which shows an increase of noise (excess level between 4 dB and 8 dB and even more 8 dB in several cells) for a persistent period (more than half a month). It is noteworthy that the cells with higher noise levels are located along the coast, and this confirms a redistribution of the fishing grounds, being mainly located near the coasts and in the proximity of the origin harbours as documented in [118]. This behaviour could be due to the possibility to reduce time at sea, limiting the fuel consumption and the related costs. In June (Figure 4.5b), there is a significant increase in fishing activities w.r.t. April, as witnessed by the number of AIS data and trips (see Table 3.1). In the two mentioned zones, near Venice and near Ancona, we observe a boost in the number of cells characterised by elevated noise levels and high persistence (around 3%, dark orange and dark red). Notably, vessels ventured further offshore towards Croatia. Additionally, increased activity is evident in the region near Rimini, indicating greater exploitation of the area. In fact, in June only 14% of the basin is characterised by low noise levels and low persistence (light pink cells). Conversely, cells with noise levels below 4 dB but with persistence greater than 50% (dark blue cells) increase by 12%. These maps clearly point out that the limited fishing activity during the lockdown has caused a reduction in underwater noise, thus proving the acoustic impact of fishery in the Northern Adriatic Sea.

4.4.3 Focus on the Most Intense Fishing Days

In order to deepen the analysis on the areas with the highest noise pressure in the month of June, we investigate how the fishing activity is performed along the days of the week. Figure 4.6 illustrates the number of AIS data in June 2020 grouped by the day of the week. The plot clearly indicates that fishing vessels

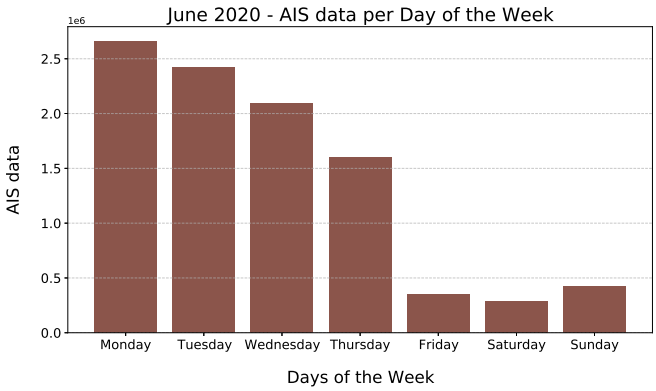
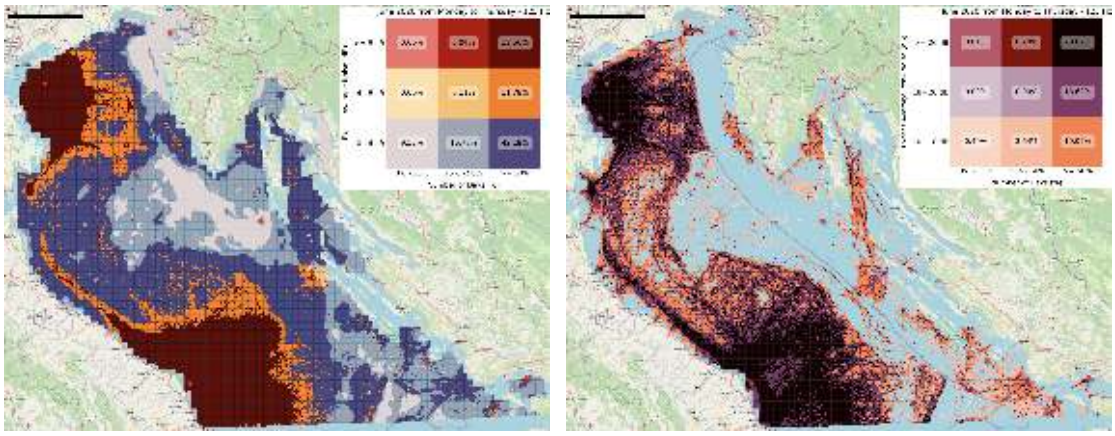


Figure 4.6: Number of AIS data per day of the week in June 2020.

concentrate their activity between Monday and Thursday whereas in the weekend a very limited number of vessels go fishing. Hence, we restrict our analysis to the days from Monday to Thursday, using only AIS data within this time frame. The resulting dataset includes 89% of the June data while covering 60% of the days. The possibility to analyse data at different granularity levels is a great advantage offered by our tool.

Figure 4.7a presents the bivariate map illustrating the average noise that exceeds the ambient noise, along with the percentage of days during the week (from Monday to Thursday) when the cells are active. This analysis allows us to highlight the areas swept by fishing vessels during the four days of their highest fishing activity. Notably, we observe an increase in excess noise levels generated during these days and a reduction of the quiet cells. Compared to Figure 4.5b, which considers all days of the week, there is a 15% increase in areas where persistence exceeds 50%: the extension of the area with excess noise levels over 8 dB (dark red) is augmented by 8% while the other two areas, i.e, between 4 dB and 8 dB (dark orange) and below 4 dB (dark blue) are enlarged by about 3.5%. On the other hand, the number of cells with excess noise levels below 4 dB and for less than 25% of days decreases by 4.4%. Clearly, the two regions most exploited by fishing vessels — the area in front of Venice and the entire area around and in front of Ancona — remain the same, although with greater noise intensity as well as increased persistence.

To further investigate the noise generated by vessels on these days, we focused on the noise peaks associated with each cell. In particular, Figure 4.7b presents a bivariate map where the first variable represents the mean of daily peaks in June



(a) Average underwater noise. (b) Underwater noise peaks.

Figure 4.7: Underwater noise in June 2020 at a frequency of 125 Hz from Monday to Thursday.

2020 (on a scale starting from 10 dB), while the second variable represents the percentage of active days throughout the month, restricted to Monday through Thursday. Considering the average peaks for the month, the resulting noise is significantly higher; in fact, the bivariate map represents only values exceeding 10 dB. We observe that 19% of the basin includes values between 10 dB and 18 dB, 18.6% are characterised by underwater noise levels between 18 dB and 26 dB, and finally, 20% exhibit noise levels greater than or equal to 26 dB, all characterised by a high persistence. Only 37% of the study area has a peak less than 10 dB. This figure clearly points out that the area in proximity of a harbour is noisier and in particular the noisiest zones are located at the south of Venice, in front of Chioggia, as well as the entire zone of Ancona and southward close to Civitanova Marche. Moreover, this map allows us to recover the main routes of the fishing activities.

4.4.4 Interactive Visualisation of the Underwater Noise Dynamics

Finally, our implementation provides also the possibility of visualising the spreading of underwater noise in time for a set of vessels. By using *QGIS TimeManager*, it is possible to generate animations which visualise the noise propagation determined by the vessels moving in the Northern Adriatic Sea. The user can choose the boats according to several criteria, such as the range of horsepower, the MMSI, the length overall, or the activity, and the time window of the analysis. In Figure 4.8 we can observe a different sound propagation at 125 Hz depending on the engine power of the vessel, its speed and its activity.

We focus on four vessels: vessel A has engine power 590.9 Hp, vessel B 613 Hp and vessel C 649.9 Hp (all three with the same SL_0 133 dB), while vessel D has engine power 215.7 Hp (SL_0 130 dB). We can observe that both vessel A and B are fishing (red dot), but vessel A is moving at a speed of 5.67 kn, while vessel B is moving at 2.5 kn. It is worth to remark that the sound propagates across more cells (covering more kilometres) for vessel A compared to vessel B, illustrating how speed affects sound propagation. Vessel C has the same source level as vessels A and B but is not fishing (green dot) and is moving at a speed of 10.75 kn. Compared to vessel A, despite vessel C is moving at a higher speed, the sound propagates less underwater. Specifically, sound propagates approximately 3 km for vessel C and 5 km for vessel A, highlighting the greater impact of fishing activity compared to the vessel's speed. Finally, vessel D has the same speed as vessel C (10.48 kn), and it is not fishing (green dot), but its SL_0 is 3 dB lower. This leads to some differences in sound propagation, as indicated by the reduced number of cells affected by underwater noise for vessel D compared to vessel C,

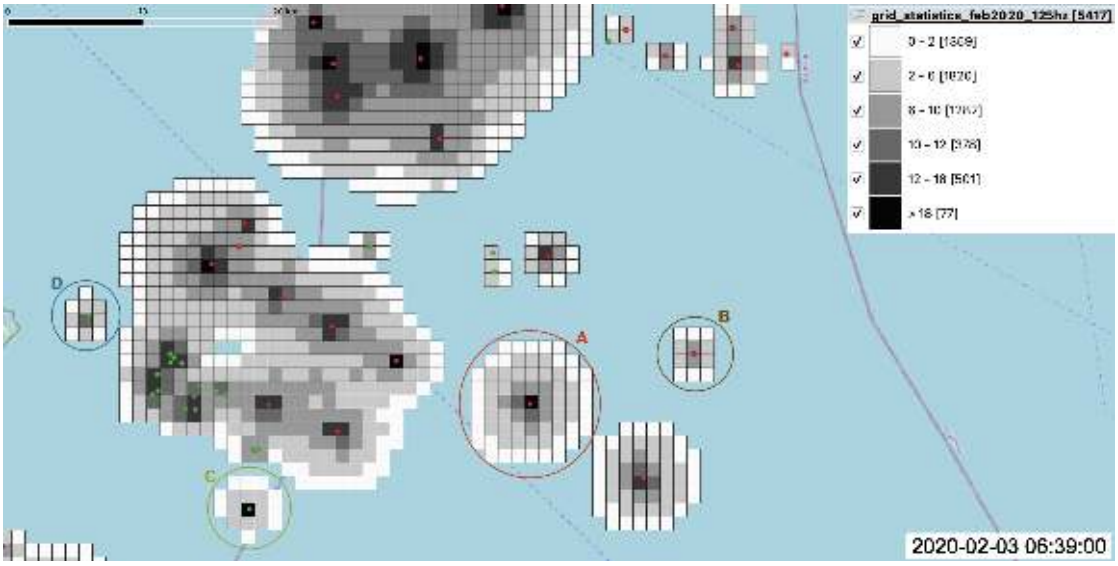


Figure 4.8: Propagation of underwater noise of fishing vessels at 125 Hz in the Northern Adriatic Sea on February 3, 2020, at 06:39 a.m. Vessels marked with a red dot are fishing, while those marked with a green dot are not fishing.

along with the lower noise levels received in those cells. For instance, the cell containing vessel C stores 19.3 dB, whereas the cell containing vessel D records only 11 dB, highlighting a difference of approximately 8 dB. These comparisons show how a fishing vessel generates more substantial underwater noise than a boat merely sailing, even when the latter has a higher speed, as well as how higher vessel speed contributes to greater underwater sound propagation.

Finally, we would like to point out that these are only a few examples of the analyses that can be performed using the spatiotemporal database of semantic trajectories. For instance, we can focus on vessels equipped with specific fishing gear (i.e., LOTB, SOTB, RAP, and PTM) and determine their impact on the underwater noise level. This fine-grained analysis could help to reveal different pollution degrees of fisheries that, in turn, could constitute a basis to implement specific management actions for these activities. Moreover, we can vary our analysis according to different periods and consider only certain sea areas. For instance, one could focus on protected areas, like the Pomo Pit or the Sole Sanctuary.

Chapter 5

Underwater Noise Model Optimisation and Scalability

Assessing underwater noise is crucial for understanding the impact of anthropogenic activities on marine ecosystems, particularly with respect to habitat degradation and behavioural disturbance in marine fauna. In Chapter 4, we introduced a sound propagation model for fishing vessels, and we estimate the impact of fishing activities on underwater noise in the Northern Adriatic Sea. However, the model proved to be computationally demanding. This is especially evident at low frequencies, where sound propagates over larger distances and consequently involves a greater number of cells. As a result, the execution time for frequencies of 63 Hz and 125 Hz reached 44 hours for a single month (June 2020). For this reason, in the present chapter we focus exclusively on the 63 Hz band, which represents the most computationally demanding case and therefore provides a robust benchmark for evaluating the performance gains introduced by the optimised workflow.

Building upon the developed underwater noise model, we introduce several enhancements aimed at improving the efficiency and scalability of its implementation. Specifically, we restructured the underwater noise propagation pipeline, introducing several optimisations aimed at improving performance. These include code-level enhancements, PostgreSQL's native parallel query execution, and table partitioning using range, hash, and list strategies. Furthermore, we implemented spatial tiling with regular, adaptive, and k-d grids, and extended the framework to a distributed environment by leveraging the Citus extension across four nodes (one coordinator and three workers). The experimental evaluation demonstrated that these optimisations substantially reduced execution time while preserving the accuracy of the original underwater noise model.

To demonstrate the potential of the optimised system, we extend the analysis to a broader area that includes both the Northern and Central Adriatic Sea, thereby covering the entire Adriatic region as defined by GSA 17, which corresponds to

more than twice the area considered in the analysis of the previous chapter. In this extended analysis, we employ a considerably larger dataset incorporating AIS data for the full vessel fleet operating in 2020, including cargo, passenger, recreational, and additional categories (see Section 3.2.1). On one hand, the inclusion of all vessel categories required revising the procedure used to estimate the source-level noise. When only fishing vessels were considered, source levels were derived from engine horsepower. Since this information is not available for the other vessel categories, the present study estimates noise levels based on the vessel's category and its length overall (LOA), following the model proposed by MacGillivray and de Jong [75]. On the other hand, the substantially larger volume of AIS data, poses significant challenges for the scalability of our approach and requires further exploration of appropriate optimisation strategies. Moreover, the inclusion of all vessel categories enabled an initial validation of the modelled noise levels in the hydrophone cells against the SOUNDSCAPE measurements during the entire year 2020.

This chapter is organised as follows. Section 5.1 presents the underwater sound model used in this chapter and, in particular, the refinements introduced with respect to the model described in the previous chapter to account for all vessel categories. Section 5.2 discusses several code optimisations applied to the sound propagation computation. Section 5.3 focuses on the implementation of data-partitioning techniques in PostgreSQL and examines the integration of the Citus extension to enable distributed processing. Section 5.4 provides a preliminary validation of our model w.r.t. the SOUNDSCAPE measurements. Finally, Section 5.5 presents a set of analyses assessing the contribution of different vessel types to underwater noise across the Northern and Central Adriatic Sea.

5.1 Underwater Noise Model for all Vessel Types

In this section, we describe how the underwater noise propagation model presented in Chapter 4 has been refined to ensure its applicability across all vessel types. Adapting the model to different vessel categories requires revising the procedure used to estimate the source level (SL). When the analysis focused only on fishing vessels, the source level was derived from the engine horsepower. Since this information is not available for the other vessel categories, we instead estimate the source level based on vessel type and length overall (LOA), following the empirical model by MacGillivray and de Jong [75]. We also modified such a model to overcome its limitations at low speed and to consider the increased noise during the fishing activity.

The source level represents the intensity of the acoustic energy emitted by a vessel and constitutes a key component of underwater noise modelling. As just men-

Vessel Class (C)	**AIS Ship Type ID**	$V_{\mathbf{C}}$ **(kn)**
Fishing	30	6.4
Tug	31, 32, 52	3.7
Naval	35	11.1
Recreational	36, 37	10.6
Government/Research	51, 53, 55	8.0
Cruise	60–69 (length $LOA_{\text{ref}} > 100$ m)	17.1
Passenger	60–69 (length $LOA_{\text{ref}} \leq 100$ m)	9.7
Bulker (Cargo)	70, 75–79 (speed $V_{\text{ref}} \leq 16$ kn)	13.9
Containership (Cargo)	71–74 (all speed); 70, 75–79 (speed $V_{\text{ref}} > 16$ kn)	18.0
Tanker	80–89	12.4
Dredger	33	9.5
Other	all other type IDs	7.4

Table 5.1: Classification of vessel classes (C) with corresponding AIS ship type IDs, and reference speed V_C (in knots), as in Table 1 of [75].

tioned, to compute the SL for all vessel types, we relied on the JOMOPANS–ECHO source level model proposed by MacGillivray and de Jong [75]. This model was developed within the framework of the Joint Monitoring Programme for Ambient Noise in the North Sea (JOMOPANS),[1] aiming at improving underwater acoustic mapping derived from AIS data. The model was designed to address the significant uncertainties involved in relating the limited information available from AIS data to vessel noise emissions.

The JOMOPANS-ECHO model was developed as an updated reference spectrum model, designed to preserve the accurate speed and length dependencies of the RANDI 3.1 model [12] while incorporating critical modifications to enhance its alignment with the measured source levels (validation data) from the ECHO dataset [125]. The resulting model calculates the ship source level spectrum, in one-third octave bands, as an explicit function of frequency (f), vessel speed (V), vessel length (LOA), and AIS ship type.

One of the most significant adjustments introduced by the JOMOPANS–ECHO model concerns the vessel speed: the generic reference speed $V_0 = 12$ knots in RANDI 3.1 is replaced by a reference speed per vessel class (V_C). This new reference speed V_C was determined by minimising the mean residual difference between model predictions and measured broadband source levels per vessel class, leading, for example, to V_C set to 18.0 kn for container ships and 6.4 kn for fishing vessels (see Table 5.1).

The model also incorporates an updated, class-specific reference spectrum de-

[1] `https://northsearegion.eu/jomopans/`

veloped individually for each vessel type by minimising model-data residuals across one-third octave bands. For certain vessel categories such as container ships, bulkers, and tankers, the revised spectrum includes an additional peak below 100 Hz, reflecting low-frequency energy components observed in the measurements. The classification of vessels is explicitly handled by integrating AIS information. The vessel class (C) is assigned on the basis of the AIS ship type ID and, where necessary, refined using additional criteria such as vessel length or operational speed.

The source level (in dB re 1 μPa $\cdot$ m) is calculated as in Equation (5.1). In this formulation, f is the centre frequency of the one-third octave bands (Hz), V_C (knots) is the reference speed associated with the vessel class (see Table 5.1), V (knots) is the vessel speed, $\mathrm{LOA_C}$ (in metres) denotes the reference length overall, set to 91.44 metres for all vessel types, LOA (in metres) is the vessel's actual length overall, and *act* is the activity of the vessel.

$$\begin{aligned}
\mathrm{SL} = {} & K - 10 \cdot (B+2) \cdot \log_{10}(f_1) \\
& + 5 \cdot B \cdot \log_{10}(f) \\
& - 10 \cdot \log_{10}\left[\left(1 - \left(\frac{f}{f_1}\right)^{0.5\cdot(B+2)}\right)^2 + D^2\right] \\
& + 60 \cdot \log_{10}\left(\frac{\underline{V}}{V_{\mathrm{C}}}\right) + 20 \cdot \log_{10}\left(\frac{LOA}{LOA_{\mathrm{C}}}\right) \\
& + 10 \cdot \log_{10}(0.231 \cdot f) \\
& + G
\end{aligned} \tag{5.1}$$

The parameters in the source level estimation are defined as follows.

$$\underline{V} = \begin{cases} V & \text{if } V \geq V_{\min} \\ V_{\min} & \text{otherwise} \end{cases}$$

$$K = \begin{cases} 208 & \text{if } f < 100 \text{ Hz and the ship is of type Cargo or Tanker} \\ 191 & \text{otherwise} \end{cases}$$

$$B = \begin{cases} 2 & \text{if } f < 100 \text{ Hz and the ship is of type Cargo or Tanker} \\ 0 & \text{otherwise} \end{cases}$$

$$f_1 = \begin{cases} 600 \cdot \dfrac{1}{V_{\mathrm{C}}} & \text{if } f < 100 \text{ Hz and the ship is of type Cargo or Tanker} \\ 480 \cdot \dfrac{1}{V_{\mathrm{C}}} & \text{otherwise} \end{cases}$$

$$D = \begin{cases} 0.8 & \text{if } f < 100 \text{ Hz and the ship is of type Cargo} \\ 1 & \text{if } f < 100 \text{ Hz and the ship is of type Tanker} \\ 4 & \text{if the ship is of type Cruise} \\ 3 & \text{otherwise} \end{cases}$$

$$G = \begin{cases} inc_f & \text{if the ship is of type Fishing and } act = \textit{fishing} \\ -15 & \text{if the ship is of type Cruise or Passenger or Cargo or Tanker and} \\ & act = \textit{in port} \text{ and } V \leq 0.5 \\ 0 & \text{otherwise} \end{cases}$$

The above equation extends the formula presented in [75] in three ways. First of all, to avoid an excessively-negative contribution of the speed term in the JOMOPANS–ECHO model at very low vessel speeds, a lower bound needs to be imposed. The fact that the JOMOPANS-ECHO model as such cannot be applied at very-low values of speed is evident from the form of the speed term, that diverges to $-\infty$ as V goes down to zero. Such model limitation is also evident in Figure 5 of MacGillivray and de Jong [75], reporting predicted and observed source levels versus speed, in the 63 Hz third-octave band. In the present study, the source model is adjusted by adopting, for all vessel classes, a floor speed value $V_{\min}$ of 5 knots. For instance, for fishing vessels $V_{\mathrm{C}} = 6.4\,\mathrm{kn}$, and thus the maximum decibel reduction due to the speed correction, ΔS, is computed as $\Delta S = 60 \cdot \log_{10}(V_{\min}/V_C) = -6.43$ dB. This implies that, for fishing vessels operating below 5 kn, a constant correction of -6.43 dB is applied, whereas above this threshold the standard formulation is used. This formulation ensures that the model remains physically consistent at low speeds, preventing unrealistic reductions in source level (Parameter $\underline{V}$ in Equation (5.1)).

Second, as described in Section 4.2.2, if the ship is of type *Fishing* and it is trawling, it typically produces higher levels of radiated noise compared to free-running vessels operating under the same machinery settings. Accordingly, depending on the frequency f, an inc_f increase is applied for vessels when they are trawling (First case of Parameter G in Equation (5.1)). In particular, for our study we consider only the frequency of 63 Hz, thus the increase is 5 dB.

Finally, for container ships, tankers, cruise and passenger vessels (i.e., those with AIS ship type identifiers between 60 and 89), we retained the points located within ports even when the recorded speed was below 0.5 knots (as described in Section 3.2). This choice was motivated by the fact that these vessels usually do not switch off their engines after being berthed; more precisely, their main propulsion systems are inactive, but the auxiliary engines remain operational to supply electrical power and other essential onboard services. For this reason, the noise was computed as the source level (SL) reduced by 15 dB [143], to account for the

fact that these vessels do not have active propellers when stationary, and the generated noise originates primarily from auxiliary machinery rather than propulsion activity (Second case of parameter G in Equation (5.1)).

5.2 Noise Modelling Optimisation

The implementation of the underwater noise model presented in Section 4.3 has been revised to incorporate the updated methodology for deriving the source level. Accordingly, the value of SL_0 is now obtained using Equation 5.1, whereas the calculations of transmission loss, ambient noise, and received noise remain unchanged. Building upon this framework, we introduce a set of optimisations aimed at improving efficiency, enhancing scalability, and reducing computational overhead.

5.2.1 Framework Overview

First of all, we extend the analysis to include AIS data from all vessel categories for the spring season (May–June 2020), rather than only fishing vessels from June 2020. This period was selected because it contains the largest volume of AIS data, and, additionally, SOUNDSCAPE measurements are available throughout the entire interval. Specifically, the dataset includes 3,778 vessels of different types, generating approximately 49.6 million AIS records. Since the AIS data are limited to the Northern and Central Adriatic Sea, we use the projected coordinate system for Italy, with the spatial reference identifier (SRID) 6876. In this analysis, we consider only the 63 Hz frequency, as it is the most computationally demanding and therefore provides an appropriate benchmark for optimisation.

As in the previous model, we partitioned the Northern and Central Adriatic Sea into a regular grid of 1 km $\times$ 1 km cells, enriched with environmental attributes such as sea depth, absorption coefficient, and ambient noise levels derived from the SOUNDSCAPE project. AIS data were used to reconstruct vessel trajectories, which were semantically enriched with information on vessel type, length overall (LOA), and the activities conducted along their paths, which are used to estimate the source level of the vessels. The reconstruction of the trajectories and their semantic enrichment leverage the temporal and spatiotemporal types of MobilityDB, as well as the functions provided by this spatiotemporal database. Subsequently, as described in Section 4.3, using the spatiotemporal functions of MobilityDB, we apply a sampling process on the vessel's trajectory at one-minute intervals to determine the boat's positions at specific temporal instants. For each position p, we estimate the decibels produced by the vessel, based on its LOA, its type, its activity and speed. Next, we calculate the propagation radius r, i.e., the distance at which the noise generated by the vessel gets drowned into ambient

```
SELECT b.point_id,g.grid_id,b.time,b.sl,b.mmsi,g.depth,
  valueAtTimestamp(g.alpha,b.time::DATE) AS alpha,
  valueAtTimestamp(g.ambient_noise, b.time::DATE)
    AS ambient_noise,
  SQRT(POWER(b.longitude - g.centroid_long, 2) +
    POWER(b.latitude - g.centroid_lat, 2)) AS dist
FROM Buffer b, Grid g
WHERE ST_Intersects(b.buffer, g.geom_centroid)
```

Listing 5.1: Returns the cells affected by noise propagation for each point.

noise, and we construct a buffer b with radius r around p. We select all the grid cells whose centroids fall within b and compute the distance between the sampled point p and these centroids. This distance is used to determine the received noise in the selected cells. Finally, by grouping by cell id and time, we combine all the received sound levels to obtain the total noise level to be associated with the cell.

One of the most computationally expensive steps in this workflow is the selection of grid cells affected by noise propagation, performed through Query 5.1. This query executes a `JOIN` between the temporary table `Buffer`, which stores each point p with its associated buffer b, and the table `Grid`, containing the grid cells. A GiST spatial index is defined on the grid centroid geometry column `geom_centroid` to improve the efficiency of the spatial intersection. The `ST_Intersects` operation identifies the cells intersecting each buffer, thereby determining which areas are influenced by the noise generated at p. For each point, the query returns the identifiers `point_id` and `grid_id` of the grid cell whose centroid lies within the point's buffer, and data required to compute the received noise level (RL) at the cell centroid as defined in Equation (4.6).

To reconstruct the trajectory and build our model, we used a distributed infrastructure composed of four virtual machines, each equipped with an AMD EPYC 7413 24-Core Processor (16 active cores running at 2.00 GHz), 32 GB of DDR4 RAM, and 150 GB of local storage, running Ubuntu 24.04 LTS. An additional 5 TB storage volume is provided for database hosting. The system hosts PostgreSQL 16.6, PostGIS 3.5.2, and MobilityDB 1.3.0, compiled with GEOS 3.12.1, PROJ 9.4.0, JSON-C 0.17, and GSL 2.7.1. For the spring season, trajectory reconstruction and enrichment take approximately 4 hours, whereas the pipeline for computing underwater noise propagation is considerably more time-consuming, requiring about 2 days 21 hours and 49 minutes (69 hours and 49 minutes). In Figure 5.9, this configuration is referred to as *Original Pipeline*. Notice that it is performed on a single machine (the coordinator node) without parallel or distributed processing.

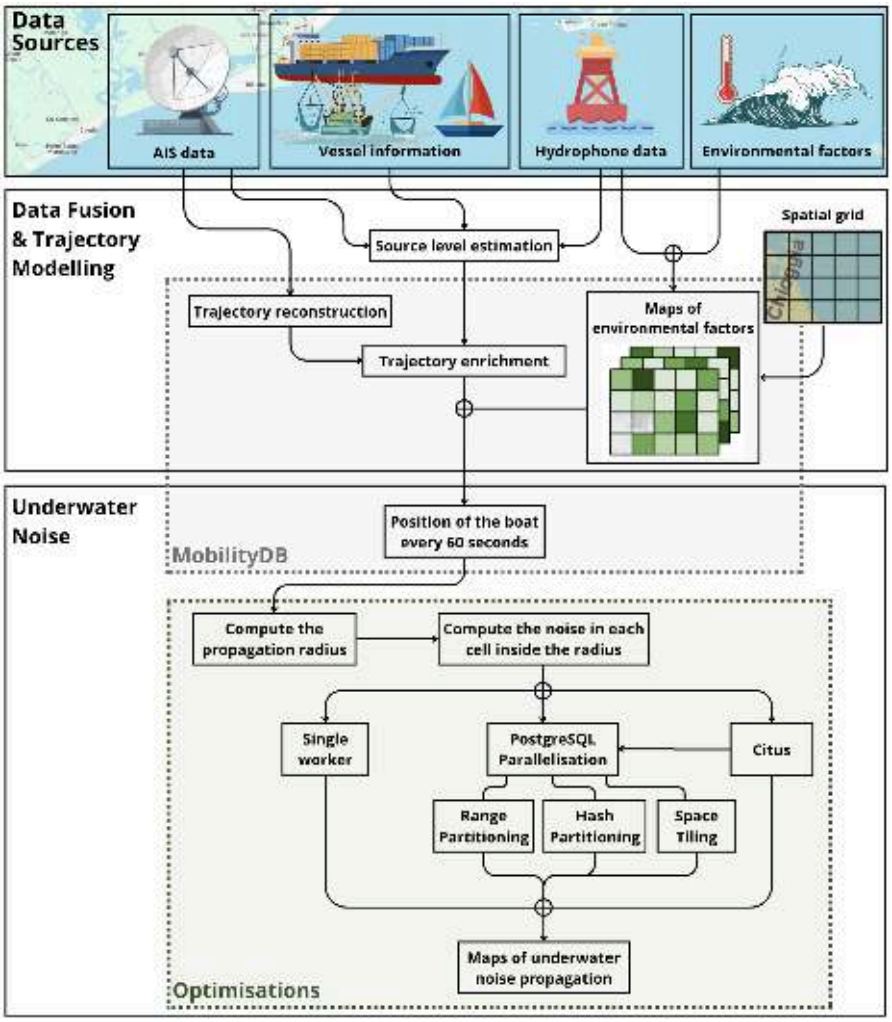


Figure 5.1: Overview of the workflow divided into three stages, with MobilityDB handling spatiotemporal processing and PostgreSQL/Citus managing optimisation and parallelisation.

Due to the high execution time required for underwater noise computation, we introduce several optimisations aimed at improving both efficiency and scalability (see Figure 5.1). First, the table storing the spatiotemporal grid of the Adriatic Sea was restructured and optimised, as detailed in Section 5.2.2. This optimisation significantly improved the efficiency of determining the grid cells affected by vessel noise propagation. Moreover, we applied a set of optimisation techniques and compared their performance. Specifically, we evaluated:

- PostgreSQL's native parallel query execution on the optimised pipeline (see Section 5.3.1).

- PostgreSQL's range partitioning on time and hash partitioning on MMSI, considering an increasing number of partitions (2, 4, 8, and 16), as detailed in Section 5.3.2.
- Spatial tiling using regular and adaptive grids, as well as the k-d tree partitioning method with an increasing number of partitions (2, 4, 8, and 12), see Section 5.3.3.
- Citus extension for distributed computation across four nodes (one coordinator and three workers), and we also employed Citus together with PostgreSQL's range partitioning on the time dimension and k-d tree spatial tiling (see Section 5.3.4).

Note that, Citus was deployed across all four machines — one acting as the coordinator and three as workers — to enable distributed query execution. For all other configurations, including those involving algorithmic optimisation, PostgreSQL-based partitioning, and space tiling, only a single machine (the coordinator) was used. These enhancements collectively enable efficient processing of large-scale spatiotemporal datasets and substantially reduce execution time, while maintaining the accuracy of the original underwater noise model.

5.2.2 Algorithmic Optimisation

The first improvement to the original pipeline involves a code optimisation that removes the `ST_Intersects` operation from Query 5.1. Since `ST_Intersects` relies on geometry types and requires computationally intensive spatial operations, its removal enhances model performance. To avoid this computational overhead, we make two significant changes: (*i*) restructure the table `Grid`, and (*ii*) use a bounding box instead of a buffer in noise propagation.

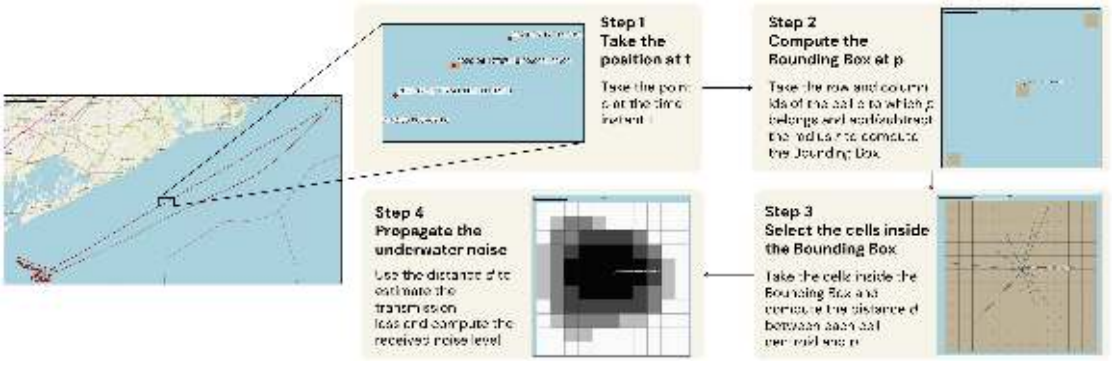


Figure 5.2: Main steps in the calculation of the noise maps.

Grid Table Restructuring. We add two new attributes to the cells of the grid: `grid_r` and `grid_c`, which indicate the row and column numbers within the grid. Hence, starting from the lower-left corner, the grid cells are numbered sequentially, so they are identified as $(1, 1)$, $(1, 2)$ and so on. This grid-based system allows for an efficient identification of the cells within a bounding box, without the need for costly spatial operations. The table `Grid` additionally contains the longitude and latitude of the cell centroid, the cell depth, the ambient noise level (which varies monthly), and the absorption coefficient α (which exhibits daily variability). These data are used for calculating sound propagation. The structure of the table `Grid` is thus as follows.

```
CREATE TABLE Grid (
  grid_id integer PRIMARY KEY,
  grid_r integer NOT NULL,
  grid_c integer NOT NULL,
  centroid_long double precision,
  centroid_lat double precision,
  depth real,
  ambient_noise tfloat,
  alpha tfloat );
CREATE INDEX idx_grid_r ON Grid (grid_r);
CREATE INDEX idx_grid_c ON Grid (grid_c);
```

Note that we also add two indexes to the `Grid` table, on the columns `grid_r` and `grid_c`, to enhance the efficiency of spatial queries.

Bounding Box for Noise Propagation. To compute the total received noise level for each cell of our grid, we proceed as illustrated in Figure 5.2. After reconstructing the vessel trajectories from the AIS data, we get the positions of all the vessels at the same time instants, i.e., every 60 seconds (Step 1 in Figure 5.2). For each point p, we determine the cell c it belongs to, by comparing the coordinates of p with the grid cell boundaries which are computed by adding or subtracting 500 metres from the coordinates of the cell centroid, thus avoiding the use of the `ST_Intersects`. We calculate the noise generated by the vessel according to Equation (5.1). Then, we compute the propagation radius r (in metres) and we build the *sound propagation bounding box* (Step 2 in Figure 5.2), defined by the minimum and maximum row and column identifiers that enclose all the cells affected by the noise generated by the vessel at p. We store the points with their bounding box in a table `PointBoundingBox` with the following structure. The attribute `point_id` identifies the spatiotemporal point, `mmsi` is the identifier of the vessel the point belongs to, `longitude` and `latitude` are the coordinates of the point, `time` specifies the date and hour of the point, `sl` denotes the decibel level

```
SELECT pbb.point_id,eg.grid_r,eg.grid_c,
  pbb.time,pbb.sl,pbb.mmsi,eg.depth,
  valueAtTimestamp(eg.alpha,pbb.time::DATE) AS alpha,
  valueAtTimestamp(eg.ambient_noise,pbb.time::DATE)
    AS ambient_noise,
  SQRT(POWER(pbb.longitude-eg.centroid_long,2) +
    POWER(pbb.latitude-eg.centroid_lat,2)) AS dist
FROM PointBoundingBox pbb, Grid eg
WHERE eg.grid_r>=pbb.r_min AND eg.grid_c>=pbb.c_min AND
  eg.grid_r<=pbb.r_max AND eg.grid_c<=pbb.c_max;
```

Listing 5.2: Optimisation of Query 5.1.

generated by the vessel at that point, and `ship_cargo_type` represents the type of vessel according to the AIS classification. The remaining attributes represent the row and column identifiers used to construct the sound propagation bounding box.

```
CREATE TABLE PointBoundingBox AS (
  SELECT point_id,mmsi,longitude,latitude,
    time,sl,ship_cargo_type,
    grid_r-radius AS r_min,
    grid_r+radius AS r_max,
    grid_c-radius AS c_min,
    grid_c+radius AS c_max
FROM UnnestTripWithCell );
```

Notice that the boundaries of the bounding box are obtained simply by adding or subtracting the propagation radius (`radius`) from the row and columns identifiers of the cell c, `grid_r` and `grid_c`. Hence, while in the pipeline illustrated in Figure 4.3 we used a circular buffer around the point p, here we build a bounding box starting from the cell the point p belongs to. This can be computed easily by using the row and column identifiers of the grid cell, avoiding the use of the `ST_Intersects` operation, which is very time consuming. Indeed, with this approach, we retrieve the cells involved in the noise calculation in just 3 minutes for the entire dataset of the spring season.

Next, we select all grid cells inside the bounding box and compute the distance between the point p and the cell centroids (Step 3 in Figure 5.2). We use this distance to estimate the transmission loss, which allows us to determine the received noise level in the selected cells. Query 5.1 is reformulated as shown in Query 5.2. It is worth noting that, we avoid the expensive `ST_Intersects` operation by checking whether a cell falls within a point's bounding box through a simple comparison

of the cell's row and column identifiers with those of the bounding box. The last step, Step 4 in Figure 5.2, combines all the contributions of the points of the different trajectories, and it is implemented by grouping by cell id and time. We thus obtain for each cell the received noise level (RL).

The optimised pipeline produces results identical to those of the original version, but with notably improved performance. In the experiment for spring 2020, it completes in 2 days, 13 hours, and 6 minutes (61 hours and 6 minutes), which is approximately nine hours faster than the original pipeline (see Figure 5.9, labelled *Optimised Pipeline*).

5.3 Partitioning and Parallelisation

To further enhance the performance of the optimised pipeline, we present an analysis of various partitioning and parallelisation techniques. In Section 5.3.1, we analyse PostgreSQL's native parallel query execution, which is also exploited in the subsequent optimisations. Section 5.3.2 explores table partitioning techniques in PostgreSQL, applying both range and hash partitioning strategies. In Section 5.3.3, we extend the previous approach by combining PostgreSQL partitioning with multidimensional tiling, focusing on the spatial dimension. Finally, in Section 5.3.4, we leverage the Citus extension of PostgreSQL to apply sharding and take advantage of its parallel query execution capabilities.

5.3.1 PostgreSQL Parallelisation

The execution of Query 5.2, selecting the cells affected by noise propagation for each point p (Step 3 in Figure 5.2), remains a computationally expensive operation. This complexity arises from the `JOIN` operation between the table `PointBoundingBox`, which contains each vessel position along with its sound propagation bounding box, and the table `Grid`, which consists of $101,113$ cells. In our experiment, the table `PointBoundingBox` stores more than 58 million points. Consequently, the `JOIN` command involves a computational effort equivalent to approximately 58 million $\times$ 101 thousand operations, making it inherently costly.

To mitigate this computational cost, we first rely on PostgreSQL's native parallel query execution. PostgreSQL supports parallel query processing, a feature that enables the database engine to exploit multiple CPU cores when executing a single query. This mechanism allows certain operations to be distributed among several worker processes. While not all queries can benefit from parallel query processing, when applicable, it can yield substantial performance improvements, often achieving two- to fourfold speed-ups over single-core execution [52]. We evaluate the impact of PostgreSQL's native parallel query mechanism on the optimised

pipeline described in Section 5.2.2. In this configuration, no additional partitioning was applied. The query planner automatically allocated six parallel workers to the task, resulting in a total execution time of approximately 12 hours and 24 minutes (see Figure 5.9, labelled as *Optimised + Parallelisation*). Compared to the single-core execution time of about 2 days and 13 hours, this represents a fivefold improvement, confirming the effectiveness of parallel query execution in enhancing the efficiency of large-scale computations.

Note that all the optimisations described in the following sections take advantage of the PostgreSQL's native parallel query mechanism.

5.3.2 PostgreSQL Partitioning

In this section, we explore the *table partitioning* technique in PostgreSQL to enhance the execution time of the code of the optimised pipeline. This method consists in dividing a logically large table into smaller physical segments, with each partition being an independent table that stores a specific subset of the original data. PostgreSQL natively supports three forms of partitioning [52]: (*i*) *Range partitioning*, where the table is divided into *ranges* based on a key column or set of columns, with each partition containing non-overlapping ranges of values; (*ii*) *List partitioning*, which explicitly assigns specific key value(s) to each partition, allowing precise control over data distribution; and (*iii*) *Hash partitioning*, where the table is divided by applying a hash function to the partition key.

Table partitioning offers several advantages that significantly improve both performance and data management. It enhances query execution by allowing the database management system to filter out irrelevant partitions, thus speeding up query processing, especially for large datasets. Additionally, partitioning simplifies data management tasks such as archiving, purging, backup and restore operations. Furthermore, data loading is also more efficient since it can be parallelised, and indexing becomes faster as partitions reduce the scope of the data being indexed [120].

Range Partitioning on Time

To enhance the performance of our pipeline, as we have already remarked, we can improve the execution time of the `JOIN` operation between the table `PointBoundingBox` and the table `Grid`. To accomplish this task we partition the table `PointBoundingBox`. We apply range partitioning to the table `PointBoundingBox` to evaluate its impact on performance, focusing on the trade-off between execution time and parallelisation overhead. To this end, we divide the dataset into 2, 4, 8, and 16 temporal partitions, each evenly spanning the period from April 1 to June 30 2020. We can create the partitioned table as follows.

```
CREATE TABLE PointBb_RangePart(LIKE PointBoundingBox)
  PARTITION BY RANGE(time);
```

Next, we create time-based partitions, each corresponding to a specific time interval. For instance, if we divide the dataset into two temporal partitions, the following tables are created.

```
CREATE TABLE PointBb_RangePart_1 PARTITION OF PointBb_RangePart
  FOR VALUES FROM ('2020-04-01') TO ('2020-05-20');
CREATE TABLE PointBb_RangePart_2 PARTITION OF PointBb_RangePart
  FOR VALUES FROM ('2020-05-20') TO ('2020-07-01');
```

After inserting the data into the partitioned table, the entries are automatically routed to the appropriate partition. Some statistics regarding the number of rows and disk usage for the configuration with two partitions are reported in Table 5.2.

No. Partition	Period	Disk Usage	Rows
1	1 April–19 May	1,908 MB	26,118,413
2	20 May–30 June	2,353 MB	32,218,155

Table 5.2: Statistics for the partitions by range on the `time` column.

Query 5.2 is the query to optimise. We replace the table `PointBoundingBox` with the partitioned table `PointBb_RangePart`. The query plan involves a combination of parallel and sequential scans to optimise the data retrieval process. The first step is a parallel append operation, which processes multiple partitions of the table `PointBb_RangePart` in parallel. Each partition (corresponding to a different time range) is accessed through a parallel sequential scan. The second part of the plan involves a bitmap heap scan on the table `Grid`, where rows are selected based on conditions that compare the grid's row and column identifiers with the corresponding bounding box identifiers from the partitions. Specifically, the query checks that the cells, identified by row `grid_r` and column `grid_c`, lie within the minimum and maximum row and column values of the bounding box. This comparison is optimised through bitmap index scans on `idx_grid_r` and `idx_grid_c`, each filtering the data based on the row and column values. In essence, the query plan performs a parallel scan of partitioned data, followed by an efficient indexed search of the grid, ensuring faster query execution by narrowing down the relevant data points through partitioning and indexing. The number of workers assigned in the query plan varies according to the partitioning configuration: PostgreSQL allocates six workers when using two partitions, and five workers when using four, eight, or sixteen partitions. This behaviour is driven by the query planner's in-

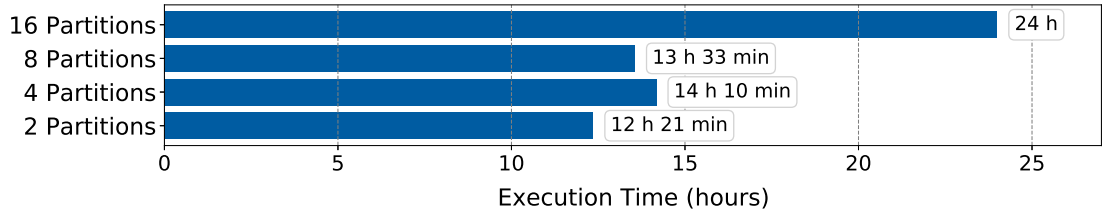


Figure 5.3: Execution times (in hours) for the range partitioning configurations with two, four, eight, and sixteen partitions.

ternal cost estimation, which determines the optimal level of parallelism based on the expected workload and available system resources.

By partitioning the table `PointBoundingBox` with range partitioning on time while leaving the rest of the code of the optimised pipeline unchanged, the entire process completes in 12 hours and 21 minutes with two partitions, 14 hours and 10 minutes with four partitions, 13 hours and 33 minutes with eight partitions, and 24 hours with sixteen partitions. Figure 5.3 illustrates the execution times for each configuration. As shown in the figure, performance does not scale linearly with the number of partitions. Although the system is configured to support up to eight parallel workers, the PostgreSQL planner allocated six workers for the configuration with two partitions and only five for the other partitions. The better performance obtained with two partitions suggests that this configuration achieves the optimal trade-off between minimising partition-management overhead and maximising parallel execution efficiency.

Hash Partitioning on MMSI

As a second partitioning experiment, we use the *hash partitioning* on the `mmsi` column of the table `PointBoundingBox`. As with the range partitioning configuration, we focus on the trade-off between execution time and parallelisation overhead, dividing the dataset into 2, 4, 8, and 16 hash-based partitions. The partitioned table can be created as follows.

```
CREATE TABLE PointBb_HashPart (LIKE PointBoundingBox)
 PARTITION BY HASH(mmsi);
CREATE TABLE PointBb_HashPart_1 PARTITION OF
 PointBb_HashPart FOR VALUES WITH (MODULUS 2, REMAINDER 0);
CREATE TABLE PointBb_HashPart_2 PARTITION OF
 PointBb_HashPart FOR VALUES WITH (MODULUS 2, REMAINDER 1);
```

Next, we insert the values into the table `PointBb_HashPart`, which are automatically distributed across the partitions. Table 5.3 presents some statistics on the number of rows and the disk usage of each partition in the configuration with

two partitions. In this case, we can observe that the data distribution across the two partitions is more balanced compared to the partitions obtained through time-based range partitioning.

No. Partition	Disk Usage	Rows
1	2,438 MB	30,260,981
2	2,262 MB	28,075,587

Table 5.3: Statistics for the partitions by hash on the `mmsi` column.

We now employ the `PointBb_HashPart` table in Query 5.2 in order to optimise its performance. The query plan is the same as that described for range partitioning and consists of a Parallel Seq Scan across the two partitions of the hash-partitioned table and a Bitmap Heap Scan on the table `Grid`. The number of workers assigned in the query plan varies according to the partitioning configuration: PostgreSQL allocates six workers when using two partitions, five workers when using four or sixteen partitions, and four workers when using eight partitions. This behaviour reflects PostgreSQL's adaptive cost-based strategy, which adjusts the number of workers according to workload expectations and resource availability.

By partitioning the table `PointBoundingBox` with hash partitioning on MMSI while leaving the rest of the code of the optimised pipeline unchanged, the entire process completes in 12 hours and 22 minutes with two partitions, 13 hours and 36 minutes with four partitions, 15 hours and 44 minutes with eight partitions, and 24 hours and 3 minutes with sixteen partitions. Figure 5.4 illustrates the execution times for each configuration. As shown in the figure, similarly to the range partitioning results, the configuration with two partitions proves to be the most efficient, suggesting that this setup provides the right balance between partition-management overhead and parallel processing efficiency.

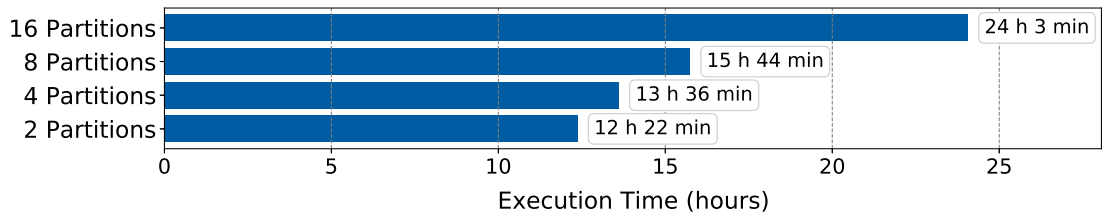
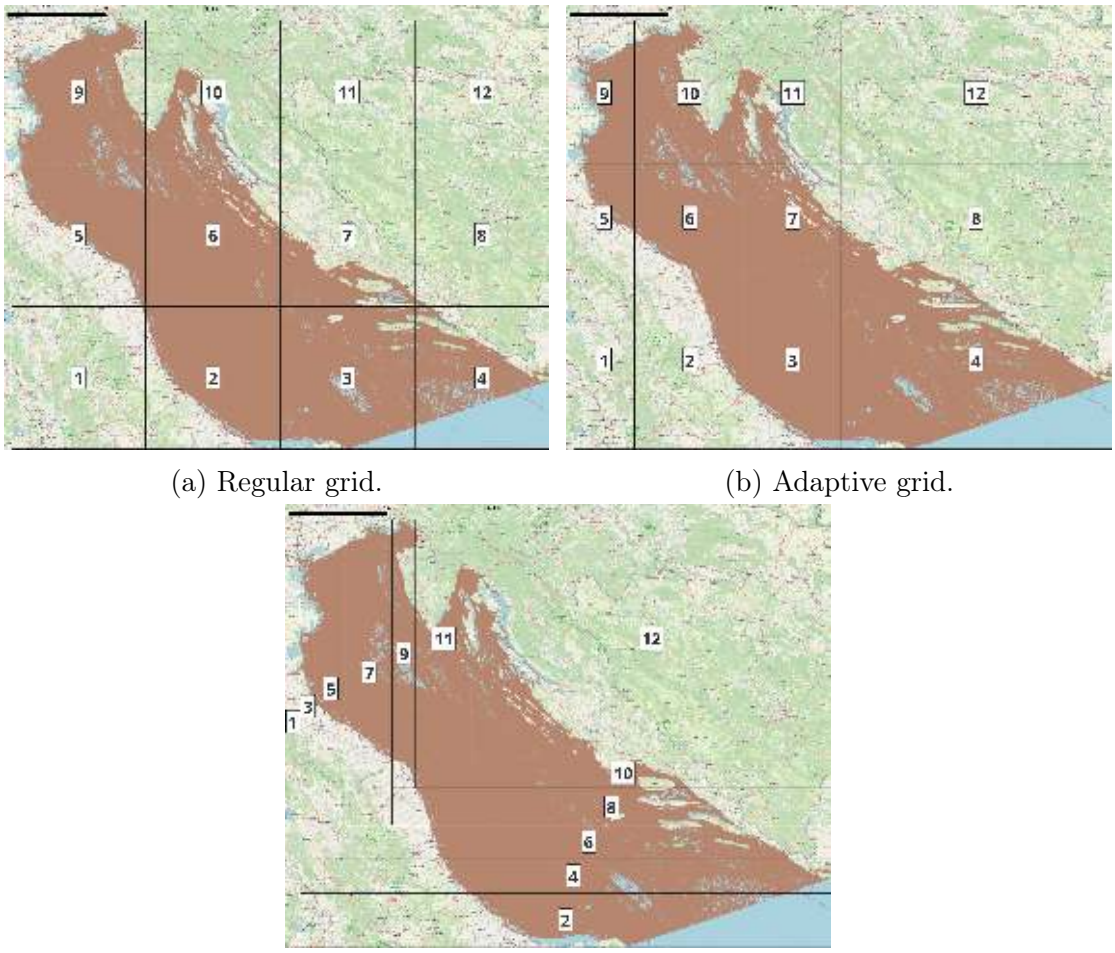


Figure 5.4: Execution times (in hours) for the hash partitioning configurations with two, four, eight, and sixteen partitions.

5.3.3 Space Tiling and Partitioning

Multidimensional tiling is a technique that partitions an n-dimensional domain into tiles of varying dimensions. This approach has several applications. For instance, multidimensional tiling can be applied to partition and/or distribute datasets across a cluster of servers. One key advantage of this partitioning mechanism is that it preserves spatial and temporal proximity, unlike traditional hash-based partitioning methods. This distribution reduces the amount of data that needs to be exchanged between nodes during query processing, a process commonly known as *reshuffling* [120].

In our work, we focus on tiling with respect to the spatial dimension. Specif-



(a) Regular grid.

(b) Adaptive grid.

(c) K-d tiling grid.

Figure 5.5: Partitioning of vessel data with a regular, an adaptive, and a k-d tiling grids.

ically, we partition the positions of vessels based on their spatial locations. We consider three different types of tiling: (*i*) *regular*, where all tiles are of equal size in each dimension; (*ii*) *adaptive*, where the size of the cells adapts to the distribution of vessel points (spatial density); (*iii*) *k-d tree* (or k-dimensional tree), which also adapts to spatial density but divides the space hierarchically, alternating between the X and Y dimensions at each recursive step.

In the first case, we employ a regular tiling, constructing a uniform grid consisting of 4×3 cells, as shown in Figure 5.5a. To generate this grid, we used the MobilityDB function `spaceTiles`. The grid size was manually tuned to balance the trade-off between the number of partitions and the data distribution within each partition. Then, we create the partitioned table along with the corresponding tables for the space tiles, by using the *list partitioning* technique.

```
CREATE TABLE PointBb_RegGrid(LIKE PointBoundingBox)
 PARTITION BY LIST(TileId);
CREATE TABLE PointBb_RegGrid_1 PARTITION OF PointBb_RegGrid
 FOR VALUES IN (1);
```

Only the creation of the first tile is specified. Once the data is inserted into the partitioned table, the entries are automatically directed to their corresponding partitions. The limitation of this type of tiling is that it does not ensure balanced workload distribution across the tiles.

A possible solution to this issue is to use an *adaptive* grid, as illustrated in Figure 5.5b. In this case, we create a grid that divides the area of interest based on the distribution of vessel points in the Northern and Central Adriatic Sea. Specifically, the coordinates of all vessel locations are ordered along each spatial dimension and divided into a predefined number of bins using PostgreSQL's native `NTILE()` window function. This function ensures that each bin contains approximately the same number of points, thereby adapting the grid resolution to the actual spatial density of the data. The resulting X and Y bins are then combined to form spatial tiles, each defined by its lower and upper coordinates and represented as a geometry. The final `AdaptiveGrid` table therefore defines a grid whose cell size varies with data density, providing a more accurate and data-driven representation of the study area compared with the regular grid. Then, we partition the table `PointBoundingBox` according to the adaptive grid structure. The process of creating the partitioned table, along with the corresponding tables for the spatial tiles, follows the same steps as for the regular grid.

An alternative way to achieve a better balance among partitions is to employ a *k-d tiling* with six levels, as shown in Figure 5.5c. To construct the k-d tree, we use the `kdtree_space_partition` function following the procedure described by Sakr et al. [120]. This function takes as input the number of levels, which serves as the stopping criterion for the splitting process, and generates the corresponding k-d

	Regular Grid		Adaptive Grid		K-d tree Grid	
Tile	Disk Usage	Rows	Disk Usage	Rows	Disk Usage	Rows
1	7,840 kB	94,729	0 bytes	0	440 MB	5,455,702
2	622 MB	7,723,244	161 MB	2,003,973	296 MB	3,676,667
3	344 MB	4,268,358	616 MB	7,648,291	369 MB	4,574,386
4	234 MB	2,898,165	789 MB	9,792,606	330 MB	4,101,677
5	700 MB	8,694,421	376 MB	4,664,978	316 MB	3,919,916
6	666 MB	8,263,615	399 MB	4,956,881	329 MB	4,080,463
7	569 MB	7,058,812	406 MB	5,033,523	303 MB	3,765,680
8	9,136 kB	110,420	386 MB	4,790,590	384 MB	4,761,643
9	1,324 MB	16,437,041	799 MB	9,919,706	400 MB	4,965,459
10	225 MB	2,787,763	614 MB	7,623,476	516 MB	6,407,275
11	0 bytes	0	153 MB	1,902,544	403 MB	5,003,764
12	0 bytes	0	0 bytes	0	614 MB	7,623,936

Table 5.4: Statistics for the partitions by list on the `tileId` column, obtained using the regular, adaptive, and k-d tree tiling approaches.

tiles. This tiling is generated using an approach similar to an adaptive grid, but the partitioning rule consists of splitting along one dimension at a time, alternating between the X and Y dimensions at each step, while progressively reducing the bounding box during the partitioning process. Specifically, the first split point is computed along the X dimension over the entire spatial domain. Once the first tile is created, its area is excluded from the remaining region to be partitioned. The next tile is then obtained by computing the split point along the Y dimension, and the process continues alternately for the subsequent tiles. After generating all tiles, we follow the same procedure as for the regular grid: we create the table `PointBb_KdGrid` and the associated tile tables using list partitioning.

Table 5.4 presents statistics on the number of rows in each tile, as well as their respective disk usage, for the regular, adaptive, and k-d tree grids. The table shows that the data partitioned according to the adaptive grid exhibits a more balanced distribution across the tiles compared to the regular tiling. However, certain tiles (specifically, tiles 1, 11, and 12) contain few or even no data points, because they mostly cover the mainland. The k-d tree tiling further improves the balance, achieving the most homogeneous distribution of both data volume and disk usage across tiles, with only minimal variation between partitions.

To optimise Query 5.2, we replace the `PointBoundingBox` table with the appropriate partitioned version — `PointBb_RegGrid`, `PointBb_AdGrid`, or `PointBb_KdGrid` — according to the chosen spatial tiling. The query plan, like the previous ones described in Section 5.3.2, combines parallel and sequential scans to optimise

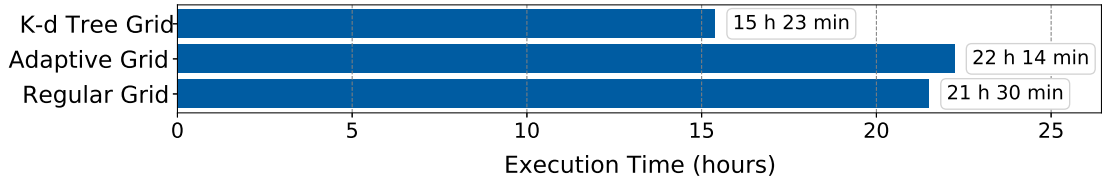


Figure 5.6: Execution times (in hours) for the regular, adaptive, and k-d tree tiling configurations with twelve partitions.

data retrieval. The first step is a parallel append operation, which processes multiple partitions of the specific partitioned table concurrently. This is followed by a bitmap heap scan on the table `Grid`, where rows are selected based on conditions that compare the grid's row and column identifiers with the corresponding bounding box identifiers from the partitions.

Figure 5.6 compares the execution times of the regular, adaptive, and k-d tree tiling configurations, each evaluated with 12 partitions. By tiling the space with the regular grid, the full pipeline is executed in 21 hours 30 minutes, while using the adaptive grid it completes in 22 hours and 14 minutes. Both configurations are slower than the range and hash partitioning approaches (approximately 10 hours), probably because the computational cost associated with handling a large number of spatial tiles outweighs the advantage of enhanced spatial selectivity. In contrast, when using the k-d tree tiling, the pipeline completes in approximately 15 hours, yielding a gain of about 7 hours with respect to the regular and adaptive grid configurations. This improvement can be attributed to the more balanced spatial distribution of data across tiles: unlike the other grids, the k-d tree tiling produces no empty partitions and maintains a relatively uniform number of records per tile. As a result, the workload is more evenly distributed among parallel tasks, reducing the time spent on partitions with sparse data and improving overall query efficiency. The number of workers assigned in the query plan is four for all tiling configurations with twelve partitions.

Since spatial partitioning with the k-d tiling balances the data more effectively across partitions, resulting in the best computational performance among the tiling methods, we also focus on the trade-off between execution time and parallelisation overhead, as in the case of range and hash partitioning. In particular, we divide the dataset into 2, 4, and 8 k-d-based partitions, in addition to the 12-partition configuration already evaluated. Also in this case, the number of workers assigned in the query plan is automatically determined by PostgreSQL: six workers are assigned for the k-d tiling with two partitions, five for the configuration with four and eight partitions, and four for that with twelve partitions. The total execution times are 12 hours and 22 minutes for the two-partition configuration, 13 hours

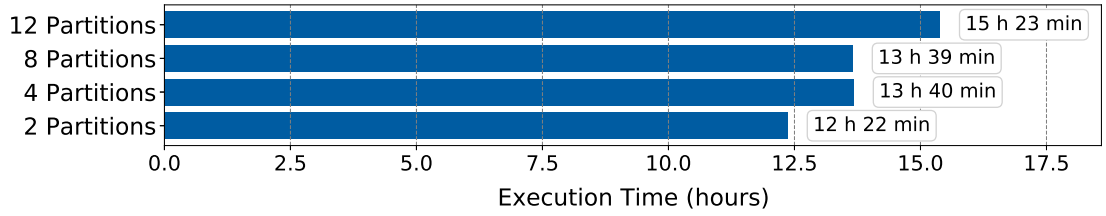


Figure 5.7: Execution times (in hours) for the k-d tree configurations with two, four, eight, and twelve partitions.

and 40 minutes for the four-partition one, and 13 hours and 39 minutes for the eight-partition configuration. Figure 5.7 shows the execution times obtained using spatial partitioning with the k-d tiling for the four configurations analysed. As can be observed, increasing the number of partitions results in longer execution times. The configuration with two partitions achieves the best performance, while additional partitions introduce computational overhead, reducing the overall efficiency of the model.

5.3.4 Using Citus for Parallelisation

Citus[2] is a PostgreSQL extension that enables horizontal scaling through distributed sharding. As introduced in Chapter 2, Citus distributes both data and queries across multiple nodes, allowing PostgreSQL to operate as a distributed database system. In this section, we focus on how Citus can be exploited to optimise the performance of the underwater noise propagation pipeline.

To this end, we first distributed the data partitions using the Citus sharding mechanism. The implementation was deployed on the machine described in Section 5.2, running Citus 12.1.9. In this configuration, Citus was employed in a four-node cluster (one coordinator and three workers) to distribute the workload across nodes, without applying any additional partitioning strategies. We adopted a four-node configuration because, as detailed in Cubukcu et al. [24], a single-node Citus setup does not yield immediate performance improvements. This behaviour was also observed in [117], where a single-node Citus configuration performed slightly worse than parallel execution with PostgreSQL. The tables `PointBoundingBox` and `Grid` were distributed using Citus functions as follows:

```
SELECT create_distributed_table('PointBoundingBox', 'point_id');
SELECT create_reference_table('Grid');
```

The first function distributes the table `PointBoundingBox` into multiple horizontal shards based on the `point_id` column. The second function distributes the table

[2] `https://www.citusdata.com/`

`Grid` into a single shard and replicates this shard to every worker node. Tables distributed in this way are called *reference tables* and are used to store data that require frequent access by multiple nodes within the cluster.

When executed using Citus, the query plan reveals that the workload is divided into multiple tasks, with a total of 32 tasks created. Each task is assigned to a specific execution node, ensuring efficient parallel processing. Within each task, a gathering operation takes place, using multiple worker threads to further parallelise the workload. The query plan performs two main operations: the *Parallel Append* retrieves data from multiple partitioned tables, and the *Bitmap Heap Scan* identifies the relevant grid cells by verifying that their positions fall within the bounding box. This step is optimised through index-based filtering on the row and column attributes, further enhancing performance. Using Citus alone, the entire pipeline is executed in 7 hours. The computation of sound propagation is nearly nine times faster than the optimised pipeline without partitioning (Section 5.2.2) and roughly twice as fast as the partitioned PostgreSQL version (Section 5.3.2).

We then adopted the same Citus configuration while also exploiting PostgreSQL range partitioning on the time dimension. As outlined in Section 5.3.2, the table `PointBoundingBox` was partitioned based on time ranges. In this case, we use the best-performing partitioning from our experiments, which divides the data into two partitions — one spanning 49 days and the other 42 days. The partitions can be created using the following Citus function.

```
SELECT create_time_partitions (
  table_name := 'PointBb_RangePart',
  partition_interval := '49 days',
  start_from := '2020-04-01 00:00:00',
  end_at := '2020-06-30 23:59:59');
```

The function above creates partitions at 49-day intervals between the specified dates. Furthermore, the tables `PointBoundingBox` and `Grid` were distributed using Citus, as described previously. Table 5.2 reports statistics on the number of rows in each partition and their corresponding disk usage. Using both Citus and PostgreSQL range partitioning on time, the entire pipeline is executed in 6 hours and 44 minutes, which is slightly faster (about 20 minutes) than the configuration using Citus alone.

We also employ Citus for the spatial tiling described in Section 5.3.3. Specifically, we partition the `PointBoundingBox` table according to the k-d tree grid structure with two partitions, as this tiling yielded the best performance in our experiments. The table is then distributed using the Citus function introduced earlier. The query plan is clearly similar to the case described above, with the workload distributed across multiple tasks. The execution time for the entire pipeline, using Citus and distributing the points according to the k-d tree grid

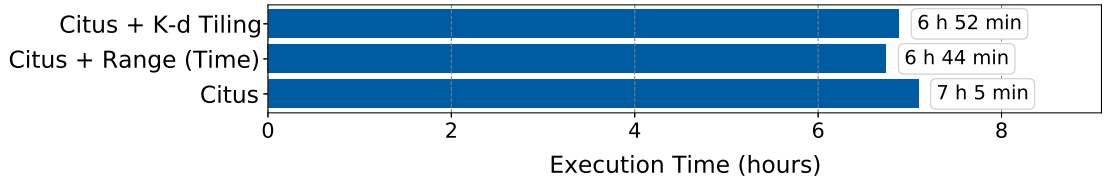


Figure 5.8: Execution times (in hours) using Citus alone, Citus with range partitioning on time, and Citus with k-d tiling partitions.

with two partitions, is 6 hours and 52 minutes, slightly slower than the range-partitioned configuration, yet marginally faster than using Citus alone. Figure 5.8 illustrates the execution times obtained using Citus for each configuration. As shown in the figure, integrating Citus with range partitioning on time or with k-d tiling further reduces the execution time to slightly less than 7 hours, yielding marginal improvements over the configuration using Citus alone.

As detailed in Cubukcu et al. [24], the use of Citus with a four-node cluster can significantly improve query performance through parallel task execution and distributed data processing. Overall, the execution with Citus is approximately nine times faster than the optimised pipeline, and around twice as fast as the range, hash and k-d tree tiling partitioning strategies.

5.3.5 Performance Evaluation

A performance evaluation is conducted to quantify the impact of the proposed data partitioning and distribution strategies on execution time. Figure 5.9 summarises the results of our experiments conducted on the spring season of 2020, reporting the best performance achieved by each technique.

We start from the baseline configuration (*Original Pipeline*), which includes the computationally expensive `ST_Intersects` operation used to identify the grid

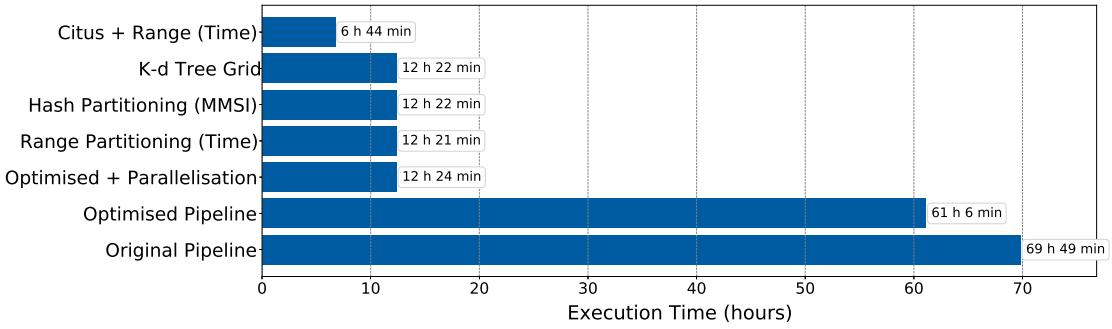


Figure 5.9: Execution times (in hours) of the different implementations.

cells affected by the noise emitted by each vessel. This model requires 2 days 21 hours and 49 minutes (69 hours and 49 minutes) to complete. Then, we consider the optimised version (*Optimised Pipeline*), which eliminates the costly geometric intersection by using the grid row and column identifiers to determine the affected cells. This configuration completes in 2 days 13 hours and 6 minutes (61 hours and 6 minutes), achieving a reduction of approximately 9 hours compared to the original pipeline.

Next, we evaluate the impact of PostgreSQL's native parallel query mechanism on the optimised pipeline, without applying any partitioning technique (*Optimised + Parallelisation*). This configuration achieves a total execution time of 12 hours and 24 minutes. Compared to the single-core execution, this corresponds to a fivefold improvement in performance. Similarly, applying range partitioning on time and hash partitioning on MMSI within PostgreSQL yields comparable results, with execution times slightly exceeding 12 hours. The use of the k-d tiling (spatial tiling) with two partitions leads to an execution time of 12 hours and 22 minutes, which is similar to the execution times observed for range and hash partitioning with two partitions.

A significant improvement is achieved with Citus, deployed on four virtual machines (one coordinator and three workers), reducing the execution time to approximately 7 hours. This corresponds to an almost tenfold speed-up with respect to the original pipeline, a ninefold improvement over the optimised version, and a twofold gain compared to PostgreSQL with range or hash partitioning. These findings confirm the scalability and efficiency of the distributed approach, showing that the adoption of Citus with four nodes significantly enhances the performance of our model.

5.4 Model Validation

In this section, we present a preliminary validation of our underwater sound model based on measured Sound Pressure Level (SPL) data from the SOUNDSCAPE project [100] for the entire year 2020 (from March to December). To this end, we identified the 1 km $\times$ 1 km grid cells containing the hydrophones and extracted the corresponding cell centroids. The SPL values computed at these centroids every 60 seconds were considered as the *model-generated* data, whereas the measurements recorded by the hydrophones at the corresponding time instants were taken as the *observed* values.

The validation approach follows the methodology used by Ghezzo et al. [49], which is based on the comparison of the *Cumulative Distribution Functions* (CDFs) of the observed and model-generated SPL data. The CDF provides a robust representation of the sound field, by capturing its overall statistical distribution while

filtering out short-term variability. The model-based CDF was obtained after removing the mean horizontal bias between the observed and model-generated values, thus aligning the two distributions along the SPL axis. This operation can be interpreted as a post-hoc calibration of the model [49, 45], to account for neglected variables that affect local propagation and ambient noise (e.g., sediment, masking islands, wind and waves), for local characteristics of vessel types, and even for possible differences in measurement setups. The bias values at the nine measurement stations are listed in Table 5.5. We note that the model-based CDFs show little bias at stations MS5, MS7, MS8, and MS9, the bias is relatively large and positive at stations MS1, MS3, MS4, and MS6, and it is moderate and negative at station MS2.

	MS1	**MS2**	**MS3**	**MS4**	**MS5**	**MS6**	**MS7**	**MS8**	**MS9**
Bias	16.45	-7.03	12.02	12.36	2.60	12.83	2.10	-2.16	1.52
RMSE	0.94	2.41	1.86	4.70	2.02	4.53	0.97	4.83	1.28

Table 5.5: Bias (dB) and RMSE (dB) between the model-based and the observed CDFs values at the measurement stations.

For validation purposes, we are interested in the shape of the statistical distributions of the measured and model-based sound pressure levels, aligned after bias removal. Figure 5.10 compares the observed (in dashed-blue) and model-based (in continuous-orange) cumulative distribution functions at the nine monitoring stations for spring season. Overall, the model-based CDFs reproduce the general shape and range of the observed ones, indicating that the model captures the main statistical features of the measurements. In particular, for hydrophones MS1, MS2, MS3, MS5, MS7, and MS9, the model-based curves closely follow the observed CDFs. For MS4 and MS8, some discrepancies appear in the tails, with model based probability distributions that look narrower than the measured ones. MS6 exhibits more pronounced differences in the shape of the CDF. Overall, the agreement between the observed and model-based CDFs is fairly good.

The quality of the model can be quantitatively evaluated by means of the Root Mean Square Error (RMSE) of the CDFs, defined as the square root of the mean of the squared differences between the observed and model-based cumulative distributions, aligned after bias removal. The RMSE, expressed in decibel, was computed as the mean squared deviation between the two CDFs for percentiles ranging from 5 to 95, to exclude extreme and rare events from computation [49]. This metric quantifies the discrepancy between the observed and model-based sound level distributions, with lower values indicating better agreement.

Table 5.5 reports the validation results in terms of RMSE between the model-

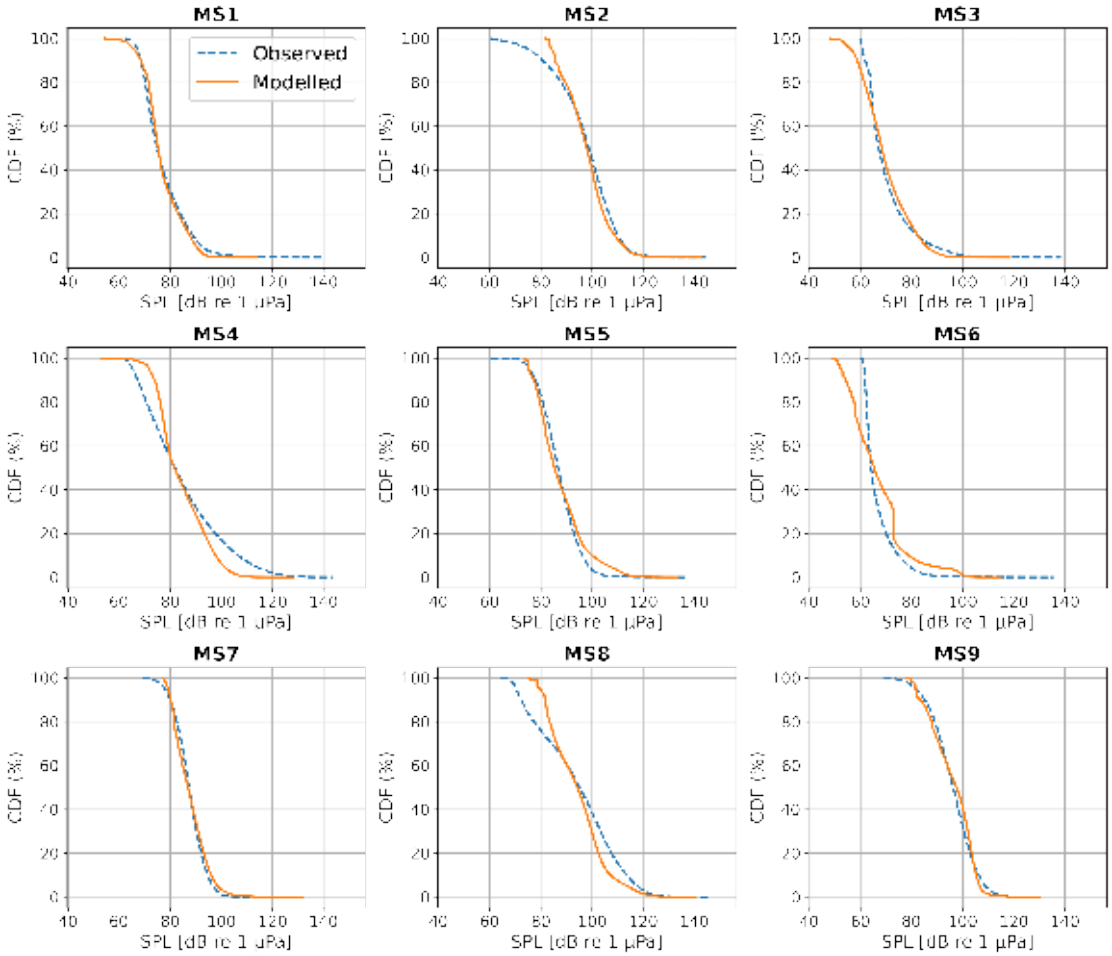


Figure 5.10: Comparison between the observed and model-based Cumulative Density Functions (CDFs) at the nine monitoring stations during spring season.

based and observed CDFs for spring 2020. Stations MS1, MS2, MS3, MS5, MS7, and MS9 show the lowest RMSE values, all remaining below or around 2 dB, indicating a very good statistical agreement between model and observations. Conversely, stations MS4, MS6, and MS8 exhibit higher RMSE values, close to 5 dB. These discrepancies may arise from local morphological and environmental factors not explicitly accounted for in the model, such as surface conditions, or from extraneous noise sources, such as non-tracked AIS-free boats.

The validation process should cover the entire duration of the SOUNDSCAPE measurements. This requires including the AIS data from January 2021 to March 2021, which are currently unavailable. Once all data have been processed and the model has been fully calibrated, it will be possible to perform a reliable comparison with the results presented by Ghezzo et al. [49].

5.5 Underwater Noise Maps

As described in Chapter 4, the proposed underwater noise model enables several types of analyses, such as assessing the impact of vessel activity on underwater noise levels at different frequencies, performing the analysis at various temporal granularities (e.g., yearly, seasonal, monthly, or daily), supporting the interactive visualisation of underwater noise dynamics, and focusing on the periods of highest activity for specific vessel categories — for instance, from Monday to Thursday for fishing vessels or during weekends for recreational ones. In this section, we present another type of analysis. Specifically, we generate: (*i*) underwater noise propagation maps for each vessel category across the four seasons of 2020; (*ii*) maps identifying, for each season, the vessel category that contributes most to underwater noise in each area; and (*iii*) seasonal noise maps obtained by considering all vessel categories together.

5.5.1 Noise Maps per Vessel Category

In this type of analysis, we focus on four different categories of vessels: recreational vessels (AIS ship type IDs 36 and 37), passenger and cruise ships (IDs 60–69), fishing vessels (ID 30), and cargo and tanker ships (IDs 70–89). The maps represent the four seasons: January–March (winter), April–June (spring), July–September (summer), and October–December (autumn). Analysing noise maps separately for each vessel category serves an important analytical purpose. This approach allows us to disentangle the contribution of individual vessel types and to characterise their temporal and spatial footprint and operational patterns. This distinction is particularly relevant from a management perspective. By examining the relative contribution of different categories, it becomes possible to identify which types of vessels exert the greatest acoustic pressure in specific areas and seasons. Such information may support targeted mitigation strategies (e.g., temporal or spatial restrictions applied to selected vessel categories), rather than imposing uniform constraints on all maritime traffic. To the best of our knowledge, previous studies have not analysed vessel-generated noise separately for different vessel categories.

As in the analyses presented in Chapter 4, all maps are bivariate. The first variable represents the average underwater noise exceeding the ambient noise level, while the second indicates the percentage of days within the analysed period (i.e., a season) in which a cell is active. Recall that, for a cell c, a day d is defined as *active* if the received sound level in c on d exceeds the ambient noise. Moreover, the red stars in the figures represent the hydrophones of the SOUNDSCAPE project. It is important to note that each set of maps is constructed using type-specific decibel and persistence scales, chosen to best represent the typical acoustic behaviour and operational patterns of the corresponding vessel category. As a consequence, the

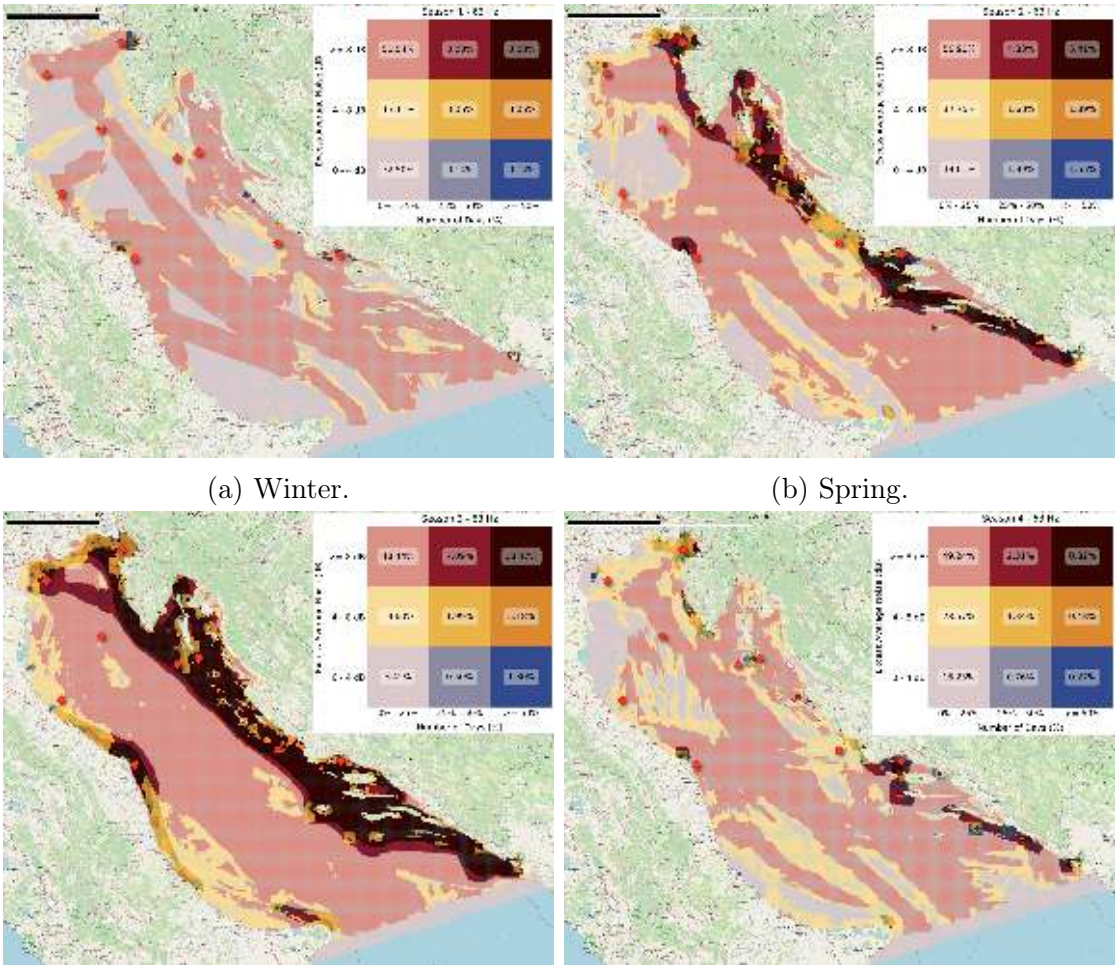
	No. of vessels			
Vessel Type	**Winter**	**Spring**	**Summer**	**Autumn**
Cargo	1,177	1,158	1,106	1,152
Passenger	158	167	220	166
Recreational	408	1,171	2,243	981
Fishing	675	691	702	705

Table 5.6: Number of vessels divided by ship type throughout the year 2020.

maps are not directly comparable across vessel types in terms of absolute noise values or persistence levels. They should instead be interpreted by focusing on the spatial distribution of the areas most affected by each category, allowing us to identify where different vessel types contribute most significantly to underwater noise.

Before proceeding with the analysis of the noise propagation maps for each vessel category, it is important to consider the seasonal distribution of vessels in the dataset. Table 5.6 reports, for each vessel category and for each season, the number of distinct vessels present in 2020. This information is useful because the number of vessels varies substantially across categories and seasons, influencing the subsequent analyses. Recreational vessels exhibit a marked seasonal pattern, with relatively few vessels in winter and a sharp increase in spring, reaching a peak in summer before decreasing again in autumn. Passenger vessels also show some variation, although to a lesser extent, with higher counts during summer and lower values in winter. Conversely, cargo and fishing vessels display a more stable presence throughout the year, with only minor fluctuations across the four seasons. These seasonal differences in vessel numbers are important for interpreting the resulting propagation maps, as categories with larger fleets are likely to generate a wider spatial footprint and, potentially, a stronger noise contribution.

Figure 5.11 shows the underwater noise propagation maps for the year 2020 for recreational vessels (AIS ship type IDs 36 and 37). Recreational vessels mainly include motorboats and sailing boats typically used by individuals for leisure purposes, such as short coastal trips or weekend outings at sea. Unlike commercial ships, they rarely undertake long journeys but rather perform short trips, often alternating navigation and stationary periods. For this reason, the persistence has been normalised by considering a cell as fully active (i.e., 100% persistence) when it is active for at least two hours per day throughout the entire season. As we can observe from Figure 5.11a, during winter there are no areas characterised by high persistence. In fact, about 99% of the cells show persistence values between 0–25%, and in less than 50% of the area the noise exceeds 8 dB. This behaviour aligns well

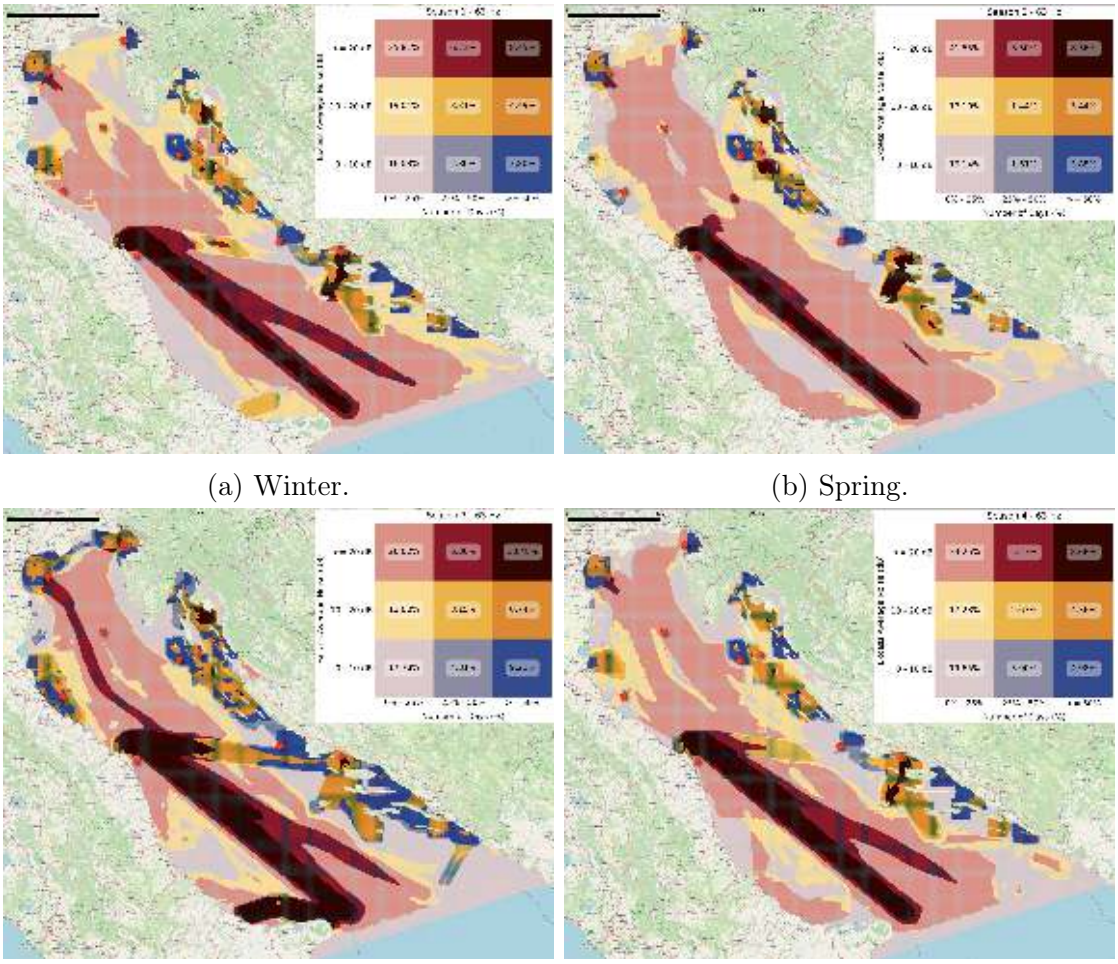


(a) Winter. (b) Spring.

(c) Summer. (d) Autumn.

Figure 5.11: Bivariate maps of underwater noise generated by recreational vessels throughout 2020.

with the expected seasonal trend, as recreational vessels typically remain in port during the winter months. Starting from spring (Figure 5.11b), an increase in both persistence and noise levels can already be observed, especially in the eastern part of the study area — i.e., the region encompassing Trieste, Slovenia, and Croatia — which is characterised by orange and dark-red areas indicating high persistence and noise values. The percentage of cells with these characteristics is noticeably higher than in winter. During summer (Figure 5.11c) this growth becomes even more pronounced. Almost the entire Croatian coastal area shows high noise levels and persistence, with more than 18% of the cells exhibiting high persistence and noise (dark-red cells), and about 5% showing maximum persistence with medium



(a) Winter. (b) Spring.

(c) Summer. (d) Autumn.

Figure 5.12: Bivariate maps of underwater noise generated by passenger and cruise ships throughout 2020.

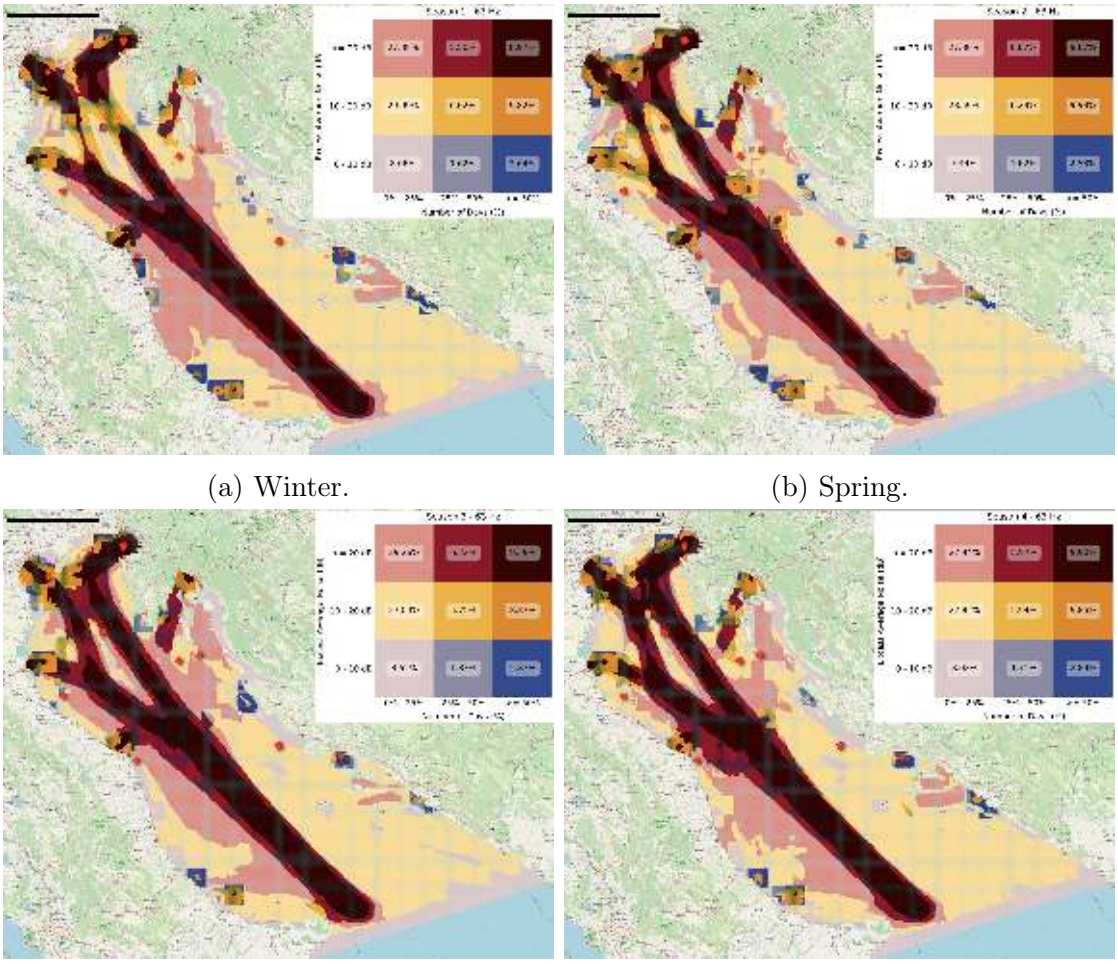
noise levels (orange cells), while in winter these values were only 0.23% and 0.06%, respectively. Moreover, a distinct route connecting Venice to the Croatian coast can also be identified, likely corresponding to recreational vessels travelling from Venice towards Croatian waters for summer holidays. Finally, in autumn (Figure 5.11d), the activity of recreational vessels decreases significantly, resulting in lower levels of both underwater noise and persistence across the area.

Figure 5.12 presents the bivariate maps for the four seasons of 2020 related to passenger and cruise ships. We recall that, over the entire year, there are only 284 vessels of this type (IDs 60–69). Given the relatively small number of vessels, a cell is considered fully active (i.e., with 100% persistence) when the

noise level exceeds the ambient noise for at least two hours per day throughout the analysed period, as for the recreational vessels. It should be noted, however, that the decibel scale used in these maps differs from that adopted for recreational vessels. This adjustment is required to accurately capture the noise propagation of this type of vessels, given that passenger and cruise ships are substantially larger (with an average LOA of about 120 metres, compared with an average LOA of about 23 metres for recreational vessels), and their propulsion systems and engine dimensions generate a broader and more intense acoustic footprint. As shown in Figure 5.12, the overall spatial pattern of underwater noise propagation appears broadly similar throughout the four seasons. The southern portion of the basin, extending from the area off Ancona towards the central and southern Adriatic, is consistently characterised by dark red cells, indicating both high persistence and high noise levels. During summer, however, there are more areas affected by high noise levels (Figure 5.12c). In particular, the area around Venice shows increased noise levels, and a clear route can be identified between Venice and Ancona, likely corresponding to passenger and cruise vessels operating along the Italian coast. Another prominent route is visible from Ancona towards the area of Split in Croatia, highlighting the intensification of maritime traffic during the summer period, when cruise activity and passenger transport between Italy and Croatia are more intense.

Compared with the maps for recreational vessels (Figure 5.11), the areas where passenger and cruise ships generate higher underwater noise levels are markedly different. The latter mainly follow routes from Ancona towards the southern Adriatic Sea, while recreational vessels are more widespread along the entire Croatian coastline, including the areas of Slovenia and Trieste. These maps are therefore crucial not only for identifying the regions most affected by underwater noise, but also for distinguishing the spatial contribution of different vessel categories to the overall acoustic environment.

Figure 5.13 shows the maps for cargo and tanker vessels (IDs 70–89) for the entire year 2020. It is important to note that the decibel scale is the same as that used for passenger vessels, while the percentage scale, representing the proportion of time a cell is active, reflects both the number of vessels and their operational patterns. In this case, a cell is considered fully active (i.e., 100% persistence) when the noise level exceeds the ambient noise for at least 16 hours per day. For these vessel categories, there is no marked difference between seasons, as the areas most affected by underwater noise remain largely the same throughout the year. This behaviour is consistent with the operational nature of cargo and tanker ships, which operate year-round to transport goods, containers, and oil or chemical products, resulting in stable traffic patterns over time. The most impacted areas are those around Trieste and Venice, from which two distinct routes depart and subsequently



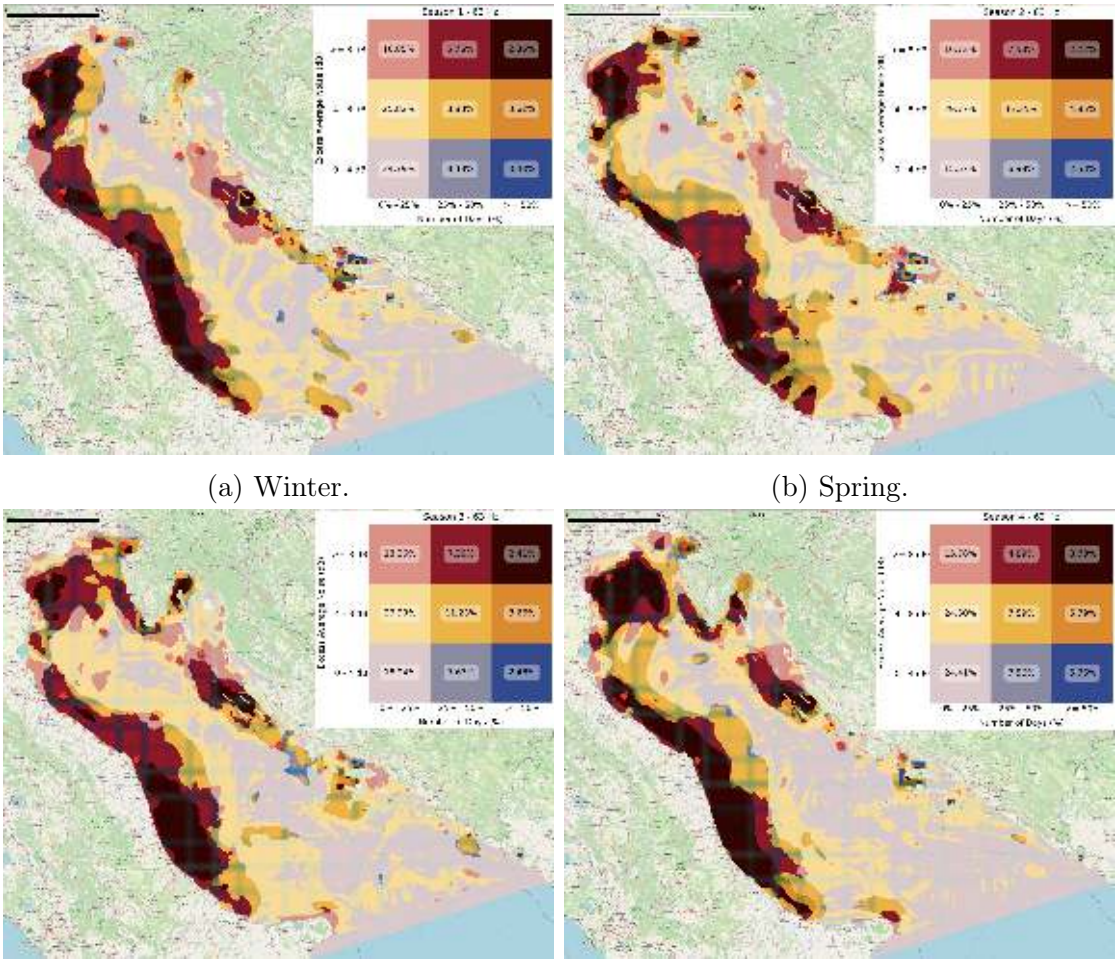
(a) Winter. (b) Spring.

(c) Summer. (d) Autumn.

Figure 5.13: Bivariate maps of underwater noise generated by cargo and tanker ships throughout 2020.

merge into a single main route towards the southern part of the Adriatic Sea (dark red cells). Two additional areas, near Ravenna and Ancona, also exhibit elevated noise levels, although the intensity is less pronounced compared with the two primary hotspots (dark red and orange cells). In contrast, the Croatian part of the Adriatic Sea shows limited underwater noise for cargo and tanker vessels throughout all seasons, with most cells displaying noise levels between 10 and 20 dB and persistence values below 25% (yellow cells).

Finally, Figure 5.14 presents the maps for the fishing vessel category (ID 30). Here, the persistence scale is defined as for cargo and tanker vessels, while the decibel scale is reduced to account for the smaller number of vessels (754 in total)



(a) Winter. (b) Spring.

(c) Summer. (d) Autumn.

Figure 5.14: Bivariate maps of underwater noise generated by fishing vessels throughout 2020.

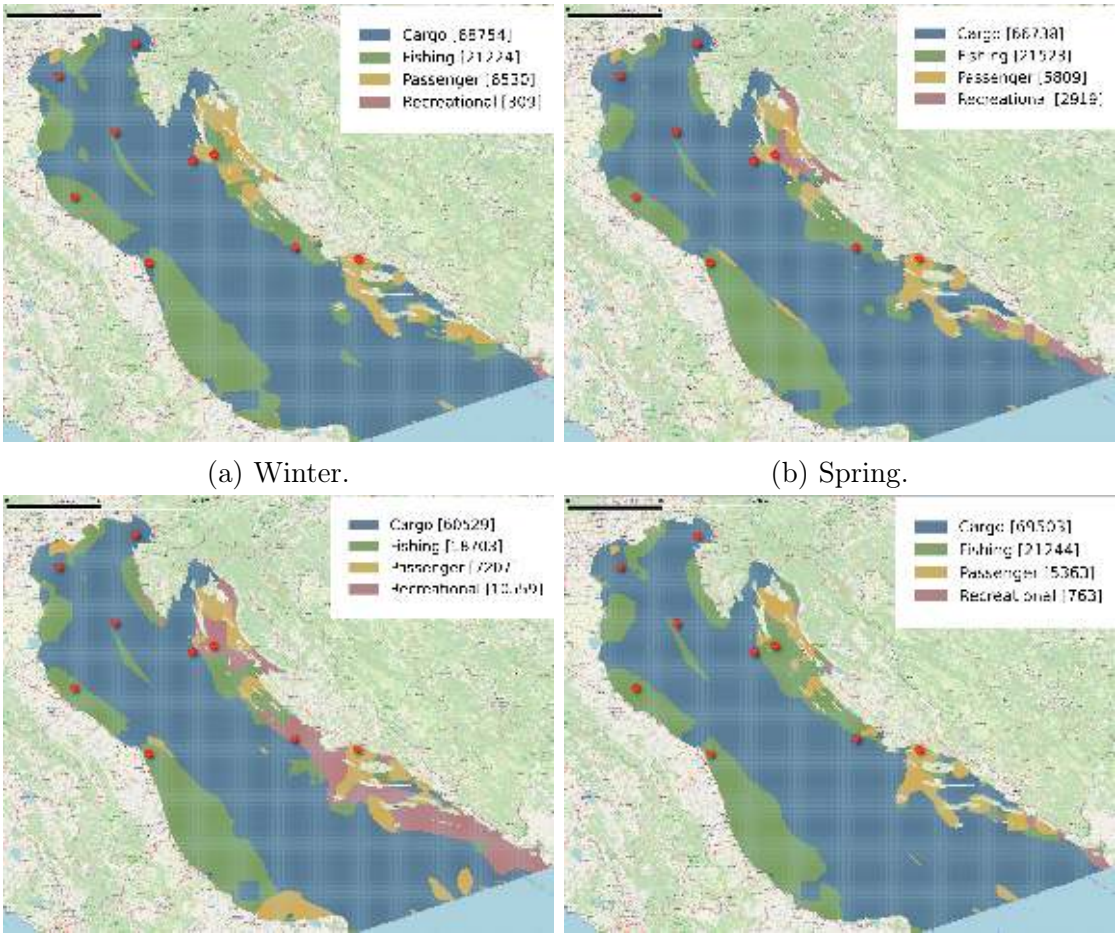
and their smaller dimensions compared with cargo and tankers. As shown in the winter map (Figure 5.14a), fishing vessels tend to remain close to the coast and rarely venture offshore. The entire coastal area from Venice down to southern Italy is characterised by high levels of underwater noise, mainly due to fishing activities. In spring (Figure 5.14b), vessels start operating farther from the coast. This can be observed both in the northern sector near Venice and in the area around Ancona. Moreover, a larger number of cells display high noise levels and persistence compared with the previous season (orange and dark-red cells). During summer (Figure 5.14c), fishing vessels operate even farther from the coast. In addition to the coastal areas affected in the previous seasons, a new area of intense

underwater noise appears in the northern Croatian waters. Finally, in autumn (Figure 5.14d), high underwater noise persists along the northern Croatian coast, together with a strong acoustic impact along the entire Italian shoreline — from Rimini to the southern Adriatic — and in the area around Venice, where fishing vessels also contribute to noise propagation farther offshore.

Although the maps presented for each vessel category are not directly comparable in absolute terms — due to the use of type-specific decibel and persistence scales — they provide essential insight into how different vessel types contribute to underwater noise pollution depending on their characteristics and operational behaviour. Recreational vessels, which are small motorboats and sailing boats typically used for leisure activities, tend to operate extensively along the Croatian coastline. Their acoustic footprint is relatively limited in magnitude, as reflected by the lower decibel ranges, yet they consistently affect broad coastal areas. Passenger and cruise ships, by contrast, generate substantially higher underwater noise levels and show a markedly different spatial pattern: the most impacted areas correspond to the main north–south corridor along the Italian coast, particularly the route from Ancona towards the southern Adriatic. Cargo and tanker vessels also represent major contributors to underwater noise. Their impact is concentrated along the major commercial routes linking the ports of Trieste and Venice to Ancona and the southern Adriatic, reflecting the movements of large commercial ships. Finally, fishing vessels produce lower noise levels than passenger and cargo ships, yet their spatial footprint differs significantly from that of the other vessel categories. Their activity is mostly concentrated near the coast, with further offshore operations appearing in the waters around Venice and Ancona.

5.5.2 Seasonal Identification of Dominant Vessel Categories

In this section, we consider the same vessel categories introduced in Section 5.5.1 (recreational, passenger, cargo, and fishing vessels). The aim of this analysis is to identify, for each grid cell, the vessel category that produces the highest underwater noise levels over the entire season. To this end, we first compute the daily noise exceedance for each cell and for each category. We then aggregate these daily values to obtain the total seasonal noise contribution per category in each cell. Finally, for every cell, we select the category with the highest seasonal contribution. Figure 5.15 shows, for each season of 2020, the vessel category that exerts the largest acoustic impact in each grid cell. The maps reveal clear and consistent spatial patterns in the dominance of cargo and fishing vessels, as well as strong seasonal variability in the contribution of recreational and passenger vessels. In addition, Table 5.7 reports, for each season, the percentage of cells dominated by each vessel category together with the corresponding percentage of the number of vessels during that season. This table provides a complementary quantitative view



(a) Winter.

(b) Spring.

(c) Summer.

(d) Autumn.

Figure 5.15: Comparison of vessel categories with the greatest acoustic contribution per grid cell across all 2020 seasons.

that helps interpret the maps more accurately, linking the spatial patterns to the underlying fleet composition.

Winter (Figure 5.15a) is characterised by a clear dominance of cargo vessels, which account for 48.68% of the fleet and correspond to 68.01% of the grid cells in terms of highest acoustic impact (blue cells). Fishing vessels represent 27.92% of the vessels and dominate 21.00% of the cells (green cells), mainly along the Italian coastline and in several offshore areas. Passenger vessels make up only 6.53% of the seasonal fleet and are associated with 6.46% of the cells (yellow cells), appearing primarily near major ports and ferry routes. Recreational vessels, despite representing 16.87% of the vessels, have a negligible spatial influence, dominating only 0.31% of the cells (red cells). Finally, 4.25% of the grid cells do not show a

dominant category, as no vessel-generated noise propagates to those areas.

	Winter		Spring		Summer		Autumn	
	Vessels	**Cells**	**Vessels**	**Cells**	**Vessels**	**Cells**	**Vessels**	**Cells**
Cargo	48.68%	68.01%	36.34%	66.00%	25.90%	59.86%	38.35%	68.76%
Fishing	27.92%	21.00%	21.68%	21.30%	16.44%	18.50%	23.47%	21.02%
Passenger	6.53%	6.46%	5.24%	5.75%	5.15%	7.13%	5.53%	5.30%
Recreational	16.87%	0.31%	36.74%	2.89%	52.52%	10.44%	32.66%	0.75%
None	–	4.25%	–	4.08%	–	4.07%	–	4.19%

Table 5.7: Percentage of vessels by ship category and percentage of grid-cells exhibiting the highest acoustic contribution from each category across the four seasons.

Spring (Figure 5.15b) shows a spatial pattern broadly similar to winter, with cargo vessels still representing the dominant contributors across most offshore areas. They correspond to 66.00% of the grid cells (accounting for 36.34% of the vessels in the dataset). Fishing vessels maintain a substantial influence, dominating around 21% of the cells, particularly along the Italian coastline and in several nearshore areas. Recreational vessels, which constitute a much larger share of the fleet in spring (36.74%), dominate only a small fraction of cells (2.89%). This represents a slight increase compared with winter, with the affected cells mainly concentrated along the Croatian coastline. Passenger vessels contribute only marginally, with dominance in about 6% of the cells, mostly around port areas.

Summer (Figure 5.15c) displays the most pronounced seasonal shift. Recreational vessels, which represent more than half of the fleet in this season (52.52%), dominate 10.44% of the grid cells — mainly along the Croatian coastline. This marks a notable expansion compared with the previous seasons. Cargo vessels remain the primary contributors offshore, corresponding to 59.86% of the cells, while fishing vessels dominate 18.50% of the grid (which is lower than in the other seasons, also because of the fishing ban during August), particularly along the western Adriatic coast. Passenger vessels show their widest seasonal presence, reaching 7.13% of cells, consistent with the increased summer ferry and cruise activity.

Finally, in autumn (Figure 5.15d), the spatial patterns are similar to those observed in winter and spring. Cargo vessels once again dominate most of the basin, corresponding to 68.76% of the cells. Fishing vessels maintain a stable influence, dominating roughly 21.02% of the grid, particularly along the Italian coastline. Recreational vessels, although still constituting 32.66% of the fleet, dominate less than 1% of the cells, indicating a substantial reduction in their spatial impact

compared with summer. Passenger vessels remain marginal, contributing to approximately 5% of the cells.

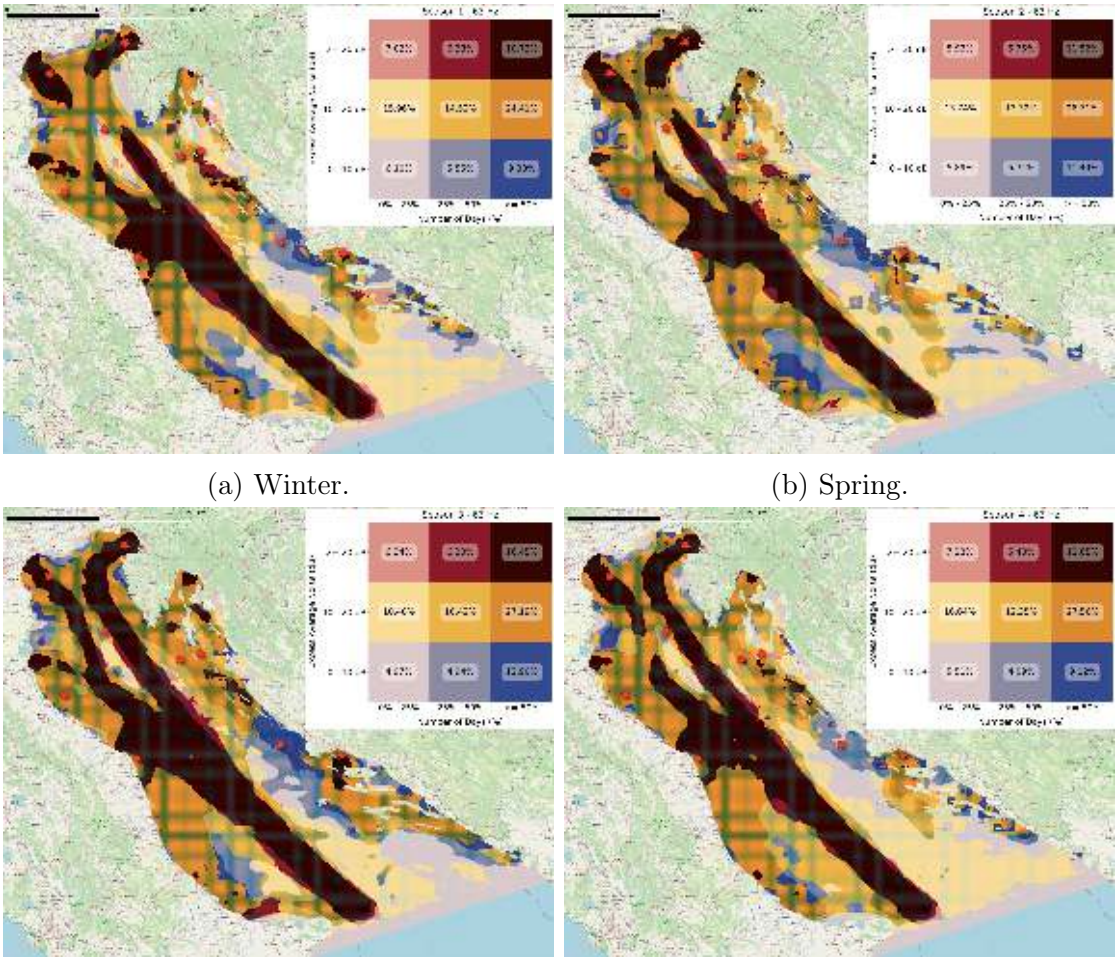
5.5.3 Seasonal Noise Landscape for All Vessels

In this section, we present the bivariate maps of noise propagation considering all vessels together, without distinguishing among categories. These bivariate maps are constructed following the approach described in Section 5.5.1, in which we have two variables: the first one represents the noise generated by vessels exceeding the ambient noise level (in dB), while the second indicates the percentage of time during which each cell is active. For these bivariate maps, a cell is considered 100% active if it receives noise levels exceeding the ambient noise for at least 16 hours every day of the season.

Figure 5.16 shows the resulting maps for the four seasons of the year 2020. Winter (Figure 5.16a) shows a pronounced high-intensity corridor running along the central axis of the basin, where several cells exceed 20 dB and exhibit persistences above 50% (dark-red cells). Similarly, the areas around Trieste and Venice, as well as the region in front of Rimini and Ancona, are also characterised by both high exceedance levels and high persistence. Surrounding zones are mainly associated with exceedances between 10 dB and 20 dB and with a persistence between 25% and 50% (orange cells). Finally, lower-intensity regions with limited persistence (0-25%) appear closer to the coast, particularly along the Croatian shoreline (the lilac and yellow cells).

Spring (Figure 5.16b) shows an acoustic footprint broadly similar to that observed in winter. The areas with the highest noise levels and persistence remain largely the same — the regions surrounding Trieste, Venice, Rimini, and Ancona. The Venice and Trieste corridors become slightly more pronounced, with a greater number of dark-red cells indicating both high exceedance and high persistence. Two areas can be observed where low noise levels coincide with relatively high persistence: one in the south of the map along the Italian coast and one near the MS7 hydrophone (blue cells).

Summer (Figure 5.16c) exhibits a marked increase in underwater noise, likely influenced by the seasonal rise in recreational and passenger/cruise vessels, in addition to the regular traffic of cargo, tanker, and other ship types. The two corridors originating from Venice and Trieste become even more pronounced and merge with the central corridor extending towards southern Italy. Indeed, more than 16% of the cells fall into the class characterised by both high noise levels and high persistence (dark-red cells), compared with about 11% in the two previous seasons. Coastal zones — particularly along the Croatian shoreline — show higher noise levels and increased persistence compared with the previous seasons. The area near the MS7 hydrophone continues to display low noise levels but an even



(a) Winter. (b) Spring.

(c) Summer. (d) Autumn.

Figure 5.16: Bivariate maps of underwater noise generated by all vessels throughout 2020.

higher persistence, highlighted by the darker blue cells.

In autumn (Figure 5.16d) there is a reduction in the spatial footprint of the exceedance classes compared with summer, with dark-red cells decreasing to around 12%. Nonetheless, the two corridors extending from Trieste and Venice towards southern Italy remain the areas most affected by underwater noise. Both persistence and noise levels slightly decrease along the coastal zones, particularly along the Croatian shoreline. In the south-eastern part of the basin, noise levels predominantly fall within 10–20 dB but with limited persistence (yellow cells).

Part II

Temporal Circular Buffer

Chapter 6

Circular Moving Region

Moving Objects Databases (MODs) are spatiotemporal databases designed to store and manage objects whose attributes vary over time [57, 120]. An example of a moving object is a vessel or a car, which is typically represented as a moving point when the spatial extent of the object is negligible and its orientation is of no importance. However, many real-world phenomena exhibit more complex spatial dynamics that cannot be adequately represented by a single moving point. Instead, they occupy a spatial region whose position, shape, or size may evolve continuously with time. Such phenomena are better described by *moving regions* — spatial objects whose geometry changes as a function of time — enabling the modelling of entities such as drifting oil spills, expanding rain clouds, or hurricanes. On an abstract level, a moving region can be formally described as a function from time into a set of points in the plane. Since such a continuous model cannot be directly handled within a database system, it is typically discretised into a finite representation, where the region is expressed as a set of filled polygons that move and possibly change their geometry over time [62].

The need for modelling moving regions also emerged clearly in the first part of this thesis. In developing the underwater noise propagation models presented in Chapters 4 and 5, we observed that acoustic fields naturally evolve as regions whose extent expands or contracts over time. This behaviour is not unique to underwater acoustics, but shared by many natural and anthropogenic phenomena, such as the spread of pollutants, wildfires, or meteorological systems. This motivated the second part of the thesis. In this chapter, we introduce a new spatiotemporal data type for representing deformable moving regions with circular geometry, whose position and radius vary continuously over time. We call this data type the *temporal circular buffer*. Alongside the type definition, we provide a suite of functions and operators enabling the manipulation and analysis of temporal circular buffers within a spatiotemporal database.

This chapter is structured as follows. Section 6.1 provides an overview of the literature on moving regions and their representations. Section 6.2 introduces the proposed data type, defining its abstract model and the set of functions and operators for manipulating and analysing temporal circular buffers.

6.1 Related Work

The concept of modelling spatial objects whose shape evolves over time traces back to the spatiotemporal data types introduced by Erwig et al. [37]. In their work, the authors propose data types for moving points and moving regions, together with the auxiliary spatial and temporal types and the set of operations required to manipulate such entities. This collection of types and operations is conceived as a foundation that can be integrated into the algebra and query language of a DBMS, thereby yielding a complete spatiotemporal data model.

Building upon this work, Tøssebro and Güting [134] started to investigate how a deformable moving region could be reconstructed from a sequence of observed static regions. Their work introduced a family of algorithms capable of generating spatiotemporal consistent representations from discrete observations. A complementary perspective was offered by McKenney and Webb [78], who demonstrated that the interpolation algorithm of Tøssebro and Güting is not robust, as it may produce invalid results in which moving segments self-intersect during the interpolation interval. They subsequently proposed an alternative algorithm for extracting valid moving regions from spatial data through topological analysis and polygon correspondences across time. Their work provided one of the first practical algorithmic approaches for constructing deformable moving regions, and later formed the basis of the *PySpatioTemporalGeom* library [77], which has played a notable role in the experimental evaluation of deformable-region algorithms.

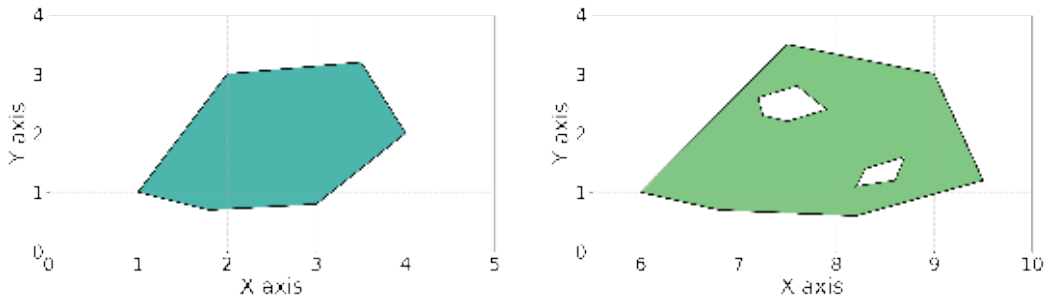
Subsequent work further expanded the field. Heinz and Güting [61] proposed a robust interpolation framework aimed at overcoming the numerical issues observed in earlier approaches, providing one of the most rigorous formal solutions to the region interpolation problem. Complementary developments emerged from the SPTMesh framework of Duarte et al. [30], which represents deformable moving objects through triangulated meshes and integrates them into PostgreSQL/PostGIS using custom data structures. More recently, machine-learning approaches have also been used in this context, such as the Conditional Variational Autoencoder model of Ribeiro et al. [110], which learns the temporal evolution of complex regions (e.g., wildfire perimeters) directly from data.

While these studies focus primarily on the algorithmic reconstruction or interpolation of deformable regions, a complementary line of research has examined how such objects can be represented within a spatiotemporal model. Consequently,

different representations have been proposed in the literature to capture the spatiotemporal evolution of moving regions. These approaches differ mainly in the assumptions made about how the geometry of the region changes over time. In the simplest case, the region preserves its shape and orientation, undergoing only translation or rotation in space. Such representations are referred to as *fixed-shape moving regions* [62] and are well suited for modelling rigid bodies such as vehicles, ships, or aircraft. In contrast, many natural or physical phenomena require modelling geometries that can expand, contract, or deform. These are known as *deformable moving regions* [46], and they provide a more flexible framework for representing dynamic spatial processes such as the spreading of oil spills, the growth of a fire front, or the propagation of underwater sound.

6.1.1 Deforming Moving Regions

Deformable moving regions constitute a subset of moving regions and are suitable for representing a wide range of real-world phenomena whose spatial extent evolves over time, such as the propagation of underwater sound, the dispersion of pollutants, or atmospheric phenomena. A *region* may consist of several disjoint parts called *faces* [134], each of which may have zero or mole holes. At discrete level, the boundaries of faces as well as holes are described by polygons [134]. Hence a region looks as shown in Figure 6.1.



(a) Region without holes. (b) Region with two holes.

Figure 6.1: Examples of regions.

A *moving region* (`mregion`) extends this notion by associating the region with an explicit temporal component: it is a region whose spatial extent evolves over time, allowing both its position and its shape to change continuously. An illustrative example is given in Figure 6.2, where a spatiotemporal phenomenon — such as an atmospheric disturbance — is shown at four different time instants. As time progresses, the object moves and its geometry deforms, highlighting the need for a data model capable of representing both movement and shape variation.

(a) t_1 (b) t_2 (c) t_3 (d) t_4

Figure 6.2: Example of a moving region.

Depending on the application, these regions may change either in discrete steps or continuously with time. When the geometry evolves at discrete temporal intervals, the primary challenge lies in storage requirements rather than in representation or computation. The region is well defined at each instant and can be interpolated between observations using a stepwise approach, allowing standard spatial operations to be extended to the temporal domain.

In contrast, when the region changes continuously, stepwise interpolation is no longer sufficient, and dedicated methods must be introduced to model the gradual geometric transformation between snapshots. In addition to defining an interpolation function, an appropriate storage model is required to efficiently compute intermediate geometries. Constructing such regions from successive observations introduces further complexity, particularly when the initial and final geometries differ substantially. The two shapes may have different numbers of vertices, or even distinct topologies, as occurs when a region splits into multiple parts or merges with another. Handling such topological and geometric changes while maintaining continuity remains one of the central challenges in modelling deformable moving regions [61, 85, 134].

Similar to other moving data types defined in [57], the moving region is represented using a *sliced representation*. The key idea is to decompose the temporal evolution of the object into a sequence of temporal fragments, called slices, within which the spatial and temporal variation of the region can be described by a simple function, typically through interpolation. Within a single slice, the simple function is essentially a region (as defined above) whose vertices move linearly in time, in such a way that at every instant within the slice the resulting geometry forms a valid region. Linear interpolation of the vertices thus guarantees a continuous and well-defined deformation between the boundary configurations at the start and end instants of the slice. This sliced representation relies on *moving segments*, defined as pairs of moving points that remain co-planar in 3d space. Since this definition does not permit a segment to rotate, a rotating segment can be represented by two moving segments (see Figure 6.3).

To accurately capture the full temporal behaviour of a moving region, or to cor-

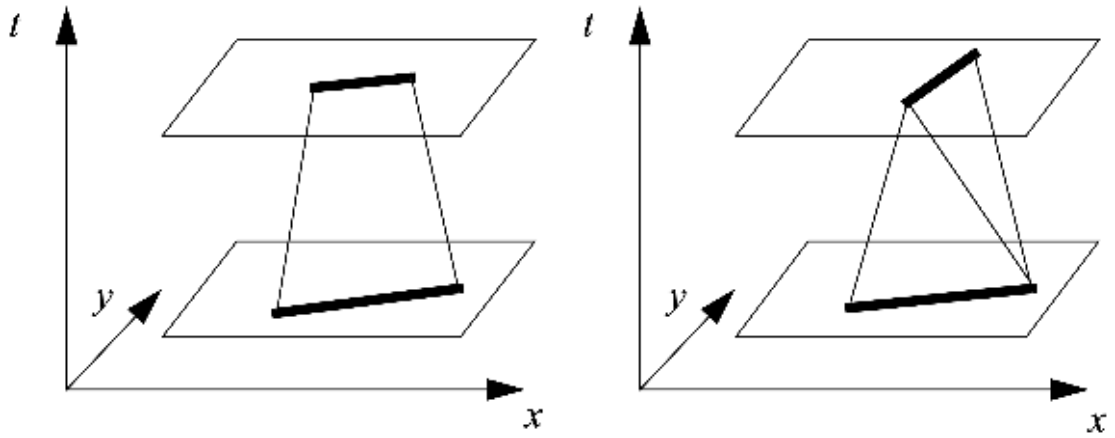


Figure 6.3: Sliced representation of moving segments, with two moving segments representing a rotating line segment [134].

rectly evaluate spatiotemporal operations, traditional approaches require a large number of two-dimensional time slices. This, however, leads to considerable storage demands and computational overhead. Heinz and Güting [63] proposed an alternative polyhedra-based framework in which moving segments are embedded directly into a three-dimensional spatiotemporal representation. Here, each moving segment is defined as a triangle in 3d space. These triangles construct a closed surface which are the facets of the border of a polyhedron representing the moving region. By performing all operations within this unified 3d model, their approach reduces the number of slices that need to be generated and processed, thereby improving efficiency and mitigating the storage limitations inherent in earlier methods.

Moreover, Heinz and Schildgen [64] extended regions and moving regions from the two-dimensional setting to objects of arbitrary spatial dimensionality by introducing the type constructors *region*(n) and *mregion*(n). The extension from two to higher dimensions is achieved by generalising the polyhedral model to a *polytopal* model, in which an n-dimensional *region*(n) is represented by an n-polytope, and an n-dimensional moving region *mregion*(n) corresponds to an $(n+1)$-polytope. The additional dimension represents time. For example, a three-dimensional moving region is represented by a four-polytope, that is, the natural generalisation of a polyhedron into four dimensions [60].

6.1.2 Fixed-Shape Moving Regions

Fixed-shape moving regions constitute a subset of all moving regions and can be employed to represent any rigid body in motion. The movement of fixed-shape regions can be described much easier than the movement of a deforming region, since

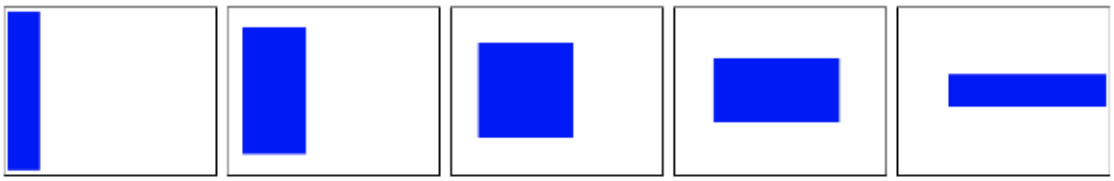


Figure 6.4: Interpolation of a moving region using the classical moving segments based model [62].

the only possible transformations are translations and rotations. Consequently, the shape of the region remains constant over time, and its movement can be expressed as a sequence of rigid transformations applied to a static geometry [62].

When managing continuous moving regions, it is crucial to compute the interpolation efficiently. The previous model for deforming regions relies on moving segments, which are assumed to be non-rotating. This assumption is contradictory to the idea of rotating (and translating) regions, and the larger the rotation of the region is, the worse the interpolation becomes. The underlying reason is that the interpolation method does not take into account the transformations that the polygon is allowed to perform. An example illustrating this limitation is shown in Figure 6.4, where the interpolation results in a considerable deformation of the region, which becomes a square at t_3. To overcome this limitation, Heinz and Güting [62] proposed a new model in which a region is represented as a sequence of geometric transformations, and makes use of these transformations to compute the correct interpolation. The resulting object, which preserves its shape while moving continuously through space, is referred to as a *fixed-shape moving region*, or `fmregion`. The interpolation results shown in Figure 6.5 accurately match the real movement of the object.

Based on this model, the authors define a collection of operations that enable query processing over fixed-shape moving regions. Among the various operations defined for this type, *atInstant* returns the interpolation of the `fmregion` at a given

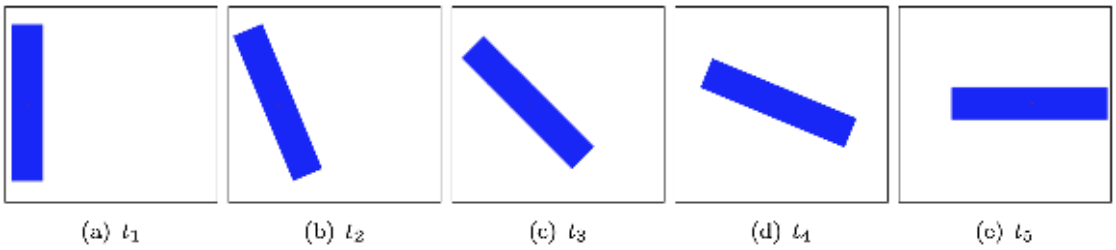


Figure 6.5: Interpolation of a moving region using the fixed-shape assumptions [62].

time instant t; *inside* determines the time intervals during which a moving point is located inside the `fmregion`; *traversedArea* denotes the set of all points covered by the moving region at some instant of the time interval, that is, the total area traversed by the region during its motion; and *intersect* which determines whether a (classical) region intersects with a geometry representing the boundary of the traversed area. These operations are also implemented in SECONDO [54].

A related line of work is the implementation of fixed-shape moving regions in MobilityDB, as presented by Schoemans et al. [124]. Their contribution consists in extending MobilityDB with a data type capable of representing fixed-moving regions, thereby supporting objects that undergo only translations and rotations. In this model, a fixed-moving region is stored as a static spatial geometry together with a temporal sequence of transformations. Each transformation specifies a rotation and a translation valid over a particular time interval, and the geometry at any instant is obtained by applying the appropriate transformation to the static shape. This representation avoids geometric redundancy and ensures temporal consistency, since the region's shape is stored only once and all spatial changes result from explicitly defined transformations.

In this work, the authors formalised an algebra for fixed-shape moving regions that includes accessor functions, transformation functions, comparison operators, and spatial functions for analysing the evolution of the region over time. The work also demonstrates how these operations can be integrated efficiently into PostgreSQL/PostGIS through MobilityDB's extensible type system.

Beyond the data model and query algebra, the authors implemented the full set of ISO 19141 operations for rigid temporal geometries and adopted a delta-encoding representation that stores movement as a compact list of translation and rotation parameters. Their experimental evaluation showed that both storage costs and query performance are comparable to those of temporal points. The implementation also benefits from integration with existing components of the PostgreSQL/PostGIS ecosystem, including GiST indexing and visualisation tools such as QGIS. Finally, the authors provided an open-source data generator for rigid temporal geometries in two and three dimensions, extending the BerlinMOD generator [31] for temporal points and enabling realistic benchmarking scenarios.

6.2 Circular Moving Regions

In this section, we introduce a new abstract data type, which we refer to as a *circular moving region*. This data type models a class of deformable moving regions whose spatial extent remains circular at every time instant, while the centre and the radius evolve continuously over time. We first introduce the base type *circular buffer*, which models a circular area and is referred to as `cbuffer`. Then, we

provide the temporal type `tcbuffer`, which represents the evolution in time of values of type `cbuffer`.

6.2.1 Circular Buffer

A *static circular buffer*, modelled by the type `cbuffer`, is defined as a pair (p, r) consisting of a two-dimensional point p, called *centre*, and a non-negative float r, called *radius*. It represents a circular area centred at the point p with radius r. This can be used to model phenomena, such as noise or pollution, caused by an object located at the centre of the buffer. The radius intuitively bounds the *impact* that the object has on its surroundings. It is defined as:

$$cbuffer = ((x, y), r)$$

where (x, y) represents the position of the moving object, and r is the radius of the buffer. An example of circular buffer value is shown in Figure 6.6, where centre is in position $(2, 2)$ and the area of impact is defined by a radius of 1.5. In general, the construction of a circular buffer requires two parameters: the location, and the buffer radius.

Since a circular buffer is defined by these two components, the `cbuffer` type provides two accessor functions to retrieve them. Specifically, the function `point` returns the centre of the circular region, while the function `radius` returns its radius. The type definition also includes support for standard comparison operators

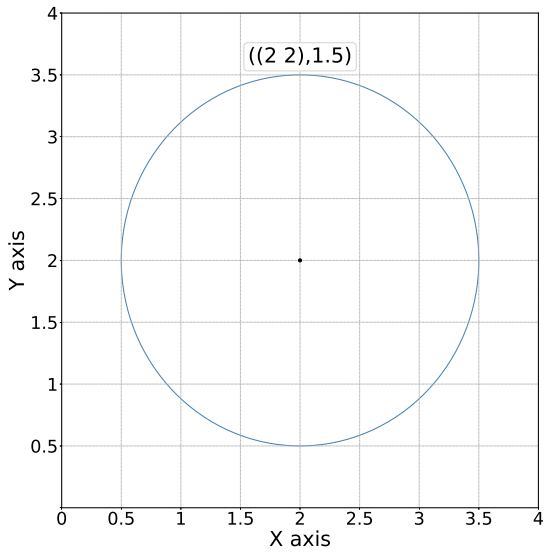


Figure 6.6: Representation of a circular buffer centred at $(2, 2)$ with radius 1.5.

(=, <>, <, >, <=, >=). They compare first the centre points and then the radii of the arguments.

6.2.2 Temporal Circular Buffer

A *temporal circular buffer*, modelled by the type `tcbuffer`, represents the movement of a circular buffer over time. It is defined as a finite sequence of static circular buffers, each associated with a timestamp:

$$tcbuffer = \langle (cbuffer_1, t_1),\ \ldots\ , (cbuffer_n, t_n) \rangle$$

where each $cbuffer_i = ((x_i, y_i), r_i)$ is a static circular buffer, and t_i is the associated timestamp. The sequence must be temporally ordered, that is, $t_i < t_{i+1}$ for all $i \in \{1, 2, \ldots, n-1\}$. This representation captures both the spatial and temporal evolution of the buffer, allowing the modelling of circular regions whose location and size can change over time. An example of a `tcbuffer` value is shown in Figure 6.7. In this case, the `tcbuffer` value is defined by three `cbuffer` values whose position and radius vary over time. Specifically, the object starts at position $(4, 4)$ with an impact radius of 2 on January 1 2001, moves to $(7.5, 10)$ with a radius of 2.5 on the following day, and reaches $(16, 15)$ with a radius of 5 on January 4 2001.

We provide three different kinds of interpolation which states how a value of type `tcbuffer` evolves between successive instants.

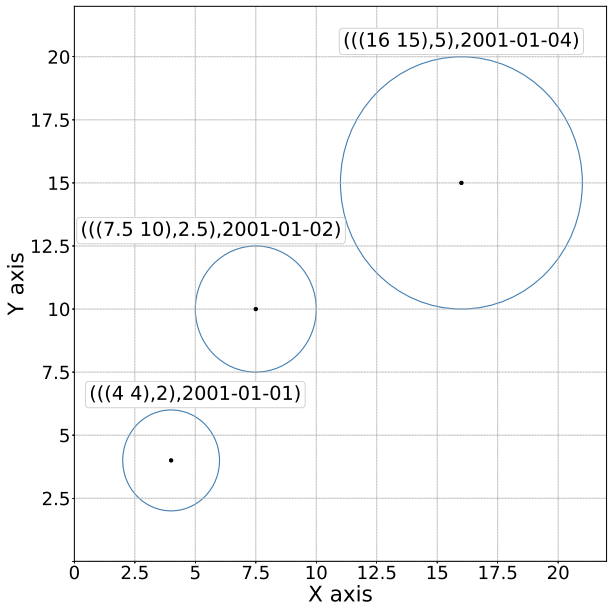


Figure 6.7: A temporal circular buffer.

- The interpolation is *discrete* when the value is unknown between two successive instants.

$$tcbuffer(t) = \begin{cases} cbuffer_i, & \text{if } t = t_i, \\ \text{undefined}, & \text{if } t_i < t < t_{i+1}. \end{cases}$$

  For example, a sonar device that emits periodic pulses perceives the environment only at the exact times of emission, with no information available in between (see Figure 6.8a).

- The interpolation is *stepwise* when the value remains constant between two successive instants.

$$tcbuffer(t) = cbuffer_i, \qquad t_i \leq t < t_{i+1}.$$

  For example, an autonomous vehicle may update its *safety area* every 3 seconds. This zone, modelled as a circular buffer around the vehicle's position (e.g., braking distance), remains constant until the next update, reflecting the assumption that the system considers the safety area stable within each time interval (see Figure 6.8b).

- The interpolation is *linear* when the value evolves linearly between two successive instants.

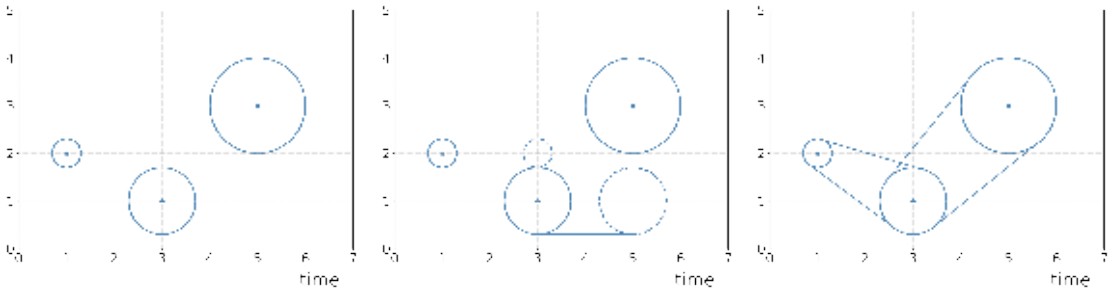
$$tcbuffer(t) = cbuffer_t \qquad t_i \leq t \leq t_{i+1},$$

  where the centre coordinates and the radius of $cbuffer_t$ evolve linearly between (x_i, y_i, r_i) and $(x_{i+1}, y_{i+1}, r_{i+1})$:

$$\begin{aligned} x(t) &= x_i + \frac{t - t_i}{t_{i+1} - t_i} \big(x_{i+1} - x_i\big), \\ y(t) &= y_i + \frac{t - t_i}{t_{i+1} - t_i} \big(y_{i+1} - y_i\big), \\ r(t) &= r_i + \frac{t - t_i}{t_{i+1} - t_i} \big(r_{i+1} - r_i\big). \end{aligned}$$

  For example, a vessel is moving, producing different levels of underwater noise based on the speed, the engine power, the kind of activity it is performing and we use linear interpolation to obtain the circular buffers between two consecutive positions (see Figure 6.8c).

We define a wide range of accessor functions for retrieving various properties of a `tcbuffer` value, including its duration, individual values, time instants, timestamps, and more. More specifically, the function `interp` returns the interpolation



(a) Discrete. (b) Stepwise. (c) Linear.

Figure 6.8: Interpolation.

method in use, i.e., {`Discrete`, `Stepwise`, `Linear`}. The `setInterp` function allows the interpolation method of a temporal circular buffer to be modified.

Additional accessor functions are available to extract specific components from a temporal circular buffer, such as individual values, timestamps, or entire sequences. The function `points` returns all centre points of the circular buffers, while `getTime` retrieves the set of temporal instants associated with the `tcbuffer` value. The function `getValues` extracts, for each timestamp, the corresponding circular buffer (i.e., centre and radius) from the `tcbuffer`. Finally, `valueAtTimestamp` returns the buffer value (centre and radius) at a specified time instant.

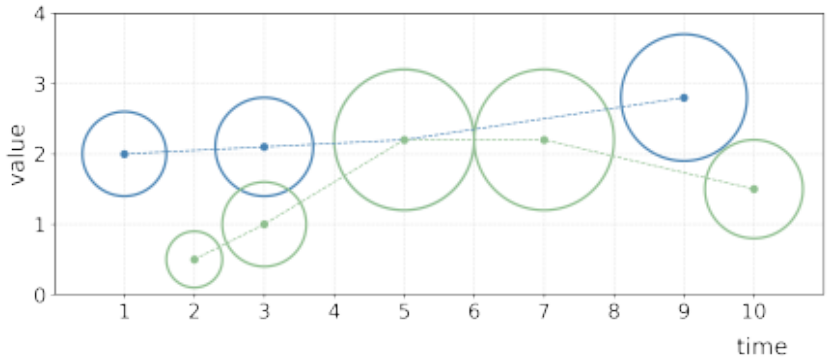
The last set of functions allow for accessing specific temporal elements of a `tcbuffer` value. Functions such as `startValue`, `startTimestamp`, and `startInstant` return, respectively, the first value of the `tcbuffer` (a `cbuffer`), the first timestamp, and the first instant (i.e., the first `tcbuffer` in the input sequence). Corresponding functions are also defined to retrieve the last value, timestamp, or instant, namely `endValue`, `endTimestamp`, and `endInstant`. Additionally, the n-th element can be accessed by specifying a positive integer n; for instance, `valueN(tcbuffer,3)` returns the third value in the sequence. Similarly, the functions `timestampN` and `instantN` are defined to return the n-th timestamp and instant, respectively. Finally, functions such as `timestamps` and `instants` take as input a `tcbuffer` and return the complete set of timestamps or instants.

A set of comparison operators is available for the `tcbuffer` type, enabling the comparison of two temporal circular buffers. These operators, denoted as `op`, can be one of {`=`, `<>`, `<`, `>`, `<=`, `>=`}. These comparison operators take two `tcbuffer` values as input and they yield a single boolean result, summarising the relationship between two temporal geometries over their entire duration. For example, the equality operator (`=`) returns *true* only if the two temporal circular buffers are equal at every timestamp in their temporal domain.

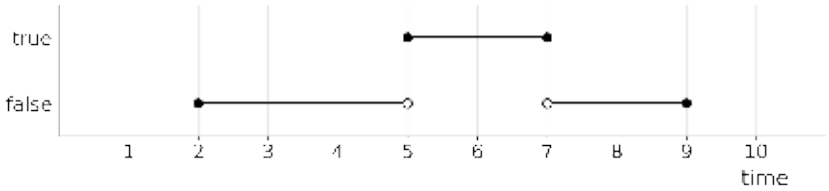
In temporal analysis it is often important to capture how this relationship

evolves over time. For this reason, temporal comparison operators have been introduced. We define two temporal operators, `#=` and `#<>`, which return a temporal boolean whose value, at each time instant, corresponds to the result of applying the equality or inequality operator to the input values, respectively. We can compare a `tcbuffer` with another `tcbuffer` value or with a static circular buffer (`cbuffer`). The comparison is applied to the underlying `cbuffer` values. For the equality operator, if the two circular buffers are not congruent (i.e., if either their centres or radii differ) the comparison evaluates to *false* at that instant.

For example, in Figure 6.9a, we consider two temporal circular buffers evolving over time at different temporal instants. The dashed lines connecting the centres highlight their movement across time. At time 1 only the blue buffer is defined, and it moves to its next position at time 3. The green buffer, instead, starts at time 2 and moves towards the blue one; at time 3 the two buffers become closer, and at time 5 they coincide, sharing both the same centre and the same radius. After that instant, they diverge again and evolve independently in space and time. In this scenario, when applying the equality operator (`#=`), the result, shown in Figure 6.9b, is *false* at time 2, becomes *true* at time 5 when the two buffers overlap, and returns to *false* afterwards, up to the final instant (time 9).



(a) Evolution of two temporal circular buffers over time.



(b) Result of the temporal equality operator.

Figure 6.9: Example of a temporal equality between two temporal circular buffers: (a) evolution of the two buffers; (b) temporal boolean returned by the temporal equality operator (`#=`).

6.2.3 Spatial and Spatiotemporal Functions

In addition to constructors, accessor functions, and comparison operators, it is necessary to define a wide range of operations for the `tcbuffer` data type. In this section we define two main spatial and spatiotemporal functions that determine the area traversed by a `tcbuffer` value and the temporal evolution of the distance between a `tcbuffer` value and another circular moving region. We also present a number of functions expressing topological relationships.

Traversed Area

The first function provided for spatial analysis is `traversedArea`. Given a temporal circular buffer, the function returns a geometry corresponding to the spatial region *impacted* by the circular buffer during its temporal evolution. The computation assumes linear interpolation between consecutive circular buffers, so that the buffer evolves continuously over time.

Given a point p and a non-negative float r, we denote by $C(p, r)$ the circular buffer with centre p and radius r:

$$C(p, r) = \{q \in \mathbb{R}^2 \mid \|q - p\| \leq r\}$$

where $\| \ \|$ denotes the Euclidean distance between two points.

Let $tcb = \langle((p_1, r_1), t_1), \ \ldots, \ ((p_n, r_n), t_n)\rangle$ be a temporal circular buffer. For each time instant t_i with $i \in \{1, \ldots, n\}$, we write C_i for the circle $C(p_i, r_i)$. The total traversed area $\mathcal{A}$ of tcb is defined as the union of the areas impacted between consecutive timestamps, namely:

$$\mathcal{A} = \bigcup_{i=1}^{n-1} \mathcal{T}_{i,i+1},$$

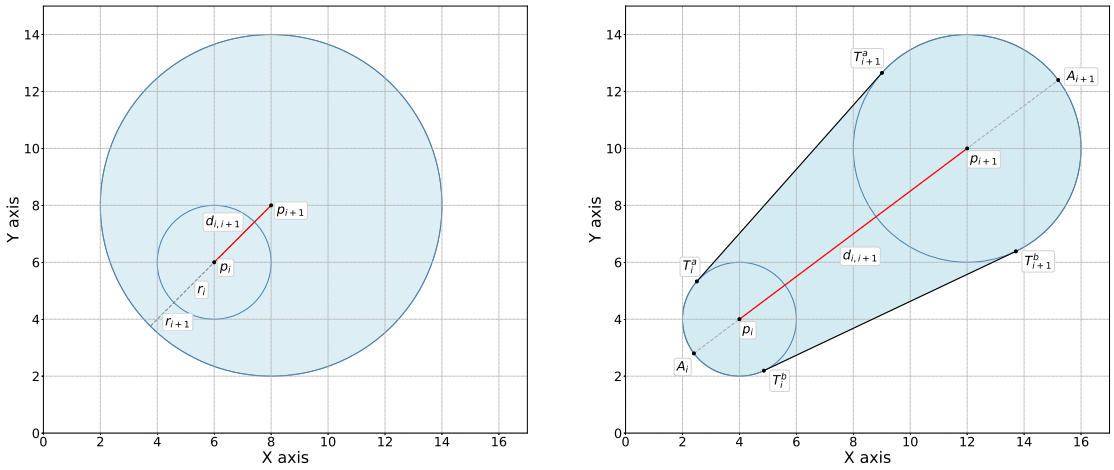
where $\mathcal{T}_{i,i+1}$ denotes the area impacted between times t_i and t_{i+1}.

For $n = 1$, the traversed area trivially reduces to C_1. For $n \geq 2$, we consider each pair of consecutive circular buffers C_i and C_{i+1}. Let $d_{i,i+1} = \|p_{i+1} - p_i\|$ the Euclidean distance between their centres. Two cases must be distinguished.

Case 1. If $d_{i,i+1} \leq |r_i - r_{i+1}|$, then one circle is entirely contained in the other between t_i and t_{i+1}. Thus, the traversed area between t_i and t_{i+1} corresponds to the circle with the larger radius, i.e.,

$$\mathcal{T}_{i,i+1} = \begin{cases} C_i & \text{if } r_i \geq r_{i+1} \\ C_{i+1} & \text{otherwise.} \end{cases}$$

Figure 6.10a shows an example.



(a) Case 1. The distance between the centres of the circular buffers is smaller than the absolute difference of their radii.

(b) Case 2. The distance between the centres of the circular buffers is greater than the absolute difference of their radii.

Figure 6.10: Traversed area (in light blue) computed between two consecutive circular buffers at times t_i and t_{i+1}, centred at p_i and p_{i+1} with radii r_i and r_{i+1}, respectively.

Case 2. Otherwise, two external tangent lines to both C_i and C_{i+1} exist, called T^a and T^b. Let θ_i be the angle formed by the segment $\overline{p_i\, p_{i+1}}$ and the positive x-axis:

$$\theta_i = \operatorname{atan2}(y_{i+1} - y_i, x_{i+1} - x_i),$$

and let δ_i be the angle between this segment and the external tangent lines:

$$\delta_i = \arccos\left(\frac{r_i - r_{i+1}}{d_{i,i+1}}\right). \tag{6.1}$$

The four tangent points are then computed by rotating the radius vectors by angles $\theta_i \pm \delta_i$. The x and y components of the points T^a_i and T^a_{i+1}, which define the external tangent line T^a, are given by:

$$T^a_i = (x_i + r_i \cos(\theta_i + \delta_i),\ y_i + r_i \sin(\theta_i + \delta_i))$$
$$T^a_{i+1} = (x_{i+1} + r_{i+1} \cos(\theta_i + \delta_i),\ y_{i+1} + r_{i+1} \sin(\theta_i + \delta_i))$$

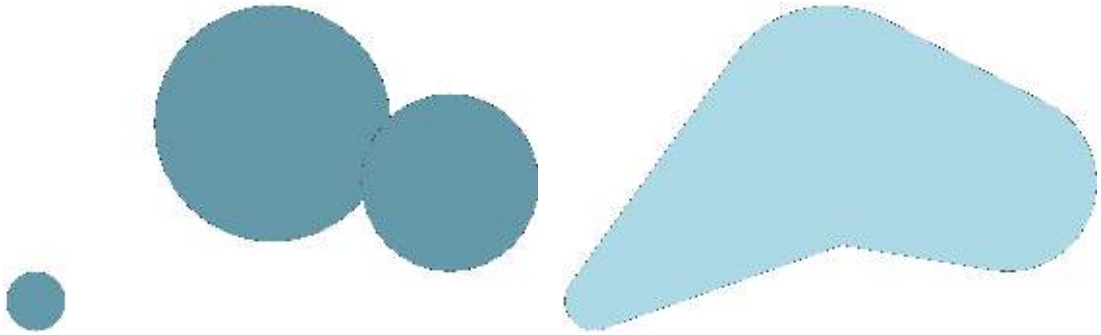
The other tangent points T^b_i and T^b_{i+1} are computed analogously by replacing the angle $\theta_i + \delta_i$ with $\theta_i - \delta_i$. These four points, as shown in Figure 6.10b, define the endpoints of the external tangent lines that form the lateral boundaries of the impacted area. The remaining parts of the boundary of the traversed area are constructed as circular arcs: one connecting T^a_i to T^b_i along

C_i, and the other connecting T^b_{i+1} to T^a_{i+1} along C_{i+1}. Each arc is defined to pass through the point on the corresponding circle that lies farthest from the centre of the other, along the direction of the segment joining the two centres. These points are denoted by A_i on C_i and A_{i+1} on C_{i+1}. The impacted region $\mathcal{T}_{i,i+1}$, illustrated in Figure 6.10b, is thus defined as a closed region bounded by the arcs $\overset{\frown}{T^a_i\, A_i\, T^b_i}$ and $\overset{\frown}{T^b_{i+1}\, A_{i+1}\, T^a_{i+1}}$, and by the two external straight segments $\overline{T^b_i\, T^b_{i+1}}$ and $\overline{T^a_i\, T^a_{i+1}}$.

Algorithm 3 outlines the procedure for computing the area traversed by a temporal circular buffer. Given a `tcbuffer` value, the algorithm iterates over each consecutive pair of `cbuffer` values composing the moving circular buffer of input, and computes the corresponding impacted area. Then the union of these geometries yields the total area covered by the temporal circular buffer.

Algorithm 3 Traversed Area Algorithm

1: **Input:** A temporal circular buffer tcb with n time instants
2: **Output:** The geometry of the traversed area $\mathcal{A}$
3: **Begin**
4: Extract the sequence of cbuffers $\langle (p_1, r_1), \ldots, (p_n, r_n) \rangle$ from tcb
5: $C_1 = C(p_1, r_1)$
6: **if** $n = 1$ **then return** C_1
7: **end if**
8: Initialise $\mathcal{R}$ as an empty list of regions
9: **for** $i = 1$ to $n - 1$ **do**
10: $C_{i+1} = C(p_{i+1}, r_{i+1})$
11: $d_{i,i+1} = \|p_{i+1} - p_i\|$
12: **if** $d_{i,i+1} \leq |r_i - r_{i+1}|$ **then**
13: $\mathcal{T}_{i,i+1}$ = the circle of larger radius between r_i and r_{i+1}
14: **else**
15: $\theta_i = \text{atan2}(y_{i+1} - y_i, x_{i+1} - x_i)$
16: $\delta_i = \arccos\left(\dfrac{r_i - r_{i+1}}{d_{i,i+1}}\right)$
17: Compute tangent points T^a_i, T^b_i, T^a_{i+1}, T^b_{i+1}
18: Compute farthest points A_i and A_{i+1}
19: $\mathcal{T}_{i,i+1}$ = the closed region bounded by arcs $\overset{\frown}{T^a_i\, A_i\, T^b_i}$ and $\overset{\frown}{T^b_{i+1}\, A_{i+1}\, T^a_{i+1}}$ and tangent segments $\overline{T^b_i\, T^b_{i+1}}$ and $\overline{T^a_i\, T^a_{i+1}}$
20: **end if**
21: Append $\mathcal{T}_{i,i+1}$ to $\mathcal{R}$
22: **end for**
23: **return** Union($\mathcal{R}$)



(a) Instants defining the `tcbuffer`. (b) Resulting traversed area.

Figure 6.11: Traversed area generated by the movement of a `tcbuffer`.

Let n be the number of time instants in the temporal circular buffer *tcb*. The algorithm begins by extracting the sequence of cbuffers (Line 4), which requires $O(n)$ time. Then the algorithm iterates over the $n-1$ consecutive pairs of circular buffers (Line 9). For each pair, it constructs the region $\mathcal{T}_{i,i+1}$ swept by the tcbuffer between the two corresponding instants (Lines 10–21) using geometric operations (distance, trigonometric functions, and the computation of tangent and farthest points) whose computational cost is constant. Then the append operator is performed in constant time. Hence, the cost of the for loop (Lines 9–22) is $O(n)$. The final step of the algorithm (Line 23) computes the geometric union of all regions in $\mathcal{R}$. This operation is executed once and dominates the overall cost. Its complexity depends on the union algorithm and on the total boundary complexity of the input regions (i.e., the total number of segments/vertices after discretisation). Therefore, the overall complexity of the algorithm is $O(n + \text{Union}(\mathcal{R}))$, and in practice it is dominated by the union operation. Finally, we observe that the algorithm requires $O(n)$ additional space to store the list of regions in $\mathcal{R}$.

Figure 6.11 illustrates the traversed area generated by a temporal circular buffer, whose temporal evolution is shown in Figure 6.11a. The corresponding geometry, shown in Figure 6.11b, is obtained using Algorithm 3.

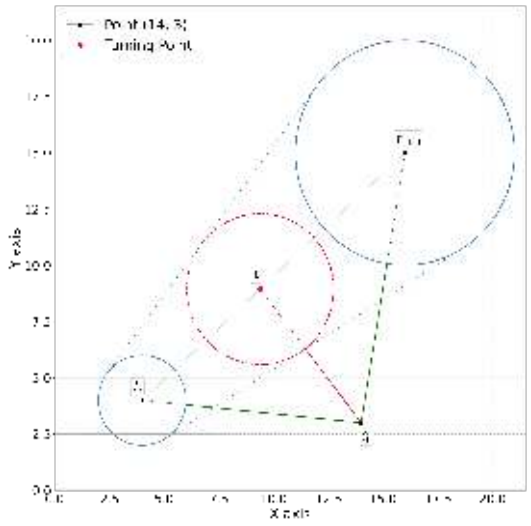
Temporal Distance

When dealing with moving regions, the time-varying (temporal) distance plays a crucial role in many applications. The *temporal distance* function (`tDistance`), computes the distance at each instant in the intersection of the temporal extents of its arguments, and results in a temporal float. Note that the distance function is the square root of a quadratic expression whereas temporal types only support linear interpolation between values. Hence, the temporal distance operator provides a linear approximation of the actual distance values for temporal sequence

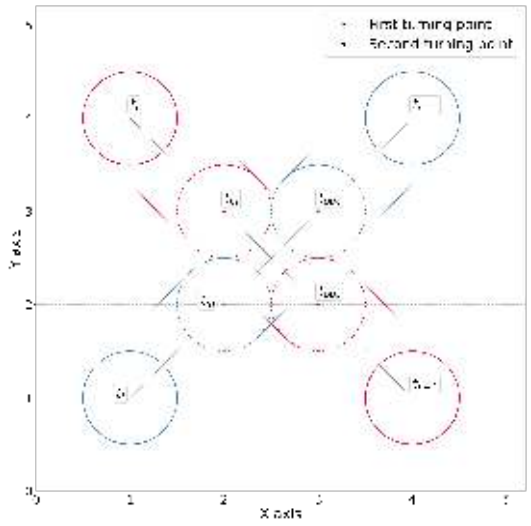
types.

A key concept underlying the computation of the temporal distance is the *turning point*. A turning point is defined as the position at which the temporal distance between two objects reaches a local minimum. Identifying such points is essential to ensure the accuracy of the computed distance, as they may not coincide with the original sampling instants of the input temporal objects. In fact, since the result of the `tDistance` function is a temporal float, which relies on linear interpolation between successive values, the presence of a turning point indicates a change in the monotonicity of the function. If such points are not explicitly included, the interpolation does not accurately represent the actual variation of the distance over time. Whether intermediate points need to be added depends on the types of the operands involved: (*i*) when computing the temporal distance between a temporal circular buffer and a non temporal type, the distance is evaluated at each instant of the temporal object; (*ii*) when both operands are temporal, they must first be synchronised, and the distance function is then applied to each pair of aligned instants.

To clarify how a turning point is computed, consider Figure 6.12a. We want to compute the temporal distance between a temporal circular buffer *tcb*, whose centre and radius evolve over time, and a static point q in the plane. The temporal circular buffer begins at time t_i with a circle centred at the coordinates $(4, 4)$ and with radius 2. At time t_{i+1}, its centre has moved to $(16, 15)$ and its radius has increased to 5. The motion between the two instants is linear both in space (the centre moves along a straight segment) and in radius (the radius interpolates linearly from 2 to 5). The fixed geometry against which the distance is computed is a point located at $(14, 3)$. The resulting temporal distance function is characterised by three instants. At t_i the distance between the point and the moving circle is approximately 8.05. This value corresponds to the Euclidean distance between the point and the initial centre (minus the initial radius). At t the distance reaches a minimum of approximately 4.17 (i.e., the turning point). At t_{i+1} when the circle has centre $(16, 15)$ and radius 5, the distance increases again to approximately 7.17. Notice that in addition to evaluating the distance at the time instants of *tcb*, a new instant t is added, representing when the distance between q and *tcb* is at its minimum. The result indicates that the distance starts at 8.05 at t_i, decreases linearly until it reaches the minimum value of 4.17 at t, and then increases linearly until it reaches 7.2 at t_{i+1}. If the turning point were not taken into account, the temporal distance would only include the values at the initial and final instants. In this case, the temporal distance function would report a value of approximately 8.05 at t_i, and a value of approximately 7.17 at t_{i+1}, yielding a function that decreases linearly from 8.05 to 7.17. Such a function does not capture the actual temporal distance between *tcb* and q, as the minimum distance is attained at an



(a) Turning point (in red) added at instant t between a `tcbuffer`, defined over the interval $[t_i, t_{i+1}]$, and a geometry point q.



(b) Two turning points (corresponding to the dashed circles) added between two `tcbuffer`s, defined over the interval $[t_i, t_{i+1}]$.

Figure 6.12: Examples of turning points in the computation of the temporal distance. (a) A single turning point is added between a `tcbuffer` and a point. (b) Two turning points are added between two `tcbuffer`s, corresponding to the start and end of their spatial overlap.

intermediate instant t.

We now consider the case in which the temporal distance is computed between two temporal circular buffers, that intersect during motion. As depicted in Figure 6.12b, the first `tcbuffer` moves from position p_i at timestamp t_i to position p_{i+1} at t_{i+1}, while the second moves from q_i to q_{i+1} over the same time interval. Both buffers have the same constant radius throughout their movements. In this configuration, the two `tcbuffer`s do not simply approach one another at a single turning point; instead, their paths intersect, leading to multiple turning points. As shown in the figure, the temporal distance between the two objects reaches zero at a specific instant t_{in}, when their spatial buffers begin to overlap. The distance remains zero throughout the intersection, and starts increasing again when they move apart, at instant t_{out}. Consequently, this scenario involves not just one but two turning points: the first occurs when the objects begin to intersect, and the second when their overlap concludes. In this case, two turning points are added, corresponding to the times of entry into and exit from the intersection, during which the distance remains zero.

We now formally define the temporal distance between two temporal circular

buffers which is the most difficult case. Given two time instants t_i and t_{i+1}, we consider two temporal circular buffers:

$$a = \langle((p_i^a, r_i^a), t_i), ((p_{i+1}^a, r_{i+1}^a), t_{i+1})\rangle$$
$$b = \langle((p_i^b, r_i^b), t_i), ((p_{i+1}^b, r_{i+1}^b), t_{i+1})\rangle$$

We compute the turning point(s), if any, that occur between the two buffers within the interval $[t_i, t_{i+1}]$. Let $\Delta t = t_{i+1} - t_i$ be the duration of the interval. Any time instant $t \in [t_i, t_{i+1}]$ can be expressed as $t = t_i + \alpha\Delta t$, where $\alpha \in [0, 1]$ denotes the relative position of t within the interval. The centres and radii of the two temporal buffers at time t are linearly interpolated as follows:

$$p^a(t) = p_i^a + \alpha(p_{i+1}^a - p_i^a), \qquad r^a(t) = r_i^a + \alpha(r_{i+1}^a - r_i^a)$$
$$p^b(t) = p_i^b + \alpha(p_{i+1}^b - p_i^b), \qquad r^b(t) = r_i^b + \alpha(r_{i+1}^b - r_i^b)$$

The temporal distance function is defined as the Euclidean distance between the centres minus the sum of the radii:

$$D(t) = \|p^a(t) - p^b(t)\| - (r^a(t) + r^b(t))$$

In order to identify turning points, we consider the first derivative of the temporal distance function $D(t)$ with respect to time. Turning points correspond to the local minima of the function and occur at the instants where its derivative vanishes. We therefore seek the roots of the equation $\dfrac{d}{dt}D(t) = 0$.

The velocities of the centres and the radii can then be computed as follows:

$$v_x^a = (x_{i+1}^a - x_i^a)/\Delta t, \quad v_y^a = (y_{i+1}^a - y_i^a)/\Delta t, \quad v_r^a = (r_{i+1}^a - r_i^a)/\Delta t$$

The same expressions apply to the circular buffer b, yielding v_x^b, v_y^b and v_r^b. The relative velocity of the centroids is obtained by subtracting the individual components, whereas the rate of change of the combined radius is computed as the sum of the individual rates, given that the temporal distance is defined as the Euclidean distance between the centroids minus the sum of the radii.

$$v_x = v_x^a - v_x^b, \qquad v_y = v_y^a - v_y^b, \qquad v_r = v_r^a + v_r^b$$

Finally, we solve the first derivative of the temporal distance function $D(t)$, which is linear over the interval $[t_i, t_{i+1}]$. The derivative vanishes at the instant:

$$t = -\frac{(x_i^a - x_i^b) \cdot v_x + (y_i^a - y_i^b) \cdot v_y - (r_i^a - r_i^b) \cdot v_r}{v_x^2 + v_y^2 - v_r^2}$$

If the computed timestamp t lies strictly within the interval $[t_i, t_{i+1}]$, the temporal distance at t is evaluated, and depending on its value, two cases are considered:

- If $D(t) > 0$, the two temporal buffers do not intersect, and t corresponds to a single turning point at which the temporal distance reaches a minimum.
- If $D(t) \leq 0$, the two temporal buffers intersect during the interval. In this case, two additional turning points are computed: the entry time t_{in} and the exit time t_{out}, corresponding to when the temporal distance becomes zero. These are obtained via linear interpolation:

$$t_{\text{in}} = t_i + (t - t_i) \cdot \frac{0 - D(t_i)}{D(t) - D(t_i)}$$
$$t_{\text{out}} = t + (t_{i+1} - t) \cdot \frac{0 - D(t)}{D(t_{i+1}) - D(t)}$$

Only values strictly within the interval (t_i, t_{i+1}) are retained as valid turning points. The function that computes the turning point(s) takes as input two temporal circular buffers defined on the same interval and returns an integer corresponding to the number of distinct turning points identified within the interval. Depending on the geometric configuration, the function may return either zero, one, or two turning points. When turning points are detected, their corresponding timestamps

Algorithm 4 Temporal distance algorithm

1: **Input:** Two temporal circular buffers, tcb_1 and tcb_2, with $p_1(t)$, $p_2(t)$, and $r_1(t)$, $r_2(t)$ their linearly time-varying centres and radii
2: **Output:** A tfloat *TDist* modelling the temporal distance between tcb_1 and tcb_2
3: **Begin**
4: $T = \emptyset$
5: *TSync* is the set of instants obtained by synchronising tcb_1 and tcb_2
6: **for all** consecutive instants t_i, t_{i+1} in *TSync* **do**
7: $T = T \cup \{t_i\}$
8: Extract the tcbuffers of tcb_1 and tcb_2 restricted to the interval $[t_i, t_{i+1}]$
9: Compute the turning point(s) for these two tcbuffers
10: If one or two turning points t_1, t_2 are found, add them to T
11: **end for**
12: $T = T \cup \{t_{i+1}\}$
13: Initialise *TDist* as an empty tfloat
14: **for all** $t \in T$ **do**
15: Compute $D(t) = \|p_1(t) - p_2(t)\| - r_1(t) - r_2(t)$
16: Append $(D(t), t)$ to *TDist*
17: **end for**
18: **return** *TDist*

are stored and then provided as input to the function that computes the temporal distance. Algorithm 4 outlines the high-level procedure for computing the temporal distance between two `tcbuffer` values.

Let n_1 and n_2 be the numbers of instants of tcb_1 and tcb_2, respectively. The synchronisation step (Line 5) produces the set of instants *TSync*, whose size is $O(n_1 + n_2)$. The algorithm then iterates over all consecutive pairs of instants t_i, t_{i+1} in *TSync* (Line 6), which requires $O(n_1 + n_2)$ time. For each interval $[t_i, t_{i+1}]$, it computes the turning point(s) of the distance function between the two corresponding tcbuffers (Lines 7–10). This computation requires arithmetic and trigonometric operations performed in constant time, and yields at most two candidate turning points; consequently, it takes constant time per interval. Then the set T contains all instants in *TSync* plus the turning points found across the intervals; hence $|T| = O(n_1 + n_2)$. Finally, the algorithm iterates over all instants in T (Line 14), evaluates the distance $D(t)$ between the two cbuffers at t, and appends the result to *TDist*. This step also requires $O(n_1 + n_2)$ time. Overall, the temporal distance algorithm runs in $O(n_1 + n_2)$ time and requires $O(n_1 + n_2)$ additional space.

Additional Spatial Functions

Beyond the `traversedArea` and `tDistance` functions, additional operations are available to further improve the expressivity and usability of the proposed data types. These include the `nearestApproachDistance`, `nearestApproachInstant`, and `shortestLine` functions. Given two moving circular buffers, the `nearestApproachDistance` returns the smallest distance ever reached between them during their movement. This function is particularly relevant in nearest-neighbour search applications. The `nearestApproachInstant` function returns the timestamp at which the minimum distance is attained. Finally, the `shortestLine` function computes the shortest line segment between the two temporal circular buffers at their point of closest approach.

Moreover, some topological relationships available in PostGIS have been generalised for `tcbuffer`. These include the functions `ST_Contains`, `ST_Disjoint`, `ST_Intersects`, `ST_Touches`, and `ST_DWithin`. Each of these PostGIS functions has two generalised versions:

- The *ever* relationships determine whether the topological or distance relationship is ever satisfied and returns a boolean.
- The *always* variant verifies whether the topological or distance relationship is always satisfied and returns a boolean.

Some of these functions are `eContains`, `eIntersect`, `eDisjoint`. The underlying semantics of each relationship follow the corresponding PostGIS definition.

Two geometries *intersects* if they share at least one point in common; they are *disjoint* when they have no point in common; and they *touches* when their boundaries meet without overlapping in their interiors. The *contains* predicate holds when one geometry lies entirely within the interior of another. Finally, *dWithin* returns true if the geometries are within a given distance.

Chapter 7

Implementation of the Circular Moving Region

In this chapter we present the implementation of the temporal circular buffer, extending MobilityDB with a new spatiotemporal data type. The chapter details the implementation of all the functions previously defined at the abstract level, including constructors, accessor functions, comparison operators, and the full collection of spatial and spatiotemporal operations. The data type, together with all associated functions and operators, is implemented within the main MobilityDB repository.[1] We also describe how these features integrate with the existing MobilityDB and PostgreSQL/PostGIS architectures.

In addition, we illustrate the use of the `tcbuffer` through a real-world case study. A set of representative queries is presented to demonstrate how the data type can be employed in practical scenarios, showcasing its expressive power and its applicability in complex spatiotemporal analyses.

The chapter is structured as follows. Section 7.1 describes the implementation of the circular buffer in MobilityDB, together with its functions and operators. Section 7.2 introduces the implementation of the temporal circular buffer, detailing the constructors, functions, and operators that support the manipulation of this spatiotemporal data type. Finally, Section 7.3 illustrates how the temporal circular buffer can be applied in a real-world case study.

7.1 Circular Buffer

This section presents the implementation of the `cbuffer` type introduced in the previous chapter. A value of type `cbuffer` is defined as a pair (p, r), where p is a two-dimensional point representing the centre of the buffer and r is a non-negative

[1] `https://github.com/MobilityDB/MobilityDB`

floating-point number denoting its radius. This structure models a circular region whose spatial extent is fully determined by its centre and radius.

7.1.1 Constructors

In MobilityDB, the abstract definition of the `cbuffer` is realised as a data type that stores both the position of the centre and the corresponding radius. Two constructors are provided: one that creates a `cbuffer` from explicit coordinates (x, y) and a radius r, and another that accepts a point `geometry` together with r. The signatures of these constructors are shown in Table 7.1.

Constructor	Signature		
`cbuffer`	`double` × `double` × `float`	→	`cbuffer`
	`geompoint` × `float`	→	`cbuffer`

Table 7.1: Constructors for `cbuffer`.

The following query shows how to construct a `cbuffer` with centre $(2, 2)$ and radius 1.5 (Figure 6.6) using the constructor function.

```
SELECT asText(cbuffer(ST_Point(2,2), 1.5));
-- Cbuffer(POINT(2 2),1.5)
```

The system automatically checks the constraints to ensure that a `cbuffer` value is well defined: the first component must be a two-dimensional point, and the second component must be a non-negative floating-point number. Examples of incorrect circular buffer type values are as follows.

```
-- incorrect point value
SELECT cbuffer 'Cbuffer(Linestring(2 2, 3 3), 3.0)';
-- incorrect 3D point
SELECT cbuffer 'Cbuffer(Point Z(1 1 1), 3.0)';
-- incorrect radius value
SELECT cbuffer 'Cbuffer(Point(1 1), -3.0)';
```

7.1.2 Accessors

The `cbuffer` type includes two accessor functions. Specifically, as illustrated in Table 7.2, given a `cbuffer` value, `point` returns the centre of the circular region as a `geometry`, while `radius` returns the radius.

Considering again the `cbuffer` centred at $(2, 2)$ with radius 1.5, we obtain the following queries.

Function	Signature		
point	cbuffer	→	geompoint
radius	cbuffer	→	float

Table 7.2: Accessors functions for `cbuffer`.

```
SELECT ST_AsText(point(cbuffer 'Cbuffer(Point(2 2), 1.5)'));
-- Point(2 2)
SELECT radius(cbuffer 'Cbuffer(Point(2 2), 1.5)');
-- 1.5
```

7.1.3 Spatial Operations

There are also three spatial operations (see Table 7.3).

Function	Signature		
SRID	cbuffer	→	integer
setSRID	cbuffer	→	cbuffer
transform	cbuffer × integer	→	cbuffer

Table 7.3: Spatial operations for `cbuffer`.

The operator `SRID` returns the spatial reference identifier, whereas `setSRID` assigns a spatial reference identifier to the circular region. If we want to return or set the spatial reference identifier we can execute the following queries.

```
SELECT SRID(cbuffer 'Cbuffer(SRID=5676;Point(2 2), 1.5)');
-- 5676
SELECT asEWKT(setSRID(cbuffer 'Cbuffer(Point(0 0), 1)', 4326));
-- SRID=4326;Cbuffer(POINT(0 0),1)
```

If we want to convert a `cbuffer` from one spatial reference system to another, we must use the `transform` function. This function applies the transformation specified by the target SRID. An error is raised if the input circular buffer has an unknown SRID (represented by 0).

7.1.4 Comparisons

In addition, the `cbuffer` type can support standard comparison operators (`=`, `<>`, `<`, `>`, `<=`, `>=`). We denote these operators as `op` in Table 7.4. They compare first the centre points and then the radii of the arguments. Actually, equality and

Function	Signature
op	cbuffer × cbuffer → boolean

Table 7.4: Comparison operators with op ∈ {=, <>, <, >, <=, >=}.

inequality are the comparison operators useful in the real world but the whole set of comparison operators is available since they allow B-tree indexes to be constructed on circular buffers. Some examples are given below.

```
SELECT cbuffer 'Cbuffer(Point(2 2), 1.5)' =
  cbuffer 'Cbuffer(Point(2 2), 1.5)';
-- true
SELECT cbuffer 'Cbuffer(Point(1 1), 0.6)' >=
  cbuffer 'Cbuffer(Point(1 1), 0.5)';
-- true
```

7.2 Temporal Circular Buffer

As defined in the previous chapter, the *temporal circular buffer*, modelled by the type `tcbuffer`, represents the movement of a circular buffer over time. An example of a `tcbuffer` value defined on January 1 2001, at position (2, 2) with radius 1.5, is shown below.

```
SELECT tcbuffer 'Cbuffer(Point(2 2), 1.5)@2001-01-01';
```

As all temporal types in MobilityDB, the temporal circular buffer comes in three subtypes, namely, *instant*, *sequence*, and *sequence set*, which states how a value of type `tcbuffer` evolves between successive instants.

- A temporal circular buffer of subtype *instant* (`tcbufferInst`) represents the circular buffer at a time instant. For example:

  ```
  SELECT tcbuffer 'Cbuffer(Point(1 1), 0.5)@2001-01-01';
  ```

- A temporal circular buffer of subtype *sequence* represents the evolution of the circular buffer during a sequence of time instants, where the values between these instants are interpolated using a discrete, step, or a linear function. In MobilityDB it is denoted as `tcbufferSeq`. These are some examples:

  ```
  -- discrete interpolation
    SELECT tcbuffer '{Cbuffer(Point(1 1), 0.3)@2001-01-01, Cbuffer(
      Point(2 2), 0.5)@2001-01-02, Cbuffer(Point(1 1), 3)@2001-01-05}';
  ```

It represents a temporal circular buffer with known values only on January 1, January 2 and January 5, 2001.

```
-- step interpolation
SELECT tcbuffer 'Interp=Step;[Cbuffer(Point(4 4), 2)@2001-01-01,
  Cbuffer(Point(7.5 10), 2.5)@2001-01-02, Cbuffer(Point(16 15), 5)
  @2001-01-05]';
```

It models a `tcbuffer` value that is defined on January 1, changes on January 2, and holds the value `Cbuffer(Point(7.5 10), 2.5)` from January 2 to January 4 and then it is set to `Cbuffer(Point(16 15), 5)` on January 5.

```
-- linear interpolation
SELECT tcbuffer '[Cbuffer(Point(4 4), 2)@2001-01-01, Cbuffer(Point
  (7.5 10), 2.5)@2001-01-02, Cbuffer(Point(16 15), 5)@2001-01-05]';
```

If no `Interp` setting is specified, linear interpolation between consecutive instants is the default. Thus, this value differs from the previous one because from January 1 to January 2 and from January 2 to January 4 it is no longer a constant value but it varies linearly.

- A `tcbuffer` value of *sequence set* subtype (briefly, a sequence set value), denoted as `tcbufferSeqSet`, represents the evolution of the value at a set of sequences, where the values between these sequences are unknown. An example is as follows:

  ```
  SELECT tcbuffer '{[Cbuffer(Point(1 1), 1)@2001-01-01, Cbuffer(
    Point(2 2), 1)@2001-01-03], [Cbuffer(Point(2 2), 2)@2001-01-05,
    Cbuffer(Point(2 2), 3)@2001-01-06]}';
  ```

  Between January 1 and January 3, the object moves linearly. There is then a gap in the data, with no information available for January 4. The object resumes linear movement from January 5 to January 6.

7.2.1 Constructors

The `tcbuffer` type can be constructed in several ways, depending on the temporal characteristics of the value to be created. Table 7.5 summarises all available constructor functions for the `tcbuffer` type.

The first four constructors correspond to temporal circular buffers whose value remains constant within each temporal instant or interval. These constructors allow the user to specify either a single *timestamp*, a *timestamp set*, a *time span*, or a *span set*. In the signatures, the parameter `interpL` indicates that linear interpolation is used between the start and end instants of the temporal interval.

Function	Signature		
tcbuffer	cbuffer × timestamptz	→	tcbufferInst
tcbuffer	cbuffer × tstzset	→	tcbufferDiscSeq
tcbuffer	cbuffer × tstzspan × interpL	→	tcbufferContSeq
tcbuffer	cbuffer × tstzspanset × interpL	→	tcbufferSeqSet
tcbufferSeq	tcbufferInst[] × interp × leftInc × rightInc	→	tcbufferSeq
tcbufferSeqset	tcbuffer[]	→	tcbufferSeqSet
tcbufferSeqSetGaps	tcbufferInst[] × maxt × maxdist × interpL	→	tcbufferSeqSet
tcbuffer	tgeompoint × tfloat	→	tcbuffer

Table 7.5: Constructors functions for `tcbuffer`.

The `tcbufferSeq` constructor creates a temporal circular buffer of sequence subtype. In this case, the `interp` parameter specifies the type of interpolation between consecutive instants, i.e., `stepwise` or `linear`, depending on whether the value remains constant or varies linearly over time. The `leftInc` and `rightInc` parameters are boolean values (*true* by default) that indicate whether the left and right bounds of the sequence, respectively, are included (*true*) or excluded (*false*).

The constructors `tcbufferSeqset` and `tcbufferSeqSetGaps` create temporal circular buffers of sequence-set subtype. These functions allow the user to specify multiple sequences, either explicitly or by automatically inserting gaps when the temporal distance between instants exceeds a given threshold.

Finally, the last constructor builds a `tcbuffer` directly from a temporal geometry point and a temporal float. Two examples are provided below.

```
SELECT asText(tcbuffer('Cbuffer(Point(1 1), 0.2)', tstzspanset '
  {[2001-01-01, 2001-01-03]}', 'step'));
-- Interp=Step;{[Cbuffer(Point(1 1),0.2)@2001-01-01, Cbuffer(Point(1 1)
  ,0.2)@2001-01-03]}
SELECT asText(tcbufferSeqSetGaps(ARRAY[tcbuffer 'Cbuffer(Point(1 1)
  ,0.1)@2001-01-01', 'Cbuffer(Point(1 1),0.3)@2001-01-03', 'Cbuffer(
  Point(1 1),0.5)@2001-01-05'], '1 day'));
-- {[Cbuffer(Point(1 1),0.1)@2001-01-01], [Cbuffer(Point(1 1),0.3)@2001
  -01-03], [Cbuffer(Point(1 1),0.5)@2001-01-05]}
```

7.2.2 Accessors

A large list of accessor functions exists to retrieve the duration, memory size, individual values, instants, timestamps, and more. These functions are listed in Table 7.6. The function `interp`, applies to temporal sequences and sequence sets,

Function	Signature		
tempSubType	tcbuffer	→	text
interp	tcbuffer	→	text
setInterp	tcbuffer × text	→	tcbuffer
memSize	tcbuffer	→	integer
points	tcbuffer	→	bigintset
getTime	tcbuffer	→	tstzspanset
getValues	tcbuffer	→	cbufferset
valueAtTimestamp	tcbuffer × timestamptz	→	cbuffer
startValue	tcbuffer	→	cbuffer
startTimestamp	tcbuffer	→	timestamp
startInstant	tcbuffer	→	tcbuffer
instants	tcbuffer	→	tcbufferInst[]

Table 7.6: Accessors functions for `tcbuffer`.

returning the interpolation method in use, i.e., {`Discrete`, `Stepwise`, `Linear`}. The `setInterp` function allows the interpolation method of a temporal circular buffer to be modified. Other accessor functions are defined to retrieve individual values, timestamps or sequences from the temporal geometry. The functions `valueAtTimestamp` and `startTimestamp` are illustrated in the examples below.

```
SELECT asText(valueAtTimestamp(tcbuffer 'Cbuffer(Point(2 2), 1.5)@2001
  -01-01','2001-01-01'));
-- Cbuffer(POINT(2 2),1.5)
SELECT startTimestamp(tcbuffer '[Cbuffer(Point(4 4), 2)@2001-01-01,
  Cbuffer(Point(7.5 10), 2.5)@2001-01-02, Cbuffer(Point(16 15), 5)@2001
  -01-05]');
--  2001-01-01 00:00:00+01
```

7.2.3 Comparison Operators

A set of comparison operators is available for the `tcbuffer` type, allowing the comparison of two temporal circular buffers. These operators, denoted as `op`, can be one of {=, <>, <, >, <=, >=}. Their signatures are reported in Table 7.7.

The equality and inequality operators introduced above return a single boolean value for the entire temporal extent of the objects. Their semantics are defined over the full duration of the temporal circular buffers: at each instant, the operators compare both the centre and the radius of the two values. Equality is satisfied only if the centres and radii are equal at every instant; conversely, inequality

Operator	Signature
op	tcbuffer × tcbuffer → boolean

Table 7.7: Comparison operators with op ∈ {=, <>, <, >, <=, >=}.

holds if there exists at least one instant at which either the centres or the radii differ. However, in many cases one may be interested in identifying the time periods during which two temporal geometries coincide or differ. To this end, the following two operators return a temporal boolean representing the result over time of the equality or inequality operator on the input values. Table 7.8 contains the signatures of these operators.

Operator	Signature
#=	tcbuffer × {tcbuffer, cbuffer} → tbool
#<>	tcbuffer × {tcbuffer, cbuffer} → tbool

Table 7.8: Temporal comparison operators.

Below we present an example of temporal equality between a temporal circular buffer and a circular buffer.

```
SELECT tcbuffer '[Cbuffer(Point(1 1), 0.2)@2001-01-01, Cbuffer(Point(1
  1), 0.4)@2001-01-03)' #= cbuffer 'Cbuffer(Point(1 1), 0.3)';
-- {[f@2001-01-01, t@2001-01-02], (f@2001-01-02, f@2001-01-03)}
```

7.2.4 Spatial Functions

In this section, we present two main spatial and spatiotemporal functions: one computing the area traversed by a `tcbuffer` value, and one capturing the temporal evolution of its distance to another object. The distance function supports several argument types, allowing the `tcbuffer` to be compared with another `tcbuffer`, with a static `cbuffer`, with a moving point (`tgeompoint`), or with a static point (`geometry(point)`). Moreover, we implement several functions that express topological relationships. In particular, Table 7.9 reports the signatures of the spatial functions defined in this section, where the regions and moving regions under consideration are defined as `regs` = {`tcbuffer`, `cbuffer`, `tgeompoint`, `geometry(point)`}.

The `traversedArea` function returns a polygon representing the area traversed by the moving region. The computation of the traversed area is described in

Function	Signature		
traversedArea	{tcbuffer, cbuffer}	→	geometry
tDistance	regs × regs	→	tfloat
nearestApproachDistance	regs × regs	→	float
nearestApproachInstant	regs × regs	→	tgeometry(Instant)
shortestLine	regs × regs	→	geometry(LineString)

Table 7.9: Spatial functions.

Section 6.2.3. The following query computes the traversed area of the `tcbuffer` shown in Figure 6.11a, producing the polygon illustrated in Figure 6.11b.

```
SELECT traversedArea(tcbuffer '[CBuffer(Point(4 4), 1)@2001-01-01,
  CBuffer(Point(12 10), 4)@2001-01-02, CBuffer(Point(18 8), 3)@2001
  -01-03]');
```

The `tDistance` function, represented by the operator `<->` in MobilityDB, computes the distance at each instant between a `tcbuffer` and another moving region (either a temporal point or a temporal circular buffer). The method used to compute this temporal distance is described in Section 6.2.3. The following query computes the temporal distance between a point and the `tcbuffer` shown in Figure 6.12a.

```
SELECT tcbuffer '[CBuffer(Point(4 4), 2)@2001-01-01,  CBuffer(Point(16
  15), 5)@2001-01-04]' <-> geometry 'Point(14 3)';
-- [8.05@2001-01-01 00:00:00+01, 4.17@2001-01-02 08:20:37.5+01, 7.17
  @2001-01-04 00:00:00+01]
```

As a second example we compute the temporal distance between the two temporal circular buffers defined in Figure 6.12b.

```
SELECT tcbuffer '[CBuffer(Point(1 1), 0.5)@2001-01-01,  CBuffer(Point(4
   4), 0.5)@2001-01-04]' <-> tcbuffer '[CBuffer(Point(1 4), 0.5)@2001
  -01-01,  CBuffer(Point(4 1), 0.5)@2001-01-04]';
--  [2@2001-01-01 00:00:00+01, 0@2001-01-02 00:00:00+01, 0@2001-01-03
  00:00:00+01, 2@2001-01-04 00:00:00+01]
```

As explained in Section 6.2.3, the temporal distance function does not only evaluate the distance at the instants explicitly defined in the `tcbuffer`, but it also computes the additional instants at which a *turning point* occurs. By incorporating the turning points, the resulting temporal distance function accurately reflects the geometric evolution of the moving region over time.

Beyond the `traversedArea` and `tDistance` functions, additional operations are available to further improve the expressivity and usability of the proposed

data types. These include the `nearestApproachDistance`, `nearestApproachInstant`, and `shortestLine` functions. The implementation of these three functions is straightforward, provided that the `tDistance` operator is already implemented. Given two moving circular buffers, the `nearestApproachDistance` returns the smallest distance ever reached between them during their movement. Its implementation first computes the temporal distance between the objects using the `tDistance` operator, and then applies the `minValue` function provided by MobilityDB to extract the minimum. This function is particularly relevant in nearest-neighbour search applications.

The `nearestApproachInstant` function returns the timestamp at which this minimum distance is attained. As in the previous case, the temporal distance is first computed, and the function then extracts the instant corresponding to the smallest distance. If multiple instants yield the same minimum value, the first one is returned. Finally, the `shortestLine` function computes the shortest line segment between the two temporal circular buffers at their point of closest approach, i.e., at the instant returned by the `nearestApproachInstant` function.

Ever and Always Spatial Relationships

Finally, several topological relationships available in PostGIS have been generalised to the `tcbuffer` type in order to support the *ever* and *always* comparisons introduced in the previous chapter. Table 7.10 lists the *ever* and *always* spatial relationships defined for the `tcbuffer` in MobilityDB. The placeholder `Func` denotes one of the spatial predicates {`Disjoint`, `Intersects`, `Touches`}, while the spatial and spatiotemporal arguments involved in these predicates belong to the set `regs` = {`tcbuffer`, `cbuffer`, `tgeompoint`, `geometry(point)`}.

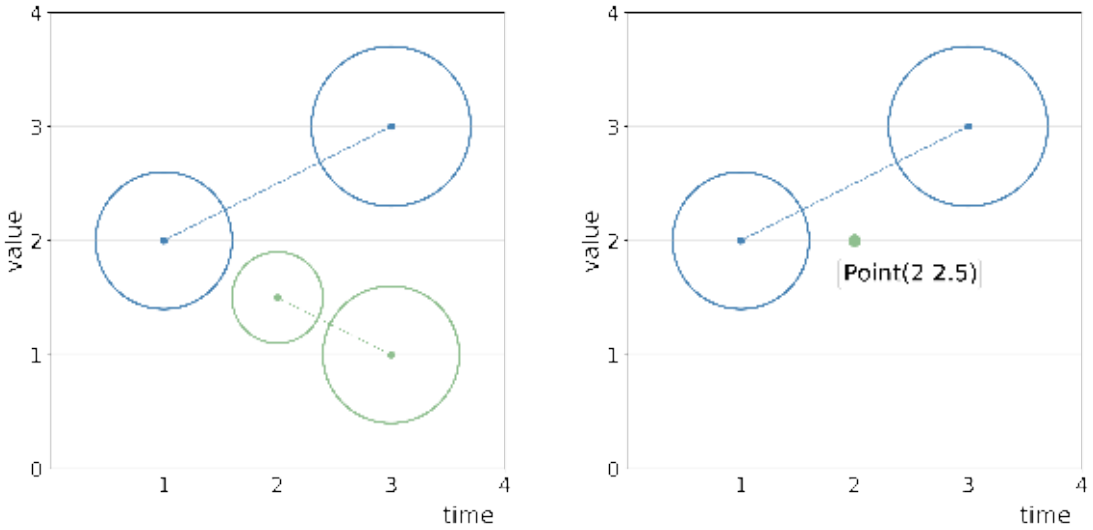
Function	**Signature**		
`eContains`	`geometry` × `tcbuffer`	→	`boolean`
`aContains`	`geometry` × `tcbuffer`	→	`boolean`
`eFunc`	`regs` × `regs`	→	`boolean`
`aFunc`	`regs` × `regs`	→	`boolean`
`eDwithin`	`regs` × `regs` × `float`	→	`boolean`
`aDwithin`	`regs` × `regs` × `float`	→	`boolean`

Table 7.10: Ever and always spatial relationships.

An example illustrating the use of these functions is provided below. Figure 7.1a shows two temporal circular buffers that move over time and are evaluated using the `aIntersects` function, which returns *false*. Figure 7.1b instead illustrates a temporal circular buffer and a static point. When applying the `eContains`

function, and assuming linear interpolation between instants, the point falls within the temporal circular buffer at some instant, and the query therefore returns *true*.

```
SELECT aIntersects(tcbuffer '[Cbuffer(Point(1 2), 0.6)@2001-01-01,
  Cbuffer(Point(3 3), 0.7)@2001-01-03)', tcbuffer '[Cbuffer(Point(2 1.5)
  ,0.4)@2001-01-02, Cbuffer(Point(3 1), 0.6)@2001-01-03)');
-- false
SELECT  eContains(tcbuffer'[Cbuffer(Point(1 2), 0.6)@2001-01-01,
  Cbuffer(Point(3 3), 0.7)@2001-01-03)', geometry 'Point(2 2.5)');
-- true
```



(a) Example of `aIntersects`.

(b) Example of `eContains`.

Figure 7.1: Examples of ever and always comparisons: (a) `aIntersects` between two `tcbuffers`; (b) `eContains` between a `tcbuffer` and a point.

7.3 Case Study

We illustrate the usefulness of the `tcbuffer` data type and its implementation in MobilityDB in a concrete application. As a case study, we consider the underwater noise pollution generated by moving vessels. In fact, in this setting a standard assumption is that the underwater noise emitted by a vessel in a given position and time instant propagates instantaneously over a bounded area of influence. This area, projected over the sea surface, is a circle centred in the vessel's position, i.e., a `cbuffer` value. By associating also the timestamp the overall data structure is exactly a `tcbuffer`. From this perspective, a moving vessel, which is usually

described by its trajectory, when its emitted underwater noise is also considered, is naturally modelled as a *moving region*.

To present realistic examples of possible queries involving the use of temporal circular buffers, we build on the underwater noise model introduced in Chapters 3, 4, and 5. In particular, we consider a case study on the underwater noise pollution generated by different types of vessels operating in the Northern and Central Adriatic Sea. We proceed by presenting some examples, which assume the availability of the following tables and attributes.

- `moving_points`, containing the `geom`, `date_time` and `radius` attributes, representing the vessel position, timestamp and noise propagation radius, respectively.
- `vessel_trip`, containing the `trip_id` and `noise_imp` attributes that identify the vessel's trip and its noise-impacted area, respectively. The `noise_imp` attribute is constructed from the information stored in the `moving_points` table.
- `natural_areas`, containing the `id` and `wkb_geometry` attributes that represent the protected area identifier and its geometry, respectively.

7.3.1 Underwater Noise Affected Area of a Vessel Trip

Table `moving_points` allows for computing a vessel's underwater noise impact throughout its trip as the area affected by noise along its trajectory. The following query constructs a `tcbufferSeq` from the trajectory of a vessel and stores it in the `vessel_trip` table. For each position in the vessel's trajectory, ordered by increasing timestamps (`date_time`), we create a `cbuffer` centred at the vessel's location, with a radius equal to the estimated noise propagation radius. Each of these `cbuffer` instances, together with its associated timestamp, forms an element of the resulting `tcbufferSeq`, which models the spatiotemporal extent of the moving region.

```
INSERT INTO vessel_trip(trip_id, noise_imp)
SELECT trip_id, tcbufferSeq(array_agg(tcbuffer(cbuffer(geom, radius),
  date_time) ORDER BY date_time))
FROM moving_points
GROUP BY trip_id;
```

Furthermore, the next query applies the `traversedArea` function to the obtained `tcbufferSeq` to compute the overall area covered during the entire trip:

```
SELECT trip_id, traversedArea(noise_imp) AS travArea
FROM vessel_trip;
```

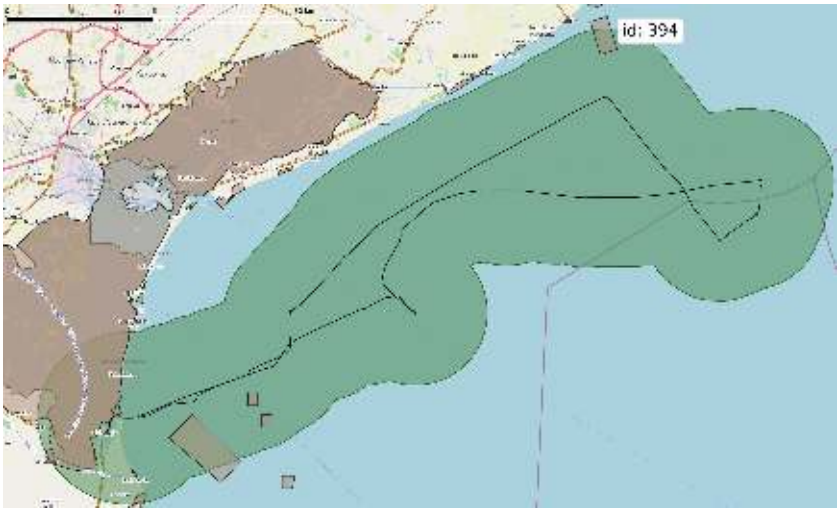


Figure 7.2: Trajectory of a fishing vessel (in black) along with the portion of the Adriatic Sea affected by its underwater noise emissions (in green). Protected areas are shown in brown.

The query result can be visualised in QGIS as shown in Figure 7.2. The vessel's trajectory is depicted as a black line, while the area of the Adriatic Sea affected by its underwater noise is highlighted in green. The image also shows some small protected areas, depicted as brown polygons. Note that although the vessel's trajectory avoids entering protected areas, as required by current maritime regulations, their emitted noise actually affects some of these zones, potentially impacting their marine ecosystems.

7.3.2 Quietness of Protected Areas

As shown in Figure 7.2 the transit and fishing bans in protected areas do not guarantee the quietness of the aquatic life in those areas, since the underwater noise produced by vessels passing nearby can be audible several kilometres away from the actual positions of the vessels. The queries presented in this section consider the vessel trip shown in Figure 7.2 (`trip_id` = 2486) and one of the protected areas (identified by `id` = 394 and shown in the top right side of the map). We use the functions `aDwithin` and `eIntersects` to check whether the vessel and its underwater noise affects the protected area along its trip.

A first question is whether the vessel's moving region is always sufficiently far away from the protected area (i.e., at least 1 km far) to ensure that its marine ecosystem is also protected from underwater noise pollution. To answer this question, the following query uses the `aDwithin` function that checks if the geometry

of the protected area and the vessel's moving region (comprising the underwater noise affected area) are always at least 1 km far from each other. In the considered example the query result is `false` since the vessel approaches the protected area during its trip and the distance of its moving region becomes less than 1 km.

```
SELECT aDwithin(na.wkb_geometry,vt.noise_imp, 1000)
FROM vessel_trip vt, natural_areas na
WHERE vt.trip_id=2486 AND na.id=394;
--  false
```

A second question is whether the vessel's approach makes its underwater noise actually audible in the protected area. To this end, we use the `eIntersects` function that checks if there exists a time interval in which the two geometries, namely the protected area and the fishing vessel's moving region, intersect. In the considered example the answer is `true` because the two geometries actually intersect each other for a time period exactly when the vessel approaches the protected area.

```
SELECT eIntersect(na.wkb_geometry,vt.noise_imp)
FROM vessel_trip vt, natural_areas na
WHERE vt.trip_id=2486 AND na.id=394;
--  true
```

7.3.3 Distance between Vessels' Moving Regions

Figure 7.3 shows two vessels navigating in the same area and period of time. Their corresponding traversed areas are shown as red and grey areas, respectively. If the two areas get too close and intersect, the corresponding marine area is more exposed to underwater noise pollution. In this section we show how to monitor their distance over time.

We first create two views of the `vessel_trip` table containing the moving regions of the vessels of interest:

```
CREATE VIEW traj193(noise_imp)
AS SELECT noise_imp
FROM vessel_trip WHERE trip_id=193;

CREATE VIEW traj303(noise_imp)
AS SELECT noise_imp
FROM vessel_trip WHERE trip_id=303;
```

The following query allows for monitoring how the distance between the two moving regions shown in Figure 7.3 varies in time. It employs the `tDistance` function, which returns an array showing, for all timestamps, the distance between the

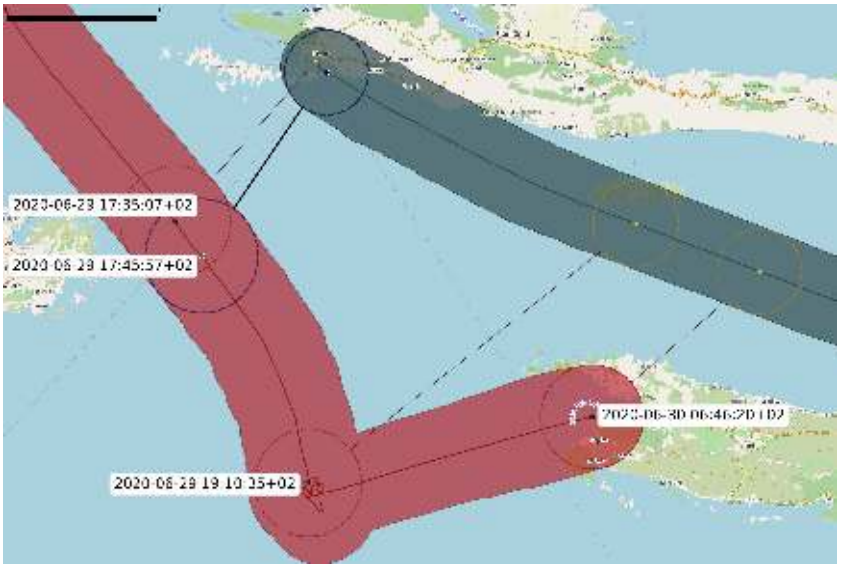


Figure 7.3: Two vessels trips and their moving regions. Dashed lines indicate some of the `tDistance` calculated values while the solid line indicates the minimal value, i.e., the shortest distance w.r.t all timestamps.

two given geometries. Both the query output and Figure 7.3 display only a subset of the resulting values, with the latter visualising the corresponding distances as dashed lines.

```
SELECT tDistance(a.noise_imp, b.noise_imp)
FROM traj193 a, traj303 b;
-- [9521.7@2020-06-29 17:35:07+02, ..., 8161.23@2020-06-29 17:45:57+02,
   ..., 21459.72@2020-06-29 19:10:35+02, ..., 8535.33@2020-06-30
  06:46:20+02]
```

It is also possible to determine the time instant when the two moving regions are at the minimal distance as well as the length of such distance. To this aim, the following queries use the `nearestApproachInstant` and `nearestApproachDistance` functions, respectively. The resulting time instant is `2020-06-29 17:45:57+02` while the minimal distance is 8161.23 m.

```
SELECT nearestApproachInstant(a.noise_imp, b.noise_imp)
FROM traj193 a, traj303 b;
-- '2020-06-29 17:45:57+02'

SELECT nearestApproachDistance(a.noise_imp, b.noise_imp)
FROM traj193 a, traj303 b;
-- 8161.23
```

Finally, the `shortestLine` function can be used to transform the minimal distance into a line geometry, which corresponds to the solid black line depicted in Figure 7.3 and which is generated by the following query:

```
SELECT ST_AsText(shortestLine(a.noise_imp, b.noise_imp))
FROM traj193 a, traj303 b;
-- LINESTRING(3360168.89 4787840.39, 3355911.58 4780877.54)
```

The examples presented in this section highlight the modelling flexibility and applicability of the proposed `tcbuffer` data type. By representing the underwater noise footprint of a vessel as a circular moving region, it becomes possible to formulate rich spatiotemporal queries that jointly reason about geometry, time, and deformation. Operations such as `traversedArea`, `tDistance`, `nearestApproachInstant`, and the family of ever and always comparisons can be expressed concisely and evaluated efficiently within MobilityDB.

Without the support of the temporal circular buffer, these analyses would require a significantly more complex processing pipeline. In particular, the trajectory of the vessel would have to be discretised into a sequence of static buffers, with all pairwise spatial relationships computed explicitly at each time instant and temporal semantics reconstructed externally. Such an approach is computationally expensive and unable to capture the object's behaviour over time, since it relies on arbitrary temporal sampling. Moreover, operations such as the minimal distance between two deforming regions or the computation of their shortest connecting segment would be extremely difficult to express without a native spatiotemporal type. By contrast, the `tcbuffer` provides an abstraction that encapsulates spatial and temporal variation within a single data type. The database system is responsible for handling temporal interpolation, geometric transformations, and the evaluation of spatiotemporal predicates, enabling the formulation of queries using built-in functions that are efficient, reliable, and optimised within the system. The case study presented in this section highlights the advantages of the temporal circular buffer in real-world analytical scenarios, illustrating how it substantially enhances the modelling and querying capabilities of MobilityDB.

Part III

Urban Mobility

Chapter 8

Analysis of Urban Mobility in Venice

Mobility encompasses a wide range of phenomena occurring across different domains each characterised by specific spatial and temporal dynamics, scales of interaction, and relationships with the surrounding environment. Understanding mobility therefore requires a comprehensive perspective that combines spatial, temporal, and contextual analysis to capture the complexity of moving entities and their interactions. After investigating maritime mobility, through the reconstruction and semantic enrichment of vessel trajectories and the development of a model for underwater noise propagation, this final part of the thesis turns to the study of *urban mobility*. Urban mobility plays a crucial role in shaping the functioning and sustainability of cities. It affects accessibility, social inclusion, and environmental quality, while also reflecting broader economic and cultural dynamics. Understanding how people move within urban environments is therefore essential to designing efficient transport systems, alleviating congestion and emissions, and fostering more sustainable urban spaces. In recent years, many cities have also been facing the challenge of *overtourism* [81], a term denoting the significant and escalating pressure tourism places on destination areas, affecting infrastructure, public services, and residents' quality of life. In this context, analysing urban mobility becomes even more important, as it supports the design of strategies to balance accessibility and sustainability, ensure public safety, and mitigate the impacts of tourism on urban ecosystems.

For the study of urban mobility, Venice represents a unique urban environment. Its unique urban structure — a network of islands separated by canals and connected by bridges — and the absence of road infrastructure make water transport the primary means of movement across the city. This distinctive configuration, combined with a strong tourist presence, creates complex mobility dynamics that differ substantially from those of other urban environments.

The whole municipality of Venice is divided into three urban areas: the *historic city* together with some smaller islands such as Murano, Burano and Torcello, *Lido*, a barrier island that serves as both a residential district of Venice and a physical boundary between the lagoon and the Adriatic Sea, and *Mainland*, the urban settlement of Mestre and Marghera. On the Lido and the mainland, mobility is supported by the regular circulation of cars and buses, whereas Venice historic city represents the largest urban area in Europe where road traffic is absent. In this unique context, the movement of passengers and goods relies primarily on boats and ferries navigating the city's canal network. The main means of transportation for passengers include motorised water buses (*vaporetti*) and private taxis, which operate on regular routes between the city's different islands, along with cargo vessels and private boats.

In addition, Venice is internationally recognised as one of the leading tourist destinations in the world, attracting millions of visitors each year due to its distinctive urban landscape and extraordinary cultural heritage. In recent years, however, Venice has garnered attention as an early example of *overtourism*. One of the most significant effects is the increasing unaffordability of housing for both residents, students, and those employed in the tourism industry [121]. According to the *"Yearbook of Tourism"*,[1] a data collection commissioned by the *Tourism Sector of the City of Venice*, tourist arrivals to Venice and its surrounding areas have shown a continuous rise in the last years. In 2024, arrivals reached nearly 6 million, surpassing the levels recorded in 2019, prior to the COVID-19 pandemic that had caused a sharp decline in visitor numbers. These figures not only highlight the rapid recovery of tourism within five years but also underscore a critical imbalance: in the historic city alone, annual tourist arrivals exceed 4 million, compared with a resident population of fewer than 50,000.

In order to explore new policies for more sustainable tourism, and to promote knowledge-based governance of a city that is exceptional and remarkably fragile, we analyse a large public transportation dataset consisting of ticket validations provided by ACTV (*Azienda del Consorzio Trasporti Veneziano*), Venice's primary public transport company. ACTV manages a multimodal network of land and water transport services, which ensures the mobility of tourists, residents, students, and workers, supporting the daily life of the city. The dataset encompasses a heterogeneous range of tickets, including short-duration passes predominantly purchased by tourists, and various long-term subscriptions typically held by residents, workers, students, and retirees. Furthermore, the dataset contains ticket validations across three distinct seasons — winter, spring, and the transition from summer to autumn — and, more interestingly, document the mobility of users during two globally renowned events, the Carnival and the Venice Film Festival, as

[1] `https://www.comune.venezia.it/it/content/studi`

well as during major holiday periods such as Easter and summer, thus providing a comprehensive representation of both ordinary and exceptional urban mobility dynamics in Venice. The dataset, together with the cleaning procedures employed to eliminate redundant and erroneous records, is described in detail, followed by some analyses designed to illustrate the richness and potential of this data collection for research and policy-making.

With regard to the analyses, we begin by examining seasonal trends to assess variations in mobility patterns across the three periods under study. We then investigate the daily distribution of validations to identify behavioural differences associated with specific events such as Carnival, Easter, and the Venice Film Festival. Finally, we analyse the hourly distribution of validations, offering insights into time-of-day mobility preferences across different user groups.

In addition, we look into individual travel dynamics by reconstructing sequences of ticket validations. This is challenging because users validate their tickets at the entrance of the means of transport but not at the exit. Consequently, the reconstructed trajectories are very sparse. For privacy reasons, in the case of subscription-based tickets, identifiers are reset on a daily basis, preventing the reconstruction of multi-day travel patterns that could compromise user anonymity. As a result, the average trajectory length for residents, students, workers, and retirees is approximately 1.5. For these user groups, our analysis therefore concentrates on pairs of consecutive validations, examining the longest dwell time (i.e., the maximum interval between two successive validations), the typical morning departure time, the afternoon return time, and the most frequently used stops. By contrast, tourist trajectories are longer and often span multiple days. This enables us to observe how tourists distribute their mobility over time and how they utilise the Venetian public transport system during their stay.

After conducting a detailed exploratory analysis of user behaviours across different time periods and event-driven contexts in Venice, we aim to identify recurrent patterns in stop usage and group stops that exhibit similar characteristics. To this end, we apply unsupervised clustering techniques to classify stops based on their usage profiles. This approach enables us to reveal meaningful spatial and behavioural structures within the transport network and to better understand how public transport is utilised across diverse temporal scenarios. In this study, we adopt two distinct clustering strategies. The first approach focuses on user categories: stops are considered similar if they are predominantly used by the same categories of users. The second approach relies on temporal activity profiles, grouping stops that exhibit comparable levels of validation activity across different time slots throughout the day. These complementary perspectives allow us to capture both the variation in user-type distribution, and the temporal patterns of stop utilisation across the transport network. Finally, we compare the clustering

results across distinct time periods to assess the stability or variability of usage patterns over time.

8.1 Related Work

The increasing availability of smart card data from Automated Fare Collection (AFC) systems has fostered a wide range of studies aimed at improving our understanding of urban mobility and public transport usage. These studies have primarily focused on three main research directions: origin-destination (OD) matrix estimation, temporal and spatial clustering of usage patterns, and user behaviour analysis over time.

A first line of work has addressed the challenge of estimating OD matrices in contexts where only boarding information is available. Massobrio and Nesmachnow [76] proposed a methodology to estimate OD matrices for Montevideo's public transport system, based on open smart card data retrieved from Uruguay's national open data catalogue and geographic layers from the Montevideo GIS portal. The dataset includes GPS-based bus location data and smart card ticket sales for the year 2015, with an average of approximately 3.31 million transactions on weekdays, around 2.19 million on Saturdays, and 1.28 million on Sundays. Their OD estimation model was validated against user survey data using Spearman correlation, demonstrating a strong correspondence between inferred and reported flows. Similarly, Lee et al. [70] proposed a probabilistic framework for estimating destinations in a tap-in-only smart card system, using long-term transactional data collected in Sejong City, South Korea. The dataset includes more than 1.3 million trip records generated by $63,014$ passengers. While standard trip chain methods were initially applied, the authors addressed the limitations posed by *unlinked trips* (i.e., those lacking inferable destinations) by developing a method based on temporal travel patterns and historical boarding records. Passengers were first clustered via k-means, and time-of-day travel behaviours were modelled using Gaussian mixture models for each cluster. This combination of clustering and probabilistic modelling significantly improved destination matching, particularly for unlinked trips.

Other studies have employed clustering to characterise patterns of mobility, both at the level of individual users and of network elements such as stops or stations. Cats and Ferranti [16] analysed smart card data from Stockholm County, Sweden, collected through a system that records both boarding and alighting events. The dataset covers 46 weeks of multimodal travel in 2019, and includes over 371 million journeys performed by more than 4.4 million users. By applying k-means and hierarchical classification, the authors identified ten daily and five weekly user profiles, capturing different commuting patterns such as regular peak-time travellers and early morning riders. Viallard et al. [138], working with

data from the *Société de Transport de l'Outaouais* (STO) in Gatineau, Canada, used weekly smart card transactions (35.4 million) to explore the evolution of user behaviour over time. The smart card system used by STO, in place since 2001, covers over 80% of transit users and served as the basis for the dataset analysed in this study. Their iterative clustering model highlighted the benefit of temporal continuity in capturing behavioural changes, such as those due to holidays or seasonal variation. Clustering techniques have also been used to explore the temporal dynamics of network elements. El Mahrsi et al. [33] analysed a real-world dataset comprising four weeks of smart card transactions in the Rennes metropolitan area (France), provided by the local transit authority. The dataset covers the month of April 2014 and includes over 5.4 million journey transactions, of which approximately 4.3 million (80%) were recorded through 134,979 smart cards. To reveal urban mobility patterns, the authors proposed two complementary clustering approaches applied to smart card data: the first classifies stations according to their temporal activity profiles, while the second identifies passenger groups based on their aggregated weekly boarding behaviour. He et al. [59] extended the idea of spatiotemporal characterisation using smart card data from the STO system in Gatineau. Their methodology combined Euclidean distance between stops with Dynamic Time Warping (DTW) on transaction times to produce hierarchical clusters of user activity locations, taking into account both the spatial and temporal dimensions of mobility. The analysis was conducted using all weekday transaction data from May 2014, comprising more than 1.1 million trips.

Several of these methodologies align with the general direction of our work, although our focus differs in scope and objectives. In contrast to OD estimation techniques, our work does not attempt to reconstruct full journeys. Rather, it focuses on the characterisation of stop usage through clustering methods based on temporal patterns and user behaviour. Similarly to [33], we explore temporal patterns at the stop level; however, our method also incorporates a component for analysing similarity in user types or time-of-day usage profiles. While He et al. [59] introduce a sophisticated similarity measure via DTW, our approach prioritises interpretability by employing standard clustering techniques based on dissimilarities in user-type distributions or time-of-day usage patterns across stops.

The impacts of external events on public transport usage have also been studied through smart card data. Jenelius and Cebecauer [67] examined daily public transport ridership during the early months of the COVID-19 pandemic across three Swedish regions — Stockholm, Västra Götaland, and Skåne — using smart card validations, ticket sales, and passenger counts provided by the respective regional transit authorities. The dataset covered the period from February to May 2020, with a comparative baseline from the previous year. Additional mobility indicators from Google and Apple were used to contextualise reductions in transit

usage. Their study provides an example of how smart card data can be integrated with external sources to assess behavioural responses to large-scale disruptions. Focusing on travel regularity and individual routines, Foell et al. [44] explored the regularity of bus usage in Lisbon by analysing data provided by *Carris*, the city's main bus operator. The dataset covered nine weeks of travel (from April to May 2010), including over 2000 stops and more than 100 bus lines. It comprised more than 24 million bus rides taken by 809, 758 users over the observation period. Using smart card data combined with Automatic Vehicle Location (AVL) data, the authors studied inter-trip intervals, daily travel rhythms, and the concentration of usage across lines and stops, revealing highly regular travel patterns for most users.

In summary, the literature provides a rich landscape of approaches to analyse smart card data, ranging from trip reconstruction and user segmentation to stop-level temporal profiling. In this context, we present a novel dataset from the city of Venice — an urban context with a distinctive public transport system based on water buses. Unlike traditional urban networks, Venice's insular structure and reliance on water-based transit present unique mobility dynamics, which remain largely unexplored in existing studies. Our analysis focuses on validations provided by ACTV, and covers three distinct periods of 2023: the Carnival season (January-March), the spring period (April–June), and the Venice Film Festival (August-October). These periods were selected to capture different mobility patterns associated with regular commuting, seasonal tourism, and large-scale cultural events. We provide a detailed temporal and spatial characterisation of public transport usage across these periods. Building on this analysis, we propose a clustering-based method to characterise stop usage patterns across the city. Rather than focusing on trip inference or individual user profiles, our contribution lies in the stop-level profiling of the transport network. By applying two clustering approaches, we aim to provide insights into the structural functioning of the public transport system in Venice, supporting service optimisation and informed planning strategies in a context where conventional urban mobility assumptions often do not hold.

Some very preliminary analyses on the mobility of tourists in Venice have been presented in [2]. The dataset used in that work was limited to two months in autumn 2018, and included only a typology of tickets usually bought by tourists. As a consequence, the analysis focused only on tourists. Furthermore, no information about special events was available.

8.2 Data Description

In this section, we describe the public transportation datasets, along with the data cleaning and preprocessing procedures applied to them. The datasets were

provided by ACTV (*Azienda del Consorzio Trasporti Veneziano*), which is the primary public transport company in Venice. ACTV provides transport services on both land and water. The company has fleets of water and land buses for services in the centre, suburbs and out of town. Operating 151 water buses, ACTV transports approximately 145 million passengers annually on the navigation network. With over 120 floating stations and 27 well-connected lines, ACTV provides essential transportation services throughout Venice. Additionally, the bus network comprises 95 routes, supported by a fleet of 568 buses and 20 trams, catering to approximately 70 million passengers. This vast transportation infrastructure facilitates the movement of tourists, residents, students, workers contributing to Venice's urban life.

8.2.1 Validation Datasets

ACTV provided us three different datasets. The first dataset pertains to the Carnival period and encompasses validations conducted between January 13 and March 14, 2023. The second dataset records validations throughout the spring season, including Easter, spanning from April 4 to June 3 2023. The third dataset covers validations from August 21 to October 20, 2023, which includes the period of the Venice Film Festival. Table 8.1 reports the number of validations for each dataset.

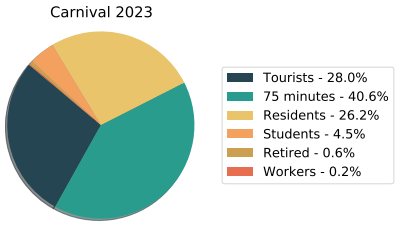
		No. of validations	
Dataset	**Period (2023)**	**Raw**	**Cleaned**
Carnival	January 13–March 14	5,537,467	4,947,477
Spring	April 4–June 3	8,644,448	7,740,907
Venice Film Festival	August 21–October 20	8,197,200	7,903,898

Table 8.1: Number of validations for the three datasets with their respective periods.

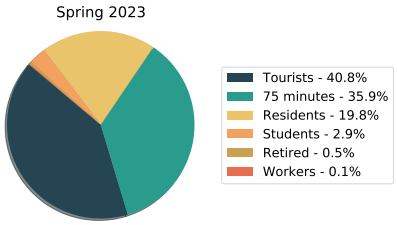
Each validation consists of a ticketing log containing: an anonymised card ID (to preserve user privacy), the ticket type and its description, the timestamp of the transaction (date and time), and details of the stop where the validation took place, including both its identifier and name.

In order to analyse the behaviour of different typologies of users we consider thirteen ticket types: (1) 75 minutes ticket; (2) one day ticket; (3) two days ticket; (4) three days ticket; (5) seven days ticket; (6) monthly ticket; (7) monthly ticket for students; (8) monthly ticket for retirees; (9) monthly ticket for workers; (10)

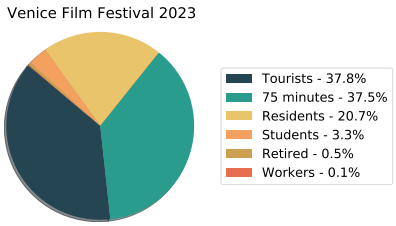
yearly ticket; (11) yearly ticket for students; (12) yearly ticket for retirees; and (13) yearly ticket for workers. Based on the knowledge of ACTV, we grouped these types into 6 categories to understand the variation in experience of daily travel conditions for six different users: *Tourists* collects tickets belonging to the second up to the fifth type (called *time limited tickets*) which are typically bought by tourists; *75 minutes* includes the first type, modelling short trips; *Residents* gathers types 6 and 10, people living in Venice and its suburbs; *Students* assembles types 7 and 11; *Retirees* collects types 8 and 12; and finally, *Workers* joins together types 9 and 13. Note that 75-minute tickets may also be purchased by tourists as well as by residents to travel into the historic city. Nevertheless, such journeys are generally characterized by their occasional and short-term nature. The category Retirees refers specifically to residents aged 75 years and older, with age constituting the defining attribute of this user group.



(a) Winter — Carnival.



(b) Spring — Easter.



(c) Late summer/early autumn — Venice Film Festival.

Figure 8.1: Proportions of the user categories for the three datasets.

Before starting the analyses, a cleaning operation has been performed aimed at removing redundant and erroneous data. Consecutive validations made by the same user at the same stop within a short time interval are removed. We fixed as temporal threshold the median of the duration of the intervals between two consecutive validations for the same user, at the same stop on the same date. In our datasets, the result is 5 minutes. Indeed, in 5 minutes it is not possible to validate the ticket, make a trip, and turn back to the same stop to validate again the ticket. Table 8.1 reports also the dimensions of the datasets after the cleaning process and Figure 8.1 illustrates the proportion of each user category in the three

datasets. Notice that in all graphs, validations for tourists and 75-minute tickets collectively exceed 65% of the total validations. Validations for residents also constitute a significant portion, around or more than 20% of the totality. Conversely, validations attributed to students, workers, and retirees represent approximately 5% of the total validations.

In order to georeference public transport stops, we used the *General Transit Feed Specification* (GTFS) data from *Open Mobility Data*, a valuable repository dedicated to gathering open mobility data worldwide. GTFS provides a series of text files compressed into a single zip file, each of these files describes a particular aspect of transport: stops, lines, routes and further data. In this study, we employed GTFS data containing details about the stops of the Venice public transport system.

The focus of our analyses is the historic centre of Venice. Hence, since our attention is on its navigation network, we do not consider all the single stops of the land transport networks, but we aggregate them by identifying five key areas of interest, as shown in Figure 8.2: (1) *Lido*, that represents the automotive service on the Lido of Venice; (2) *Airport*, which contains the stops associated with activities at Marco Polo Airport; (3) *Piazzale Roma (Bus)*, that includes the bus stops at Piazzale Roma in Venice; (4) *Mestre station*, a crucial reference point for the urban settlement of Mestre; (5) *Mainland*, which covers all other land transportation stops, is geographically represented by the coordinates of Mestre's

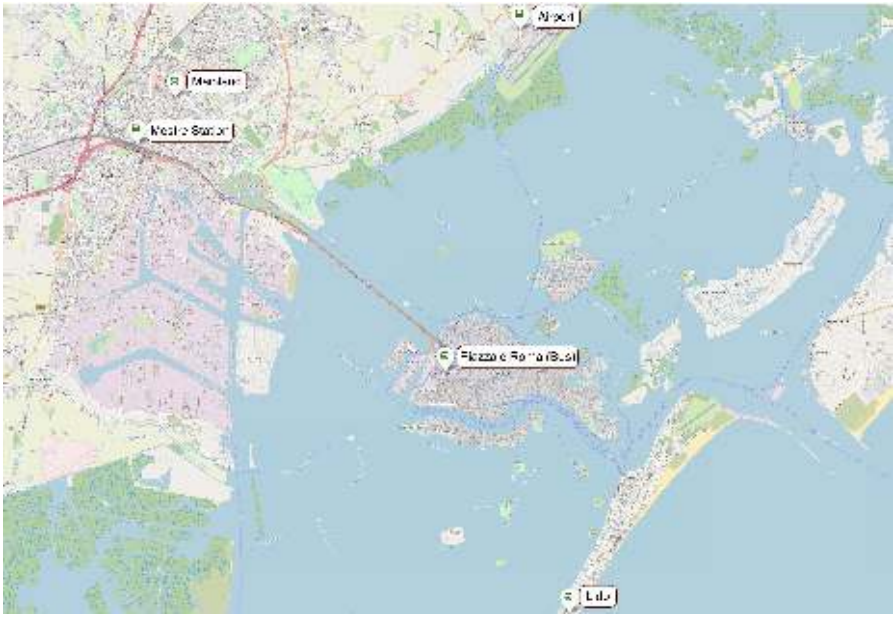


Figure 8.2: Representation of the historic centre of Venice and the five aggregated land transport areas used in the analysis.

city centre. By aggregating tram and bus stops within these five areas, the resulting dataset comprises 160 stops in total, of which 155 correspond to water bus services and 5 to bus services. This grouping allows us to emphasise the unique aspects of the Venice island navigation network. In fact, for the stops relating to the Venice island (water bus stops) no aggregation process was carried out.

8.2.2 Barrier Dataset

In addition to the validation records, ACTV also provided a dataset describing the presence of physical barriers (hereafter referred to as *barriers*) at water bus stops. These barriers consist of automated gates designed to regulate access to the platforms, and they play a crucial role in our analysis of ticket validations, as they enforce validation at the physical access points to the platform. At these stops, passengers are required to validate their tickets in order to pass through the gate and access the water bus. As a result, validation data from these locations are generally more reliable. This dataset specifies, for each stop, whether barriers are installed and, if so, whether they operate at entry only or in both directions (entry and exit). It is worth noting that ticket validation is not mandatory upon exit. A camera detects the presence of a person and automatically opens the barrier. This gate is primarily intended to prevent individuals from entering through the exit to avoid ticket validation, rather than to enforce validation upon exiting. Overall, barriers are installed at 53 stops across the network, out of a total of

Figure 8.3: Water bus stops equipped with barriers installed either at entry only (in red) or in both directions (in black).

155 water bus stops. As shown in Figure 8.3, 23 of these stops are equipped with entry-only barriers (highlighted in red), while 30 barriers operate in both directions (highlighted in black). Many of these barriers have been installed at strategic locations throughout the city, such as the main access points to Venice (e.g., Piazzale Roma and the railway station stops), the gateway stops in the islands (e.g., those serving Lido, Burano, or Murano) and Punta Sabbioni (a coastal locality at the northern edge of the Venetian Lagoon), and also the most frequently used stops in the city centre (including San Marco and Rialto).

Given the reliability of validation data at barrier-equipped stops — where ticket validation is enforced as a prerequisite for platform access — we conducted a preliminary analysis of the distribution of validations across user categories, differentiating between stops with and without barriers. Table 8.2 presents the total number of validations for each user category at both types of stops, along with the corresponding percentages. The distribution of validations between stops with and without barriers varies considerably across user categories. Workers exhibit the highest proportion of validations at barrier-equipped stops (98.83%), indicating that they tend to validate their tickets almost exclusively when required — i.e., when access to the platform is physically controlled. Residents also display a high proportion of validations at such stops (82.57%), indicating that they also tend to validate primarily when access is physically controlled. However, this tendency is less exclusive than in the case of workers, who validate almost only in such conditions. A similar but less marked pattern is observed among students (74.81%), tourists (61.90%), and retirees (61.30%), indicating that these groups also tend to validate mainly when required. By contrast, the validation behaviour of passengers using 75-minute tickets displays a different trend: only 39.71% of validations for this ticket type occur at barrier-equipped stops. This behaviour is likely driven by the nature of the ticket itself: since it is time-limited, passengers

	No. of validations		**Percentage of validations**	
User type	**With barriers**	**Without barriers**	**With barriers**	**Without barriers**
Workers	25,359	301	98.83%	1.17%
Residents	3,687,180	778,149	82.57%	17.43%
Students	533,632	179,691	74.81%	25.19%
Tourists	4,661,672	2,869,052	61.90%	38.10%
Retirees	64,159	40,507	61.30%	38.70%
75 minutes	2,392,539	3,633,207	39.71%	60.29%

Table 8.2: Number and percentage of validations by user category at stops with and without barriers, considering all three data collection periods (winter, spring, and late summer/early autumn 2023).

are required to validate it in order to activate the validity period, regardless of the presence of barriers.

8.3 Analysis at Different Temporal Levels

Figure 8.1 gives us an overview of the mobility in Venice during specific periods of winter, spring and late summer/early autumn. Tourists are dominant in the last two periods with an increase in ticket validations by more than 10% compared to winter. In the winter period, there is the highest usage of 75-minute tickets, due to Carnival which attracts a lot of people of the neighbourhood for the planned events and, probably, because of cold weather which push people to take public transport rather than walking. On the other hand, in spring and summer/autumn periods, people enjoy walking and use public transports less. Similar reasons can explain a slight decrease in ticket validations for residents and students. In particular the lowest usage for students during the spring period is probably related to the fact that during Easter holidays schools and university are closed and many students return to their native towns in order to spend some time with their families. For the retirees and workers categories, there is no relevant difference in the three periods at this level of aggregation.

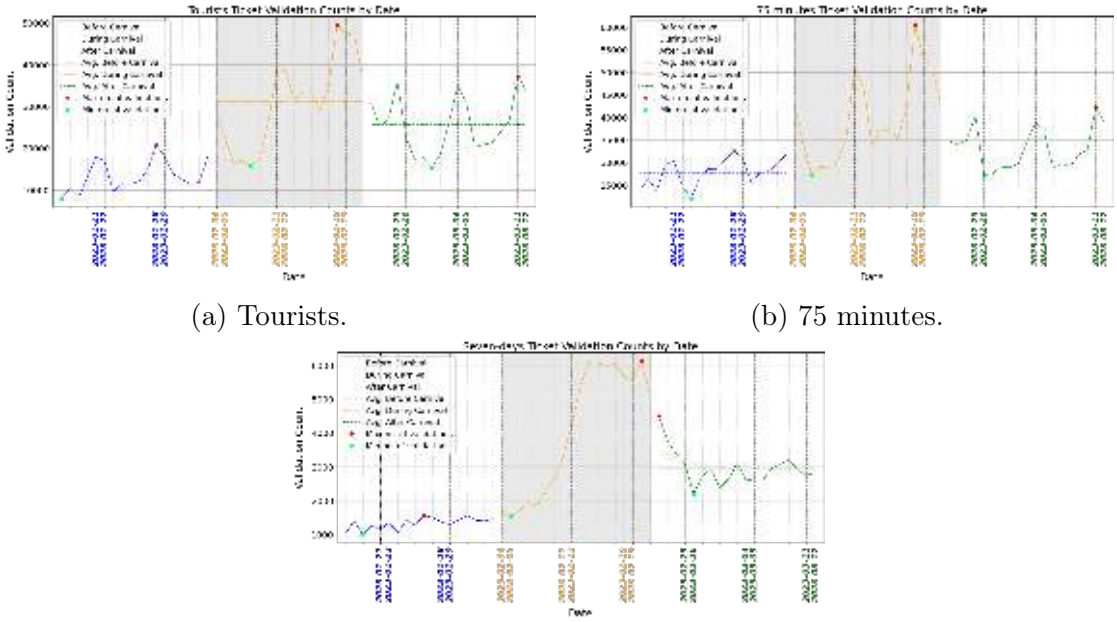
In this section, first we focus on the distribution of the daily validations, and then we point out the distribution of the validations at each hour of a day still with the aim of detecting different behaviours for each user category.

8.3.1 Daily Analysis

We want to explore and compare the travel patterns of the different user categories during three significant events for the city of Venice: Carnival, Easter and Venice Film Festival. This allows us to determine the implications of these iconic events on public transport infrastructure.

Carnival 2023

The Venice Carnival is an internationally renowned cultural event held annually in February and it draws tourists from around the globe and significantly impacts tourism and local transport in Venice. To carry out a focused analysis of ticket validation patterns, the dataset for the period of interest is segmented into three distinct time frames: (1) the *Carnival Period*, spanning from February 4 to February 21, 2023; (2) the *Pre-Carnival Period*, encompassing the 18 days leading up to the Carnival (from January 17 to February 3, 2023); (3) and the *Post-Carnival Period*, comprising the 18 days following the conclusion of the Carnival (from



(a) Tourists.

(b) 75 minutes.

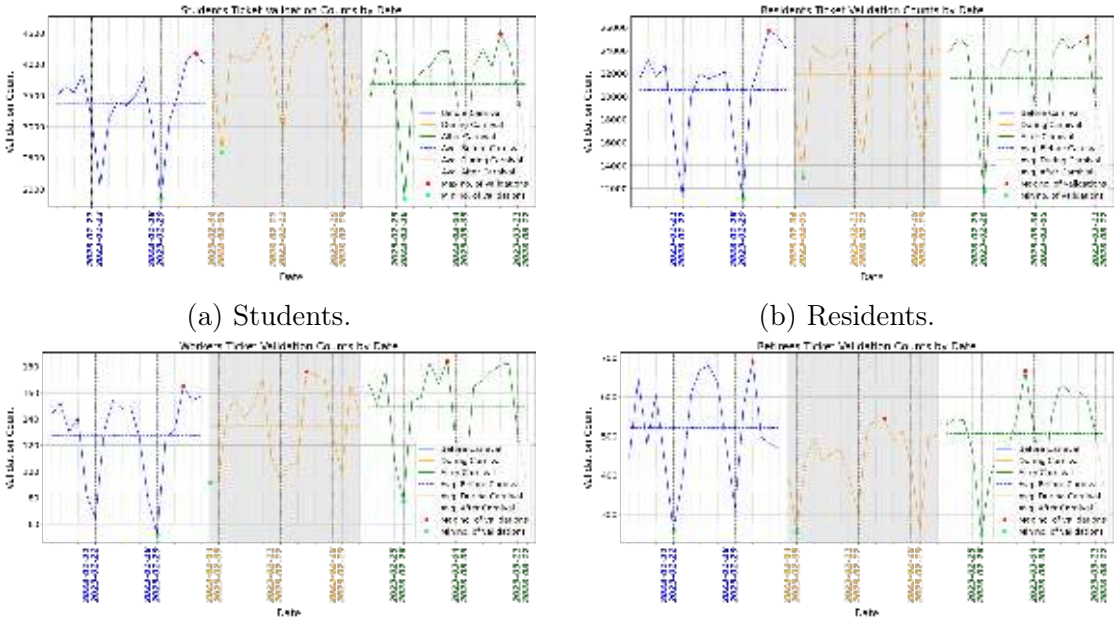
(c) Seven-day ticket.

Figure 8.4: Number of daily validations during Carnival 2023 for Tourists, 75 minutes and for Tourists who purchased 7-day tickets.

February 22 to March 12, 2023). Such a partition enables a detailed analysis of ticket validations before, during and after Carnival.

Figures 8.4 and 8.5 show the trends of the daily validation counts during the these three periods for the six categories of users we identified. Specifically, the plot of daily validation counts during the period *before* Carnival is depicted in blue and the average number of validations for the period is represented by a horizontal dotted line. The plot of the validations *during* Carnival are shown in orange on a gray background, along with their average, whereas the validations for the period *after* Carnival and the corresponding average are displayed in green. Furthermore, the minimum number of validations for each period is indicated by a cyan circle, while the maximum number of validations is highlighted by a red diamond, and the weekends are shown with dotted vertical lines.

Tourists and 75 minutes are the user categories with the highest increase in validations during Carnival (see Figures 8.4a and 8.4b) and they have a similar trend. For the category of Tourists, and thus for the *time limited tickets* associated with this group of users, we observe a significant growth in average validations compared to the previous and subsequent periods, around 135%, with percentage increases between 101.9% and 206.7% according to the types of timed limited



(a) Students. (b) Residents.

(c) Workers. (d) Retirees.

Figure 8.5: Number of validations during Carnival 2023 for Students, Residents, Workers, and Retirees.

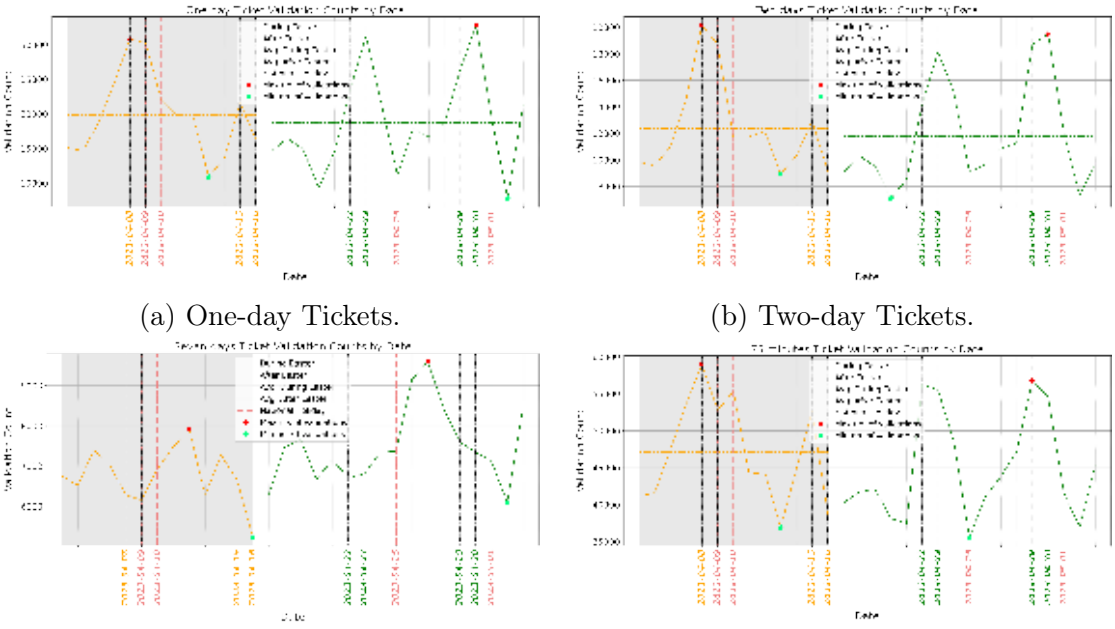
tickets. In particular, as we can see in Figure 8.4c, the tourists who purchased 7-day tickets present a notable boost of 206.7%, highlighting a strong preference for this type of ticket during the event. This phenomenon clearly indicates that tourists choose to stay in Venice for several days to partake in this event. This behaviour was observed for Carnival and is not so marked for the other events during the rest of the year. Even the 75-minute tickets (Figure 8.4b) shows a relevant increase in validations during the Carnival period, about 44%, and the greatest number of validations occurs on weekends. This behaviour suggests that users are mainly visitors interested in specific events located in specific locations, thus they do not need one-day tickets but only single trip ones.

Figure 8.5a shows that students have a limited interest in Carnival: there is an increase of validations of 16% with respect to the previous period and of 6% compared to the following month. Moreover, it is worth noting that students exhibit a notable increase in minimum validations during Carnival, despite weekends typically display low validation numbers. There is a slight increase in validations for residents and workers — approximately 7% and 6%, respectively — compared to the period before Carnival. For workers, this appears to be part of an upward trend, as validations rise further by a more consistent 11% after Carnival. As

observed for students, the Carnival period also shows an increase in weekend validations for residents and workers, even though these days still exhibit the lowest overall peaks (see Figures 8.5b and 8.5c). Retirees, on the other hand, exhibit a different trend (see Figure 8.5d). During Carnival, their average number of validations is noticeably lower compared to both the preceding and following periods, with decreases of 15.3% and 14.9%, respectively. This suggests that retirees reduce their travel during Carnival, likely to avoid crowded public transport.

Easter 2023

The Easter period represents a significant time for tourism and mobility in Venice, with numerous people visiting the city to celebrate the holidays. In this analysis, we examine data relating to ticket validations during the Easter period, spanning from April 4th to April 16th, 2023, including April 9 (day of the Easter celebration), and the 17 days after the Easter period (from April 17 to May 3, 2023). Figures 8.6 and 8.7 show the trends of the daily validation counts during these two periods, Easter and post-Easter, for some of the categories of users we identified, where the plots for the two periods are depicted in orange and green, respectively. Specifically, for Tourists we focus on short-term tickets (1/2-day tickets) and long-



(a) One-day Tickets.

(b) Two-day Tickets.

(c) Seven-day Tickets.

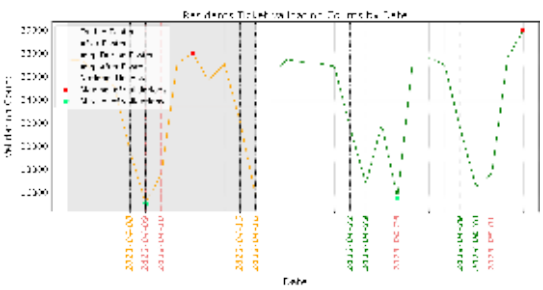
(d) 75 minutes.

Figure 8.6: Number of validations during spring 2023 for one day, two days, seven days, and 75 minutes tickets.

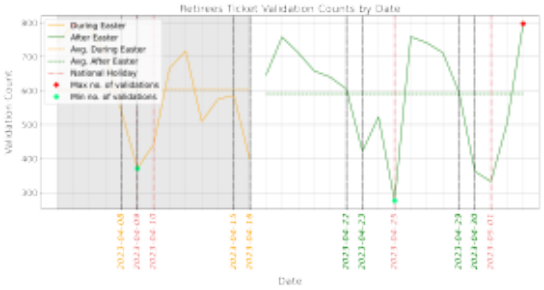
term ones (7-day tickets), because the tourists who bought 7-day tickets exhibits a very different behaviour. The plots, besides the dates of the weekends, highlight also a couple of Italian national holidays, April 25 and May 1 (pink coloured), which typically permits Italian tourists to extend the previous weekend.

As reported in Figure 8.6, short-term tickets, such as one-day, two-day, as well as 75-minute tickets, showed higher validation rates during Easter, with increases of 5.67%, 4.76% and 4.72%, respectively. Instead, seven-day tickets experienced a decrease of 10.74%. These findings highlight a preference for short-term travel during Easter. In particular, the analysis reveals significant spikes in validations for short-term tickets over the Easter weekend. In contrast, Figure 8.7 shows that the groups of residents and students increased validations after the Easter period, with respective rises of 2.11% and 3.86%. This pattern aligns with the temporary suspension of academic activities during the Easter break, which typically reduces student presence in the city. In particular, many university students return to their native towns during the holidays. Similarly, many residents leave the city for holidays during this period, contributing to the decline in validations observed at that time. Retirees, by contrast, experience a slight decrease in validations after Easter (−1.94%), suggesting more stable mobility habits throughout the period.

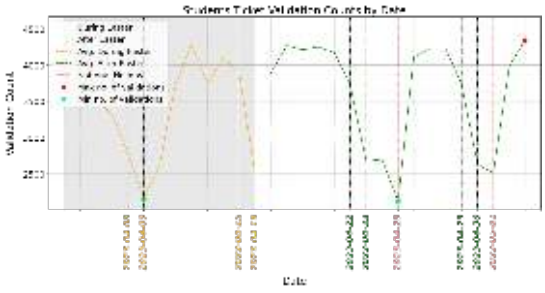
On Italian holidays, such as April 25 and May 1, almost all the categories of



(a) Residents.



(b) Retirees.



(c) Students.

Figure 8.7: Number of daily validations during spring 2023 for Residents, Retirees and Students.

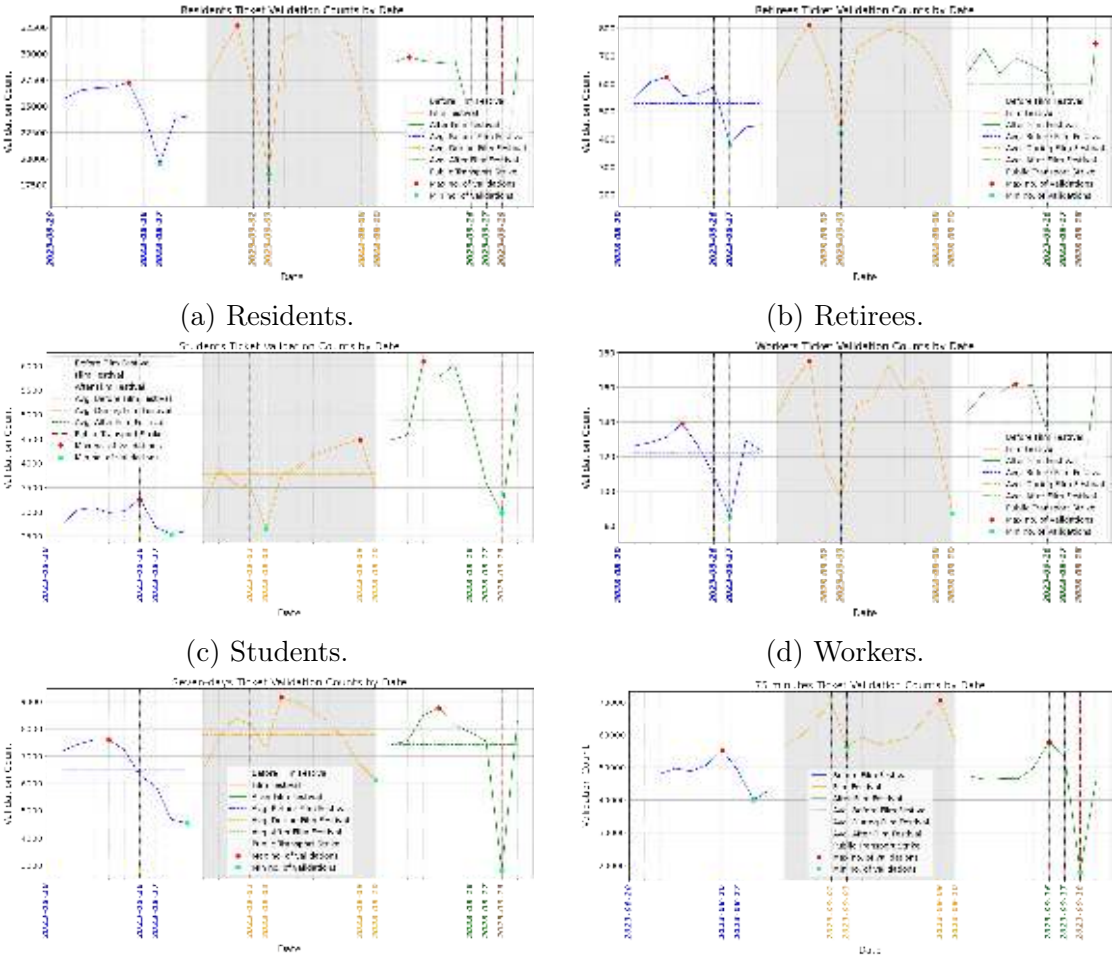
tickets show a significant decrease in usage compared to the average number of validations recorded during the period following Easter, specifically between April 17 and May 3, 2023. These usage patterns can be attributed to the nature of these holidays, which typically do not draw as many international tourists, resulting in a general decrease in demand for public transport services. Notably, seven-days tickets, as already remarked, are an exception, exhibiting an increase of 10.74% with respect to the Easter period and in particular the peak is reached between April 25 and May 1, suggesting that Italian tourists choose this time for a longer stay in Venice.

Venice Film Festival 2023

The Venice Film Festival is an annual international film festival held on the island of Lido, Venice. Established in 1932, it is the world's oldest film festival and one of the *Big Five* International film festivals worldwide. The event attracts thousands of visitors, professionals, and international guests, significantly impacting mobility across the Venetian lagoon. The large influx of visitors, journalists, and industry professionals during the event generates a substantial increase in passenger demand, requiring careful planning of public transport services across the lagoon. Indeed, during the festival period, ACTV enhances its public transport services to facilitate access to the Lido and the main screening venues. In particular, a dedicated line MC (*Mostra del Cinema*) operates every 20 minutes from *Piazzale Roma*, *Zattere*, and *San Marco–San Zaccaria* to *Lido Casinò–Mostra del Cinema* (the stop serving the Film Festival) from late afternoon until after midnight. Additionally, Line 20, connecting *San Marco–San Zaccaria* and *San Servolo*, is extended to the Lido for the entire day.

The increased passenger demand and modified service schedules make the Film Festival period especially suitable for studying mobility and ticket validation behaviour. To carry out a focused analysis of ticket validation patterns, the dataset for the period of interest is segmented into three distinct time frames: (1) the *Film Festival*, spanning from August 30 to September 10, 2023; (2) the *Pre-Festival Period*, encompassing the 9 days leading up to the Festival (from August 21 to August 29, 2023); (3) and the *Post-Festival Period*, including the 9 days following the conclusion of the Festival (from September 11 to September 19, 2023). Figure 8.8 shows the trends of the daily validation counts during the these three periods, represented by blue, yellow, and green plots, respectively.

Figure 8.8) unveils that the Pre-Festival period is characterised by a decline in validations for all categories of people, more noticeable for those living, studying and working in Venice. This is due to the period considered, which in Italy is typically a holiday one, during which schools, universities and several working places are closed. During the Film Festival period, retirees show a strong appeal:



(a) Residents.

(b) Retirees.

(c) Students.

(d) Workers.

(e) Seven-Day tickets.

(f) 75 minutes tickets.

Figure 8.8: Number of daily validations during the Film Festival Cinema for Residents, Retirees, Students, Workers, Seven-Days tickets and 75 minutes tickets.

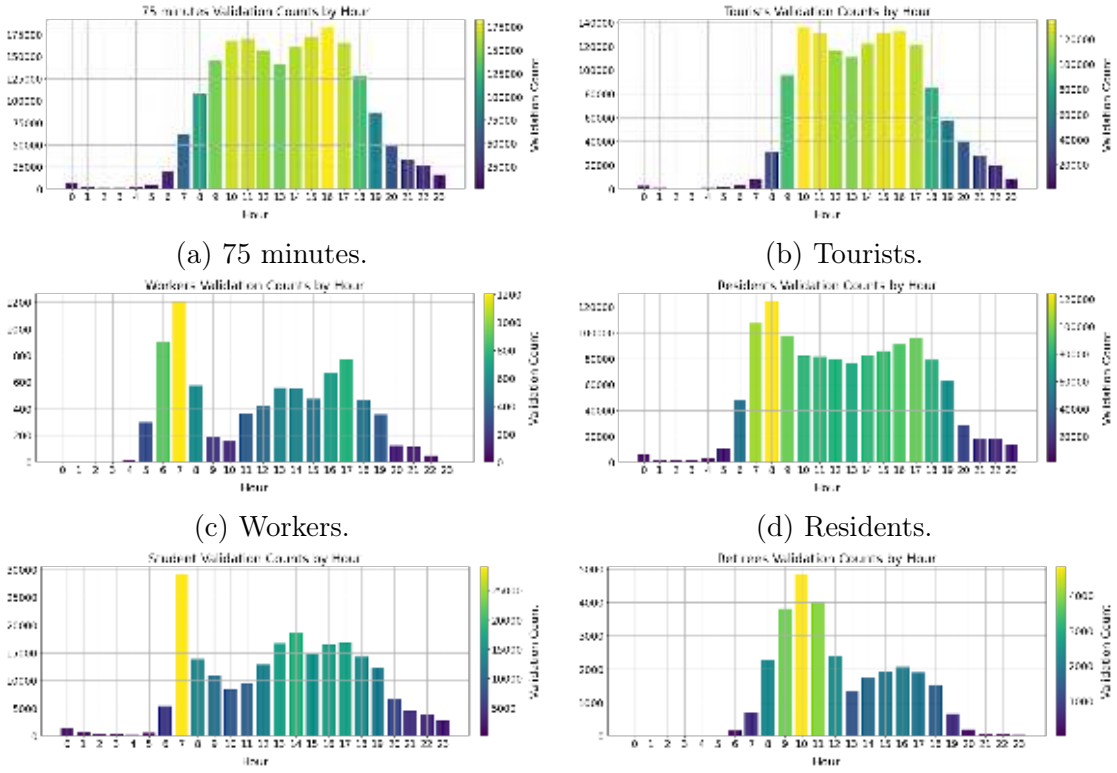
there is an increase in validations of 29.86% with respect to the previous period and of 13.17% with respect to the following period. Among the users in the category Tourists, the seven-day tickets exhibit the highest increase 20% compared to the pre-Festival period, suggesting that such people are interested in enjoying a longer stay in Venice (Figure 8.8e). However, there is also a relevant increase for the 75 minutes tickets: +27.18% compared to the previous period, and +25.34% compared to the post-Festival period. This reveals another trend: commuting, that is, stay in Mestre or hinterland, and go to Lido to see films and attend events (Figure 8.8f). An anomaly in validations occurs on September 3 2023: it is Sunday but it is a minimum for almost all categories. This is strange in particular for

tourists and users of 75-minute tickets, who usually exhibit high rate on weekends. The reason is the event *Regata Storica* that is the main event in the annual *Voga alla Veneta* rowing calendar. There are different races for various types of boats preceded by a spectacular historical water pageant in the Grand Canal. The event takes place in the afternoon and this prevents the ordinary transit of boats through this canal that is one of the busiest ways in Venice. On this day, the navigation service through the Grand Canal is suspended from approximately 3 p.m. to 7 p.m., and alternative routes are activated to ensure connections between *Piazzale Roma* and *San Marco*.

Finally, it is worth to point out the increasing trend for students: +30.66% compared to the pre-Festival period and +29.41% compared to the period during the Film Festival, as shown in Figure 8.8c. At the beginning of September students have exams at university and the peak is reached on September 13 when primary, middle and high schools started. This figure shows clearly that the student population using public transport attends mainly high school.

8.3.2 Hourly Analysis

In order to understand the behaviour of the different user categories, it is also important to analyse how the number of validations varies over the course of the day. To this end, we group the validations by user category and hour of the day. Figure 8.9 shows the histograms related to the Carnival period for our six user categories. The figure clearly shows that each user category exhibits a distinct pattern of validations throughout the day. In particular, 75-minute ticket users (Figure 8.9a) move around the city of Venice mainly from 10 a.m. to 12 p.m. and from 2 p.m. to 5 p.m., with a peak at 4 p.m., with little to no activity in the early morning, or at night. Tourists (Figure 8.9b) exhibit similar behaviour to 75-minute ticket users, travelling mostly in the central hours of the day from 9 a.m. to 5 p.m., with a peak at 10 a.m., and there are also many validations at 4 p.m. Workers (Figure 8.9c) start commuting very early, beginning at 5 a.m. and peaking at 7 a.m. They also travel around lunchtime (1–2 p.m.) and at the end of the workday (around 5 p.m.), with minimal activity from 11 p.m. to 3–4 a.m. Residents (Figure 8.9d) show different patterns, moving slightly later than workers with a peak at 8 a.m., and continuing to use public transport throughout the day without a distinct peak period. Students (Figure 8.9e) exhibit some movement during the night hours, but the majority of travel occurs between 7–8 a.m. as they head to school or university. The number of validations rises again around 2 p.m. when school ends and most students return home. Additionally, there is a significant number of validations around 5 p.m., which then gradually decreases throughout the evening. Finally, Retirees (Figure 8.9f) move differently compared to other categories. Similar to Workers but even more accentuated, validations



(a) 75 minutes.

(b) Tourists.

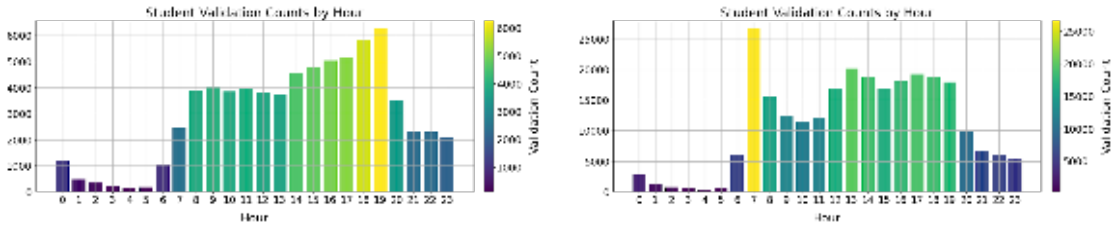
(c) Workers.

(d) Residents.

(e) Students.

(f) Retirees.

Figure 8.9: Number of validations per time slot during the Carnival period.



(a) Holidays.

(b) School period.

Figure 8.10: Number of validations per time slot during the Venice Film Festival period, during school holidays (from 21 August to 10 September) and the school period (from 11 September to 20 October) for the student category.

at night (until 6 a.m.) are very few. These users mainly move during the central hours of the day, particularly from 9 a.m. to 11 a.m. with a peak at 10 a.m. There are also some validations in the afternoon, about half the number of those in the morning, and they are more evenly distributed throughout the afternoon time slots.

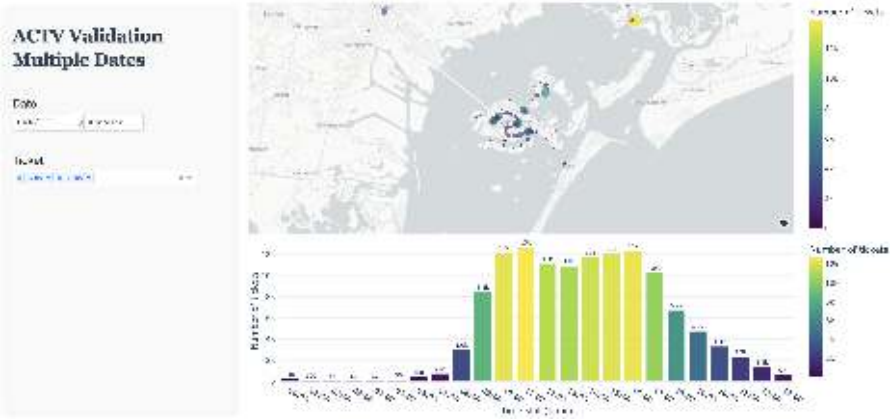


Figure 8.11: Validations of the users in a single day with tickets of 24 and 48 hours. In the map it is possible to see how the validations are distributed in space in all the ACTV stops, while the bar chart shows the distribution over time (for each time of day).

The behaviour of the various user categories does not vary significantly across the three periods analysed. Only students exhibit a relevant difference in the use of public transportation during summer holidays compared to the school period. Figure 8.10 shows the holiday period from August 21 to September 10 (on the left) and the back-to-school season from September 11 to October 20 (on the right). Despite some slight variations in the number of validations, the school period in September (Figure 8.10b) is similar to the behaviour during the Carnival period (Figure 8.9e): there is always a peak at 7 a.m. and a high number of validations at 1–2 p.m. and again at 5-6 p.m. The summer period presents a completely different pattern. Figure 8.10a reveals that students are more active at night, waking up much later in the morning and beginning their activities around 2 p.m., with a steady increase in validations until a peak at 7 p.m.

Also we developed a tool to visualise validations in both space and time. The city manager can choose a specific date range within a broader event period (such as Carnival, Easter, or the Film Festival), along with one or more user categories. Then, the tool generates an image consisting of two parts, as depicted in Figure 8.11. The upper portion displays the spatial distribution of validations, where the colour and size of the circles represent the number of validations proportionally. In the bottom part, a histogram shows the temporal distribution of the validations along the different hours of the day. In addition, the prototype offers a functionality that enables exploration of users' validations in both space and time by producing animations that display heatmaps at three-hour intervals.

8.4 Trajectories Analysis

In the previous analyses we consider validations separately and we group them by user category. In this section, instead, we aim to study user trajectories, i.e., the *sequence* of validations associated with a user. Notably, for privacy reasons, the serial numbers of subscription-type tickets (monthly or yearly) are anonymised and change every day, in order to prevent identification of users based on their daily routines. Instead, for tourists, who are occasional visitors for a few days, and for whom no personal information is stored, the serial number is maintained unchanged for all the days they use the time limited tickets. Due to these constraints, tourist trajectories are generally longer than those of other user categories and can extend over multiple days (see Table 8.3 and Table 8.4). Moreover, all these trajectories are sparse since users with both time limited tickets or monthly/yearly tickets are not used to validate their tickets. As a consequence, the tickets are stamped mainly when there are barriers closing the entrance of the platforms for water buses. Figure 8.3 shows the positions of these barriers. Finally, notice that the ticket validation is at the entrance of the bus or of the platform for water buses, but there is no validation at the exit. Therefore, it is difficult to reconstruct the complete trip followed by users.

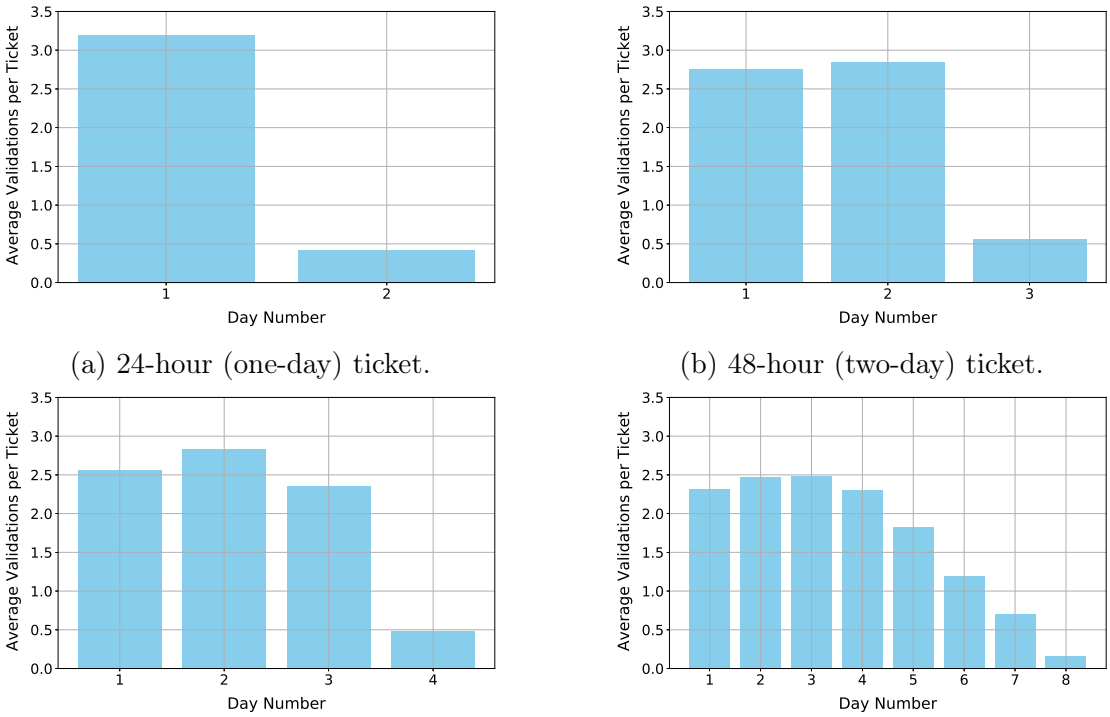
8.4.1 Tourist Trajectories

An insightful analysis regarding tourists involves examining their travel patterns during their stay in Venice to understand their utilisation of public transportation across different days. Specifically, tickets are categorised by their validity periods (one-day, two-day, three-day, and seven-day), and we computed the average number of validations for each day.

Figure 8.12 depicts, for each ticket category, the average daily validations throughout the ticket's validity period. It is noteworthy that each ticket type

		Validations	
Ticket type	**No. of Tourists**	**Average**	**Maximum**
24-hour (one-day)	752,945	3.66	25
48-hour (two-day)	269,882	6.21	51
72-hour (three-day)	250,121	8.22	45
168-hour (seven-day)	78,556	13.27	98

Table 8.3: Number of users, average number of validations, and maximum number of validations for tourist ticket types.



(a) 24-hour (one-day) ticket.

(b) 48-hour (two-day) ticket.

(c) 72-hour (three-day) ticket.

(d) 168-hour (seven-day) ticket.

Figure 8.12: Average number of daily tourist validations based on 24-hour (one-day), 48-hour (two-day), 72-hour (three-day), or 168-hour (seven-day) ticket during the Venice Film Festival.

considers the subsequent day, as tickets remain valid until the same time on the following day (e.g., a 24-hour ticket stamped at 5 p.m. remains valid until 5 p.m. the next day). From these plots, we observe that when tourists stay for multiple days there is an increasing usage of their time-limited ticket in the first days and then there is a decline.

To deep into the behaviour of tourists, we implemented also a prototype which allows for the visualisation of the locations (i.e., the stops) of the validations along the different days (first day, second day, etc.) of their stay. The city manager can select one of the periods of analysis and one of the time limited tickets. In the upper part, the tool builds an animation showing the heatmaps of the validations for each day of the validity of the ticket. As already observed, since the tickets last 24/48/72/168 hours, the period can spread over a further day. At the bottom, on the left part, a histogram illustrates the number of validations for each day of the validity of the ticket, whereas, on the right part, the city manager can select a stop, and a pie chart shows the percentage of validations for that stop in the different

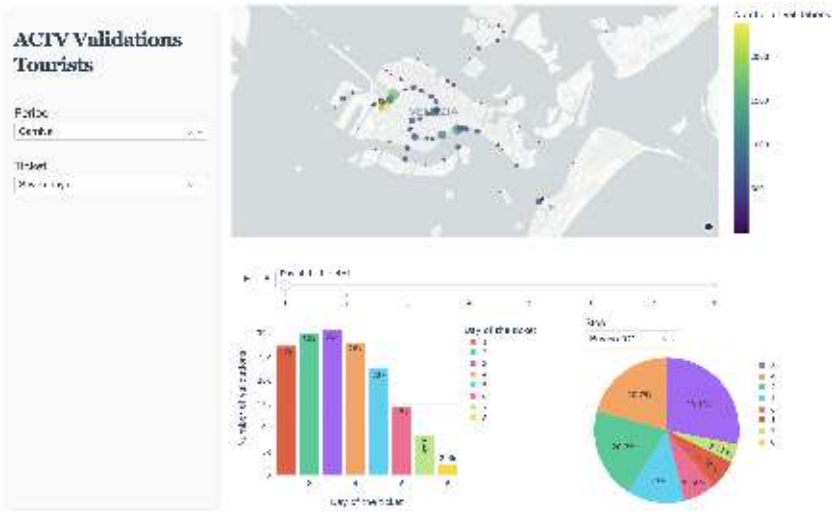


Figure 8.13: Movements for seven-days tickets.

days. For instance, Figure 8.13 highlights the use of the seven-days ticket. Tourists exploit the ticket mostly in the first days, reaching a peak on the third day, and from the fifth day there is a rapid decrease. An explanation for this phenomenon could be that tourists stay less than one week, or they go visiting other places, like Padua, or they prefer walking the last days. As far as the selected stop is concerned, *Burano C* in our example, tourists prefer visiting Burano in the middle of their stay in Venice, not the first day neither the last days.

8.4.2 Trajectory Analysis using Validation Pairs

We recall that, for privacy reasons, the ticket's serial number for workers, students, residents, and pensioners, who hold season tickets for public transport, changes daily. Consequently, for these user categories we can only construct trajectories on a daily basis. As shown in Table 8.4, the average number of validations for each category is 1.5. Hence, only two validations per user are typically available each day. This is why for these categories our analysis concentrates on pairs of consecutive validations. Specifically, for each category, we conducted two types of analysis: (i) we compute the time elapsed between pairs of validations; and (ii) we focus on the longest stay. This allows us not only to detect the time intervals when users start and end their trip but also to find out the locations of greatest interest for each category. In particular, the first validation gives us hints when a certain user category starts its activity, while the second validation reveals details about the time when the activity ends and also on the location where the activity could

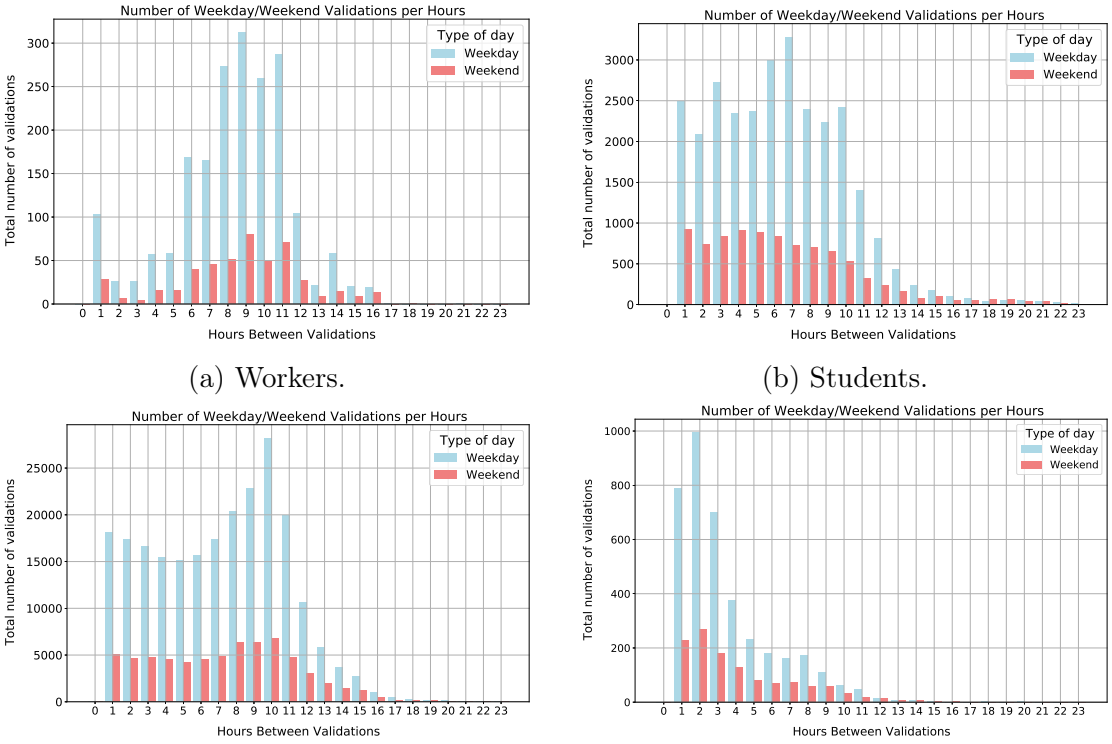
		Validations per day	
Ticket type	**Average no. of daily users**	**Average**	**Maximum**
Workers	85.61	1.64	10
Students	2,651.62	1.47	14
Residents	15,664.87	1.56	37
Retirees	377.24	1.52	13

Table 8.4: Average number of daily users, average and maximum number of daily validations for workers, students, residents, and retirees.

take place. In fact, one of the greatest challenges related to these datasets is that the user validates entering the means of transport but not at the exit. Therefore, we know the starting point of the trip but not the arrival stop.

Let us start computing the *dwell time* between pairs of validations, which is the time elapsed between two consecutive validations. Figure 8.14 shows the number of pairs having a certain dwell time for the spring period. We distinguish between pairs in weekdays, shown in blue, and those on weekends shown in red, across the categories of workers, students, residents, and retirees. As shown in Figure 8.14a, workers exhibit a peak dwell time of 9 hours, with secondary peaks at 11 and 8 hours. This pattern reflects typical worker behaviour, where individuals leave in the morning for an 8–9 hour workday and then take public transport to return home. There are relatively few validations for shorter intervals. In contrast, Figure 8.14b shows a different pattern for students. The peak of dwell time is 7 hours, which aligns with the typical school or university schedule. Notably, there is also a significant 3-hour dwell time, which could be attributed to university classes, afternoon sports, or leisure activities. For residents, as illustrated in Figure 8.14c, the peak is reached at a 10-hour interval, likely reflecting the working schedules of these individuals. Finally, Figure 8.14d depicts a distinct behaviour for retired people. They typically go out for no more than 1–2 hours, and therefore they stay away from home for much shorter intervals compared to other categories.

As the results for the spring period closely align with those of the other periods, they are not reported here. The following analyses focus on the identified peaks in dwell time. Specifically, we examine the hours at which the first and second validations occur within the pairs associated with the peak dwell times. We distinguish between weekdays and weekend days, and to ensure a fair comparison, the validations are normalised by the number of weekdays and weekend days occurring in the analysis period. Furthermore, we investigate the most frequently used stops linked to these validations.



(a) Workers.

(b) Students.

(c) Residents.

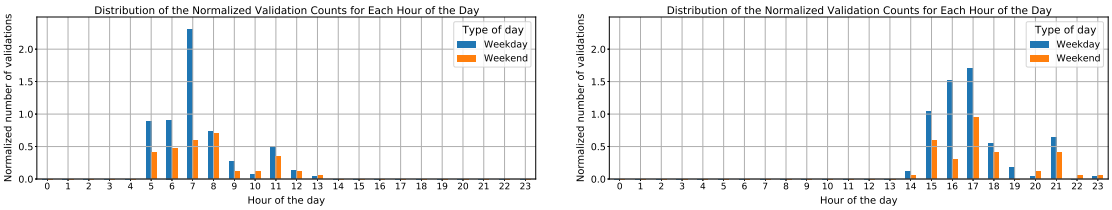
(d) Retirees.

Figure 8.14: Number of couples with their dwell time for the spring period. Weekdays are in light blue while weekends are in red, across the categories of workers, students, residents, and retirees.

Workers

Figure 8.15 depicts pairs of validations with a time interval of 9 hours punched by workers during the spring period. Specifically, in Figure 8.15a, we observe that the initial validations of these pairs occur between 5 a.m. and 1 p.m., with the highest number of validations between 5 a.m. and 8 a.m., peaking at 7 a.m. There is also a noticeable difference in the number of validations between weekends and weekdays, indicating that workers travel more during weekdays and prefer to stay at home on weekends. Regarding the second (and last) validation of the day, Figure 8.15b shows peaks again during hours when people typically finish work, with the highest number of validations at 4 p.m. and 5 p.m. Naturally, the number of validations is higher during weekdays in this case as well.

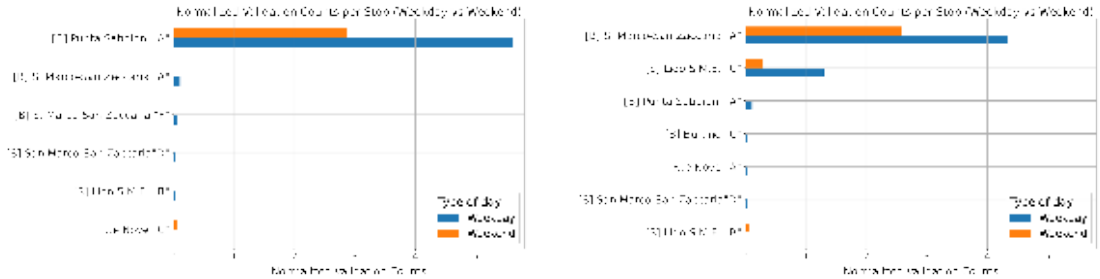
Another analysis we can conduct on this data is to understand which stops workers depart from in the morning and which ones they use to return home in the evening. From the first stop, we can deduce where workers live, and from



(a) First daily validations. (b) Last daily validations.

Figure 8.15: Normalised number of validations during the hours of the day grouped by weekdays (in blue) and weekends (in orange). These refer to pairs of one-day validations by workers during the spring period within a 9-hour interval.

the second, where they work. In Figure 8.16, we can observe the seven most frequently used stops by workers and the normalised number of validations across weekdays and weekends. Figure 8.16a clearly shows that the majority of workers depart from *Punta Sabbioni*. Punta Sabbioni is located at the southern end of the



(a) Stops related to the first validation. (b) Stops related to the last validation.

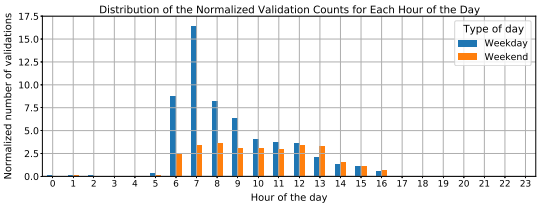
Figure 8.16: Normalised number of validations for the 8 most used stops by workers during the spring period, grouped by weekdays (in blue) and weekends (in orange). The stops reported are those used by workers within a 9-hour interval.

Cavallino–Treporti peninsula and constitutes a strategic mobility hub connecting the Venetian Lagoon with the Adriatic coast. It functions as a primary gateway for commuters and students residing in Cavallino, Jesolo, and surrounding municipalities, providing regular ferry services to Venice and Lido. Punta Sabbioni is also home to a community that has settled there after moving from the island of Burano. A smaller number begin their journey from *San Marco*, indicating they live on the main island, while a minority depart from *Lido*. Lido is a barrier island situated between the Venetian Lagoon and the Adriatic Sea, functioning both as a residential area and as a major destination for tourism and leisure. On weekends, only *Punta Sabbioni* and *F.te Nove C* serve as a common departure point for workers. The first one provides access to Venice whereas the second one is

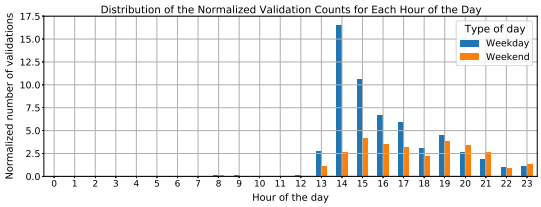
a key waterfront area on the northern edge of Venice, serving as a major transportation hub that connects the city with the lagoon islands, including Murano, Burano, Torcello and Lido and also with the railway station, facilitating travel out of Venice. Figure 8.16b highlights the locations where workers begin their journey home, reflecting their workplaces. *San Marco* is the most frequented stop, followed by *Lido*, whereas *Punta Sabbioni* and *Burano* are associated with a smaller number of workers. San Marco allows one to reach various commercial activities and administrative offices. In Lido, there is a consistent number of healthcare personnel from the nursing and hospital facilities located in this island, and many of them commute to Lido. On weekends, the most frequent destinations are San Marco and Lido. The Carnival period and the Film Festival period for workers show a similar distribution of validations and also the same stops used.

Students

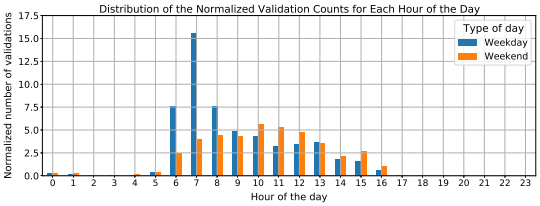
The same type of analysis is performed for students, focusing on pairs of validations separated by 7 hours, which is the peak of dwell time for this user category. As shown in Figure 8.17, validations appear more widely spread across different times of the day. In fact, the first validations of the day start from midnight until 4 p.m. Likewise, the second validations of the day start as early as 7 a.m. and end



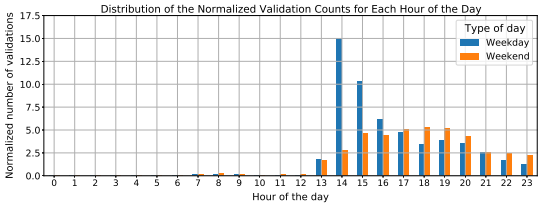
(a) Stops related to the first validation during the Carnival period.



(b) Stops related to the last validation during the Carnival period.



(c) Stops related to the first validation during the spring period.

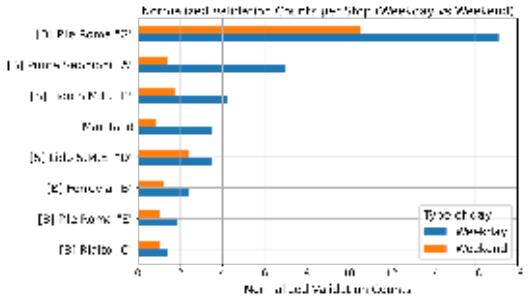


(d) Stops related to the last validation during the spring period.

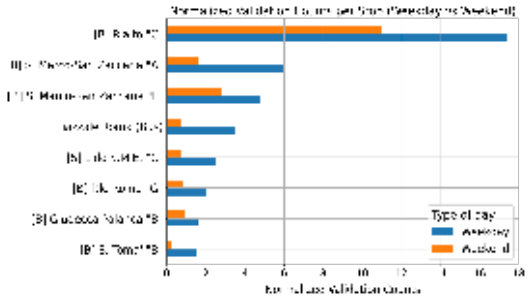
Figure 8.17: Normalised number of validations during the hours of the day grouped by weekdays (in blue) and weekends (in orange). These represent pairs of student validations occurring within a 7-hour time window during the Carnival and spring periods.

at 11 p.m. This clearly illustrates that students exhibit different mobility patterns compared to workers, who typically do not travel at night. In particular, Figures 8.17a and 8.17c show that during the weekdays students leave in the morning to attend school or university, with a pronounced peak at 7 a.m. The second validation (Figure 8.17b and 8.17d) also reflects the academic schedule, with a high concentration of validations occurring at the end of school hours (2–3 p.m.) or university hours (around 4–5 p.m.). It is important to note the difference in how students move during the weekends in the Carnival period compared to the spring period. In the spring period, we can observe that, except for the time slot used by students to go to school (from 6 a.m. to 8 a.m. in Figure 8.17c) and to return from school (from 2 p.m. to 3 p.m. in Figure 8.17d), the number of validations is higher during the weekend in all other time slots. This clearly indicates that with the arrival of warmer, sunnier, and milder days, students are more likely to go out during the weekend compared to the winter period, when they tend to stay at home. In addition, in both periods, students go out at night mainly on weekends.

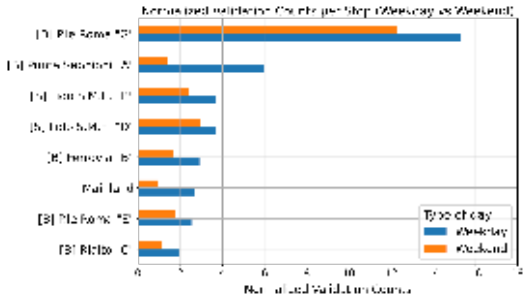
Figure 8.18 illustrates the eight most frequently used stops by students for the Carnival and spring periods. Figures 8.18a and 8.18c depict the stops used for the first validation during the day. The most frequently used stop is *Piazzale Roma*, which plays a strategic role due to its access to multiple water bus lines. Also students arrive from *Punta Sabbioni*, *Lido*, and Mestre (*Mainland*), as well as from more distant areas connected by the train network (*Ferrovia*). Finally, some students start their trip in the centre of Venice, i.e., from *Rialto*. The majority of students enter Venice and only a limited number goes studying in the mainland. Figures 8.18b and 8.18d illustrate the locations of the second validation occurring after a 7-hour interval. *Rialto "C"* is the stop predominantly used by students returning home, likely due to its central location within Venice, as remarked previously. Students travelling from Lido and Punta Sabbioni commonly use the *S. Marco–San Zaccaria* stop to reach high schools in the Celestia area, which can be accessed on foot, thereby avoiding congested water buses. *Piazzale Roma (bus)* is noteworthy, as it represents bus validations directed to the mainland and serves as a complementary counterpart to the *Mainland* stop recorded during the first validation. Moreover, there are two stops which are typical for students: *Giudecca Palanca* (in the Carnival period) and *Zattere* (in the spring period). In Giudecca there are middle schools and some university places, while at Zattere there is the largest library for Ca' Foscari University. In spring there is an increase in validations at Lido stop on weekends. Lido is a popular destination for beach trips, and as a result, often serves as a common afternoon departure point after a day spent by the sea. The same trend is observed during the Film Festival period, which spans the end of August and early September. Of particular note is the rise in validations at the *Zattere* stop in this time, likely associated with the academic cal-



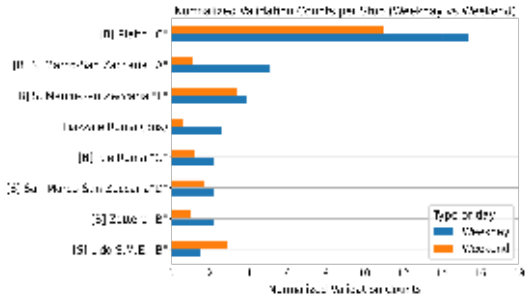
(a) Stops related to the first validation during Carnival.



(b) Stops related to the last validation during Carnival.



(c) Stops related to the first validation during spring.

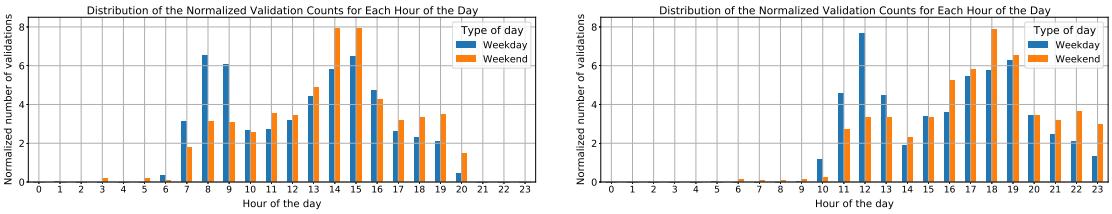


(d) Stops related to the last validation during spring.

Figure 8.18: Normalised number of validations for the 8 most used stops by students during the Carnival and spring periods, grouped by weekdays (in blue) and weekends (in orange). The stops reported are those used by students within a 7-hour interval.

endar, as this period overlaps with university exam sessions and increased library attendance by students.

Since Figure 8.14b shows a significant number of validations by students occurring at 3-hour intervals, which do not align with school hours, we also analyse the validations that occur during this time period. From Figure 8.19, it is clear that validations occurring at 3-hour intervals are predominantly on weekends. In particular, students tend to go out around 2–3 p.m. and return around 6–7 p.m., with a smaller number going out later in the evening, though this is still predominantly on weekends. During the week, there is a peak in validations around 8–9 a.m. and again around 2–3 p.m. for initial validations (Figure 8.19a), which corresponds to a peak in return times around 12 p.m. and from 5–7 p.m. (Figure 8.19b). This pattern aligns perfectly with the schedule of university students, who have more spread-out classes throughout the day, with lectures starting later in the morning and continuing into the afternoon. Figure 8.20 shows the stops used by

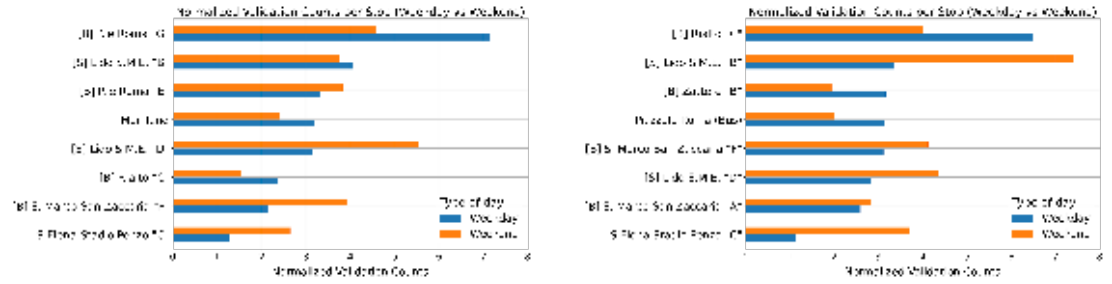


(a) Stops related to the first validation.

(b) Stops related to the last validation.

Figure 8.19: Normalised number of validations during the hours of the day grouped by weekdays (in blue) and weekends (in orange). These represent pairs of student validations occurring within a 3-hour time window during the spring period.

students during the spring period for 3-hour interval validations, which display patterns similar to those observed in the other two periods. First, we can observe differences in stops compared to Figure 8.18: *Punta Sabbioni* and *Ferrovia* do not appear as stops for short validation periods, as they serve individuals who live far from Venice, and a 3-hour window is too short for a feasible round-trip commute. *S. Elena* appears as both the first and second validation in this case, but is absent in the longer validation interval. This stop provides access to Lido and is also frequently used by supporters of the Venice football team, given the proximity of the stadium. On weekends, it typically sees increased validations due to football matches. *S. Marco-San Zaccaria "F"* is now also a stop for first validations and can be used to go to Lido. Furthermore, the frequent use of *Lido* stops for both initial and subsequent validations reflects the significant level of passenger traffic involving the island. Finally, we observe that all stops are predominantly used on weekends, consistent with the earlier observation that 3-hour trips are more common during this time of the week.



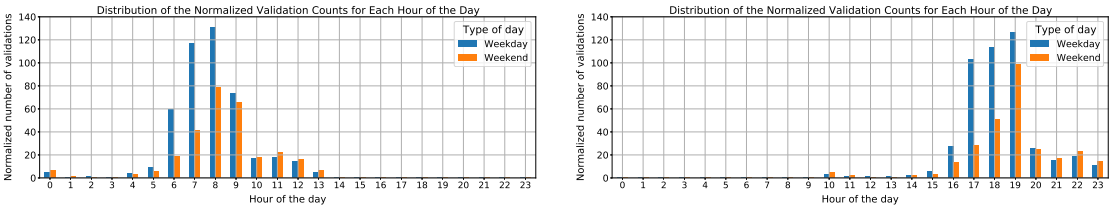
(a) Stops related to the first validation.

(b) Stops related to the last validation.

Figure 8.20: Normalised number of validations for the 8 most used stops by students during the spring period, grouped by weekdays (in blue) and weekends (in orange). The stops reported are those used by students within a 3-hour interval.

Residents

For residents, the peak of dwell time is 10 hours, as shown in Figure 8.14, across all periods. Therefore, we analysed pairs of validations with a 10-hour interval for the spring period. Figure 8.21a illustrates that on weekdays, residents begin departing from home as early as 4 a.m. A marked increase is observed at 6 a.m., followed by a further rise at 7 a.m., culminating in a peak at 8 a.m. Subsequently, there is a decrease in validations at 9 a.m., with a low number of validations between 10 a.m. and 1 p.m. These users start returning home around 5 p.m., with validations gradually increasing until 7 p.m. (Figure 8.21b). On weekends, they tend to leave much later in the morning and consequently return later in the evening, with nighttime activities being more common during this period. From this distribution of validations, it can be inferred that most residents are likely workers who leave early in the morning, travel to work, stay at their workplace for 8–9 hours, and then return home.

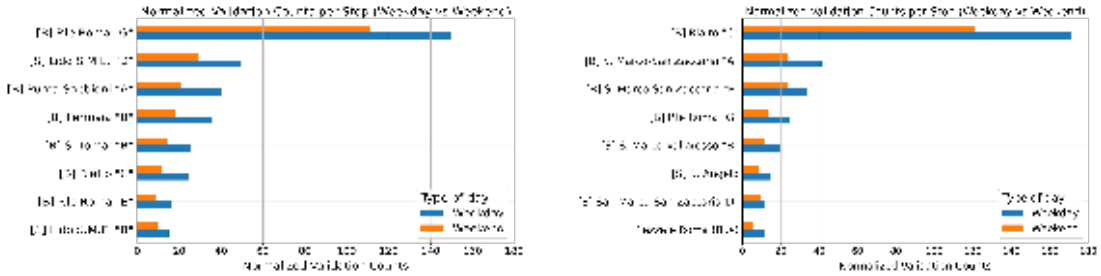


(a) First daily validations.

(b) Last daily validations.

Figure 8.21: Normalised number of validations (10-hour time interval) during the hours of the day grouped by weekdays (in blue) and weekends (in orange). These refer to pairs of one-day validations by residents during the spring period.

Figure 8.22a displays the stops used by this category of users, for validation pairs with a 10-hour interval, in the morning. *Piazzale Roma* and *Ferrovia* are strategic stops as they are connected to the mainland, while *Lido* and *Punta Sabbioni* are areas outside the city of Venice where it is necessary to take a water bus to reach the centre of Venice. *S. Tomà* is located not far from Piazzale Roma and it is used to avoid the crowded water buses leaving from Piazzale Roma, and it is used to reach Rialto. Conversely, *Rialto* is located within the city of Venice, at its very centre, and it is the stop used by people going working outside Venice. The main stops used to return home (Figure 8.22b) are *Rialto*, *San Marco*, and *P.le Roma*. Among these, it is noteworthy that the stops *S. Angelo* and *S. Marco Vallaresso* do not appear in the stops of other user categories. They are typical for residents and they are used to move within the city centre. It is worth noticing that the stop *Mainland* does not occur among the most frequent stop in the morning but *Piazzale Roma (bus)* is in the list of the most used in the afternoon. Indeed,



(a) Stops related to the first validation. (b) Stops related to the last validation.

Figure 8.22: Normalised number of validations during the hours of the day grouped by weekdays (in blue) and weekends (in orange). These represent pairs of residents validations occurring within a 10-hour time window during the spring period.

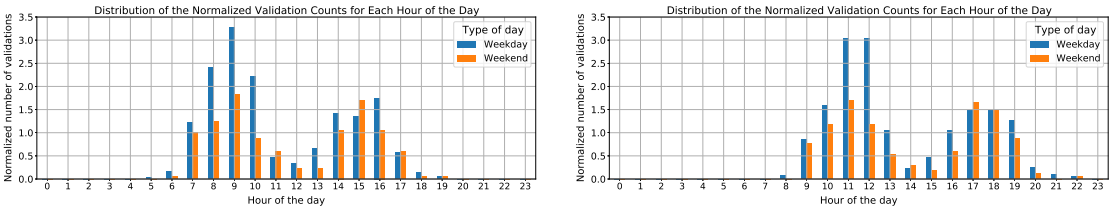
Mainland is the tenth most frequent stop in the morning and it is not in a higher position because residents generally avoid validating their tickets unless compelled by physical barriers.

A further distinction from workers and students is the lack of Lido stops among the most frequently used in the afternoon. Indeed, Lido is at the ninth position and in the Film Festival period is in the eighth position.

Retirees

Regarding pensioners, our analysis of the time intervals between consecutive stamps (Figure 8.14) shows that most validations occur in 2-hour periods. We examined this interval to analyse validation patterns both on weekdays and during the weekends of the spring period. The data shows that pensioners' short trip behaviours are markedly different from those of students (Figure 8.19). Pensioners primarily travel in the morning, typically leaving home between 8 a.m. and 10 a.m. (Figure 8.23a) and returning between 11 a.m. and 12 p.m. (Figure 8.23b). In the afternoon, they generally travel between 2 p.m. and 4 p.m. and return between 5 p.m. and 7 p.m., although this movement is less frequent compared to their morning trips. During lunchtime, around 12–1 p.m., pensioners make very few trips, unlike students, who start increasing their validations reaching a peak at 2–3 p.m. Additionally, there is a clear difference in validation patterns between weekdays and weekends: validations are more prevalent in the morning during the week, while there is a slight increase in afternoon validations on weekends.

With regard to the stops utilised by pensioners for two-hour journeys, Figure 8.24 displays the eight most frequently used stops during the spring period. Relevant variations observed in other periods are discussed when present. For the initial validation, pensioners frequently use stops such as *Lido*, *Rialto*, *S.Elena-*

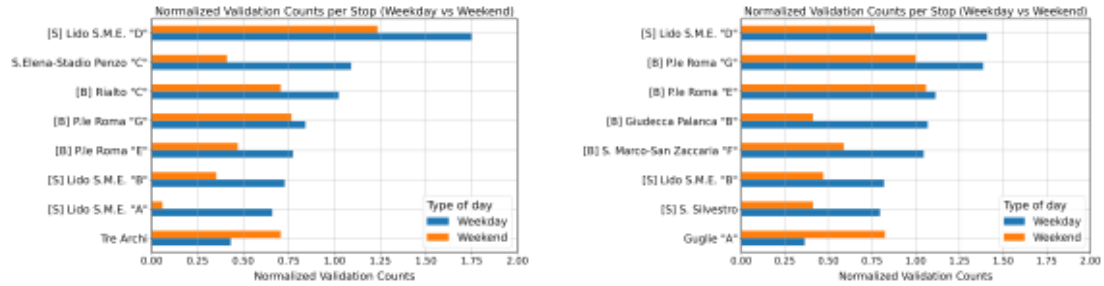


(a) First daily validations.

(b) Last daily validations.

Figure 8.23: Normalised number of validations during the hours of the day grouped by weekdays (in blue) and weekends (in orange). These refer to pairs of one-day validations by retirees during the spring period.

Stadio Penzo and *Piazzale Roma*. Then, depending on the season, other stops appear: *Mainland* in the Carnival and Film Festival periods, *Tre Archi* in spring and *Arsenale* in the Film Festival period. These latter stops and *S.Elena-Stadio Penzo* are areas inhabited mainly by locals. As already noted for students, *Punta Sabbioni* and *Ferrovia* do not appear as stops for short validation periods. For the return journey, the most frequent stop is *Lido*, even during the weekdays. Moreover, there are stops that are not commonly used by other user categories, such as *Giudecca Palanca* and *San Silvestro* in the Carnival and spring periods, *Guglie* in the spring period, *Arsenale* in the Carnival period and *Ospedale* in the Film Festival period. In particular, the *Ospedale* stop is located at the entrance of the Venice hospital, while *Guglie* is near *Tre Archi*, where an elder care home is situated. Finally, *San Silvestro* is used to reach the nearby fish, fruit, and vegetable market, located close to the Rialto Bridge.



(a) Stops related to the first validation.

(b) Stops related to the last validation.

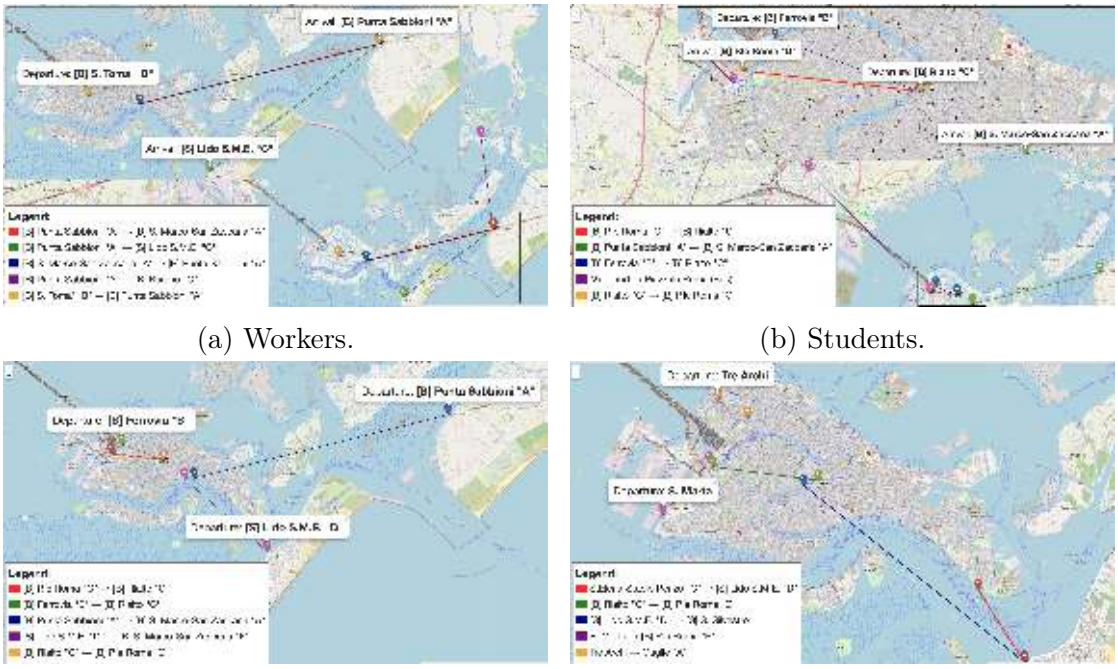
Figure 8.24: Normalised number of validations for the 8 most used stops by retirees during the spring period, grouped by weekdays (in blue) and weekends (in orange). The stops reported are those used by retirees within a 2-hour interval.

Visualisation of the Most Popular Two-Stop Trips

After a detailed analysis of validation pairs during the weekends and the weekdays across various user categories, we want to visualise on the map the most frequently used validation pairs for each category. Specifically, in Figure 8.25, we can observe the five most used validation pairs by workers, students, residents, and retirees during the Carnival period. An upward arrow represents the first validation of the day, while a downward arrow represents the second validation of the day. The first validation indicates the point where people start their journey, whereas the second validation shows the location from which they depart to return home. In the legend, the pairs of stops are listed in order, from the one with the highest number of validations to the one with the least. Each trajectory (lines connecting two validation pairs) has a different colour only to ensure better visualisation, and the thicker the line, the greater the number of validations for that departure-arrival pair. Regarding workers, in Figure 8.25a, we can observe that the most frequently used stop, both as the first validation of the day and the second validation of the day, is *Punta Sabbioni*. Specifically, workers start from *Punta Sabbioni* and travel to *San Marco* (in red), *Lido* (in green), or *Burano* (in purple). These are workers who live in *Punta Sabbioni* and surroundings and commute to these areas for work. For example, in the first case, which is the most frequently used route, workers arrive at *San Marco*, a central area of the city from which they can reach other zones. There are also trajectories where workers start from *San Marco* (in blue) or *San Tomà* (in orange) to reach *Punta Sabbioni*. These routes are likely used by workers who reside in the city centre and work in *Punta Sabbioni*. An additional route identified during the Film Festival period connects *Punta Sabbioni* to *Lido*, highlighting Lido as another major work destination.

Regarding students, as shown in Figure 8.25b, we can observe a completely different usage of the city. Students start from strategic points such as *P.le Roma* (in red), *Ferrovia* (in blue), and *Mainland* (in purple). The *Punta Sabbioni* stop (in green) is also used as a starting point since some students reside in that area and need to reach the centre of Venice to attend school or university. Furthermore, the primary arrival point is *Rialto*, a strategic stop for reaching schools and the university. The route taken by students commuting to the mainland for academic purposes is shown in orange. Typically, they depart from *Rialto* and return to Venice via *Piazzale Roma*. It is worth noticing that students arriving by bus from Mestre city centre (*Mainland*) also return home by bus from *Piazzale Roma (bus)*. In addition to the aforementioned routes, the spring period reveals a recurrent travel pattern from *Lido S.M.E. "D"* to *S. Marco–San Zaccaria "F"*, evidencing the movement of students residing in Lido who attend school or university in Venice.

Residents (Figure 8.25c) also start from strategic points with abundant trans-



(a) Workers. (b) Students.

(c) Residents. (d) Retirees.

Figure 8.25: Five most used routes by workers, students, residents, and retirees during the Carnival period. An upward arrow represents the first validation of the day, while a downward arrow represents the second validation of the day, and the thicker the line connecting two stops, the greater the number of validations for that departure-arrival pair.

portation options, such as *P.le Roma* (in red) and *Ferrovia* (in green), to arrive at *Rialto*, similarly to students. This is because, coming from different parts of the city, they head towards the city centre where they can then reach their workplaces. Additionally, residents depart from *Punta Sabbioni* (in blue) and *Lido* (in purple) to reach the city, stopping at *San Marco* in this case. As previously noted for students, the most commonly used route for commuting to the mainland (highlighted in orange) involves departing from *Rialto* and returning via *Piazzale Roma*. Finally, an additional route observed during the Film Festival period connects *S. Tomà* to *S. Angelo* — two stops that allow travellers to bypass Piazzale Roma and Rialto, which are likely to be heavily congested during this time.

For retirees, all three periods are analysed, as their travel patterns include distinct routes not observed among other user categories. Unlike workers, students and residents, retirees tend to move predominantly within the city, rarely utilising peripheral stops such as *Mainland*, *Ferrovia*, or *Punta Sabbioni*. However, they frequently travel to *Lido*. Indeed, the most commonly used route across all

(a) Spring period. (b) Film Festival period.

Figure 8.26: Five most used routes by retirees during the spring and the Film Festival period. An upward arrow represents the first validation of the day, while a downward arrow represents the second validation of the day, and the thicker the line connecting two stops, the greater the number of validations for that departure-arrival pair.

periods is from *Sant'Elena* to *Lido* (in red). During the Film Festival period (Figure 8.26b), which is in the summer season, two additional routes toward *Lido* are observed: *S. Tomà* to *Lido* (in blue) and *S. Marco–S. Zaccaria* to *Lido* (in purple). Furthermore, there are stops and routes that are unique to retirees. Notably, in Figure 8.25d in orange, we see the trajectory from *Tre Archi* to *Guglie*, and the stop at *Santa Marta* (in purple), a typically residential area, leading to *P.le Roma*. Although these are short trips, they are commonly undertaken by elderly people. The route from *Rialto* to *P.le Roma* (in green in Figure 8.26a) and the opposite, i.e., from *P.le Roma* to *Rialto* (in yellow in Figure 8.26a) are used by other categories as well, but a completely new stop is *S. Silvestro* (in purple), which is used as an arrival point after departing from *Lido*. Also the route from *Rialto* to *S. Tomà* in the spring period (in blue in Figure 8.26a) is typical only for this category.

From this analysis, we observed distinct patterns and preferred stops among the different user categories. Workers primarily use the *Punta Sabbioni* and *San Marco* stops, commuting from the outer parts of the city to more central areas. Students also exhibit this pattern, albeit from different stops such as *Mainland*, moving from peripheral regions to the city centre, particularly the *Rialto* area. Similarly, residents leverage strategic points like *Piazzale Roma*, which are well-connected with various transportation options, to access central parts of the city. In contrast, retirees tend to move within the city itself, avoiding peripheral areas and utilising other stops such as *Santa Marta*, *Sant'Elena*, and *S. Silvestro*.

8.5 Pattern Analyses of Stop Usage

Building on the extensive analysis of data and user behaviour during three key events in Venice, our objective is to identify significant patterns in user movements. To this end, we conduct two distinct *clustering analyses of stops*, each addressing a different aspect of mobility.

The first clustering analysis focuses on user categories and aims to characterise how different types of users — such as residents, students, tourists, and occasional users — interact with the transportation network during the three periods under investigation. Specifically, we examine the usage patterns of each stop across the various user groups, and apply clustering techniques to group together stops that exhibit similar usage profiles. In this context, similarity is defined in terms of how stops are used by the different categories: two stops are considered similar if they are used in similar way by the various user groups. This analysis provides insights into how different areas of the city are used by distinct types of users.

The second clustering analysis addresses the temporal dynamics of mobility. In this case, the objective is to identify recurrent patterns in how stops are used throughout the day, regardless of user category. To this end, we divide the day into six time slots, each representing a specific portion of daily mobility — from early morning to late evening. For each stop, we compute the relative number of validations occurring within each time slot, and apply clustering to group together stops that share similar temporal profiles. This allows us to distinguish, for instance, stops that are primarily active during commuting hours from those used more frequently in the afternoon or evening. By capturing the temporal dimension of mobility, this analysis contributes to a deeper understanding of when different parts of the network are most intensively used.

Together, these two clustering analyses provide a multidimensional perspective on urban mobility: the first highlights *who* uses specific stops or areas, while the second reveals *when* these movements occur. This additional analysis further demonstrates the potential of the dataset introduced in this study to investigate patterns of urban space utilisation across both spatial and temporal dimensions. Specifically, it enables a detailed examination of how different population groups access and traverse the city throughout the day and across distinct periods of the year.

To perform both clustering analyses, we first need to associate a distinct vector representation with each stop, aiming to capture either *who* used the stop or *when* the stops were used. Adopting these two distinct vector representations of stops, we then employ a Hierarchical Cluster Analysis (HCA), whose effect is a hierarchical decomposition of a given set of data objects [58]. We adopt a typical bottom-up HCA approach, where each data point initially forms a cluster, and the algorithm

iteratively merges the pair of closest clusters until all points belong to a single root cluster or until a given stopping criterion is met. More specifically, we measure the dissimilarity between vector representations of stops by Euclidean distance. In addition, we measure the distance between clusters as the greatest distance between any pair of their respective elements, thus adopting the complete linkage method, which tends to produce compact and well-separated clusters, and is less sensitive to outliers compared to alternative linkage strategies [21]. Finally, the iterative aggregation process of HCA is terminated once the distance between the two nearest clusters exceeds a predefined threshold, indicating that no sufficiently similar clusters remain to warrant a meaningful merge. The threshold is established through empirical evaluation and iterative experimentation.

In the following sections, we present the two clustering approaches in detail, starting from the two distinct vector representation of stops adopted. We first examine the clustering based on user categories, followed by the analysis of temporal usage patterns. For each case, we discuss meaningful results that reveal distinctive patterns in stop usage. We conclude this analysis by comparing the outcomes obtained from both clustering strategies across the different periods under investigation.

8.5.1 Vector Representations of Stops

Starting from the validations collected for each time period, e.g., during Carnival or Film festival, each stop s is represented through a vector $\vec{v}_s \in \mathbb{R}^d$, thus obtaining a distinct $\vec{v}_s$ for each time period. For user-category-based clustering, the vectors have $d = 4$ dimensions, representing the validation counts performed by each category of users in a given time period. For time-based clustering, the vectors have $d = 6$ dimensions, corresponding to 6 time-slots, spanning throughout the day, used for binning the validation histograms collected for the time period under investigation. In both cases, the validation counts of the vectors are expressed in percentage, i.e., the vectors are L_1 normalised to allow for meaningful comparisons across stops.

Vectors based on user category usage

Given a time period, for each stop s we count the number of validations performed by each category of users. The validation counts selected for vector $\vec{v}_s$ are four ($d = 4$ dimensions), corresponding to the following categories: residents (which also include retired individuals), students, 75-minute tickets, and tourists. For this analysis, we exclude the workers category as this group is under-represented. Finally, we apply the L_1 normalization to $\vec{v}_s$, thus representing the proportions of validations belonging to these four categories.

Vectors based on temporal usage

Given a time period, for each stop s we record for each time of the day, from 00:00:01 to 24:00:00, the count of all validations performed by all users, regardless of their category. We then discretise the time variable, by creating a set of contiguous intervals — also called bins or time slots — used to aggregate for each bin all the validations performed in the corresponding time interval. Finally, we L_1-normalize the resulting vectors.

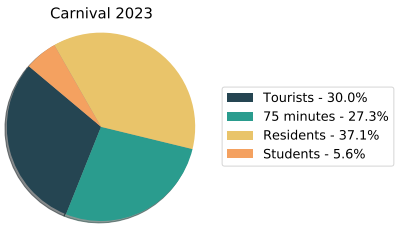
More specifically, for each vector $\vec{v}_s$, we chose six time slots ($d = 6$ dimensions) that comprehensively better describe the different periods of the day. Although we can discretise the time variable, thus identifying contiguous time slots, by adopting heuristic techniques such as equal width or equal frequency binning, we experimentally found that a form of *data-driven binning* is more valuable for our analysis. Therefore, the time-series data, collected for all validations in a given time period, are partitioned into six clusters using K-means (K=6), where each cluster finally corresponds to a distinct time slot used to produce $\vec{v}_s$.

We also apply this process of first collecting validations for each time of the day, and then discretising the time, not only to all user categories together, but also by projecting such validation data with respect to the following four categories: 75-minute tickets, Residents, Students, and Tourists.

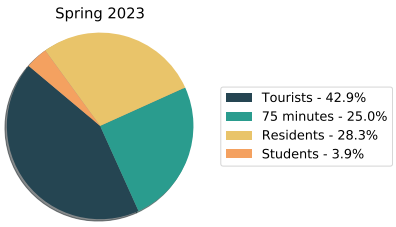
8.5.2 Clustering based on User Categories

For this clustering analysis, we focus on the stops equipped with barriers, where the validation is mandatory for all types of tickets considered. Including stops without barriers would lead to an underestimation all categories, except for 75-minute ticket users, who are forced to validate at every stop, even those without barriers, to start the validity period of their tickets. Figure 8.27 shows the percentage of validations associated with each of the four user categories included in the analysis, considering only validations recorded at stops equipped with barriers.

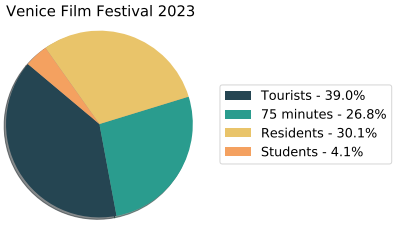
Recall that for each time period, we associate with each stop s a distinct vector $\vec{v}_s \in \mathbb{R}^4$, where the four dimensions corresponds to the proportion of validations performed by users of (1) 75-minute tickets, (2) Residents, (3) Students, and (4) Tourists, respectively. For example, for stop *S. Angelo* during the Carnival period, we obtain the vector $(0.25, 0.41, 0.04, 0.30)$, indicating that during this period this stop is predominantly utilised by Residents. Conversely, for stop *Burano "C"* during the same period, the vector obtained is $(0.17, 0.11, 0.02, 0.70)$, clearly revealing its pronounced touristic function. Based on this vectorial representation of stops, the clustering algorithm computes the Euclidean distances between all stop pairs. All these pairwise distances are exploited by the HCA algorithm during the iterative aggregation. We stop the iterative aggregation when the closets



(a) Winter — Carnival.



(b) Spring — Easter.



(c) Late summer/early autumn — Film Festival.

Figure 8.27: Number of validations for 75-minute ticket, tourist, resident, and student categories in the three periods, considering only stops with access-control barriers.

cluster distance, computed according to the complete linkage approach, exceeds a threshold of 0.21, which was set after an empirical evaluation.

As a first result, we focus on the clustering outcome during the Carnival period, a time characterised by a significant shift in mobility patterns due to the increased presence of tourists and occasional users. Figure 8.28 illustrates the spatial distribution of the clustered stops. The stops equipped with barriers are coloured according to their assigned cluster. Out of the 13 clusters produced by the algorithm, four clusters (namely clusters 4, 7, 9, and 10) account for the majority of the stops — specifically, 29 out of 52. Table 8.5 presents a summary of these clusters, including the number of stops they contain, the total number of validations, and the mean and standard deviation of the usage proportions for each user category.

Each cluster reflects a distinctive user profile. Cluster 4 (in orange in Figure 8.28), for example, is characterised by a very high proportion of 75-minutes ticket users (82.75%) and very low values for the other categories, suggesting a strong prevalence of short-term validations, possibly linked to occasional visitors. Conversely, clusters 7 (in red) and 9 (in purple) are dominated by residents, with average shares exceeding 41%, accompanied by a moderate presence of both 75-minute tickets and tourists. Finally, cluster 10 (in lilac) — which includes the largest number of stops — is marked by a more balanced distribution: tourists account for 39.41%, 75-minute ticket users for 31.69%, and residents for 23.76%.

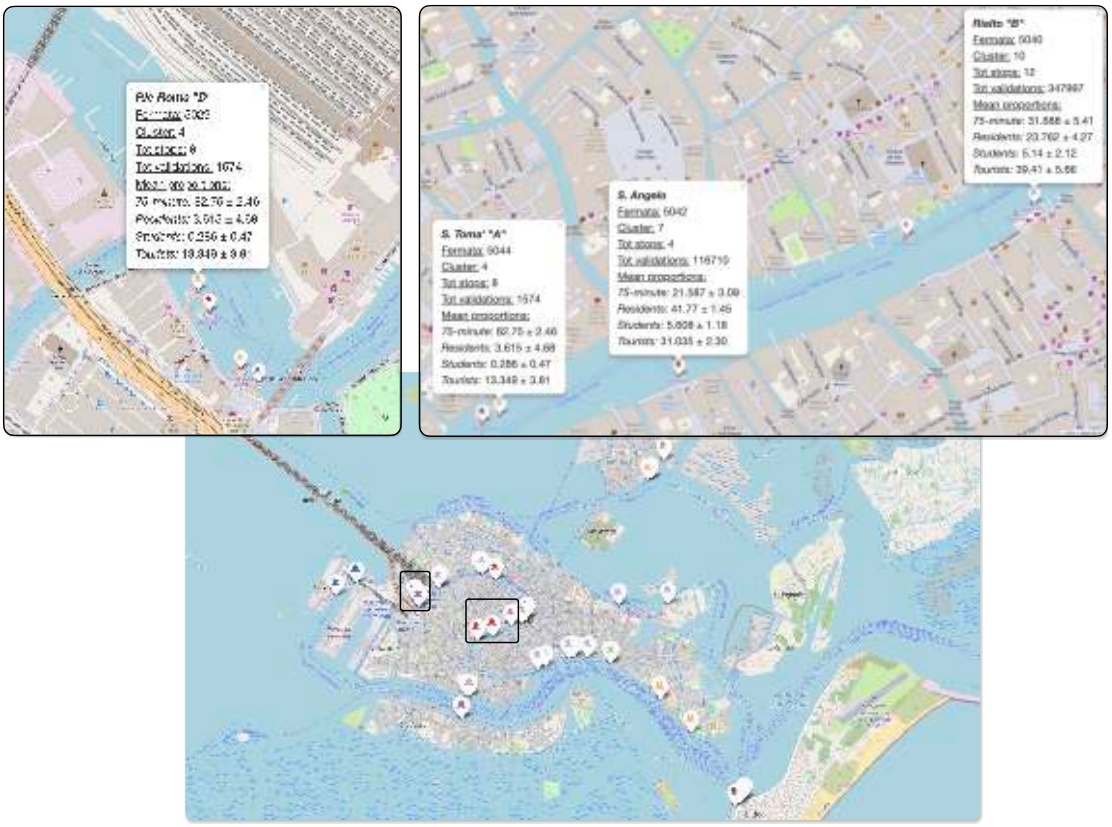


Figure 8.28: User-based clustering of stops with barriers during the Carnival period. The map displays stops coloured according to the cluster they belong to, based on similarity in usage patterns across four user categories. Two zoomed-in views highlight the areas around *P.le Roma* (left) and central Venice along the Grand Canal (right).

This indicates areas of mixed usage by both local and visiting passengers.

Figure 8.28 also includes two zoomed-in areas to provide more detailed insight. The zoom on the left focuses on the area surrounding *P.le Roma*, while the one on the right highlights the stops located in the central part of Venice along the Grand Canal. In the left zoomed area, we observe a concentration of neighbouring stops that belong to different clusters, highlighting distinct usage patterns despite their close geographical proximity. For instance, stops *P.le Roma "D"* and *P.le Roma "F"* both belong to cluster 4 (orange), which is characterised by a high prevalence of 75-minute ticket users. In contrast, *P.le Roma "E"* is assigned to cluster 7 (red), and *P.le Roma "G"* to cluster 9 (purple), both of which are primarily associated with resident users. This contrast reveals how the clustering captures differences in

	No. of		Ticket Type			
Cluster	**Stops**	**Validations**	**75-minutes**	**Residents**	**Students**	**Tourists**
4	8	1,574	82.75 ± 2.46	3.61 ± 4.68	0.29 ± 0.47	13.35 ± 3.81
7	4	116,710	21.59 ± 3.09	41.77 ± 1.45	5.61 ± 1.18	31.03 ± 2.30
9	5	272,588	28.80 ± 3.55	42.18 ± 3.51	7.31 ± 1.80	21.70 ± 2.60
10	12	347,997	31.69 ± 5.41	23.76 ± 4.27	5.14 ± 2.12	39.41 ± 5.66

Table 8.5: Summary of the main clusters identified during the Carnival period. For each cluster, the table reports the number of stops, the total number of validations, and the mean and standard deviation for each user category.

usage patterns even among stops located within the same transportation hub. For instance, stops *P.le Roma "D"* and *"F"*, both assigned to cluster 4, are primarily served by Lines 2/, 3, 4.2, 5.2, and the night service N. These lines follow key tourist corridors, connecting *P.le Roma* and the railway station with high-demand destinations such as *Rialto*, *San Marco*, *Lido*, and *Murano*. The service patterns of these routes suggest that they are frequently used by short-term or occasional users. This aligns with the high proportion of 75-minute ticket validations observed at these stops, reinforcing the idea that they act as major gateways for tourist flows entering the historic centre of Venice. In contrast, stop *P.le Roma "E"*, assigned to cluster 7, is served by Line 4.1, which connects *P.le Roma* to residential areas such as *Giudecca Palanca*, and *Sacca Fisola*. These destinations are primarily inhabited by local residents and are less frequented by tourists, supporting the observed predominance of resident users at this stop. Similarly, stop *P.le Roma "G"*, part of cluster 9, is served by Line 2, which connects both residential areas (e.g., *Giudecca Palanca* and *Sacca Fisola*) and central touristic zones such as *San Marco*. Despite its access to tourist attractions, the route also serves as a functional connection for daily mobility, which may explain the relatively high proportion of residents (over 41%) found in this cluster. This stop appears to fulfil two distinct functions, serving local commuters as well as visitors, with usage varying by time of day and season.

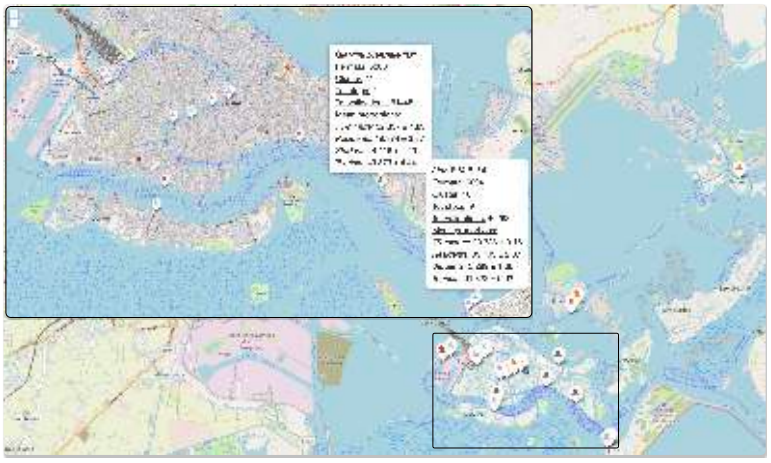
In the zoomed-in area on the right side of Figure 8.28, we observe a similar scenario. This part of the map includes several stops located along the Grand Canal in the historic centre of Venice, such as *S. Tomà "A"*, *S. Tomà "B"*, *S. Angelo*, *Rialto "B"*, and *Rialto "C"*. These stops, although geographically close, are assigned to different clusters, reflecting distinct patterns of use. For instance, *S. Tomà "A"* belongs to cluster 4, which is characterised by a strong prevalence of 75-minute ticket validations. This stop is served by Line 1, a highly touristic route that travels the full length of the Grand Canal and includes key attractions such as *Rialto*, *Accademia*, and *San Marco*. The high share of short-term users is

therefore consistent with the touristic nature of the line and its destinations. In contrast, the adjacent stops *S. Tomà "B"* and *S. Angelo* are assigned to cluster 7, which is dominated by residents. Despite serving the same line as *S. Tomà "A"*, these stops are located on the opposite quay and serve the reverse direction of travel — *S. Tomà "A"* serving water buses heading towards *P.le Roma*, while *S. Tomà "B"* those heading towards *San Marco*. Moreover, they are also part of the night service Line N, which is typically used by both locals returning home and tourists moving between accommodation and nightlife areas. This mixed service pattern aligns with the more balanced usage profile of cluster 7, which, while resident-dominated, also includes a substantial share of tourist validations. In the Rialto area, two closely located stops are assigned to distinct clusters, highlighting how the clustering process captures nuanced differences in user behaviour. *Rialto "B"* belongs to cluster 10, which is characterised by a predominant share of tourists (39%), followed by 75-minute ticket users (31%) and a smaller proportion of residents (24%). This stop is served by Line 1, a highly frequented route that runs through the entire Grand Canal, connecting major tourist attractions such as *Accademia*, *S. Marco*, and *Lido*. The direct access to such emblematic destinations likely explains the predominance of visitors and occasional users observed at this stop. In contrast, *Rialto "C"* (located to the right of stop *Rialto "B"* on the same side of the Grand Canal) is part of cluster 9, which shows a higher share of resident users. This stop is mainly served by Line 2 and its variant 2/, both of which connect to key interchange and residential areas such as *Giudecca*, *Zattere*, and areas commonly used as departure points from the city, such as the railway station, *P.le Roma* and *Tronchetto*.

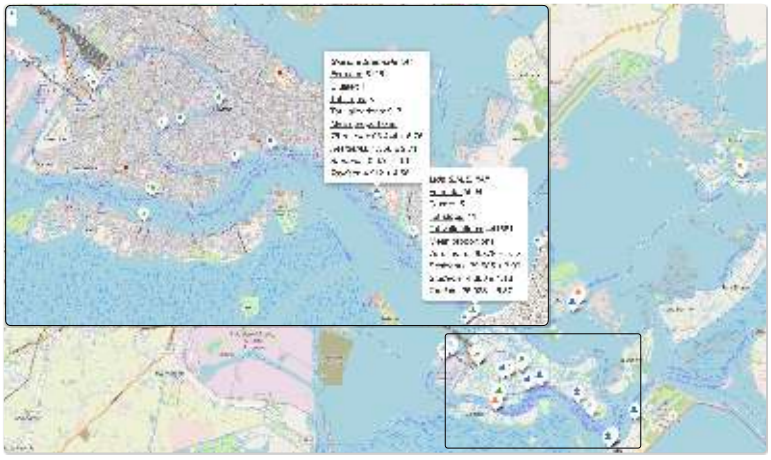
As a second result, we examine the differences in stop usage across user categories during the Easter and Film Festival periods. Figure 8.29 illustrates the spatial distribution of the clustered stops during the above periods. As already remarked, the barrier-equipped stops are coloured according to their assigned cluster.

	No. of		**Ticket Type**			
Cluster	**Stops**	**Validations**	**75-minutes**	**Residents**	**Students**	**Tourists**
3	4	111,889	15.39 ± 2.83	11.38 ± 5.07	1.13 ± 0.40	72.10 ± 3.48
6	2	55,350	44.21 ± 3.94	11.86 ± 1.37	1.64 ± 0.37	42.28 ± 2.20
7	2	15,756	56.78 ± 0.68	4.26 ± 4.26	0.37 ± 0.37	38.60 ± 5.31
10	9	402,832	23.79 ± 3.15	39.10 ± 2.37	5.29 ± 1.66	31.82 ± 5.62
11	4	61,445	32.33 ± 1.83	19.87 ± 3.37	4.42 ± 1.40	43.37 ± 4.44

Table 8.6: Summary of the main clusters identified during the Easter period. For each cluster, the table reports the number of stops, the total number of validations, and the mean and standard deviation for each user category.



(a) During Easter.



(b) During Film Festival.

Figure 8.29: Comparison of user category clusters identified during the Easter and Film Festival periods.

Table 8.6 and Table 8.7 summarise the most representative clusters identified for the two periods, highlighting the number of stops, the total number of validations, and the mean and standard deviation of the share of each user category.

Clusters dominated by tourist users are clearly identifiable in both periods. During Easter, cluster 3 (in orange in Figure 8.29a) stands out with an average of over 72% of validations attributed to tourists, while during the Film Festival, cluster 3 (in orange in Figure 8.29b) similarly shows a dominant tourist presence, with nearly 69%. Importantly, these clusters include many of the same stops, particularly those located in highly touristic areas such as *Murano* and *Burano*. This consistency across different time periods suggests that, despite slight varia-

	No. of		Ticket Type			
Cluster	**Stops**	**Validations**	**75-minutes**	**Residents**	**Students**	**Tourists**
1	8	917	93.44 ± 5.75	1.57 ± 2.71	0.08 ± 0.14	4.91 ± 4.58
2	6	36,508	67.95 ± 7.39	7.25 ± 4.05	0.79 ± 0.47	24.02 ± 6.01
3	3	57,731	23.99 ± 6.70	6.48 ± 4.61	0.68 ± 0.48	68.85 ± 2.27
5	11	441,681	28.88 ± 5.73	39.58 ± 3.96	4.61 ± 1.16	26.93 ± 5.37

Table 8.7: Summary of the main clusters identified during the Film Festival period. For each cluster, the table reports the number of stops, the total number of validations, and the mean and standard deviation for each user category.

tions in usage proportions, these stops are systematically grouped together and remain primarily used by tourists. In both tables, we also observe the presence of a large cluster characterised by a higher proportion of resident users. In the Easter period, this corresponds to cluster 10 (in liliac in Figure 8.29a), with nearly 39% of validations by residents, whereas during the Film Festival, cluster 5 (in green in Figure 8.29b) emerges with a similar resident-dominant profile (39.5%). These clusters include a comparable set of stops, mostly located in residential areas of the city, such as *Giudecca* or the island of *Lido* and key intermodal hubs like *P.le Roma*, frequently used by residents for daily commuting or travel in and out of the city. This suggests that, while some seasonal dynamics are present, the functional role of stops with respect to these two categories tends to remain unchanged.

In contrast, notable changes emerge in the usage patterns associated with 75-minute tickets. During the Film Festival period, an increase in this type of validation is observed, particularly at specific stops such as those on the Lido island. For example, the *Lido S. Nicolò* stop — located to the east of the main Lido area (in dark blue) — is absent from the Easter clustering but appears prominently during the Film Festival. Its assignment to cluster 1, which is characterised by a 93% share of 75-minute tickets, highlights its functional role in accessing the festival area. This, along with the nearby *Lido S.M.E. "D"* stop also being part of the same cluster, indicates that the Film Festival attracts many short-term or occasional users who rely on single-journey tickets to reach the venue.

8.5.3 Clustering based on Time Slots

The second clustering method relies exclusively on temporal usage patterns, identifying stops with similar levels of activity across corresponding time slots during the day. Table 8.8 displays the temporal clusters obtained by applying k-means to the entire dataset, which includes all three temporal periods and all user categories, but only considers stops equipped with barriers. Each cluster corresponds to a

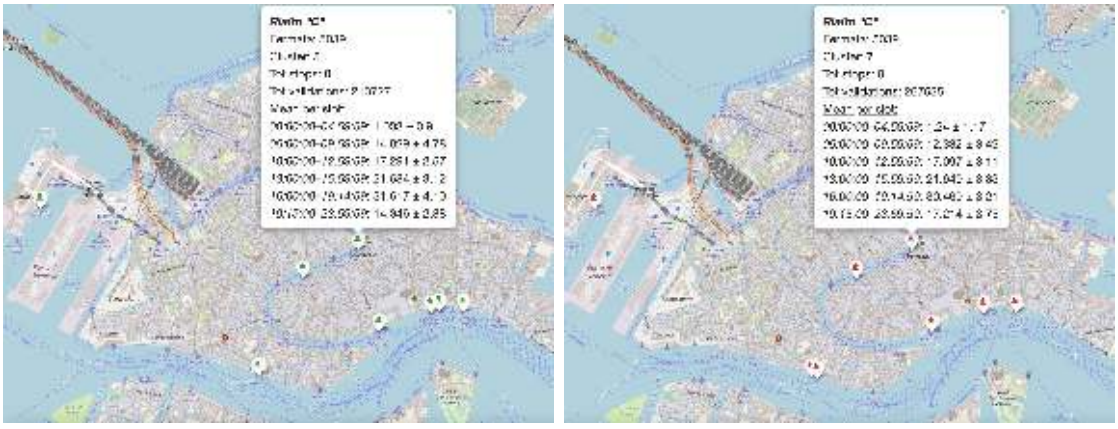
Slot	Start	End	No. of validations
1	00:00:00	04:59:59	113,887
2	05:00:00	09:59:59	2,187,967
3	10:00:00	12:59:59	2,678,139
4	13:00:00	15:59:59	2,767,378
5	16:00:00	19:14:59	2,984,946
6	19:15:00	23:59:59	1,475,896

Table 8.8: Time slots obtained through k-means clustering applied to the entire dataset — limited to stops equipped with validation barriers — encompassing all three temporal periods and all user categories, along with the corresponding number of validations.

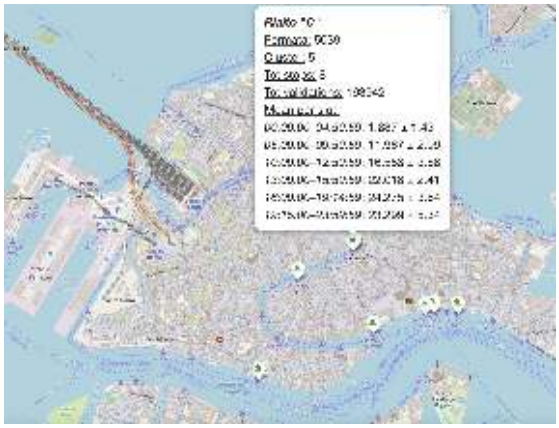
specific time slot that captures homogeneous patterns of validation activity across the day. The number of validations associated with each slot is also reported, highlighting the variability in travel demand throughout the day.

Once the time slots have been established, the next step involves constructing a distance matrix. To this end, each stop is associated with a vector that encodes the percentage of validations corresponding to each time slot. Each stop s_i is encoded as a vector $s_i = (p_{i1}, p_{i2}, \ldots, p_{i6})$, where each component p_{iz} is the percentage of validations stamped at stop i during the time slot z. Different analyses have been performed. First, we consider all user categories, and we build a distance matrix based on the pairwise Euclidean distance between all stops. As a result, we obtain clusters that characterise each stop according to its temporal usage profile throughout the day. Then, we apply this process separately to the four categories: 75-minute tickets, Residents, Students, and Tourists. We will present the results for Residents and Tourists since interesting insights emerge.

For residents, distinct variations in temporal validation patterns can be observed across the different periods. Figure 8.30 depicts a cluster comprising approximately the same eight stops consistently identified across the clustering outputs for the pre-Carnival, Easter, and pre-Film Festival periods. Although the spatial composition of this cluster remains largely stable, with minor variations, its temporal usage profile changes significantly between periods. In the pre-Carnival period (Figure 8.30a), the green cluster includes stops such as *Zattere "B"*, *S. Marco Vallaresso "B"*, *S. Marco-San Zaccaria "F"*, *S. Marco-San Zaccaria "A"*, *San Marco-San Zaccaria "D"*, *S. Angelo*, *Rialto "C"*, and *Tronchetto "B"*. During this winter period, only about 14% of the validations at these stops occur in the last time slot (19:15:00–23:59:59), reflecting reduced evening mobility among residents, likely due to seasonal and climatic conditions. In the Easter period



(a) Pre-Carnival. (b) During Easter.



(c) Pre-Film Festival.

Figure 8.30: Clustering time-slot residents.

(Figure 8.30b), the red cluster includes nearly the same set of stops, with *Zattere "A"* appearing in place of *San Marco-San Zaccaria "D"*. The evening usage rises to approximately 17%, suggesting a gradual increase in residents' activity during later hours. Although this reflects only a 3% increase, it indicates a slight shift in residents' behaviour towards later activities, possibly encouraged by milder weather and holiday dynamics. The most significant variation occurs in the pre-Film Festival period (Figure 8.30c), where the green cluster again retains a similar structure. In this case, *Rialto "D"* and *Zattere "A"* are included, while *Tronchetto "B"* and *Zattere "B"* are no longer part of the group. Evening usage in the last time slot rises sharply to 23%, marking an increase of 6% compared to Easter and 9% relative to the pre-Carnival period. This pronounced change aligns with the summer season, during which residents are more likely to use these stops in the evening, possibly for social or leisure activities.

(a) During Carnival.

(b) During Easter.

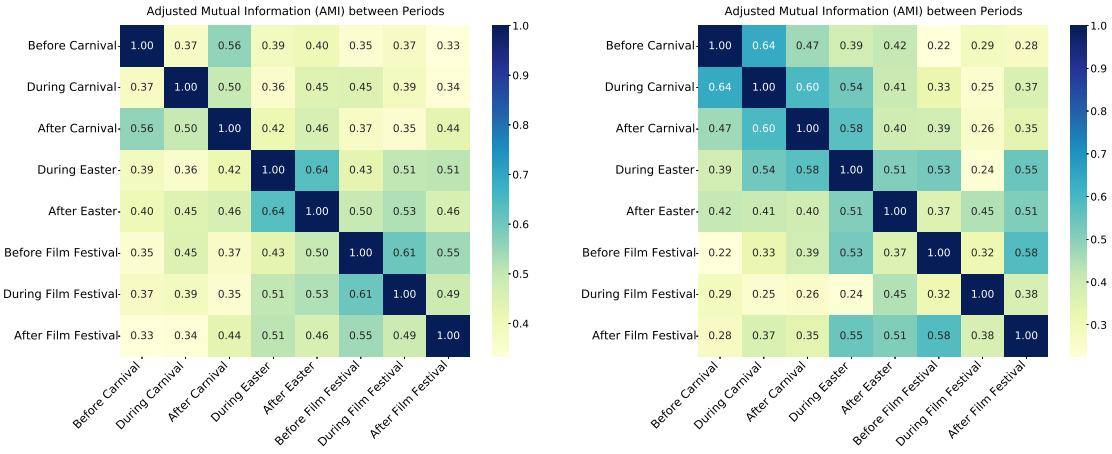
Figure 8.31: Clustering time-slot tourists.

Another noteworthy result emerges from the time slot analysis focusing on tourists. As shown in Figure 8.31, two distinct clusters can be identified across the during-Carnival and during-Easter periods, both of which include a shared stop: *Bacini - Arsenale Nord*. In particular, Figure 8.31a shows that during the Carnival period, this stop is clustered together with *S. Marco Giardinetti "A"* and is characterised by a 44% share of validations in the last time slot (7:15 p.m. and 11:59 p.m.). This behaviour is strongly linked to the nature of the Carnival itself, which features a rich programme of evening events and performances, including masquerade parades, concerts, and other cultural initiatives that typically includes at least two main events each evening (around 7 p.m. and 9 p.m.), as well as five or six nights of open-air parties and DJ sets. Moreover, combining the percentages of validations recorded in the last two time slots (from 4 p.m. to 11:59 p.m.) reveals

that nearly 70% of the stop's total daily activity occurs during the latter part of the day, further highlighting the importance of these two stops in managing late-afternoon and evening tourist flows. *Bacini - Arsenale Nord* plays a strategic role in this context: it provides direct access to the Arsenale area, a key location for major Carnival exhibitions and shows. Likewise, *S. Marco Giardinetti "A"* serves as one of the nearest access points to the main events in S. Marco Square. The high volume of evening validations at these stops indicates that tourists tend to stay out later to attend events and return afterwards, reinforcing the importance of these landing points in managing night-time tourist flows during peak festivities. By contrast, in the Easter period (Figure 8.31b), *Bacini - Arsenale Nord* is clustered with two stops located on the island of Murano and one on the island of Burano. In this case, the evening usage drops sharply, with the last time slot accounting for only 1.3% of validations. This shift suggests that the stop serves a markedly different function during Easter, acting primarily as a departure point for daytime excursions to the outer islands of the Venetian Lagoon. The limited evening activity indicates a reduced participation in night-time events and a greater interest in cultural and artisanal attractions, such as the glassmaking tradition of Murano and the distinctive built environment of Burano. Overall, this contrast highlights how the role of a single stop can vary substantially across seasons and events, depending on the temporal dynamics of tourist flows and the spatial distribution of attractions.

8.5.4 Comparison of Clustering Results

Figure 8.32 presents the pairwise similarity of clustering results across different temporal periods, measured using the *Adjusted Mutual Information* (AMI). The AMI [139, 140] quantifies the agreement between two independent clusterings over the same set of elements while adjusting for chance. It provides a normalised score ranging from 0 (no agreement beyond chance) to 1 (perfect agreement), making it particularly suitable for comparing clustering results derived under varying temporal or methodological conditions. In particular, Figure 8.32a presents the AMI values calculated between the clustering distributions of user categories, highlighting how the cluster structures vary across the different time periods. A few notable patterns can be observed. First, the clustering structures for the Easter and post-Easter periods exhibit a relatively high degree of similarity, with an AMI value of 0.64, suggesting a certain stability in user distribution across these consecutive periods. This similarity may be explained by the presence of extended holiday breaks in the post-Easter period, including public holidays such as April 25 and May 1, often associated with long weekends that influence mobility patterns in a similar way to Easter itself. A comparable similarity is also observed between the before and during Film Festival periods, with an AMI value of 0.61. This



(a) Clustering based on user categories.

(b) Clustering based on time slots.

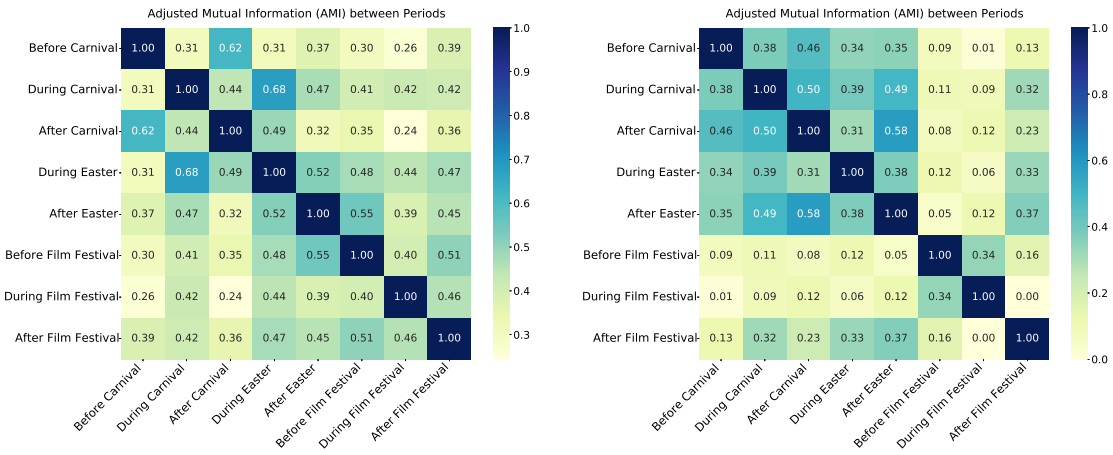
Figure 8.32: Adjusted Mutual Information of the clustering results based on user categories and time-slot usage across winter, spring, and late summer/early autumn 2023 periods.

indicates a stable pattern in stop usage across user categories, suggesting that its spatial distribution remains largely unchanged from pre-event conditions. By contrast, lower AMI values are found between more temporally distant periods, such as before Carnival and after the Film Festival (0.33), pointing to substantial shifts in the distribution of user categories between the winter and summer seasons. The Carnival period itself appears relatively distinct from all others, with moderate similarity values (e.g., 0.37 with before Carnival, 0.45 with after Easter), likely due to the unique and concentrated tourist dynamics associated with this major winter event. Overall, the AMI analysis highlights both seasonal continuity (e.g., spring and summer) and event-specific disruptions (e.g., Carnival), providing a deeper understanding of how user category distributions evolve in response to both temporal and contextual factors.

Figure 8.32b presents the Adjusted Mutual Information values computed between the clusterings based on time slot distributions across different periods. The matrix reveals moderate variability in the clustering structures, with notable similarities between temporally adjacent periods. For instance, the highest similarity is observed between the before-Carnival and during-Carnival periods (0.64), as well as between during-Carnival and after-Carnival (0.60), suggesting that time-of-day mobility patterns tend to remain relatively stable over short time horizons, even in the presence of major events. Similarly, a relatively strong similarity emerges between during-Easter and after-Carnival (0.58), pointing to consistent usage patterns around the spring holidays. A relatively high similarity is observed

between the during-Easter and before-Film Festival periods (0.53), and between after-Easter and after-Film Festival (0.51), possibly indicating a continuity in time-of-day behaviours during the late spring and summer periods, which are less influenced by major events and characterised by favourable weather conditions. By contrast, the during-Film Festival period stands out as the most distinct, with the lowest AMI values overall — 0.25 with during Carnival and 0.24 with during Easter. This indicates that the temporal usage of stops during the Film Festival differs considerably from other periods, possibly due to concentrated activities at specific times (e.g., afternoon or evening screenings) and atypical travel behaviours associated with the event.

In order to examine how time-of-day usage patterns vary across different user groups, Figure 8.33 reports the AMI values derived from time slot-based clusterings for two specific categories: tourists and students. This focused analysis provides insight into the distinct temporal dynamics exhibited by each group across the various periods, allowing for a more nuanced interpretation of the patterns observed in the aggregate analysis. Figure 8.33a shows the AMI values computed between the clustering results of tourists based on time-of-day usage across various periods. The matrix reveals a number of patterns that align well with known tourist dynamics and seasonal variations in Venice. First, the highest AMI values are observed between periods characterised by major events. For instance, the clustering structures for the during-Carnival and during-Easter periods exhibit a high degree of similarity (0.68), suggesting that tourists tend to follow compara-



(a) Tourists. (b) Students.

Figure 8.33: Adjusted Mutual Information of the clustering results based on time-slot usage for the tourist and student categories across winter, spring, and late summer/early autumn 2023 periods.

ble temporal usage patterns during festive times, regardless of the season. This behaviour likely reflects the influence of scheduled activities, and increased presence of tourists in central areas during the afternoon and evening. Similarly, the before-Carnival and after-Carnival periods also exhibit a relatively high AMI value (0.62), indicating a consistent pattern in tourists' time-of-day usage outside of the main event. This suggests that, in the absence of festive activities, tourists tend to follow more regular daily routines, likely oriented towards daytime visits and structured itineraries, resulting in similar temporal profiles before and after the Carnival period. By contrast, before Carnival and during the Film Festival exhibit low similarity (0.26), similarly to after Carnival and during the Film Festival (0.24), indicating that tourists' time-of-day usage patterns differ substantially between a low-tourist winter period and an event that, although highly attended, is spatially and temporally concentrated in specific venues (e.g., afternoon/evening film screenings in the Lido area). Notably, the during-Film Festival period appears to be the most temporally distinct overall, as it shows consistently low AMI values when compared with all other periods, further confirming the uniqueness of tourist mobility patterns associated with this specific event.

Figure 8.33b reports the AMI values computed from time slot-based clusterings for the student category across the different periods. The matrix reveals a remarkably clear separation between the school term and the summer break, as reflected in the clustering structure of temporal mobility. The clustering structure remains relatively stable across the school term, ranging from before Carnival to after Easter, with AMI values indicating moderate to high similarity between these periods (e.g., 0.46 between before and after Carnival, and 0.58 between after Carnival and after Easter). This regularity is likely associated with class schedules and structured daily routines, typically concentrated in the morning and early afternoon. In contrast, the before and during Film Festival periods, which correspond to the summer break, show very low AMI values with all the school-time periods (e.g., an AMI of 0.09 between before Film Festival and before Carnival, and 0.05 with after Easter), highlighting a major shift in temporal usage during the months when school and university lectures are not held. This change may reflect a significant drop in student activity or a shift towards less predictable behaviour during the summer months. Interestingly, the after Film Festival period, which coincides with the start of the new academic year, shows increasing similarity with the school periods again (e.g., with an AMI of 0.32 during Carnival,and 0.37 after Easter), suggesting a resumption of regular patterns once the academic calendar resumes. Furthermore, the period during the Film Festival shows moderate similarity only with before Film Festival (0.34), reinforcing the idea that these two phases are temporally cohesive yet distinct from the rest of the year. Notably, the similarity between during and after the Film Festival drops to 0, indicating a

complete structural break in student temporal behaviour. This abrupt transition is likely attributable to the start of the new school year, which brings a radical shift from unstructured summer routines to the more regular and time-constrained mobility associated with academic schedules.

Chapter 9

Concluding Remarks

This doctoral thesis examined mobility as a multidimensional phenomenon, encompassing maritime movements, the evolution of spatially extended environmental processes, and urban mobility patterns within a complex transport network. Across these domains, this work demonstrated how semantic enrichment, spatiotemporal modelling, and the introduction of a new data type for deforming moving regions within spatiotemporal database systems can be systematically combined to derive meaningful insights from large mobility datasets.

The first part of the thesis addressed maritime mobility by reconstructing and enriching vessel trajectories derived from AIS data. Through the integration of navigational attributes, behavioural indicators, and vessel activity information, the enriched trajectories provided a detailed and expressive representation of vessel movements. This semantic characterisation constitutes a key contribution of the thesis, as it provides a framework for interpreting vessel behaviour and identifying trip anomalies within the database environment.

Building on vessels' enriched trajectories, the thesis introduced a novel spatiotemporal model for underwater noise, which integrates acoustic propagation principles with vessel behaviour to estimate the spatial and temporal distribution of noise generated by maritime traffic. This model advances existing approaches by directly relating acoustic emissions to semantic enriched trajectories rather than relying solely on raw AIS data or direct measurements.

A further contribution lies in the complete integration of the modelling pipeline into MobilityDB. An extensive study of database partitioning strategies was conducted to optimise the performance of large-scale noise computations. Range, hash, and list partitioning were evaluated alongside space-tiling techniques in PostgreSQL, with their impact on storage organisation and query performance carefully assessed. In addition, the distributed architecture of Citus was exploited by deploying the model on a four-node cluster, enabling inter-node parallelism and

substantially improving computational performance. This combined approach allowed the execution of large-scale underwater noise models efficiently over millions of spatiotemporal records.

The thesis also conducted a detailed analysis of underwater noise across multiple frequency bands (63, 125, 400, and 4,000 Hz), offering insight into how different components of the acoustic spectrum propagate in the marine environment. The analytical potential of the proposed model was further demonstrated through several analyses, including the construction of bivariate maps to visualise noise propagation patterns, the assessment of environmental impacts associated with different vessel categories, the study of seasonal variations in noise pollution, and the evaluation of the acoustic effects of the COVID-19 lockdown. The thesis additionally developed interactive visual tools designed to support policymakers in interpreting and managing underwater noise.

The second part of the thesis contributed to the broader field of spatiotemporal databases by introducing a new data model for deforming moving regions. The proposed data type, the temporal circular buffer, captures circular moving regions whose position and radius vary continuously over time, providing an expressive framework for representing spatial phenomena with deforming extents. The abstract data type was formally defined together with the full set of functions and operators required to support its semantics, including methods for distance computation, turning point detection, and the evaluation of spatiotemporal relationships.

The temporal circular buffer was fully implemented within MobilityDB, along with the complete collection of analytical functions and topological operators necessary to manipulate, query, and analyse circular moving regions efficiently. This integration extends the expressive power of existing spatiotemporal database systems, enabling the direct management of deformable geometries that evolve continuously in space and time.

The applicability of the proposed model was demonstrated through a case study in which temporal circular buffers were used to represent spatially evolving phenomena in the maritime domain. This example illustrated how the new data type can be employed to model spatially evolving phenomena, highlighting its ability to represent processes whose impacts extend and deform over time.

The third contribution of the thesis extends the study of mobility to the urban domain through the analysis of a large and uniquely valuable ticket validation dataset from the public transport network of Venice. Collected within the framework of the iNEST project, this dataset offers a unique setting to examine mobility in a city whose exceptional morphology, reliance on waterborne transport, and exposure to strong seasonal and event-driven fluctuations set it apart

from conventional urban environments. Its richness — encompassing multiple user categories, diverse ticket types, three distinct seasons, and major events such as Carnival, Easter, and the Venice Film Festival — offers a remarkably detailed representation of both ordinary and exceptional mobility dynamics.

An extensive exploratory analysis was carried out to investigate temporal and spatial patterns of mobility across different user groups. Seasonal variations, daily behaviours during major events, and hourly validation distributions were examined to provide a comprehensive overview of how residents, workers, students, retirees, and tourists interact with the transport system throughout the year. Building on this exploratory analysis, in the thesis we reconstructed individual travel sequences from ticket validations and examined user trajectories. This enabled the identification of dwell times, morning departure and afternoon return patterns, and the most frequently used stops for different user categories, while also capturing multi-day mobility behaviour for tourists. These insights offer a nuanced understanding of how different populations navigate the city's transport network.

To uncover broader structural patterns, unsupervised clustering techniques were applied to group stops along two complementary dimensions: their predominant user categories and their temporal validation profiles. This dual perspective revealed meaningful spatial and behavioural structures within the network, highlighting how stops function differently depending on their location, usage patterns, and user categories. Finally, clustering results were compared across multiple time periods to assess the variability of stop usage patterns. By examining patterns across different time periods, we gain insight into how mobility demand evolves in response to seasonal changes, holiday periods, and major events, offering evidence that can support transport planning and mobility management in a uniquely complex and sensitive urban environment.

Directions for Future Work

There are several promising directions to extend and refine the work presented in this thesis.

- **Maritime Mobility and Underwater Noise Modelling**
  - **Scalability and distributed computation.** A first direction for future work concerns the scalability of the modelling pipeline. While the current implementation exploits a four-node Citus cluster, larger deployments would allow for a more systematic evaluation of how performance scales with increasing computational resources. Increasing the number of nodes would make it possible to assess limits, bottlenecks, and opportunities for further optimisation in large-scale spatiotemporal analytics.

- **Traffic impact assessment.** The model can be used to analyse the effects of alternative traffic management measures, including changes in vessel speed regulations, adjustments in fishing activity, or the introduction of traffic restrictions, in order to evaluate their influence on underwater noise and inform policy development.

- **Deforming Moving Regions**

  - **Extending analytical functionality.** Future work may focus on expanding the set of functions and operators defined for the temporal circular buffer. Examples include intersection operations between a temporal circular buffer and arbitrary static geometries, as well as the definition of *ever* and *always* predicates to determine, for instance, whether two temporal circular buffers ever coincide or remain equal throughout a time interval. Such extensions would further enhance the expressive and analytical power of the data type, broadening its applicability to a wider range of spatiotemporal queries.
  - **Generalising deforming geometries.** Building on the temporal circular buffer introduced in this thesis, future research may explore the support for more complex deforming geometries. Prior work has proposed approaches for the storage and manipulation of deforming polygons [46, 63], but it remains unclear how such models can be integrated into the MobilityDB type system. Extending the current framework to polygonal or multi-part deforming regions would significantly enhance the expressive power of spatiotemporal databases.
  - **Towards standardisation.** A further challenge arises from the absence of international standards describing deforming moving regions. As a result, the design requirements and interoperability considerations for such data types remain open research questions. Future work could contribute to the formalisation of such standards and investigate how deforming geometries could be consistently represented across different database systems and applications.

- **Urban Mobility Analysis in Venice**

  - **Analyses across multiple years.** Incorporating data from multiple consecutive years enables the study of long-term trends, including the effects of increasing tourism pressure, changes in residential mobility, and the impact of policy interventions.
  - **Integrating navigation data.** A promising direction for future research involves integrating vessel navigation data from ACTV routes

with ticket validation records. Moreover, if the ticket validation data also included tap-out information — potentially through the introduction of exit barriers — it would then become possible to reconstruct full origin–destination movements at the trip level. This, in turn, would enable the estimation of passenger flows along specific routes, connections, and interchange patterns, and would further support the analysis of load levels, service regularity, and network efficiency.

– **Predictive and behavioural modelling.** Machine learning models — such as clustering over time, anomaly detection for exceptional events, or demand forecasting — could be applied to better characterise mobility patterns and predict pressure on the transport network, particularly during high-tourism periods.

Bibliography

[1] David W. Adler. DB2 Spatial Extender - Spatial data within the RDBMS. In *Proceedings of the 27th International Conference on Very Large Data Bases*, VLDB '01, page 687–690, San Francisco, CA, USA, 2001. Morgan Kaufmann.

[2] Héctor Cogollos Adrián, Santiago Porras Alfonso, Bruno Baruque Zanon, Alessandra Raffaetà, and Filippo Zanatta. Discovery of Tourists' Movement Patterns in Venice from Public Transport Data. In *SAC '22: The 37th ACM/SIGAPP Symposium on Applied Computing, Virtual Event, April 25 - 29, 2022*, pages 564–568. ACM, 2022.

[3] Michael Ainslie. *Principles of Sonar Performance Modelling*. Springer Praxis Books. Springer, Berlin, Heidelberg, 2010.

[4] Md Mahbub Alam, Luis Torgo, and Albert Bifet. A Survey on Spatio-temporal Data Analytics Systems. *ACM Comput. Surv.*, 54(10s), 2022.

[5] Walid G. Aref and Ihab F. Ilyas. SP-GiST: An Extensible Database Index for Supporting Space Partitioning Trees. *Journal of Intelligent Information Systems*, 17:215–240, 2001.

[6] Mohamed Bakli, Mahmoud Sakr, and Taysir Hassan A. Soliman. Hadoop-Trajectory: a Hadoop spatiotemporal data processing extension. *Journal of Geographical Systems*, 21(2):211–235, 2019.

[7] Mohamed Bakli, Mahmoud Sakr, and Esteban Zimányi. Distributed moving object data management in MobilityDB. In *Proceedings of the 8th ACM SIGSPATIAL International Workshop on Analytics for Big Geospatial Data*, BigSpatial '19, New York, NY, USA, 2019. ACM.

[8] Mohamed Bakli, Mahmoud Sakr, Esteban Zimányi, Nils Dijk, and Marco Slot. Distributed MobilityDB: A Scalable Moving Object Database Management System. *ACM Transactions on Spatial Algorithms and Systems*, 11(2), 2025.

[9] Konstantina Bereta, Guohui Xiao, and Manolis Koubarakis. Ontop-spatial: Ontop of geospatial databases. *Journal of Web Semantics*, 58:100514, 2019.

[10] Vania Bogorny, Chiara Renso, Artur Ribeiro de Aquino, Fernando de Lucca Siqueira, and Luis Otavio Alvares. CONSTAnT - A Conceptual Data Model for Semantic Trajectories of Moving Objects. *Transactions in GIS*, 18(1):66–88, 2014.

[11] Bruno Brandoli, Alessandra Raffaetà, Marta Simeoni, Pedram Adibi, Fateha Khanam Bappee, Fabio Pranovi, Giulia Rovinelli, Elisabetta Russo, Claudio Silvestri, Amilcar Soares, and Stan Matwin. From multiple aspect trajectories to predictive analysis: a case study on fishing vessels in the Northern Adriatic sea. *GeoInformatica*, 26:551–579, 2022.

[12] J. E. Breeding and L. A. Pflug. Research Ambient Noise Directionality (RANDI) 3.1 Physics Description. Technical Report NRL/FS/7176–95-9628, Naval Research Laboratory, Stennis Space Center, MS, USA, 1996.

[13] Steffen Brüggemann, Katerina Bereta, Guohui Xiao, and Manolis Koubarakis. Ontology-Based Data Access for Maritime Security. In *Proceedings of the European Semantic Web Conference (ESWC 2016)*, pages 741–757. Springer, 2016.

[14] Francesca Cagnacci, Luigi Boitani, Roger A Powell, and Mark S Boyce. Challenges and opportunities of using GPS-based location data in animal ecology. *Philosophical transactions of the Royal Society of London. Series B, Biological sciences*, 365(1550):2155, 2010.

[15] Diego Calvanese, Benjamin Cogrel, Sarah Komla-Ebri, Roman Kontchakov, Davide Lanti, Martin Rezk, Mariano Rodriguez-Muro, and Guohui Xiao. Ontop: Answering SPARQL queries over relational databases. *Semantic Web*, 8(3):471 – 487, 2017.

[16] Oded Cats and Francesco Ferranti. Unravelling individual mobility temporal patterns using longitudinal smart card data. *Research in Transportation Business & Management*, 43:100816, 2022.

[17] Junyi Cheng, Xianfeng Zhang, Peng Luo, Jie Huang, and Jianfeng Huang. An unsupervised approach for semantic place annotation of trajectories based on the prior probability. *Information Sciences*, 607:1311–1327, 2022.

[18] Clément Chion, Dominic Lagrois, and Jérôme Dupras. A Meta-Analysis to Understand the Variability in Reported Source Levels of Noise Radiated by Ships From Opportunistic Studies. *Frontiers in Marine Science*, 6:714, 2019.

[19] Citus Data, Inc. Citus data. `https://www.citusdata.com/`. Accessed on November 2025.

[20] Christophe Claramunt, Cyril Ray, Elena Camossi, Anne Jousselme, Mirsad Hadzagic, Gennady L. Andrienko, Natalia V. Andrienko, Yannis Theodoridis, George A. Vouros, and Lionel Salmon. Maritime data integration and analysis: Recent progress and research challenges. In *Proceedings of the 20th International Conference on Extending Database Technology (EDBT 2017)*, pages 192–197, 2017.

[21] Pedro Contreras and Fionn Murtagh. Hierarchical clustering. In Christian Hennig, Marina Meila, Fionn Murtagh, and Roberto Rocci, editors, *Handbook of Cluster Analysis*, pages 103–120. Chapman and Hall/CRC, 1st edition, 2015.

[22] Couchbase Team. Couchbase: A distributed nosql document database. `https://www.couchbase.com/`, 2025. Accessed October 2025.

[23] Erica Cruz, Thomas Lloyd, Johan Bosschers, Frans Hendrik Lafeber, Pedro Vinagre, and Guilherme Vaz. Study on inventory of existing policy, research and impacts of continuous underwater noise in Europe. *EMSA report EMSA/NEG/21/2020. WavEC Offshore Renewables and Maritime Research Institute Netherlands*, 2021.

[24] Umur Cubukcu, Ozgun Erdogan, Sumedh Pathak, Sudhakar Sannakkayala, and Marco Slot. Citus: Distributed PostgreSQL for data-intensive applications. In *Proceedings of the 2021 International Conference on Management of Data*, SIGMOD '21, page 2490–2502, New York, NY, USA, 2021. ACM.

[25] Alex De Robertis and Christopher D Wilson. Walleye pollock respond to trawling vessels. *ICES Journal of Marine Science*, 63(3):514–522, 2006.

[26] Xin Ding, Lu Chen, Yunjun Gao, Christian Jensen, and Hujun Bao. UlTraMan: a unified platform for big trajectory data management and analytics. *Proceedings of the VLDB Endowment*, 11:787–799, 2018.

[27] Paolo Diviacco, Antonio Nadali, Massimiliano Iurcev, Mihai Burca, Rodrigo Carbajales, Matteo Gangale, Alessandro Busato, Fabio Brunetti, Lorenzo Grio, Alberto Viola, and Nikolas Potleca. Underwater Noise Monitoring with Real-Time and Low-Cost Systems, (The CORMA Experience). *Journal of Marine Science and Engineering*, 9(4):390, 2021.

[28] Renata Dividino, Anderson Soares, Stan Matwin, Andrew W. Isenor, Scott Webb, and Marc Brousseau. Semantic Integration of Real-Time Heterogeneous Data Streams for Ocean-Related Decision Making. In *Big Data and Artificial Intelligence for Military Decision Making*. NATO Science and Technology Organization (STO), 2018.

[29] Carlos M. Duarte, Lucille Chapuis, Shaun P. Collin, Daniel P. Costa, Reny P. Devassy, Victor M. Eguiluz, Christine Erbe, Timothy A. C. Gordon, Benjamin S. Halpern, Harry R. Harding, Michelle N. Havlik, Mark Meekan, Nathan D. Merchant, Jennifer L. Miksis-Olds, Miles Parsons, Milica Predragovic, Andrew N. Radford, Craig A. Radford, Stephen D. Simpson, Hans Slabbekoorn, Erica Staaterman, Ilse C. Van Opzeeland, Jana Winderen, Xiangliang Zhang, and Francis Juanes. The soundscape of the Anthropocene ocean. *Science*, 371(6529):eaba4658, 2021.

[30] José Duarte, Paulo Dias, and José Moreira. A framework for the management of deformable moving objects. In Ali Mansourian, Petter Pilesjö, Lars Harrie, and Ron van Lammeren, editors, *Geospatial Technologies for All*, pages 327–346, Cham, 2018. Springer.

[31] Christian Düntgen, Thomas Behr, and Ralf Hartmut Güting. BerlinMOD: a benchmark for moving object databases. *The VLDB Journal*, 18(6):1335–1368, 2009.

[32] Max J. Egenhofer and Robert D. Franzosa. Point-set topological spatial relations. *International Journal of Geographical Information Systems*, 5(2):161–174, 1991.

[33] Mohamed K. El Mahrsi, Etienne Côme, Latifa Oukhellou, and Michel Verleysen. Clustering Smart Card Data for Urban Mobility Analysis. *IEEE Transactions on Intelligent Transportation Systems*, 18(3):712–728, 2017.

[34] Christine Erbe, Alec Duncan, Lauren Hawkins, John M. Terhune, and Jeanette A. Thomas. *Introduction to Acoustic Terminology and Signal Processing*, pages 111–152. Springer, Cham, 2022.

[35] Christine Erbe, Alec Duncan, and Kathleen J. Vigness-Raposa. *Introduction to Sound Propagation Under Water*, pages 185–216. Springer, Cham, 2022.

[36] Christine Erbe, Alexander MacGillivray, and Rob Williams. Mapping cumulative noise from shipping to inform marine spatial planning. *The Journal of the Acoustical Society of America*, 132(5):EL423–EL428, 2012.

[37] Martin Erwig, Ralf Hartmut Güting, Markus Schneider, and Michalis Vazirgiannis. Spatio-Temporal Data Types: An Approach to Modeling and Querying Moving Objects in Databases. *GeoInformatica*, 3(3):269–296, 1999.

[38] Mohammad Etemad, Amílcar Soares Júnior, Arazoo Hoseyni, Jordan Rose, and Stan Matwin. A Trajectory Segmentation Algorithm Based on Interpolation-based Change Detection Strategies. In *Proceedings of the 2nd International Workshop on Big Mobility Data Analytics (BMDA 2019)*, volume 2322 of *CEUR Workshop Proceedings*, 2019.

[39] Paul C. Etter. *Underwater Acoustic Modeling and Simulation*. CRC Press, Boca Raton, 4th edition edition, 2013.

[40] European Union Interreg Italy–Croatia Programme. SOUNDSCAPE Project. `https://www.italy-croatia.eu/web/soundscape`, 2019-2021. Accessed on October, 2025.

[41] Yi Fang, Marc Friedman, Giri Nair, Michael Rys, and Ana-Elisa Schmid. Spatial indexing in microsoft SQL server 2008. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, SIGMOD '08, page 1207–1216, New York, NY, USA, 2008. ACM.

[42] Adrian Farcas, Claire F Powell, Kate L Brookes, and Nathan D Merchant. Validated shipping noise maps of the Northeast Atlantic. *Science of the Total Environment*, 735:139509, 2020.

[43] Adrian Farcas, Paul M Thompson, and Nathan D Merchant. Underwater noise modelling for environmental impact assessment. *Environmental Impact Assessment Review*, 57:114–122, 2016.

[44] Stefan Foell, Santi Phithakkitnukoon, Marco Veloso, Gerd Kortuem, and Carlos Bento. Regularity of Public Transport Usage: A Case Study of Bus Rides in Lisbon, Portugal. *Journal of Public Transportation*, 19(4):161–177, 2016.

[45] Thomas Folegot, Erwan Hemon, Georg Nehls, Mara Schmiing, Stefan Bräger, Mickaël Bellmann, Stephan Gerlach, Rainer Matuschek, and Judith Flamme. *Near Real-Time Underwater Sound Modeling of Dredging Noise to Meet Regulatory Noise Thresholds*, pages 1–19. Springer, Cham, 2024.

[46] Luca Forlizzi, Ralf Hartmut Güting, Enrico Nardelli, and Markus Schneider. A data model and data structures for moving objects databases. *SIGMOD Rec.*, 29(2):319–330, 2000.

[47] Robert E. Francois and George R. Garrison. Sound absorption based on ocean measurements: Part I: Pure water and magnesium sulfate contributions. *Journal of the Acoustical Society of America*, 72(3):896–907, 1982.

[48] Alessandro Furieri. Spatialite: An open-source spatial extension of sqlite. `https://www.gaia-gis.it/fossil/libspatialite/index`, 2020. Accessed October 2025.

[49] Michol Ghezzo, Antonio Petrizzo, Fantina Madricardo, Thomas Folegot, Roger Gallou, Dominique Clorennec, Robert Chavanne, Erwan Hemon, Christian Ferrarin, Hrvoje Mihanović, Kristina Pikelj, Mauro Bastianini, Alice Pari, Sauro Pari, Stefano Menegon, William J. McKiver, Giulio Farella, Sofia Bosi, Andrea Barbanti, and Marta Picciulin. Natural and shipping underwater sound distribution in the Northern Adriatic Sea basin and possible application on target areas. *Marine Pollution Bulletin*, 207:116852, August 2024.

[50] Li Gong, Xi Liu, Lun Wu, and Yu Liu. Inferring trip purposes and uncovering travel patterns from taxi trajectory data. *Cartography and Geographic Information Science*, 43(2):103–114, 2016.

[51] The PostGIS Global Development Group. Postgis 3.6.1 documentation. `https://postgis.net/stuff/postgis-3.6-en.pdf`, 2025. Accessed on October, 2025.

[52] The PostgreSQL Global Development Group. Postgresql 16.6 documentation. `https://www.postgresql.org/files/documentation/pdf/16/postgresql-16-A4.pdf`, 2024. Accessed on October, 2025.

[53] Ralf Hartmut Güting, Thomas Behr, Victor T. de Almeida, Zhiming Ding, Frank Hoffmann, and Markus Spiekermann. SECONDO: An Extensible DBMS Architecture and Prototype. Informatik-Report 313, FernUniversität in Hagen, 2004.

[54] Ralf Hartmut Güting, Thomas Behr, and Christian Düntgen. SECONDO: A Platform for Moving Objects Database Research and for Publishing and Integrating Research Implementations. *IEEE Data Engineering Bulletin*, 33:56–63, 2010.

[55] Ralf Hartmut Güting, Michael H. Böhlen, Martin Erwig, Christian S. Jensen, Nikos A. Lorentzos, Markus Schneider, and Michalis Vazirgiannis. A foundation for representing and querying moving objects. *ACM Trans. Database Syst.*, 25(1):1–42, 2000.

[56] Ralf Hartmut Güting and Jiamin Lu. Parallel SECONDO: scalable query processing in the cloud for non-standard applications. *SIGSPATIAL Special*, 6(2):3–10, 2015.

[57] Ralf Hartmut Güting and Markus Schneider. *Moving objects databases*. Academic Press, 2005.

[58] Jiawei Han, Jian Pei, and Hanghang Tong. *Data mining: concepts and techniques*. Morgan Kaufmann, 2022.

[59] Li He, Martin Trépanier, and Bruno Agard. Space–time classification of public transit smart card users' activity locations from smart card data. *Public Transport*, 13(3):579–595, 2021.

[60] Florian Heinz. Robust interpolation of arbitrary-dimensional moving regions in databases. In *Proceedings of the 3rd ACM SIGSPATIAL International Workshop on Searching and Mining Large Collections of Geospatial Data*, GeoSearch '24, page 13–21, New York, NY, USA, 2024. ACM.

[61] Florian Heinz and Ralf Hartmut Güting. Robust high-quality interpolation of regions to moving regions. *GeoInformatica*, 20(3):385–413, 2016.

[62] Florian Heinz and Ralf Hartmut Güting. A data model for moving regions of fixed shape in databases. *International Journal of Geographical Information Science*, 32(9):1737–1769, 2018.

[63] Florian Heinz and Ralf Hartmut Güting. A polyhedra-based model for moving regions in databases. *International Journal of Geographical Information Science*, 34:1–33, 2019.

[64] Florian Heinz and Johannes Schildgen. A Data Model and Operations for Higher-Dimensional Moving Objects in Databases. In *Proceedings of the 1st ACM SIGSPATIAL International Workshop on Methods for Enriched Mobility Data: Emerging Issues and Ethical Perspectives 2023*, EMODE '23, page 30–39, New York, NY, USA, 2023. ACM.

[65] Joseph M. Hellerstein, Jeffrey F. Naughton, and Avi Pfeffer. Generalized search trees for database systems. In *Proceedings of the 21th International Conference on Very Large Data Bases*, VLDB '95, page 562–573, San Francisco, CA, USA, 1995. Morgan Kaufmann.

[66] Jukka-Pekka Jalkanen, Lasse Johansson, Mathias H. Andersson, Elisa Majam aki, and Peter Sigray. Underwater noise emissions from ships during 2014-2020. *Environmental Pollution*, 311:119766, 2022.

[67] Erik Jenelius and Matej Cebecauer. Impacts of COVID-19 on public transport ridership in Sweden: Analysis of ticket validations, sales and passenger counts. *Transportation Research Interdisciplinary Perspectives*, 8:100242, 2020.

[68] Lasse Johansson, Jukka-Pekka Jalkanen, and Jaakko Kukkonen. Global assessment of shipping emissions in 2015 on a high spatial and temporal resolution. *Atmospheric Environment*, 167:403–415, 2017.

[69] Rihab Larayedh, Bruce D. Cornuelle, George Krokos, and Ibrahim Hoteit. Numerical investigation of shipping noise in the Red Sea. *Scientific Reports*, 14(1):5851, 2024.

[70] Inmook Lee, Shin-Hyung Cho, Kyoungtae Kim, Seung-Young Kho, and Dong-Kyu Kim. Travel pattern-based bus trip origin-destination estimation using smart card data. *Plos one*, 17(6):e0270346, 2022.

[71] Scott Leutenegger, Mario Lopez, and Jeffrey Edgington. STR: A Simple and Efficient Algorithm for R-Tree Packing. In *Proc. VLDB Conf*, pages 497–506, 1997.

[72] Ruiyuan Li, Huajun He, Rubin Wang, Sijie Ruan, Yuan Sui, Jie Bao, and Yu Zheng. TrajMesa: A Distributed NoSQL Storage Engine for Big Trajectory Data. In *2020 IEEE 36th International Conference on Data Engineering (ICDE)*, pages 2002–2005. IEEE, 2020.

[73] Lin Liao, Dieter Fox, and Henry Kautz. Location-based activity recognition using relational markov networks. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence*, IJCAI'05, page 773–778, San Francisco, CA, USA, 2005. Morgan Kaufmann Publishers Inc.

[74] A. MacGillivray, C. McPherson, G. McPherson, J. Izett, J. Gosselin, Z. Li, and D. Hannay. Modelling underwater shipping noise in the Great Barrier Reef Marine Park using AIS vessel track data. In *Proceedings of the 43rd International Congress on Noise Control Engineering (Inter.noise 2014)*, 2014.

[75] Alex MacGillivray and Cor de Jong. A Reference Spectrum Model for Estimating Source Levels of Marine Shipping Based on Automated Identification System Data. *Journal of Marine Science and Engineering*, 9(4):369, 2021.

[76] Renzo Massobrio and Sergio Nesmachnow. Urban Mobility Data Analysis for Public Transportation Systems: A Case Study in Montevideo, Uruguay. *Applied Sciences*, 10(16), 2020.

[77] Mark McKenney, Niharika Nyalakonda, Jarrod McEvers, and Mitchell Shipton. Pyspatiotemporalgeom: a python library for spatiotemporal types and operations. In *Proceedings of the 24th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, SIGSPATIAL '16, New York, NY, USA, 2016. ACM.

[78] Mark McKenney and James Webb. Extracting moving regions from spatial data. In *Proceedings of the 18th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 438–441. ACM, 2010.

[79] Ronaldo dos Santos Mello, Vania Bogorny, Luis Otavio Alvares, Luiz Henrique Zambom Santana, Carlos Andres Ferrero, Angelo Augusto Frozza, Geomar Andre Schreiner, and Chiara Renso. MASTER: A multiple aspect view on trajectories. *Transactions in GIS*, 23(4):805–822, 2019.

[80] Nirvana Meratnia and Rolf A. de By. Spatiotemporal compression techniques for moving point objects. In Elisa Bertino, Stavros Christodoulakis, Dimitris Plexousakis, Vassilis Christophides, Manolis Koubarakis, Klemens Böhm, and Elena Ferrari, editors, *Advances in Database Technology - EDBT 2004*, pages 765–782, Berlin, Heidelberg, 2004. Springer.

[81] Claudio Milano, Joseph M Cheer, and Marina Novelli. *Overtourism: Excesses, discontents and measures in travel and tourism*. CABI, 2019.

[82] RB Mitson. Underwater noise radiated by research vessels. In *ICES Marine Science Symposium*, volume 196, pages 147–152, 1993.

[83] MongoDB Inc. Mongodb geospatial queries. `https://docs.mongodb.com/manual/geospatial-queries/`, 2025. Accessed November 2025.

[84] Kate Moran, Brigitte Boutin, S. Kim Juniper, Benoît Pirenne, and Adrian Round. A multi-use and multi-stakeholder ocean observing platform system. In *OCEANS 2019 MTS/IEEE SEATTLE*, pages 1–5, 2019.

[85] João Moreira, Carlos Rodrigues, and Ralf Hartmut Güting. Modeling and Querying Moving Regions in Spatial Databases. *GeoInformatica*, 23(1):1–36, 2019.

[86] MySQL. Mysql 8.0 reference manual. `https://dev.mysql.com/doc/refman/8.0/en/`, 2020. Accessed November 2025.

[87] Sarah TV Neenan, Paul R White, Timothy G Leighton, and Peter J Shaw. Modeling vessel noise emissions through the accumulation and propagation of

Automatic Identification System data. *Proceedings of Meetings on Acoustics*, 27(1), 2016.

[88] Neo4j Inc. Neo4j: Graphs for everyone. `https://neo4j.com/`, 2025. Accessed November 2025.

[89] Jan Nidzwetzki and Ralf Hartmut Güting. Distributed SECONDO: A Highly Available and Scalable System for Spatial Data Processing. In *Advances in Spatial and Temporal Databases*, SSTD 2015, pages 491–496, Cham, 2015. Springer.

[90] Tiago CA Oliveira, Ying-Tsong Lin, and Michael B Porter. Underwater sound propagation modeling in a complex shallow water environment. *Frontiers in Marine Science*, 8:751327, 2021.

[91] Open Geospatial Consortium. OGC Implementation Standard for Geographic Information – Simple Feature Access – Part 1: Common Architecture. Technical Report OGC 06-103r4, Open Geospatial Consortium, 2011.

[92] Open Geospatial Consortium. OGC Implementation Standard for Geographic Information – Simple Feature Access – Part 2: SQL Option. Technical Report OGC 06-104r4, Open Geospatial Consortium, 2011.

[93] Oracle Corporation. Sql for oracle nosql database. `https://docs.oracle.com/en/database/other-databases/nosql-database/25.1/sqlfornosql/introduction-sql.html`, 2025. Accessed November 2025.

[94] Oracle Spatial Team. Oracle spatial and graph features. `https://www.oracle.com/database/technologies/spatialandgraph.html`, 2020. Accessed October 2025.

[95] Christine Parent, Stefano Spaccapietra, Chiara Renso, Gennady Andrienko, Natalia Andrienko, Vania Bogorny, Maria Luisa Damiani, Aris Gkoulalas-Divanis, José Antônio Fernandes de Macêdo, Nikos Pelekis, Yannis Theodoridis, and Zhixian Yan. Semantic trajectories modeling and analysis. *ACM Computing Surveys (CSUR)*, 45(4):1–32, 2013.

[96] Nikos Pelekis, Elias Frentzos, Nikos Giatrakos, and Yannis Theodoridis. HERMES: aggregative LBS via a trajectory DB engine. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, SIGMOD '08, page 1255–1258, New York, NY, USA, 2008. ACM.

[97] Nikos Pelekis, Elias Frentzos, Nikos Giatrakos, and Yannis Theodoridis. HERMES: A Trajectory DB Engine for Mobility-Centric Applications. *Int. J. Knowledge-Based Organ.*, 5(2):19–41, 2015.

[98] Nikos Pelekis and Yannis Theodoridis. *Mobility Data Management and Exploration*. Computer Science, Computer Science (R0). Springer, New York, NY, 1 edition, 2014.

[99] Nikos Pelekis, Yannis Theodoridis, Spyros Vosinakis, and Themis Panayiotopoulos. Hermes – A Framework for Location-Based Data Management. In *Proceedings of the VLDB Endowment*, volume 3896, pages 1130–1134, 2006.

[100] Antonio Petrizzo, Andrea Barbanti, Giulia Barfucci, Mauro Bastianini, Ilaria Biagiotti, Sofia Bosi, Michele Centurelli, Robert Chavanne, Antonio Codarin, Ilaria Costantini, Marinela Cukrov Car, Vlado Dadić, Francesco M. Falcieri, Raffaela Falkner, Giulio Farella, Mario Felli, Christian Ferrarin, Thomas Folegot, Roger Gallou, Daphnie Galvez, Michol Ghezzo, Aleksandra Kruss, Iole Leonori, Stefano Menegon, Hrvoje Mihanović, Stipe Muslim, Alice Pari, Sauro Pari, Marta Picciulin, Grgur Pleslić, Marko Radulović, Nikolina Rako-Gospić, Davide Sabbatini, Giulia Soldano, Jarosław Tęgowski, Tihana Vučur-Blazinić, Predrag Vukadin, Jakub Zdroik, and Fantina Madricardo. First assessment of underwater sound levels in the Northern Adriatic Sea at the basin scale. *Scientific Data*, 10(1):137, 2023.

[101] Antonio Petrizzo, Giulia Barfucci, Mauro Bastianini, Michele Centurelli, Antonio Codarin, Marinela Cukrov Car, Vlado Dadić, Raffael Falkner, Michol Ghezzo, Iole Leonori, Hrvoj Mihanović, Stipe Muslim, Marta Picciulin, Marko Radulović, Nikolina Rako-Gospić, Davide Sabbatini, Tihana Vučur-Blazinić, Predrag Vukadin, Jakub Zdroik, and Fantina Madricardo. SOUNDSCAPE North Adriatic Underwater Noise Sound Pressure Levels. `https://zenodo.org/records/7472152`. Accessed on October, 2025.

[102] Lucas May Petry, Amilcar Soares, Vania Bogorny, Bruno Brandoli, and Stan Matwin. Challenges in Vessel Behavior and Anomaly Detection: From Classical Machine Learning to Deep Learning. In *Advances in Artificial Intelligence – 33rd Canadian Conference on Artificial Intelligence (Canadian AI 2020)*, volume 12109 of *Lecture Notes in Computer Science*, pages 401–407. Springer, 2020.

[103] Dieter Pfoser, Christian S. Jensen, and Yannis Theodoridis. Novel Approaches in Query Processing for Moving Object Trajectories. In *Proceedings of the 26th International Conference on Very Large Data Bases*, VLDB '00,

page 395–406, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc.

[104] Dieter Pfoser and Yannis Theodoridis. Generating semantics-based trajectories of moving objects. *Computers, Environment and Urban Systems*, 27(3):243–263, 2003. Emerging Technologies for Geo-Based Applications.

[105] Marta Picciulin, Chiara Facca, Riccardo Fiorin, Federico Riccato, Matteo Zucchetta, and Stefano Malavasi. It Is Not Just a Matter of Noise: Sciaena umbra Vocalizes More in the Busiest Areas of the Venice Tidal Inlets. *Journal of Marine Science and Engineering*, 9(2):237, 2021.

[106] Marta Picciulin, Antonio Petrizzo, Fantina Madricardo, Andrea Barbanti, Mauro Bastianini, Ilaria Biagiotti, Sofia Bosi, Michele Centurelli, Antonio Codarin, Ilaria Costantini, Vlado Dadić, Raffaela Falkner, Thomas Folegot, Daphnie Galvez, Iole Leonori, Stefano Menegon, Hrvoje Mihanović, Stipe Muslim, Alice Pari, Sauro Pari, Grgur Pleslić, Marko Radulović, Nikolina Rako-Gospić, Davide Sabbatini, Jaroslaw Tegowski, Predrag Vukadin, and Michol Ghezzo. First basin scale spatial–temporal characterization of underwater sound in the Mediterranean Sea. *Scientific Reports*, 13(1):22799, 2023.

[107] A. Prasad Sistla, Ouri Wolfson, Sam Chamberlain, and Son Dao. *Querying the uncertain position of moving objects*, pages 310–337. Springer, Berlin, Heidelberg, 1998.

[108] RedisLabs. Redis: An in-memory database that persists on disk. `https://redis.io/`, 2025. Accessed November 2025.

[109] Chiara Renso, Stefano Spaccapietra, and Esteban Zimányi. *Mobility Data: Modeling, Management, and Understanding*. Cambridge University Press, USA, 2013.

[110] Tiago F. R. Ribeiro, F. J. M. Silva, and Rui L. C. Costa. Modelling Forest Fire Dynamics Using Conditional Variational Autoencoders. *Information Systems Frontiers*, 2024. Online first.

[111] W John Richardson, Charles R Greene Jr, Charles I Malme, and Denis H Thomson. *Marine Mammals and Noise*. Academic Press, London, UK, 1995.

[112] Peter H. Rogers and Mardi Cox. Underwater sound as a biological stimulus. In Jelle Atema, Richard R. Fay, Arthur N. Popper, and William N. Tavolga, editors, *Sensory Biology of Aquatic Animals*, pages 131–149, New York, NY, 1988. Springer.

[113] Giulia Rovinelli. A Spatiotemporal Framework for Underwater Noise Modelling. In *Proceedings of the 19th International Symposium on Spatial and Temporal Data*, SSTD '25, page 132–136, New York, NY, USA, 2025. ACM.

[114] Giulia Rovinelli, Stan Matwin, Fabio Pranovi, Elisabetta Russo, Claudio Silvestri, Marta Simeoni, and Alessandra Raffaetà. Multiple aspect trajectories: a case study on fishing vessels in the Northern Adriatic sea. In *Proceedings of the 4th International Workshop on Big Mobility Data Analytics (BMDA 2021) - EDBT/ICDT Workshops*, volume 2841, 2021.

[115] Giulia Rovinelli, Davide Rocchesso, Marta Simeoni, and Alessandra Raffaetà. Using semantic trajectories for spatio-temporal characterisation of underwater noise. In *Proceedings of the 6th International Workshop on Big Mobility Data Analytics (BMDA 2024) - EDBT/ICDT Workshops*, volume 3651, 2024.

[116] Giulia Rovinelli, Davide Rocchesso, Marta Simeoni, Esteban Zimányi, and Alessandra Raffaetà. Spatiotemporal characterisation of underwater noise through semantic trajectories. *Geoinformatica*, 29:845–876, 2025.

[117] Giulia Rovinelli, Esteban Zimányi, Marta Simeoni, Davide Rocchesso, and Alessandra Raffaetà. A Scalable Model for Vessel-Generated Underwater Noise: Enhancing Efficiency through Parallelisation. In *Proceedings of the 7th International Workshop on Big Mobility Data Analytics (BMDA 2025) - EDBT/ICDT Workshops*, 2025.

[118] Elisabetta Russo, Marco Anelli Monti, Giacomo Toninato, Claudio Silvestri, Alessandra Raffaetà, and Fabio Pranovi. Lockdown: How the COVID-19 pandemic affected the fishing activities in the Adriatic Sea (Central Mediterranean Sea). *Frontiers in Marine Science*, 8:685808, 2021.

[119] Mahmoud Sakr and Ralf Hartmut Güting. Spatiotemporal Pattern Queries in SECONDO. In Nikos Mamoulis, Thomas Seidl, Torben Bach Pedersen, Kristian Torp, and Ira Assent, editors, *Advances in Spatial and Temporal Databases*, pages 422–426, Berlin, Heidelberg, 2009. Springer.

[120] Mahmoud Sakr, Alejandro Vaisman, and Esteban Zimányi. *Mobility Data Science*. Data-Centric Systems and Applications. Springer, Cham, 1 edition, 2025.

[121] Giacomo-Maria Salerno and Antonio Paolo Russo. Venice as a short-term city. Between global trends and local lock-ins. *Journal of Sustainable Tourism*, 30(5):1040–1059, 2022.

[122] Georgios M. Santipantakis, Apostolos Glenis, Kostas Patroumpas, Akrivi Vlachou, Christos Doulkeridis, George A. Vouros, Nikos Pelekis, and Yannis Theodoridis. SPARTAN: Semantic integration of big spatio-temporal data from streaming and archival sources. *Future Generation Computer Systems*, 110:540–555, 2020.

[123] Quiet Ocean SAS. Quonops online services. `http://qos.quiet-oceans.com/`. Accessed on October, 2025.

[124] Maxime Schoemans, Mahmoud Sakr, and Esteban Zimányi. Implementing Rigid Temporal Geometries in Moving Object Databases. In *2021 IEEE 37th International Conference on Data Engineering (ICDE)*, pages 2547–2558. IEEE, 2021.

[125] Laughlin Siceloff and W. Huntting Howell. Fine-scale temporal and spatial distributions of Atlantic cod (Gadus morhua) on a western Gulf of Maine spawning ground. *Fisheries Research*, 141:31–43, 2013. Reconciling Spatial Scales and Stock Structures.

[126] Hans Slabbekoorn, Niels Bouton, Ilse van Opzeeland, Aukje Coers, Carel ten Cate, and Arthur N Popper. A noisy spring: the impact of globally rising underwater sound levels on fish. *Trends in ecology & evolution*, 25(7):419–427, 2010.

[127] Tom Alexander Smith and Margarita Kourounioti. Underwater radiated noise from small vessels in shallow water: propagation modelling and experimental measurements. *Conference Proceedings of International Naval Engineering Conference (INEC)*, 2023.

[128] Anderson Soares, Renata Dividino, Fernando Abreu, Marc Brousseau, Andrew W. Isenor, Scott Webb, and Stan Matwin. CRISIS: Integrating AIS and Ocean Data Streams Using Semantic Web Standards for Event Detection. In *Proceedings of the International Conference on Military Communications and Information Systems (ICMCIS 2019)*, 2019.

[129] Stefano Spaccapietra and Christine Parent. Adding meaning to your steps (keynote paper). In Manfred Jeusfeld, Lois Delcambre, and Tok-Wang Ling, editors, *Conceptual Modeling – ER 2011*, pages 13–31, Berlin, Heidelberg, 2011. Springer.

[130] Stefano Spaccapietra, Christine Parent, Maria Luisa Damiani, Jose Antonio de Macedo, Fabio Porto, and Christelle Vangenot. A conceptual view on trajectories. *Data & Knowledge Engineering*, 65(1):126–146, 2008.

[131] Laura Spinsanti, Fabrizio Celli, and Chiara Renso. Where you stop is who you are: understanding people's activities by places visited. In *The proceedings of Behaviour Monitoring and Interpretation (BMI) workshop*, 2010.

[132] Knut Stolze. SQL/MM Spatial: The Standard to Manage Spatial Data in Relational Database Systems. In Gerhard Weikum, Harald Schöning, and Erhard Rahm, editors, *BTW 2003 – Datenbanksysteme für Business, Technologie und Web*, pages 247–264, Bonn, 2003. Gesellschaft für Informatik e.V.

[133] The Apache Software Foundation. Apache cassandra: Manage massive amounts of data, fast, without losing sleep. `https://cassandra.apache.org/`, 2025. Accessed November 2025.

[134] Erlend Tøssebro and Ralf Hartmut Güting. Creating Representations for Continuously Moving Regions from Observations. In *Advances in Spatial and Temporal Databases (SSTD 2001), Lecture Notes in Computer Science, vol. 2121*, pages 321–344, Berlin, Heidelberg, 2001. Springer.

[135] Robert J. Urick. Principles of underwater sound 3rd edition. *Peninsula Publising Los Atlos, California*, 22:23–24, 1983.

[136] Nienke C.F. van Geel, Denise Risch, and Anja Wittich. A brief overview of current approaches for underwater sound analysis and reporting. *Marine Pollution Bulletin*, 178:113610, 2022.

[137] Michalis Vazirgiannis and Ouri Wolfson. A spatiotemporal model and language for moving objects on road networks. In Christian S. Jensen, Markus Schneider, Bernhard Seeger, and Vassilis J. Tsotras, editors, *Advances in Spatial and Temporal Databases*, pages 20–35, Berlin, Heidelberg, 2001.

[138] Alexis Viallard, Martin Trépanier, and Catherine Morency. Assessing the Evolution of Transit User Behavior from Smart Card Data. *Transportation Research Record*, 2673(4):184–194, 2019.

[139] Nguyen Xuan Vinh, Julien Epps, and James Bailey. Information theoretic measures for clusterings comparison: is a correction for chance necessary? In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, page 1073–1080, New York, NY, USA, 2009. ACM.

[140] Nguyen Xuan Vinh, Julien Epps, and James Bailey. Information Theoretic Measures for Clusterings Comparison: Variants, Properties, Normalization and Correction for Chance. *Journal of Machine Learning Research*, 11:2837–2854, 2010.

[141] Yuanqiao Wen, Yimeng Zhang, Liang Huang, Chunhui Zhou, Changshi Xiao, Fan Zhang, Xin Peng, Wenqiang Zhan, and Zhongyi Sui. Semantic Modelling of Ship Behavior in Harbor Based on Ontology and Dynamic Bayesian Network. *ISPRS International Journal of Geo-Information*, 8(3), 2019.

[142] R. Williams, A.J. Wright, E. Ashe, L.K. Blight, R. Bruintjes, R. Canessa, C.W. Clark, S. Cullis-Suzuki, D.T. Dakin, C. Erbe, P.S. Hammond, N.D. Merchant, P.D. O'Hara, J. Purser, A.N. Radford, S.D. Simpson, L. Thomas, and M.A. Wale. Impacts of anthropogenic noise on marine life: Publication patterns, new discoveries, and future directions in research and management. *Ocean & Coastal Management*, 115:17–24, 2015.

[143] Dietrich Wittekind. A Simple Model for the Underwater Noise Source Level of Ships. *Journal of Ship Production and Design*, 30:7–14, 2014.

[144] Ouri Wolfson, Sam Chamberlain, Son Dao, Liqin Jiang, and Gisela Mendez. Cost and Imprecision in Modeling the Position of Moving Objects. In *Proceedings 14th International Conference on Data Engineering*, pages 588–596, 1998.

[145] Ouri Wolfson, A. Prasad Sistla, Steven Chamberlain, and Yelena Yesha. Updating and Querying Databases that Track Mobile Units. *Distributed and Parallel Databases*, 7(3):257–387, 1999.

[146] Ouri Wolfson, Bo Xu, S. Chamberlain, and L. Jiang. Moving objects databases: Issues and solutions. In *Proceedings on the Tenth International Conference on Scientific and Statistical Database Management*, pages 111–122, 1998.

[147] Kexin Xie, Ke Deng, and Xiaofang Zhou. From trajectories to activities: a spatio-temporal join approach. In *Proceedings of the 2009 International Workshop on Location Based Social Networks*, LBSN '09, page 25–32, New York, NY, USA, 2009. ACM.

[148] Zhixian Yan, Dipanjan Chakraborty, Christine Parent, Stefano Spaccapietra, and Karl Aberer. Semitri: a framework for semantic annotation of heterogeneous trajectories. In *Proceedings of the 14th International Conference on Extending Database Technology*, EDBT/ICDT '11, page 259–270, New York, NY, USA, 2011. ACM.

[149] Zhigang Zhang, Cheqing Jin, Jiali Mao, Xiaolin Yang, and Aoying Zhou. Trajspark: A scalable and efficient in-memory management system for big trajectory data. In *Web and Big Data - 1st International Joint Conference, APWeb-WAIM 2017, Proceedings*, pages 11–26. Springer, 2017.

[150] Yu Zheng, Yukun Chen, Quannan Li, Xing Xie, and Wei-Ying Ma. Understanding transportation modes based on GPS data for web applications. *ACM Trans. Web*, 4(1), 2010.

[151] Esteban Zimányi. MobilityDB 1.3 User's Manual. `https://mobilitydb.com/documentation.html`, 2025. Accessed on November, 2025.

[152] Esteban Zimányi, Mahmoud Sakr, and Arthur Lesuisse. MobilityDB: A Mobility Database Based on PostgreSQL and PostGIS. *ACM Trans. Database Syst.*, 45(4), 2020.

[153] Esteban Zimányi, Mahmoud Sakr, Arthur Lesuisse, and Mohamed Bakli. MobilityDB: A Mainstream Moving Object Database System. In *Proceedings of the 16th International Symposium on Spatial and Temporal Databases*, SSTD '19, page 206–209, New York, NY, USA, 2019. ACM.