UNIVERSITY
INSTITUTE
OF LISBON

A Spatiotemporal Crowding Visualization Platform

Rodrigo José Bravo Simões

Master in Computer Science

Supervisor:
Doctor Fernando Brito e Abreu, Associate Professor,
Iscte-IUL

Co-supervisor:
Doctor Adriano Lopes, Assistant Professor,
Iscte-IUL

October, 2024

[This page has been intentionally left blank]

TECHNOLOGY
AND ARCHITECTURE

Department of Information Science and Technology

A Spatiotemporal Crowding Visualization Platform

Rodrigo José Bravo Simões

Master in Computer Science

Supervisor:
Doctor Fernando Brito e Abreu, Associate Professor,
Iscte-IUL

Co-supervisor:
Doctor Adriano Lopes, Assistant Professor,
Iscte-IUL

October, 2024

[This page has been intentionally left blank]

A Spatiotemporal
Crowding Visualization Platform

[This page has been intentionally left blank]

Acknowledgements

I would like to express my gratitude to my supervisors, Professors Fernando Brito e Abreu and Adriano Lopes, for their unwavering support and encouragement, which inspired me to give my best in my work. I am also grateful to my friends for the countless enjoyable moments we shared over drinks and coffee, bringing so much joy to this journey. Lastly, I extend heartfelt thanks to my family for their steadfast support in so many ways.

Lisbon, October, 2024
Rodrigo José Bravo Simões

[This page has been intentionally left blank]

Abstract

Overtourism degrades the tourist experience and negatively impacts the environment and local communities, potentially making tourism in popular destinations unsustainable.

To help destination managers plan crowding mitigation measures, we have developed a spatiotemporal 3D crowding visualization platform. It allows users to explore historical crowding data, visualize patterns, and understand trends. It also allows viewing of near real-time data to support short-term decision making. In addition, it can generate and visualize predictive crowding data to better understand future scenarios.

The platform has an extensible, connector-based architecture that supports independent crowding time-series data sources, with either online streaming or offline processing, from databases, APIs, and queuing systems. The data can come from sensors (e.g., motion sensors, Wi-Fi sensors), video analytics, mobile operator data, and other sources. The platform's forecasting capabilities also follow an extensible architecture, allowing new predictive models to be incorporated without modifying existing source code.

Two illustrative case studies are presented, one based on data from the Melbourne pedestrian counting system in Australia, and another based on mobile network data in Lisbon, Portugal. Evaluations of the developed platform were performed in four domains: usability, performance, predictive ability, and peer-review.

Keywords: Tourism overcrowding; spatiotemporal 3D visualization; tourism sustainability; smart tourism; smart cities; time-series forecasting.

[This page has been intentionally left blank]

Resumo

O excesso de turismo degrada a experiência do turista e tem um impacto negativo no ambiente e nas comunidades locais, tornando potencialmente insustentável o turismo em destinos populares.

Para ajudar os gestores de destinos a planear medidas de atenuação de multidões, desenvolvemos uma plataforma de visualização de apinhamento 3D espaciotemporal. Esta plataforma permite aos utilizadores explorar dados históricos de aglomeração, visualizar padrões e compreender tendências. Também permite a visualização de dados quase em tempo real para apoiar a tomada de decisões a curto prazo. Além disso, pode gerar e visualizar dados preditivos de aglomeração para melhor compreender cenários futuros.

A plataforma tem uma arquitetura extensível, baseada em conectores, que suporta fontes de dados independentes de séries temporais de apinhamento, com streaming online ou processamento offline, a partir de bases de dados, APIs e sistemas de filas. Os dados podem provir de sensores (por exemplo, sensores de movimento, sensores Wi-Fi), análise de vídeo, dados de operadores móveis e outras fontes. As capacidades de previsão da plataforma também seguem uma arquitetura extensível, permitindo a incorporação de novos modelos de previsão sem modificar o código-fonte existente.

São apresentados dois casos de estudo ilustrativos, um baseado em dados do sistema de contagem de peões de Melbourne, na Austrália, e outro baseado em dados de redes móveis em Lisboa, Portugal. As avaliações da plataforma desenvolvida foram efectuadas em quatro domínios: usabilidade, desempenho, capacidade de previsão e revisão por pares.

Palavras-chave: Sobrelotação turística; visualização 3D espácio-temporal; sustentabilidade do turismo; turismo inteligente; cidades inteligentes; previsão de séries temporais.

[This page has been intentionally left blank]

Contents

[This page has been intentionally left blank]

List of Figures

[This page has been intentionally left blank]

List of Tables

[This page has been intentionally left blank]

Glossary

ArcGIS Pro	A commercial GIS platform developed by Esri. More information here.
bcrypt	Library that implements a secure hashing algorithm. More information here.
Chart.js	Library for interactive graphs and charts that works on top of React. More information here.
CSV	Comma-Separated Values. File format for tabular data.
CUDA	Platform for general computing on a Graphics Processing Unit (GPU). More information here.
deck.gl	Visualization library that works on top of a mapping library such as MapLibre. More information can be found here.
Docker	Containerization technology. More information available here.
Flask	Python framework for defining a REST server. More information available here.
Flask-Limiter	Library that works on top of Flask for rate limiting endpoints. More information here.
flask-login	Library that works on top of Flask for securing endpoints through authorization mechanisms. More information here.
Flux	Functional query language designed for working with time series data. Supports operations like aggregations, joins, and custom functions directly within the language. It is used across the InfluxDB ecosystem and integrates with visualization tools like Grafana to enhance time-series analysis and monitoring. More information available here.
GeoJSON	File format based on JSON, used for defining geometric entities based on geographic coordinates. More information here.
GraphHopper	Tool based on geographic data with services such as route planning, navigation and traffic aware route optimization. More information here.
HTTP	HyperText Transfer Protocol. Network protocol used in web applications and web services.

InfluxDB	Time-series database. More information available here.
JSON	JavaScript Object Notation (JSON). File format for defining structured data. More information here.
KDE	Kernel Density Estimation. KDE is the application of kernel (window function) smoothing for probability density estimation, i.e., a non-parametric method to estimate the probability density function of a random variable based on kernels as weights. KDE answers a fundamental data smoothing problem where inferences about the population are made based on a finite data sample..
LxDataLab	An initiative by the municipality of Lisbon with several challenges aiming to improve the city's management. More information here.
MapLibre	Mapping library. More information here.
MariaDB	A Relational Database Management System (RDBMS). More information here.
Material UI	Library that implements User Interface (UI) widgets, making use of React. It is used across InfluxDB's ecosystem and integrates with visualization tools like Grafana to enhance time-series analysis and monitoring. More information here.
Memcached	In-memory database. More information here.
MongoDB	Document-oriented database. More information here.
nginx	HTTP server. More information here.
Opendatasoft	An API platform for accessing various datasets. More information here.
OpenStreetMap	Abbreviated as OSM, it is an open geographic database containing mapping information for the whole planet. More information here.
Osmium Tool	Tool containing several utilities for processing OpenStreetMap data. More information here.
Parsimonious	Python library for parsing text based on a grammar definition. More information here.
PostGIS	Plugin for the PostgreSQL relational database that extends it by adding support for storing, indexing, and querying geospatial data. More information here.

PyTorch	Python library for implementing artificial neural networks. More information here.
QGIS	An open-source GIS platform. More information here.
React	Javascript library used for implementing dynamic web frontends. More information available here.
REST	Representational State Transfer. Architectural style for defining web services, making use of HTTP.
Scopus	Database of peer-reviewed literature, available here.
statsmodels	Python library with various statistical functionalities. More information here.
uWSGI	Server that implements the Web Server Gateway Interface (WSGI) protocol. More information here.
WebGL	Web Graphics Library. Library provided by web browsers for graphics rendering, making use of a computer's Graphics Processing Unit (GPU).
YAML	YAML Ain't Markup Language. File format for defining structured data, it is an alternative to JSON that can be more suitable for configuration files. More information here.

[This page has been intentionally left blank]

Acronyms

ABM	Agent-Based Modeling.
API	Application Programming Interface.
CSV	Comma-Separated Values.
DSR	Design Science Research.
GIS	Geographic Information System.
GPU	Graphics Processing Unit.
HTTP	HyperText Transfer Protocol.
JSON	JavaScript Object Notation.
KDE	Kernel Density Estimation.
MAE	Mean Absolute Error.
NASA-TLX	NASA-Task Load Index.
OSM	OpenStreetMap.
PEG	Parsing Expression Grammar.
RDBMS	Relational Database Management System.
RESETTING	Relaunching European smart and SustainablE Tourism models Through digitalization and INnovative technoloGies.
REST	Representational State Transfer.
RMSE	Root Mean Squared Error.
RNN	Recurrent Neural Network.
RQs	Research Questions.

SME	Small and Medium-sized Enterprise.
SUS	System Usability Scale.
TCP	Transmission Control Protocol.
TLS	Transport Layer Security.
UEQ	User Experience Questionnaire.
UI	User Interface.
UML	Unified Modeling Language.
WebGL	Web Graphics Library.
WMAPE	Weighted Mean Absolute Percentage Error.
WSGI	Web Server Gateway Interface.
WYD	World Youth Day.
YAML	YAML Ain't Markup Language.

Chapter 1

Introduction

This chapter outlines the context and motivation behind this dissertation, highlighting the key problems it addresses and introducing the artifact that was built to solve them. It also presents the research questions and methodology, along with the overall structure of the dissertation.

This chapter is organized as follows: section 1.1 outlines the motivation behind this dissertation. Section 1.2 lists different scenarios in which the platform presented in this dissertation can be deployed. Section 1.3 presents the research questions that will be answered in the last chapter. Section 1.4 describes the research methodology employed while also mapping the steps of the methodology to the relevant chapters and sections of this dissertation. Finally, section 1.5 summarizes the following chapters of this dissertation.

[This page has been intentionally left blank]

Chapter 1
Introduction

1.1 Motivation

Overtourism is described as *"the situation in which the impact of tourism, at certain times and in certain places, exceeds physical, ecological, social, economic, psychological and/or political capacity thresholds"* [34], i.e., when the resources of a tourist destination are being depleted. Overtourism can displace locals, strain infrastructure, harm the environment, and disrupt communities while degrading the tourist experience and eroding cultural heritage [2]. Moreover, tourism in popular destinations can become unsustainable, giving tourism-related businesses a vested interest in reducing overtourism and its negative impacts.

Smart tourism solutions have been long identified to scaffold the work of destination managers in identifying areas of concern and implementing measures to mitigate the negative impacts of tourism while promoting more sustainable tourism practices [16]. In the scope of the RESETTING project, funded by the European COSME Programme, several smart tourism tools that deal with overcrowding were developed, namely for supporting simulation-based participatory discussions among tourism destination stakeholders [41]. In the same scope, Wi-Fi activity sensors were developed to enable local monitoring of tourism crowding in near real-time data, allowing short-term decision-making, such as reinforcing or freeing up human and technical resources assigned to managing tourist crowding on the ground [32]. An overview of the overall strategy of building a smart tourism toolset to mitigate overcrowding can be found in [9].

This dissertation presents one such smart tourism tool, developed under the same project, for spatiotemporal crowding visualization. It allows users to explore historical and live crowding data, enabling the visualization of past trends and real-time crowd conditions. Additionally, the platform generates and displays predictive data, using recent data to forecast future crowding levels. This functionality supports medium- and short-term decision-making. For instance, in tourism, the tool can help identify less crowded routes or attractions and suggest when to avoid busy locations. It is also helpful for organizing large public events, such as music festivals or fireworks shows. Event organizers can leverage historical data for planning logistics like urban cleaning, policing, and paramedic deployment while using real-time monitoring and crowding predictions to detect potential safety issues and manage crowd flow effectively.

The platform includes many features that enable a more effective understanding of the crowding phenomenon. In particular, it leverages the physical carrying capacity of destinations to present information that is more relevant to understanding the perception of crowding. For example, consider the difference between an open space, such as a park, and an easily congested space, such as a narrow street in a historic neighborhood. The perception of crowding is different in these two scenarios.

The platform has an extensible, connector-based architecture that allows it to ingest data from different types of data sources without changing the existing source code. Examples

of these data sources include queuing systems and Application Programming Interface (API) platforms. The platform's forecasting capabilities is also extensible, as new forecasting models can be added to the platform, again without modifying the existing source code.

1.2 Deployment scenarios

This section lists four possible deployment scenarios for the platform, presented in Table 1.1, assuming the latter is loaded with crowding data. For smaller areas, the best solution may be to use sensors to collect that data (e.g., Wi-Fi device detection, infrared cameras, or laser detectors). For larger areas (e.g., with a radius above 100 meters), purchasing data from mobile operators may be a better option, as deploying and maintaining many sensors may exceed the cost of obtaining data from mobile operators.

Table 1.1: Deployment scenarios characterization

	Mass Events in Delimited Spaces Scenario	**Mass Events in Open Public Spaces Scenario**	**Crowded Indoor Scenario**	**Smart Areas Management Scenario**
Location	Outdoor	Outdoor	Indoor	Outdoor
Visitation	Intensive within a given time frame, with controlled access	Intensive within a given time frame and with uncontrolled access	Indoor space with controlled access all year round	Wide area public space, all year round
Examples	Large cultural events, e.g., rock festivals, archaeological sites visiting	Political rallies, fireworks, video mapping, pilgrimage sites	Museums, historical monuments, tech summits and fairs	Local management of crowded historical neighborhoods, natural parks management
Range	tens of ha	tens of ha	< tens of ha	tens of ha, town, city, village, or neighborhood
Geo-mapping	OpenStreetMap	OpenStreetMap	Custom map	OpenStreetMap

1.3 Research questions

Although in this section we define the research questions that will inform some of the content of this dissertation, the work that has been developed is more than what is strictly necessary to answer those research questions. In other words, this work is as much an engineering product as it is research. Recall from section 1.1 that this work had the goal of implementing a smart tourism tool in the context of the RESETTING project.

In the quest to select appropriate visualization techniques that effectively communicate insights from data, namely by following best practices as suggested in [33], some research questions have been raised. These concern the visualization paradigm that should be used, as many different idioms can be used for spatiotemporal visualizations. Many such examples

have been collected in [3]. Exploring existing solutions in the literature related to information visualization is important for this purpose, as is exploring software libraries for rendering interactive visualizations.

A requirement of the platform's visualization paradigm is that it should be able to deal with irregularities in the geographic location of crowding data collection points. One of the sources of crowding data used throughout the development of the platform is Melbourne's Pedestrian Counting System, a related platform for visualizing urban pedestrian counts that has open data. This system, as well as others that could be created by deploying sensors, consists of crowding data for point-based locations that are distributed in an irregular manner. To understand the importance of this, consider the heatmap, a popular visualization idiom for displaying data on a map. A heatmap visualization is more accurate the higher its resolution (i.e., the density of the points), just like any picture, screen, or printer. When the point-based locations are irregular, some sort of spatial interpolation is required for the heatmap metaphor to be effective.

Another important issue is how to quantify the perception of crowding, since raw counts of pedestrians alone do not provide information about how crowded the space actually is. The concept of tourism carrying capacity [11] can be used to define a threshold of visitors above which a place is considered overcrowded.

In summary, three research questions have been posed and are presented below. They will be answered at the end of this dissertation and will help shape the structure of this dissertation.

- **RQ1:** Which visualization paradigm is most suitable for exploring crowding data in cities, both in time and space?

- **RQ2:** How to deal with the irregular geographical location of crowding data collection points?

- **RQ3:** How to quantify the perception of crowding in a given location?

1.4 Methodology and dissertation structure

The structure of this dissertation has been determined to follow the steps of DSR, a research methodology primarily applicable to disciplines where the creation of artifacts, systems, or solutions is a key focus. This methodology is a systematic, iterative, and scientific outcome-based approach that aims to acquire and develop the necessary knowledge to create new artifacts, processes, or systems to solve a well-defined problem in a specific research area and provides specific guidelines for evaluation within research projects [21, 35, 50]. Figure 1.1 illustrates the typical DSR Process Model, highlighting its iterations and outputs.

The following steps of the DSR process model were reified as described in the identified sections of this dissertation.

- *Identify Problem & Motivate* → Consists of identifying the research problem, understanding the context, motivation, and insights into relevant concepts. This step is partially covered in section 1.1, as well as in chapter 2, where related work on crowding visualization is reviewed;

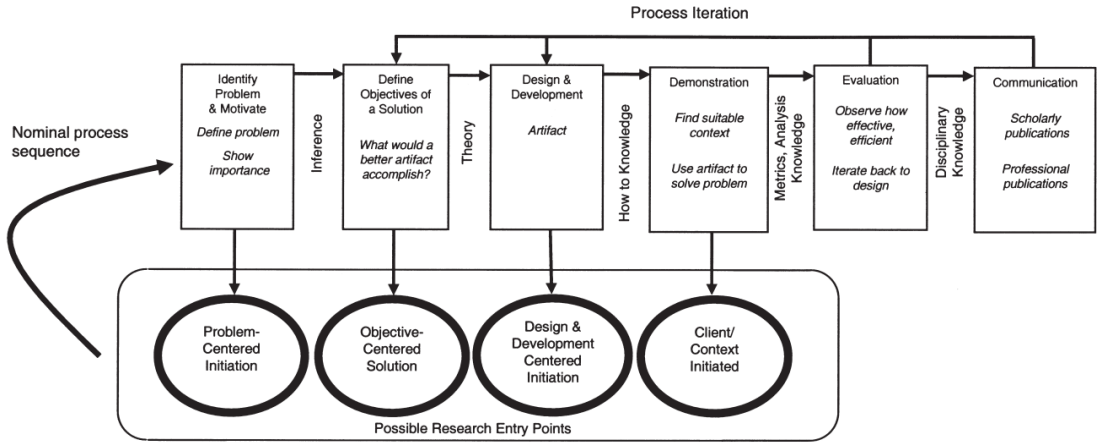


Figure 1.1: DSR Methodology Process Model (source: [35])

- *Define the Objectives of a Solution* → Consists of clearly defining the objectives the designed solution should achieve. Each requirement that led to the implementation of the platform is listed on chapter 3, accompanied by a brief rationale that justifies it;
- *Design & Development* → Consists in the development of the artifact - a 3D spatiotemporal visualization platform - that addresses the research problem. Its architecture is presented in section 3.5, and chapter 4 lists its various functionalities from an end-user perspective while also delving into some technical details that were deemed important to highlight the design choices made during this step;
- *Demonstration* → Consists of presenting and demonstrating the functionality and capabilities of the designed artifact to various parties to obtain constructive feedback. The subsection 5.1.1 shows how the platform can be used to visualize crowding in two different scenarios;
- *Evaluation* → Consists of evaluating the effectiveness, usability, and performance of the developed solution in addressing the identified problem. Evaluations of the platform in four different domains: usability, performance, forecasting ability, and peer-review, are presented in section 5.2;

1.5 Summary of chapters

Although the previous section provides an overview of the structure of this dissertation, a summary of each chapter is presented here so that the reader can have a clearer picture of the contents of this dissertation.

Chapter 2 presents a review of the academic literature on the use of Geographic Information System (GIS) to deal with overcrowding in the context of tourism. The review is systematic and its protocol is described in section 2.1.

Chapter 3 presents the requirements that led to the design and implementation of the platform, as well as its architecture. The functional requirements, which explain what the platform can do, are described in section 3.3, while the non-functional requirements, which

explain how the system should achieve its goals, are described in section 3.4. In both cases, a short rationale is included with each requirement description. The chapter concludes with an overview of the architecture of the platform in section 3.5.

In chapter 4, the various features of the platform are described, both from an end-user perspective and from a more detailed technical perspective when relevant to the discussion of the platform's design. The visualization paradigm is described in section 4.2. Section 4.3 discusses the platform's ability to use derived metrics, as well as its use of carrying capacity. Section 4.4 describes the data ingestion process, as well as the details of loading both historical and live data into the platform's client. An entire section, section 4.5, is dedicated to the forecasting capabilities of the platform, discussing the available models, the quantification and visualization of the uncertainty of the predictions, and some of the implementation challenges that had to be overcome. Finally, section 4.6 describes security and deployment issues, such as access control mechanisms, configuration scripts, and the deployment process.

Chapter 5 begins with descriptions of case studies to demonstrate the platform in action. Then, section 5.2 presents evaluations of the platform in four different domains: usability, performance, forecasting ability, and peer review.

The dissertation concludes with chapter 6, which summarizes the key points of the dissertation, answers the research questions, and outlines future work.

[This page has been intentionally left blank]

Chapter 2

Related work

This chapter presents existing work on the use of Geographic Information System (GIS) to manage overcrowding in the context of tourism. The studies were selected by conducting a rapid review, the protocol of which is presented in section 2.1. Section 2.2 presents a summary of the results, which are discussed in section 2.3.

[This page has been intentionally left blank]

Chapter 2
Related work

2.1 Review protocol

We conducted a rapid review [49] to identify tools and methodologies for dealing with overcrowding in the context of tourism through Geographic Information System (GIS), whether the GIS is used from an end-user perspective, where the authors use it for geovisualization purposes, or from a more technical perspective, where they extend them via plugins or even create their own platform. The corresponding search string for querying the Scopus database of peer-reviewed literature was as follows:

```
( GIS OR "geographic information system") AND ( crowding OR
congestion OR occupancy ) AND tourism
```

We enclosed this search string in the TITLE-ABS-KEY operator so that the search only considers the documents' titles, abstracts, and keywords, allowing for a more precise scope. The abstracts of the search results (a collection of papers) were read, and off-topic papers were excluded. The resulting papers were submitted to a selection process using inclusion and exclusion criteria.

Inclusion criteria:

- paper must have been published in the last 20 years;
- paper must be written in English;
- paper must be a primary study.

Exclusion criteria:

- paper is a duplicate;
- paper is outdated by a later and more complete version;
- there is no access permission to the paper.

After applying the selection criteria, a quality assessment was conducted based on five questions. The questions were answered with grades between 0 points (strongly disagree) and 4 points (strongly agree), resulting in a potential maximum score of 20 points.

Quality assessment questions:

- Were the goals and research questions clearly stated?
- Was related work exposed and compared with claimed research results?
- Was the research design clearly specified?
- Do the authors assess the limitations of the study?
- Was the study relevant for crowd-monitoring solutions using GIS?

2.2 Review summary

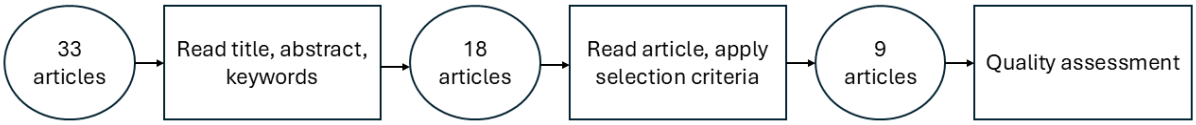


Figure 2.1: Diagram with steps of the literature review

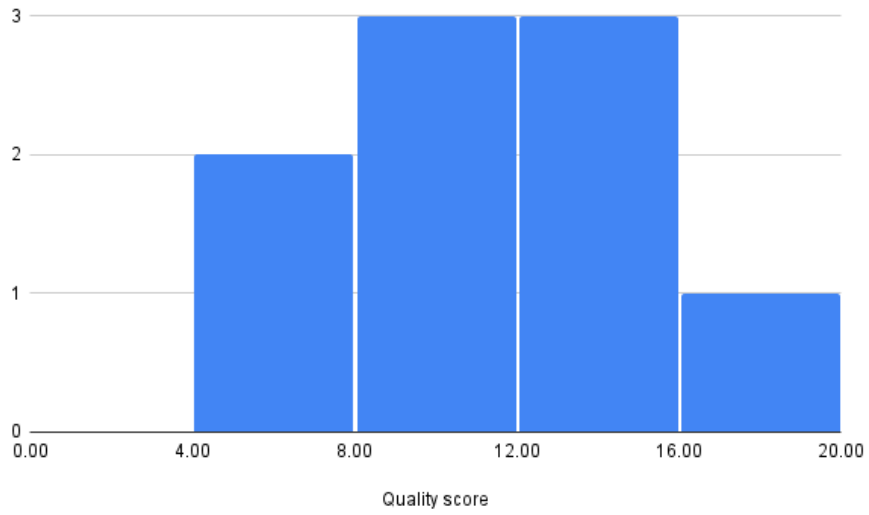


Figure 2.2: Histogram of quality scores for selected papers

Table 2.1: Summary of the selected articles

Article	Domain	Data sources	Presents tool?	Considers CC?
[40]	Urban pedestrian dynamics	Wi-Fi sensors	No	TCC
[25]	Touristic itineraries	Online information of travel operators	No	No
[4]	Traffic congestion	Satellite imagery	No	No
[31]	Light rail transit	Governmental organizations	No	Other CC
[55]	Visitor flows	Cameras and tourist panels	Yes	TCC
[51]	Hiking routes recommendation	National and regional topographic maps	Yes	No
[39]	Soil carbon stock	Governmental organizations	No	No
[37]	Quality of touristic experience	Web-based surveys	No	No
[53]	WebGIS development	*N/A*	Yes	No

As seen in Figure 2.1, the search returned 33 papers. After reading the abstracts of each, 14 were rejected for being off-topic, and another was rejected for being older than 20 years. The remaining 18 papers were read whenever possible, depending on language availability and access to the full text. The inclusion and exclusion criteria were applied, resulting in our final nine papers. The quality assessment of these papers was carried out, and its results are shown in Figure 2.2.

Table 2.1 shows a summary of the selected articles across four dimensions: the domain addressed by the study, the sources of the data employed in the study, whether the study presents a tool that can be applied to different scenarios, and whether the study considers Carrying Capacity (CC). For the CC dimension, three values are possible: "TCC" indicates that Tourism Carrying Capacity was considered, while "Other CC" indicates that other forms of Carrying Capacity were used, and finally "No" indicates that no types of Carrying Capacity were taken into account in the study.

2.2.1 Threats to validity

It should be noted that there are several threats to the validity of this review. The most critical issue is the small number of articles selected, which makes this review potentially non-comprehensive, despite its systematic nature. The use of a single search engine contributes to this problem, as other important related papers may appear in other academic search engines. In addition, it should be noted that the search string used is very focused, which may result in relevant material not being considered for selection. Finally, the selection criteria also contribute to the low number of articles selected, as several papers were excluded due to lack of access permission, and another was excluded because it was not written in English.

Another threat to validity is the fact that only peer-reviewed content was analyzed, whereas a multi-vocal review including grey literature might be more appropriate given the technical nature of the topic.

2.3 Review report

Most of the selected papers do not present the development of a crowd monitoring solution that can be reused but instead make use of GIS to analyze specific scenarios, such as analyzing pedestrian dynamics in a city [40], planning alternative roads to reduce traffic congestion [4], visualizing spatial patterns in touristic itineraries [25], exploring the consequence of land-use change on soil carbon stock [39], conducting surveys on the quality of the touristic experience [37], and visualizing the result of Agent-Based Modeling (ABM) for analyzing the relations between light rail transit development and tourism [31].

Some papers present a crowd-monitoring solution. One such paper presents a solution that consists of a manager platform application that can use real-time and historical data on visitor flows and a mobile application for tourists so that they can be more evenly distributed among the different places of interest [55]. As for the manager platform application, it is a web application that uses 3D visualization of urban environments based on a 3D model, and the monitored areas and monuments are color-coded based on the average daily number of

visitors. The application's visualization paradigm includes another visualization idiom: a bar chart aggregating the average number of visitors per month. The paper does not describe the visualization paradigm for real-time data. Another paper presents a recommendation system for hiking routes, which has the potential to keep hikers interested throughout the routes and reduce the impact of human activities on protected areas [51]. The GIS used is PostGIS for storing geographical data rather than for its visualization. Finally, [53] presents a WebGIS-based solution that can display real-time road traffic conditions on an interactive map, although the specifics of its visualization paradigm are not given.

Since this dissertation is about a visualization platform, it is also relevant to look at the presentation of data in the literature reviewed. In two papers [40, 55], crowding levels for a given period are represented by line graphs, where different lines correspond to the crowding levels of their respective locations. These visualization idioms are among the most relevant to us among those presented in the selected literature, as they deal with crowding data. However, many other visualizations are used in the selected papers, primarily based on cartographic maps, as would be expected from studies using GIS, although none of these geovisualizations incorporate a temporal dimension; the only way to implement spatiotemporal visualizations based on these idioms would be to link different spatial visualizations, for example by playing an animation or allowing the time to be selected through an interactive slider, showing different spatial visualizations for each moment in time.

A critical step in the viability of a crowd-monitoring solution is collecting crowding data, which consists of people counts. These counts can be obtained in several ways. In one study, Wi-Fi sensors are used to detect pedestrians [40]. In another, people counting cameras are used, and interaction with tourist panels also contributes to crowding counts [55].

Regarding collecting other types of data, some studies [31, 39] obtain the necessary geographic data through governmental organizations. Satellite imagery is used by one study [4] to aid in planning alternative roads to decrease the traffic congestion partly caused by tourism.

In mitigating overcrowding, the concept of carrying capacity, used to quantify the perception of overcrowding, is crucial and is mentioned in several selected papers. A paper describing a methodology for a smart tourism tool for the management of tourism flows [55] includes calculating the carrying capacity of tourist destinations in its approach. A case study implementation in the same paper uses a 3D city model from which the carrying capacities of locations in the city can be obtained. Another paper uses a GIS to digitize and quantify land use to determine the walkable area of a location, from which its carrying capacity is calculated [40].

Chapter 3.

Requirements & Architecture

Contents

This chapter presents the platform's main functional and non-functional requirements. Their specifications were presented in natural language, without using templated structures, and were described as what could be called 'user requirements', that is, requirements that are more abstract and high-level than 'system requirements'. Implementation details of the various features that support these requirements are described in the following chapter. In addition to the requirements specifications, a high-level overview of the platform's architecture was presented.

[This page has been intentionally left blank]

Chapter 3
Requirements & Architecture

3.1 Introduction

The platform described in this dissertation is a software artifact. As such, elaborating a clear picture of what it should do before the implementation phase is crucial. The adopted DSR methodology naturally implied an iterative process. New requirements were considered or adjusted as the produced artifact evolved from an initial prototype into a relatively sophisticated application.

Requirements can be grouped into "user requirements" or "system requirements", the former being more abstract and high-level, and the latter being more detailed [46]. The requirements presented in this chapter fall into the more abstract and high-level category. Note that detailed specifications are more relevant to communicating the expectations between stakeholders.

3.2 Requirements overview

Before delving into the requirements specifications, a UML use case diagram[1] is presented in Figure 3.1.

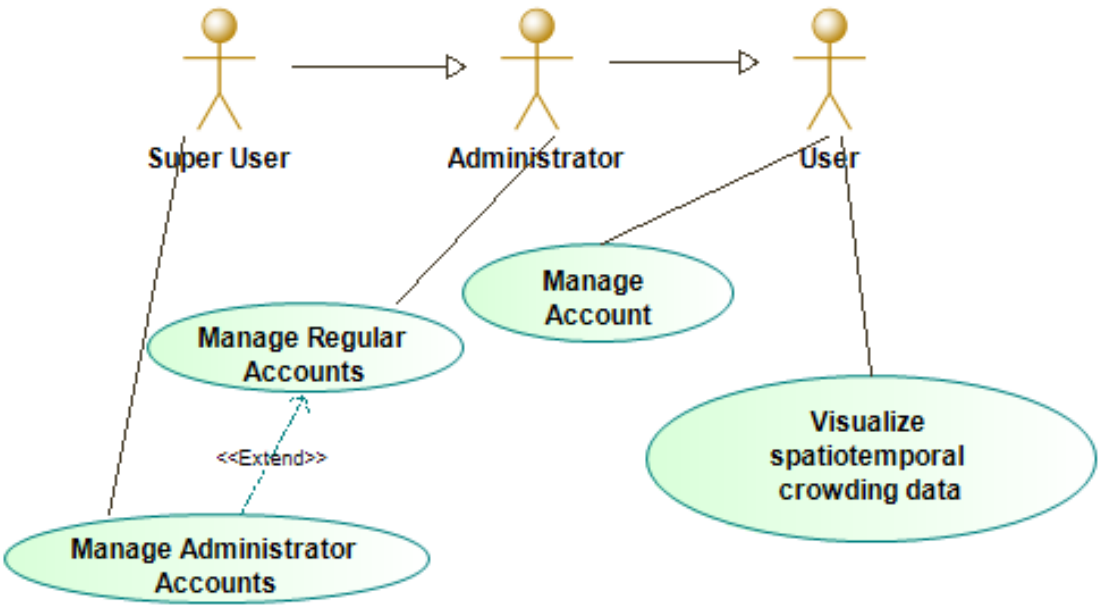


Figure 3.1: UML use case diagram for the platform

The actors in Figure 3.1 represent different types of users. Due to the inheritance links, the Administrator can do everything the User can. Similarly, the Super User can do anything the

[1] This type of diagram contains the use cases of a system and the actors that can participate in them. Further details can be seen in e.g. [8]

Administrator can do. Extensions of use cases are also employed here, as "Manage Administrator Accounts" extends "Manage Regular Accounts", meaning the former's definition is based on the latter's. These use cases represent scenarios that map to functional requirements. Many platform requirements are omitted to keep the diagram short and simple, following the guideline proposed in [13] that the diagram should be simple and that the focus is on the textual description of the various scenarios. The most important requirement for the platform, from which all its other requirements emerge, is to have visualization capabilities for spatiotemporal data. All requirements described below that could be represented as extensions of this main functional requirement, represented in the use case "Visualize spatiotemporal crowding data", are omitted in the diagram.

3.3 Functional requirements

The functional requirements of a system specify what it should do or what a user can do with it. This section contains textual descriptions of each functional requirement we have elicited. They are:

1. Visualize spatiotemporal crowding data
2. Visualize historical, live, and predictive data
3. Visualize crowd density in a location relative to its carrying capacity
4. Select zones of interest
5. Show different crowding metrics
6. Show critical points
7. Allow uncertainty quantification and visualization
8. Provide access control mechanisms

3.3.1 Visualize spatiotemporal crowding data

The platform's highest priority requirement is the ability to display interactive visualizations to explore spatiotemporal crowding data. Most of the other requirements described here support this capability.

This visualization capability allows the end-user to gain an understanding of crowding data that, while not necessarily 'big data,' contains too many data points to be understood in rudimentary forms of data visualization, such as tables.

Crowding data are given as numerical time series for different locations. These locations can be specified as geographic points or polygons[2].

As this is the most fundamental part of the platform, two of the Research Questions (RQs) introduced in section 1.3 are directly related to this requirement. Recall that RQ1 is *"Which visualization paradigm is most suitable for exploring crowding data in cities, both in time and space?"*, and RQ2 is *"How to deal with the irregular geographical location of crowding data collection points?"*.

[2]Polygon-based locations allow for the display of grid-based topologies, as well as more complex layouts.

The visualization paradigm chosen for the platform should be designed considering the related work presented in chapter 2. This allows us to answer RQ1. The possibility of irregularities in the spatial locations of the crowding data is another concern expressed in RQ2 that needs to be addressed here.

Note that the visualization idioms employed by the platform do not necessarily need to have both a spatial and temporal dimension. A visualization idiom is a specific, established method for visually representing data, which leverages particular graphical elements (like bars, lines, or shapes) and interactions to communicate certain types of information effectively. Visualization idioms are grounded in perceptual and cognitive principles, making data patterns or insights more intuitive and accessible for viewers. While many such idioms exist for displaying both dimensions at the same time, an effective spatiotemporal visualization paradigm might make use of linked but distinct spatial and temporal visualizations.

3.3.2 Visualize historical, live, and predictive data

The platform should support three different functioning modes concerning the temporal aspect of the data to be displayed: historical data, near real-time data, and predictive data. Each of these modes brings challenges to the design of the platform, notably its user interface, which must be adapted to each mode. However, the goal of the visualization paradigm is to make the look and feel of the different modes as similar as possible. This similarity can improve the platform's usability - specifically, its learnability - since a user accustomed to one of its visualization modes can more easily understand the other modes. In other words, there should be no conflict in the user's mental model.

3.3.3 Visualize crowd density in a location relative to its carrying capacity

Visualizing the density of crowding values and their absolute values is critical. By density, we mean the value relative to the carrying capacity of the location, as discussed in [5, 12]. This density provides the user with more relevant information for managing crowded spaces, as two locations with the same number of people may have very different levels of perceived crowding. For example, consider the difference between an open space, such as a park, and an easily congested space, such as a narrow street in a historic neighborhood. The perception of crowding in these two scenarios is different.

3.3.4 Select zones of interest

Since the platform must display information for multiple locations, there may be scenarios where the number of locations is large. Selecting zones of interest, i.e., the locations the user is interested in, allows the visualizations to be filtered rather than cluttered with less relevant information.

3.3.5 Show different metrics

This requirement extends the data type definition the platform should work with, as given in subsection 3.3.1. Each location may have several time series because the data may have different

crowding properties. For example, different values may correspond to different directions captured when a pedestrian walks through a motion sensor. The platform's user interface should display these different metrics, present descriptive names for each, and allow users to choose which to visualize.

3.3.6 Show critical points

This requirement is motivated by the possibility that the number of date/time points to be visualized is large, thus stretching the visualization paradigm to a point where its effectiveness is reduced. In addition, the critical points of the crowding data, i.e., local maxima and minima, are more likely to interest the user concerned with managing crowded spaces. Having the platform highlight critical points for the selected region of interest and metric is an important functionality. Another solution would be to allow jumping between critical points rather than highlighting them visually.

3.3.7 Uncertainty quantification and visualization

As mentioned in subsection 3.3.2, visualizing predictive data is one possible platform mode. This implies that the platform should have forecasting capabilities that feed predictive data into the platform's visualizations. While most of the differences between the three modes of the platform (historical data, real-time data, predictive data) are implementation details not covered in these requirements specifications, forecasting opens up the possibility of significantly extending the visualization paradigm. This is because time series forecasting models may consider uncertainty quantification in their predictions. Visualization of uncertainty gives the user an idea of the "credibility" of the forecast data, especially as the error compounds over time.

3.3.8 Access control mechanisms

Let us consider the scenario where the platform is hosted in the cloud and accessible via a public IP address. If we want to maintain the confidentiality of the data in the cloud, access control mechanisms are a necessary part of the architecture. It should be role-based, as administrators should be able to create and delete non-privileged accounts. Another privilege exists for the superuser, who can create and delete administrator accounts. Given this role-based authentication and authorization method, which uses encryption mechanisms, the confidentiality of data is maintained.

3.4 Non-functional requirements

While functional requirements describe *what* a system can do, non-functional requirements describe *how* the system should accomplish it. Or, put in another way, what the system should *be*. These non-functional requirements are that the system should:

1. be a web application
2. be responsive

3. use open technologies
4. be extensible regarding data source
5. be deployable through containers

3.4.1 Web application

Technically, the platform should be a web application, as this facilitates the deployment of the frontend so that the end user can use it on any reasonably modern computer without having to install the application. Furthermore, this increases the chances that the platform will be more widely adopted. Other well-known advantages of this approach include (i) reducing the level of trust required by the end user since web applications run in a sandbox within the browser and (ii) the ability to run on other types of devices, such as mobile phones and tablets. While mobile phones typically have a screen that is too small for this type of platform, a tablet could be a suitable target device in addition to laptops and desktop PCs.

3.4.2 Responsiveness

The interactive components of the web application should have as little time delay as possible in producing visible image effects. No quantified time limits are imposed here, except that the effects produced by interaction with the user interface should be near *instantaneous* (as perceived by a human) whenever possible. Scenarios where instant effects are hard to achieve include loading a reasonably large amount of data from the server to the client. What this requirement means in practice is that a visualization idiom used in the application that is not responsive on a low-end modern PC should be discarded in favor of more responsive idioms.

3.4.3 Use of open technologies

Due to the requirements of the RESETTING project under which this platform was developed, the platform itself should be publicly available under a free and open-source software license. In addition, the platform's supporting technologies, namely software libraries, should all be free and open-source.

3.4.4 Extensibility regarding data source

Data can be made available to a web application in various ways. Geographic location data can be stored in the server's file system or a document-oriented database. Multiple storage options exist, including specialized time-series databases for the time-series data related to crowding across locations and metrics. It was decided early in the DSR process that the platform should be able to draw from multiple data sources. It is possible to imagine an architecture with different modules for different types of data sources, and the application dynamically loads the appropriate module based on what is specified in a configuration file. This makes it possible to extend the platform to fetch data from different sources by creating new modules without modifying the existing source code.

3.4.5 Containerization

Containerization technologies such as Docker have the potential to reduce the complexity of the deployment process significantly. Indeed, running the various platform's components as services should be possible based on pre-built images, avoiding the typical scenario in which differences between different environments create difficulties in deployment under some circumstances. Given the requirement in subsection 3.4.3, that specifies that open technologies should be used, it is very likely that any necessary software components that are external to the platform's codebase, such as databases, already have pre-built images available online. Artifacts internal to the platform should also have images available online when appropriate.

3.5 Architecture

Given the abovementioned requirements, the envisaged architecture will be discussed in more detail. The platform's implementation details are left for the next chapter.

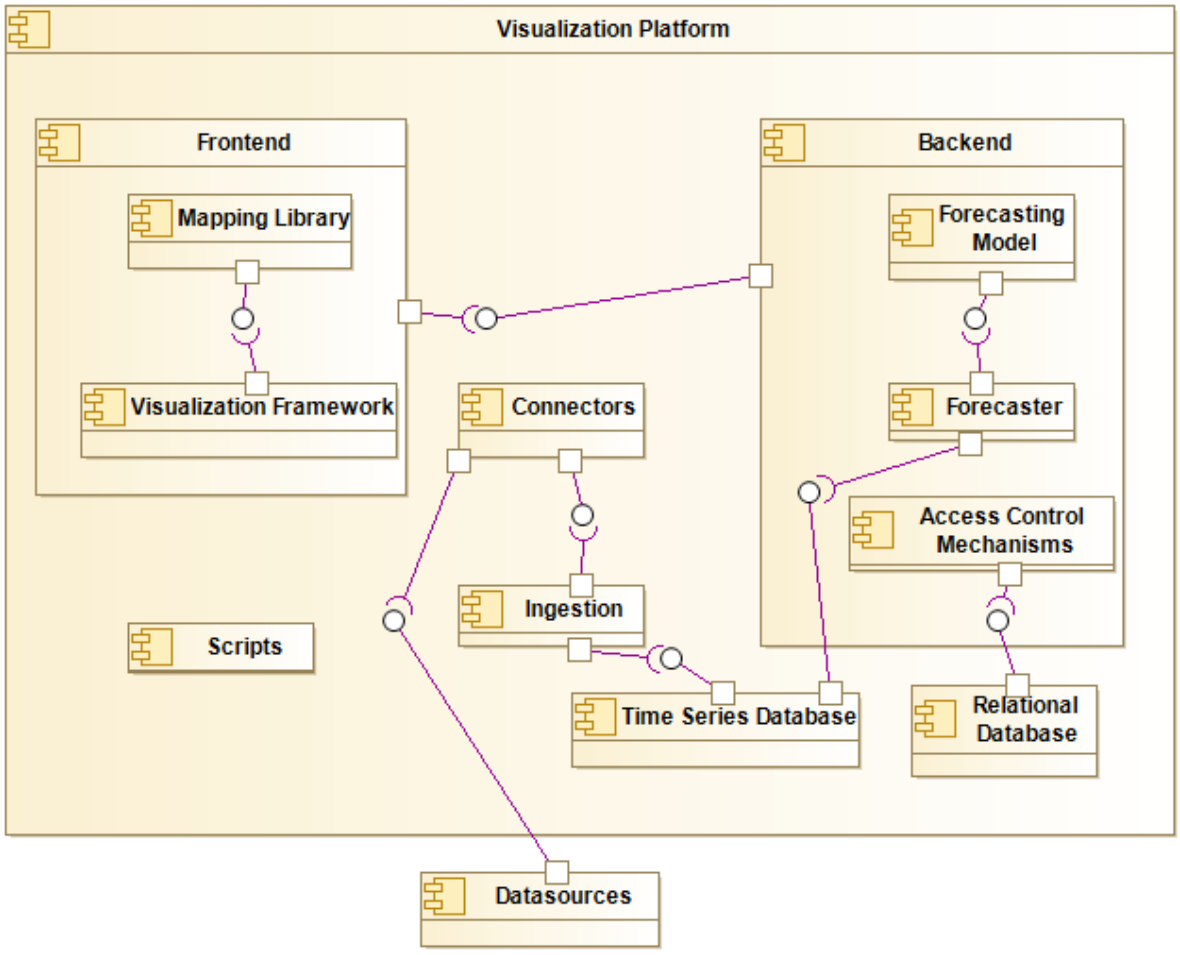


Figure 3.2: UML component diagram for the platform

The platform is ultimately a web application with decoupled frontend and backend. Figure 3.2 shows a component diagram of the application. There are components and subcomponents, with a ball and socket notation for interfaces, where the ball represents a provided

interface (some functionality or service that other components can use), and the socket represents a required interface (functionality or service from other components that a component needs to fulfill its functionality).

One critical component is the time-series database, where the crowding data resides. The "Scripts" component in the diagram is a set of scripts to help configure the platform, described in subsection 4.6.2. The two main components of the platform, Backend and Frontend, are discussed in the following subsections.

3.5.1 Backend

The backend exposes Representational State Transfer (REST) endpoints to be consumed by the frontend. There are four primary endpoints: to load historical data, live data, predictive data, and retrieve metadata.

The crowding data is assumed to reside on a time-series database for the first three endpoints. For data stored using a different technology, the Ingestion component is used. This is a script that runs on the server and polls the original data source periodically if real-time data is available; otherwise, it fetches a specified date/time range for historical analysis. In both cases, the fetched data is uploaded to the time-series database for later use.

Since support for multiple data sources is a requirement, the Ingestion script instantiates one of the available Connectors according to a configuration file. Each connector module is responsible for retrieving data from a different data source.

Another important component is the Forecaster. This includes a job that trains and runs forecasting models as new data is made available by the ingestion component. The results are stored in memory and are served to the frontend when the latter invokes the REST endpoint for predictive data. The Forecaster's design is extensible, as was the case for Ingestion, so different Forecasting Models may exist. A configuration file specifies which of them should be used, along with any required parameters.

The backend contains access control mechanisms as described in subsection 3.3.8. A Relational Database Management System (RDBMS) stores user data and encrypted credentials. The four REST endpoints mentioned earlier require a valid session cookie, which can be obtained from a login endpoint. Additional endpoints exist for account management, such as creating, modifying, or deleting user accounts.

3.5.2 Frontend

As expected, most of the codebase will be dedicated to the frontend implementation. The component diagram shows two subcomponents: a visualization framework and a mapping library. These are the core components upon which the frontend is built, although the latter depends on additional libraries for functionalities such as date manipulation, mathematical functions, and widgets.

A single web page is dedicated to the spatiotemporal visualizations and control interfaces. Several other pages exist for account management purposes.

[This page has been intentionally left blank]

Chapter 4

Implementation

Contents

This chapter discusses the features of the platform that implement the requirements specified in the previous chapter.

The chapter is structured as follows: section 4.1 provides a brief introduction to the implementation details of the platform. Section 4.2 describes the platform's geographic and temporal features. Section 4.3 details the platform's use of carrying capacity and its ability to define metrics derived from other existing metrics. Section 4.4 discusses the data ingestion process and historical and live data loading. Section 4.5 explains how the platform forecasts data, and section 4.6 details the configuration and deployment of the platform and its access control mechanisms.

[This page has been intentionally left blank]

Chapter 4
Implementation

4.1 Introduction

The platform meets the requirements specified in chapter 3 through several features. This chapter describes these from both a technical and an end-user perspective.

To give a high-level technical perspective, recall that the platform is a web application with decoupled frontend and backend. The frontend was built with Javascript/React and the backend with Python/Flask. In addition to those technologies, many other third-party libraries are used, as is natural for any non-trivial software product. For example, the widgets employed in the frontend's user interface are provided by the Material UI library. Other libraries the platform uses will be referenced in this chapter when appropriate.

The communication between frontend and backend is done through invocating the Representational State Transfer (REST) endpoints defined in the backend, and the data transmitted between these components uses the JSON format. There are endpoints for loading metadata, such as the locale and timezone to be used, and for loading crowding data, whether historical, live, or predictive.

Besides the description of the platform's various features, this chapter contains information regarding the configuration and deployment of the platform. The platform has been successfully deployed in a production server for demonstration purposes and for one of the usability assessments that will be described in subsection 5.2.1.

The code developed for the platform is free and open source and can be accessed in the RESETTING GitHub repository.

4.2 Visualization features

The visualization platform presents three linked spatiotemporal visualizations, which can be seen in Figure 4.1 and are described in detail in this section. The data presented in the visualizations include geographic locations – defined as polygons or points – date/time, and numerical values for various metrics. The data loaded into the platform can be historical data, near real-time data, or predictive data. For the latter two, the frontend automatically fetches new data as it becomes available and then updates the visualizations.

4.2.1 Geographical features

The main area holds a three-dimensional visualization based on a 2D cartographic map, providing the spatial context. MapLibre is the mapping library used for presenting an interactive map rendered as a plane in 3D space using WebGL. The map is populated with tile data from OpenStreetMap.

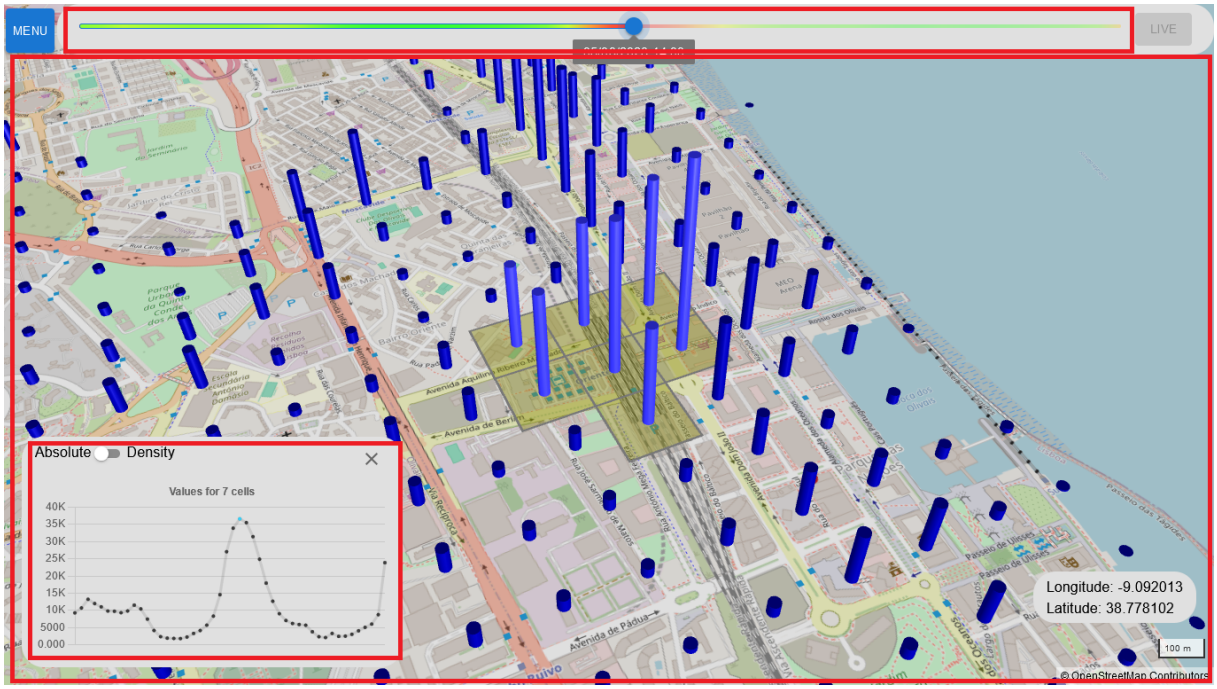


Figure 4.1: Screenshot of the platform with its three spatiotemporal visualizations (outlined in red)

As the focus is on highlighting values and locations of metrics of interest, cylinders are superimposed as visual metaphors at locations in the map where data is available. The definition of the placement of a cylinder's lower base depends on the definition of its location. For a location defined as a point, the point's coordinates define where the cylinder is placed. For a location defined as a polygon, the placement can be specified via the GeoJSON feature's properties 'latitude' and 'longitude', if such properties are defined, otherwise the geometric center of the polygon is considered.

The height of a cylinder represents the value of a metric, while its color could represent another metric or the density of the same metric. Hence, we can visualize two metrics at once. The user can choose which metrics to map to these visual properties. Furthermore, to show more detail on demand, hovering over a cylinder will display a tooltip with the exact values of the selected metrics.

Regarding scale mapping, the height of a cylinder is defined in meters and is scaled based on the metric's value, the metric's 'cap' value, and the level of zoom. A metric's 'cap' value is statically defined and represents a value considered to be very high for that metric. One of the purposes of this value is to have an upper limit on the height of the cylinders so that the 3D visualization does not include cylinders too big when displaying outliers. To indicate that a location's value crossed the cap value and the cylinder is cut short, the upper base of the cylinder is filled with pure red, or $(255, 0, 0)$ in RGB notation (Red, Green, Blue). Note also that the height of the cylinders relative to the map is automatically adjusted when zooming so that high values fit into the viewport at high zoom levels, and small variations in values are easier to see at low zoom levels.

As mentioned, the user can choose to map a second metric onto the color of the cylinders. When such a mapping exists, the hue of the cylinder in the HSL (Hue, Saturation, and Lightness)

model ranges from 0 (pure red) to 120 (pure green). The hue component is calculated as in the formula below, where V_{MLT} is the crowding value for metric M at location L at time T, C_M is the 'cap' value of metric M:

$$hue = 120 - \min\left(\frac{120 \times V_{MLT}}{C_M}, 120\right)$$

This means that the hue is set to 0 for values greater than or equal to 'cap', to 120 for a value of 0, and to a value in between, calculated linearly, for values between 0 and 'cap'.

Concerning the rendering of graphic primitives, particularly cylinders, the deck.gl library is used to render the cylinders layer with WebGL efficiently. The platform's code base extends the functionality of the library's ColumnLayer, adding new properties and modifying its shaders. One such modification enables filling the top base of the cylinders with a color that is different from the rest. This indicates a location has crossed the metric's 'cap' value. Other modifications to ColumnLayer exist to transform the cylinder into a different shape used for visualizing forecasting uncertainty, as described later in this chapter.

The user can select regions of interest by clicking on individual cylinders, selecting predefined sets of locations such as parishes, or by drawing a polygon on the map. In the latter case, the region of interest includes all locations whose cylinder is contained or intersected by the polygon. When a location is selected, its cylinder's saturation in the HSL model is higher than the cylinders of locations not selected. Selected locations defined by polygons can have another visual cue indicating their selection by having their area on the map filled with a solid, translucid color. This last cue is optional, as the user can toggle off this behavior. Locations can be deselected by clicking on individual cylinders or pressing a button that clears the whole selection.

4.2.2 Temporal features

A key feature of the leading visualization is that it displays spatial data for a single point in time. A slider allows the user to select the moment to be visualized, and to convey further the information, the date and time of the selected moment are displayed when the thumb of the slider is hovered or moved. Following a video player metaphor, there are buttons to play forward and backward, pause and resume, change speed (accelerating the playback speed in either direction), and next/previous scene buttons that make the slider thumb jump to the next/previous local maximum. The local maxima are determined by summing the values of the selected metric for the current region of interest if some locations are selected. Otherwise, it considers all locations.

The timeline slider also serves a purpose beyond browsing the time dimension itself. Its background is a color gradient that encodes the sum of the values for the selected locations or the whole map if no locations have been selected. This is the platform's second area of visualization. The colors of the slider's background are determined using the HSL model. The moment with the lowest value on the timeline is assigned a hue of 120 (pure green), while the moment with the highest value is assigned a hue of 0 (pure red), with the values in between being calculated linearly.

Finally, the third visualization area comprises a line graph with time on the horizontal axis and the sum of the metric values on the vertical axis. As with the slider's background, the values considered are for the selected locations or the whole map if no locations have been selected. If the user selects a metric to be mapped to the hue of the cylinders, the graph becomes a dual-axis line graph, with two vertical axes and two lines, one for the 'height metric' and another for the 'hue metric.' Hovering over a point in the line graph gives a tooltip showing the date and time of the point and the sum of the metric values at that time. The graph also supports zooming and panning. The library used for the graph implementation is Chart.js.

4.3 Metrics

As mentioned, the crowding data supported by the platform can have various metrics. For example, data collected by a movement sensor that can sense in two directions can have one count for one direction and a different count for the other.

4.3.1 Derived metrics

The platform supports defining derived or computed metrics based on the values of existing metrics. The usefulness of this feature can be exemplified by some of the metrics that exist in a real dataset that was used throughout the development of the platform. Consider a metric named 'C1' that measures the amount of mobile devices detected at a given space and time, and a different metric, named 'C2', that measures the amount of mobile devices roaming. An interesting derived metric could be obtained by subtracting the values of C2 from C1, thus giving the number of mobile devices that were *not* in roaming, most likely local residents.

To this end, the backend's configuration file can optionally contain a YAML dictionary where its keys are the names of the new metrics, and their respective values are arithmetic expressions that define how the derived metrics are calculated. These expressions are parsed with the help of Parsimonious, a Python library for parsing text according to a Parsing Expression Grammar (PEG). The following listing shows the grammar used by the platform:

```
sum = product plus_or_minus?
plus_or_minus = ("+" / "-") sum
product = atom mult_or_div?
mult_or_div = ("*" / "/") product
atom = paren_expr / identifier / constant
paren_expr = "(" sum ")"
identifier = ~r"[A-Z_][A-Z_0-9]*"i
constant = int / float
float = ~r"-?[0-9]*\.[0-9]+"
int = ~r"-?[0-9]+"
```

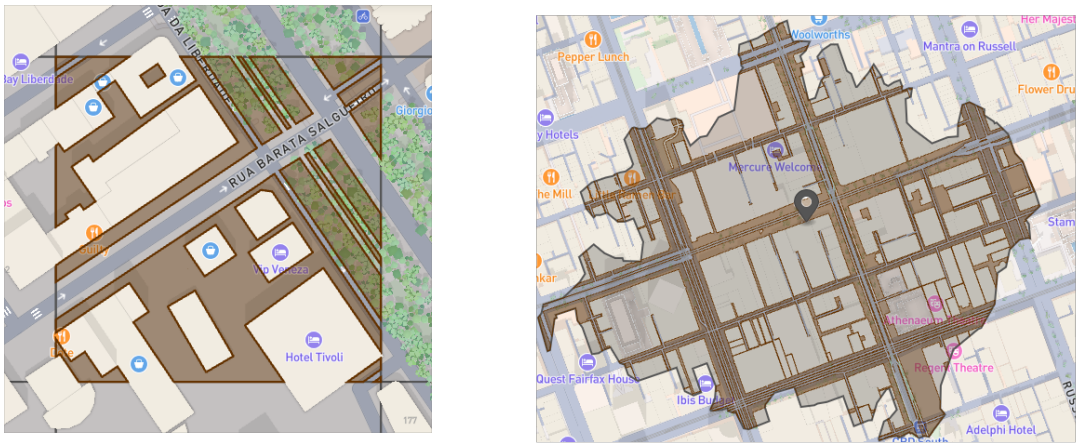
This grammar defines a language that can be said to be 'the language' of arithmetic expressions, as it exists in many mainstream programming languages, including those with C-style

syntax and Python. The four basic arithmetic operations are allowed, as is grouping with parentheses for higher precedence. The values can be integer or floating-point literals, or they can be identifiers. These identifiers must be names of existing metrics.

These expressions are evaluated before the crowding data is sent to the frontend. The expressions are evaluated once for each $(Date/Time, Location)$ pair, and each identifier in the arithmetic expression evaluates to a numeric value that is the value for $(Date/Time, Location, Metric)$ where $Metric$ corresponds to the identifier.

4.3.2 Carrying capacity

As mentioned before, it is possible to map the 'Density' values to the hue of the cylinders in the main visualization. By density, we mean the value of the metric relative to the location's physical carrying capacity [12], which we obtain by dividing the given value by the location's walkable area.



(a) Polygonal area (200x200m) in Lisbon

(b) Isochrone area in Melbourne

Figure 4.2: Carrying capacity visualization in two scenarios

For locations defined as polygons, such as in Figure 4.2(a), we obtain the walkable area (shown in brown) by applying the algorithm described in [5] directly to the polygon. In addition, the platform includes a configuration script that defines carrying capacities for topologies defined by points where it does not make sense to talk about areas. Given a point, we considered its area of influence to be within its isochrone, hereafter referred to as the "isochrone area". An isochrone is a line on a map that connects points reached within the same timeframe from a given starting point and at a given speed. Given the pedestrian's starting point and average walking speed, we use GraphHopper to calculate an isochrone area. These isochrone areas are then fed into the same algorithm used in the script to determine the carrying capacity for polygon-defined locations. An example of the application of this technique is represented in Figure 4.2(b), where the walkable area is also shown in brown. The point of origin of the isochrone area is shown as a pin marker in the center of the figure.

When the aforementioned 'Density' visualization is selected, the Hue component of a cylinder is calculated as in the formula below, where V_{MLT} is the crowding value for metric M at location L at time T, C_M is the 'cap' value of metric M, and A_L is the walkable area of location L:

$$hue = 120 - \min\left(\frac{120 \times V_{MLT}}{A_L \times C_M}, 120\right)$$

Both subfigures in Figure 4.2 were obtained by screenshotting a web application for calculating and visualizing the carrying capacities of GeoJSON polygons. The author of this dissertation participated in developing that application, which is discussed in [5].

4.4 Data management & visualization

4.4.1 Data ingestion

The crowding data for use in the platform is stored in an InfluxDB instance, which is a time-series database. The platform includes a script to ingest data from a data source into InfluxDB. It has an extensible architecture where a configuration file defines the name of the module ('connector') that is dynamically loaded at runtime to perform the actual fetching of the data, while a core module scaffolds the process. The configuration file includes other parameters. Some of these parameters are required regardless of the chosen connector, such as the URL and credentials of the InfluxDB instance, while others are specific to the connector. At the time of writing, there is only one connector in the platform's codebase, which can be used to ingest data from Opendatasoft (an API platform for accessing specific datasets).

The script can be used to retrieve data relative to a given date/time range, or it can be in live mode, where it periodically polls the data source to automatically retrieve new data as it becomes available.

The ingestion script's live mode is meant for real-time data and predictive data visualization, as described later in this chapter. These two types of visualization make the platform a real-time system. One concern with this type of system is the handling of late-arriving data. The Opendatasoft API has a limitation in that it is possible to query for the timestamp of the recorded events but not for the timestamp of the arrival of the data. This means that the naive approach of storing the latest timestamp to be used as the beginning of the time range of the next query does not work for the late-arriving data scenario.

The workaround that has been implemented in the platform's Opendatasoft connector is to require a configuration parameter called 'ignore_before', which is defined as a time duration. When making a Flux query, typically it is defined a range of date/times through a 'start' timestamp and a 'stop' timestamp. When the ingestion script polls the API in live mode, instead of defining the 'start' of the date/time range to be the timestamp for the latest available data, it is defined as the difference between the last received timestamp and 'ignore_before'. For instance, if the latest timestamp is for 2024-08-02 at 1PM, and 'ignore_before' is 24 hours, the difference between them is the timestamp for 2024-08-01 at 1PM. Late-arriving data for events that happened after that timestamp will be ingested. Timestamps that are even earlier than that will be forever 'lost' or 'ignored' in this ingestion process, hence the parameter's name. This design is inspired by Apache Spark's "watermarking" technique for late-arriving data.

4.4.2 Loading historical data

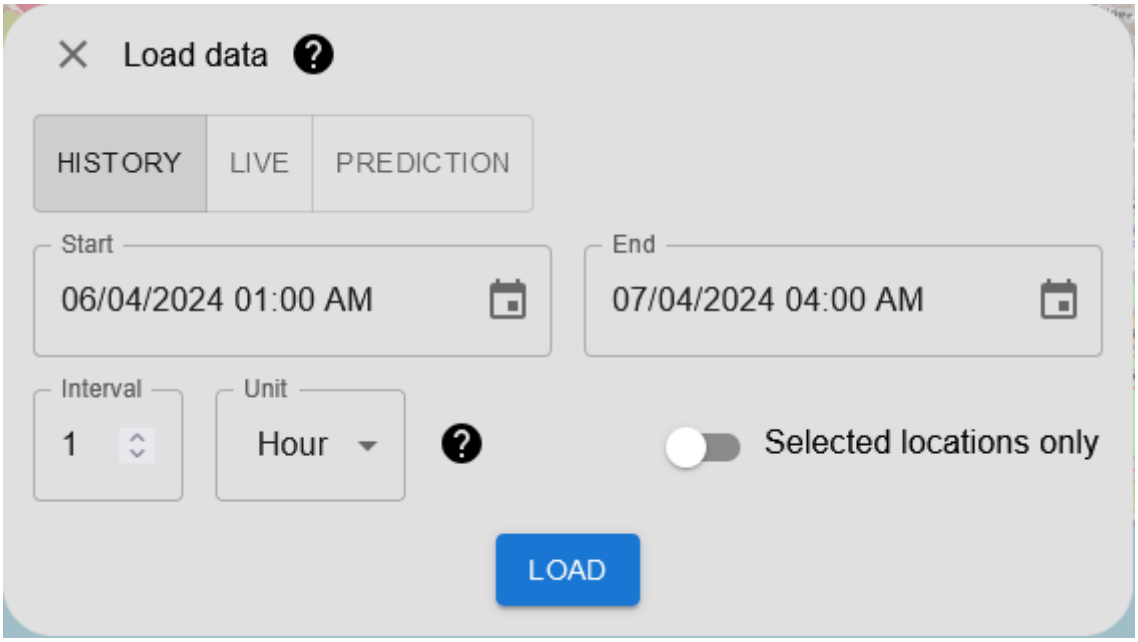


Figure 4.3: Screenshot of the UI for loading historical data

The user interface of the platform contains a menu for opening panes containing widgets for the various configuration purposes that are described in this chapter. For example, selecting metrics to be visualized and playing an animation. One of these panes, named "Load data", supports several loading modes (historical data, real-time data, or predictive data). The pane contains a toggle button group for each of these modes. When one of the toggle buttons is clicked, the pane displays the available options for the selected mode.

Figure 4.3 shows this "Load data" pane with the options for loading historical data. Two date/time pickers exist to select the start and end of the time range. Two other widgets exist to define the time interval used to aggregate and average the data. One of them allows specifying the numerical part of the interval while the other is a dropdown menu for selecting the corresponding unit (minute, hour, day, or week). Another widget that is specific to the historical data mode is the switch for "selected location only". When this switch is toggled, crowding data will be fetched only for the selected region of interest. This can significantly decrease loading times for the scenarios in which the selected region is small compared to the totality of available locations.

When the user presses the blue "load" button, the frontend invokes the backend's REST endpoint for retrieving historical data. The options specified in the pane's widgets are transmitted to the backend through URL query parameters. The backend then executes a Flux query that is dynamically constructed based on these parameters. The 'start' and 'end' parameters define the arguments to the Flux *range* function, while the interval parameter is passed to the Flux's *aggregateWindow* function's *every* argument. If the parameter that is mapped to the switch is toggled on, then the backend receives a list of location IDs, which will be converted into a *filter* for the Flux query, applied between the *range* and *aggregateWindow* invocations. In addition to the parameters defined by the user, a parameter in the backend's configuration

file is available to specify additional custom filters. One use case for these custom filters is filtering out metrics with non-numerical values the platform cannot handle.

4.4.3 Live data

The platform supports visualization of near real-time data. When the user clicks on the button for live data in the loading pane, the frontend invokes the backend's REST endpoint for live data. The backend then queries the database for the most recent timestamp across all locations and all metrics. A second query is then executed to fetch the crowding data. This query depends on three variables: 'last_timestamp', the timestamp received as the result of the previous query, an 'offset' value, and an 'interval' value. The last two values are read from a configuration file and are specified as time duration. The query is constructed by selecting a date/time range spanning from $last\_timestamp - offset$ until $last\_timestamp$, and then aggregating the results according to $interval$, finally applying the *mean* function to the aggregated groups to average the data.

The frontend then periodically invokes the same endpoint, which continues to behave the same way as in the first invocation. The idea is for the state variable that holds the crowding data to be replaced by the data in the backend's response. Updating a state variable in React triggers a rerender, which means that the visualizations are updated to show the new data. Before doing that, however, a check is made to see if the timestamp currently selected is contained in the timestamps of the new data. If so, in addition to updating the crowding data's state variable, the state variable holding the index of the selected timestamp is also updated so that the chosen timestamp stays the same. This is necessary as the timestamp index may differ in the new list of timestamps. Otherwise, two scenarios are possible. Due to how aggregation works in InfluxDB, the most recent timestamp may not be aligned according to the specified $interval$ time duration. For instance, for $interval = 5min$, starting from 14:00, the result contains time steps for 14:00, 14:05, 14:10, ..., 14:55, 15:00, but the last time step may be 15:03 or some other non-aligned time value. Consider the case when the selected time is 15:03 and the new data contains 15:00 and 15:05 timestamps, while the 15:03 timestamp is not present anymore. In this case, the intended behavior is to update the selected timestamp to the nearest higher timestamp, which in this case would be 15:05. Finally, there is the scenario where the selected timestamp is not contained in any of the new data's time steps, which can happen when the user is visualizing data for older timestamps. For this case, the frontend stops polling the backend for real-time data, and the state variable holding the crowding data is not updated. The real-time visualization can be said to have "paused". A button to the right of the slider can be pressed to resume the fetching of real-time data. This will result in updating the crowding data state variable with the new data, and the new selected timestamp becomes the latest timestamp available in the new data.

The initial implementation of real-time visualization had a different approach. Note that the query's result returned by the backend may be the same as a previously returned result. To save computing time and bandwidth, the alternative approach was for the backend to only return new data. The advantage of the approach described in the previous paragraphs versus this initial approach is that it is easier to handle late-arriving data, as data arriving for a period already in the frontend may be part of the result of the above-described query. In other words,

late-arriving data is handled "for free" without requiring any additional logic.

4.5 Forecasting

The platform can provide predictive data through forecasting models described later in this section. These models can be used to predict values in a time series. In the context of this platform, each location has its instance model that is used to predict the evolution of crowding at that location.

The backend runs a job in a separate thread that periodically polls the time-series database to check if new data has arrived. When new data is available, new predictions are made using forecasting models. The results are stored in memory and sent to the frontend when the latter invokes the REST endpoint for predictive values. As is the case for the live data mode, the user does not control the range of dates to be visualized nor the interval used for aggregation; these values depend on parameters defined in the backend's configuration file.

4.5.1 Forecasting models

The platform uses the same extensible design for forecasting models as the one presented for the ingestion script in subsection 4.4.1. Different models can be defined as modules that can be dynamically loaded according to the backend's configuration file. Some parameters are always required, while others are specific to the chosen model. The decision to allow for different models in the platform stems from a desire to compare their performance. Two models are present in the platform's codebase: LSTM and SARIMA. Evaluations of these models using crowding data were carried out and are presented in subsection 5.2.3.

LSTM (Long Short-Term Memory) is a type of Recurrent Neural Network (RNN). RNN architectures enable data propagation from earlier events to current processing steps, making them suitable for time-series modeling [48]. Older RNNs, however, fail to learn over periods greater than 5-10 discrete time steps due to the vanishing error problem [15]. LSTM is one solution that addresses this problem.

ARIMA (AutoRegressive Integrated Moving Average) models are statistical models capable of handling many time-series patterns [24]. SARIMA (Seasonal ARIMA) is an extension of ARIMA that includes seasonal information, making it better at adapting to the seasonality that we know a priori to probably exist in some scenarios, such as city-wide pedestrian/mobile device counts.

As LSTM and SARIMA do not use spatial information for their predictions, one model is trained for each location. Some models account for spatial information, such as deep learning architectures with spatiotemporal graphs [54]. For STGCN, one of the first such models, there exists an implementation on GitHub with hundreds of stars at the time of writing, which indicates some degree of quality. Tests were carried out to investigate the viability of its integration with the platform. However, the training times were much higher than those of the other models, even with GPU acceleration enabled, although measurements of this training time were not recorded. Another drawback of this implementation is that it has incompatibilities with the Python version and with other libraries used in our platform. A solution to this problem would

be to adopt a different architecture where the forecasting models are microservices running on different containers. Due to these factors and time constraints that made it unfeasible to address them, this model was not included in the platform.

Recall that the backend of the presented platform is implemented in Python. This is advantageous for implementing forecasting capabilities, as many widely used statistical and machine-learning libraries exist for Python. For the SARIMA model, the SARIMAX implementation of the statsmodels library was used. SARIMAX is an extension of SARIMA that can integrate exogenous (explanatory) factors [28]. Our implementation does not account for exogenous factors, effectively making the model a SARIMA model, even though the class used in the platform's code is called SARIMAX. For LSTM, PyTorch was used. This library provides high-level constructs to build complex neural networks. It can use CUDA for parallelization when a compatible NVIDIA Graphics Processing Unit (GPU) is available, which has the potential to increase the speed of training of the neural networks dramatically.

Regarding the parameters of the models, the platform's SARIMA implementation requires only an integer defining the seasonality of the time series. In contrast, the LSTM implementation has three parameters: the size of the neural network's hidden layer, the learning rate, and the number of training epochs. These last two parameters are used for the optimization during training. LSTM, unlike SARIMA, requires itself to be trained before being able to output more accurate predictions. When training, the input data is split into a train set and a validation set. The Adam optimizer [26] is used for gradually improving the model's accuracy by comparing its predictions with the data in the validation set. This optimizer is parametrized by a learning rate and the number of training epochs (steps) it will take.

4.5.2 Implementation challenges

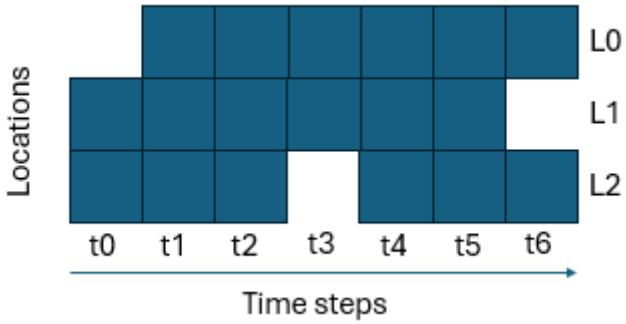


Figure 4.4: Illustration of how different locations may have missing data on some time steps, with $limit = 6$

One challenging aspect of the implementation is that new data may arrive at different times for different locations, allowing for the scenario where different locations have different date ranges for which data is available. Figure 4.4 shows examples for three locations, with different time steps being available for each, represented with blue squares. We can call a configuration parameter $limit$, which determines the maximum number of steps to be fetched for each location. In the figure, the value of $limit$ is 6. For location L0, the time step t0 is not loaded as there are 6 more recent time steps. L1, on the other hand, is missing data for the latest

time step (t6), so it loads t0, so its time step count is 6. The *minSteps* configuration parameter defines the minimum number of steps to predict. The locations whose data contains the latest time step will only forecast this minimum amount, as is the case of L0. If $minSteps = 3$, then 3 steps will be predicted for L0. L1, on the other hand, has a "gap" between its last time step and the most recent time step, making it so t6 needs to be predicted, in addition to the following 3 steps. This results in the platform displaying both predicted and real values for some time steps, albeit in different locations. While this may be counterintuitive, the alternative would be for L1 to miss data in t6.

Figure 4.4 also illustrates another scenario. L3 has missing time steps in the "middle" of the time range. There may be several different approaches to deal with these missing values, depending on the type of model used for forecasting, as some models can deal with missing values (e.g. SARIMAX [36]). Other approaches include estimating the missing value with some other model that can handle missing values, estimating the missing value by determining the means between the values present in the surrounding time steps, or dropping the location entirely from the result. The developed platform uses this last approach for the sake of simplicity, although in many real world scenarios this would not acceptable.

There is yet another scenario that must be taken into consideration. Consider again location L0 in Figure 4.4. The time steps correspond to the latest date range for which there is data in the InfluxDB instance, having some duration specified for aggregation purposes. Let us consider the duration used for aggregation is one hour and that the 6 time steps of L0 correspond to the last 6 hours counting from the current time. Imagine another location, L3, for which the most recent data is from a month ago. This scenario is similar to L1 regarding the "shape" of its time steps in that there is a gap between its last time step and t6. However, applying the same method would not make sense, as forecasting one month's time steps would result in an enormous uncertainty relative to the other locations. For this reason, a configuration parameter specifies a maximum time duration for that gap. A location that has a gap with a higher duration is discarded from the result.

4.5.3 Uncertainty quantification

As mentioned in subsection 3.3.7, the platform must be able to determine and present a measure of "wrongness" of the predictions, which we may call uncertainty.

One way of understanding uncertainty is through explanations of its different types. There are at least two types of uncertainty in a forecasting model: aleatoric (or statistical) uncertainty and epistemic (or systematic) uncertainty [23]. Aleatoric uncertainty refers to random events that are inherently unpredictable, such as the result of flipping a coin. Epistemic uncertainty, conversely, concerns the error that stems from not considering relevant variables to make a correct prediction or not having a model good enough even in the presence of perfect information. Considering forecasting based on the crowding datasets meant to be used in our platform, it can be seen how both types of uncertainty are present. The crowding phenomenon is too complex to be modeled with a single variable (number of pedestrians), resulting in epistemic uncertainty. But even in the presence of perfect information and the best possible model, the free will of pedestrians is a source of aleatoric uncertainty.

Neither the SARIMA or LSTM implementations provided by libraries mentioned in subsection 4.5.1 have a native concept of uncertainty. While the implementation details of uncertainty quantification for the two models are different, the general approach is the same: have several iterations of predicting for the same location instead of just one. If the predictions are not deterministic, we obtain sets of different values for each step. The sets are then ordered and their quartiles are calculated. Five quartiles are determined: Q0 (minimum value), Q1 (25th percentile), Q2 (median), Q3 (75th percentile) and Q4 (maximum). The visualization idioms in the frontend make use of these quartile values.

Regarding the implementation specifics for SARIMA, the library-provided SARIMAX class has a method called "simulate" that can be called after training to obtain predictive values. This method takes as a parameter the number of steps to forecast. The prediction mechanism consists of iterating a fixed number of times on the simulate method with the same parameters, resulting in different values each time.

For LSTM, the solution is not as straightforward, as predicting with a trained model on the same input results in the same output. Altering the neural network architecture to obtain uncertainty quantification is a possible solution (see [29, 47]). However, such models' implementations are not easy to find on the internet. A simpler solution is using the Monte Carlo (MC) dropout technique [1, 14]. This technique is easy to implement in PyTorch using its dropout layer module. Given some probability, a dropout layer randomly drops units in the neural network. The dropout layer was initially intended as a regularization technique and to avoid co-adaptation of neurons [22]. For these purposes, the dropout was performed during training and suppressed when predicting. But for MC dropout, the dropout is performed at prediction time [43]. This way, we obtain different values for the same input, as each prediction can potentially have a different network configuration.

4.5.4 Uncertainty visualization

Two different visualization idioms are used in the platform to display uncertainty quantification. Both can be seen in Figure 4.5. A switch widget is available to enable/disable the uncertainty visualizations.

One of the two idioms affects the line chart on the bottom left. As when the uncertainty visualization is disabled, it shows black dots representing the sum of the values for the selected locations. The approach for the uncertainty visualization is similar. However, the black dots can represent both real values and the Q2 values of the simulations, depending on whether real or predicted data is available. In addition, red points are displayed for the sums of Q0 (minimum) and Q4 (maximum) values, and the area between the lines connecting these points is filled with pink. These modifications to the line graph visualization allow the user to see the increase in uncertainty as it accumulates over time. This can be seen in Figure 4.5, for which four days of data were loaded into the application. The data for the first two days are real data, while the data for the last two days are predicted values. Notice how the peaks of the Q4 line are higher on the last day compared to the previous day, even though the Q2 line has similar peaks.

Figure 4.5: Screenshot of the platform displaying uncertainty measures in the 3D visualization and in the line chart

The other uncertainty visualization idiom transforms the 3D cylinders into a 3D representation inspired by box plots, also known as box-and-whisker plots. A semi-transparent cylinder encloses the 3D box plot. The latter consists of several cylinders with different radii. The cylinders with the larger radius are analogous to the boxes in box plots, representing the range of values between Q1 and Q2, and between Q2 and Q3, with a differently colored cylinder of fixed size in between representing Q2. The cylinders with the smaller radius are analogous to the whiskers in a box plot, representing the range of values between Q0 and Q1, and between Q3 and Q4.

4.6 Platform configuration & security management

This section presents aspects related to the platform's configuration and deployment, as well as the mechanisms that enable its secure operation. First, the access control mechanisms to manage user permissions and safeguard sensitive data are described. Next, the configuration scripts used to automate the setup of system components are outlined. Finally, the deployment process is presented, explaining the tools involved in deploying the platform across different environments.

4.6.1 Access control

Since the backend uses Flask as its REST framework, flask-login, a plugin for Flask, is used for protecting the backend's endpoints. This plugin stores the user's ID in the Flask Session. It provides decorators for endpoints so that only logged-in users can access them, which are applied to the endpoints used for fetching historical, real-time, or predictive data.

Additional endpoints exist for access control management, such as logging in and out, user creation and deletion, resetting and defining a new password in case it is forgotten. The platform employs role-based access control, such that the initial user is defined to be the super user, having unrestricted access to all endpoints, while other users can have administrator privileges or be regular accounts.

MariaDB, a relational database management system, is used for storing user data, including names, emails, role, and credentials. The credentials are stored as hashed and salted passwords using the bcrypt library that implements the hashing algorithm of the same name [38].

Administrators can create new accounts. When creating a new account, an email with a validation link for activating the account is sent. Super users can do all that administrators can do and can additionally grant or remove administrator privileges to existing accounts.

Another Flask plugin is used in the backend, Flask-Limiter, which is used to rate-limit specific endpoints to prevent brute-force attacks. The number of allowed requests per unit of time for each unique IP can be parameterized through the backend's configuration file. For storage, Flask-Limiter requires an in-memory database, such as Memcached.

The frontend provides several web pages that implement the interfaces for the access control management mechanisms, namely logging in, activating an account, resetting an account's password, and managing existing accounts.

4.6.2 Configuration scripts

The platform's codebase includes Python scripts for configuration purposes described in the following subsections. These scripts serve the following purposes:

1. Determine 'cap' values for metrics
2. Map locations to parishes
3. Calculate carrying capacities

In the backend's configuration file, a property called 'cap' is defined for each available metric in the crowding data, representing what would be considered a very high value for that metric. This is needed to scale the cylinders in the main visualization to the values of the metric. It is also used to calculate the color of the cylinders if the 'Hue' option has been selected by the user. The derivation of values for this 'cap' property is left to the platform administrator to decide. Using Flux's 'quantile' function, we have created a script that automates this process for data residing in InfluxDB. Although this approach is systematic, the choice of quantile is arbitrary. Interaction with the application is required to verify that the chosen 'cap' values are reasonable. Gathering feedback from users could be one way of validating these values.

As mentioned in subsection 4.2.2, locations can be grouped into regions, which we call parishes. A script is available that given a GeoJSON file with a list of parishes and another GeoJSON file with a list of locations, creates a JSON file with a list of location IDs for each parish. The backend's configuration file expects a filepath for this resulting file that maps locations to parishes, which is then used by a dropdown menu in the frontend to allow the selection of whole parishes at a time.

For the calculation of carrying capacities, there are two different scripts as follows:

- Given a GeoJSON file with a list of locations defined as polygons, the first script calculates their carrying capacities, outputting another GeoJSON file with the locations' carrying capacities defined in their features' properties;

- Given a GeoJSON file with a list of locations defined as points, the second script calculates their carrying capacities by determining their areas of influence on the map.

As for the latter, these areas of influence are their isochrone areas as described in subsection 4.3.2, using GraphHopper. The time limit that defines the isochrones is specified as a parameter to the script. The script outputs a GeoJSON file with the locations' isochrone areas as the features' geometries, and the original points' coordinates and carrying capacities defined in the features' properties. This script also takes as input a binary file containing OpenStreetMap data, and makes use of Osmium Tool to reduce the size of this file by reducing its area to a bounding box that contains all possible isochrone areas given the input GeoJSON and the time limit parameter. This file size reduction can greatly decrease the time and space requirements of the isochrone calculation service of GraphHopper.

Both carrying capacity scripts use MongoDB for its geographic indexing capabilities.

4.6.3 Deployment

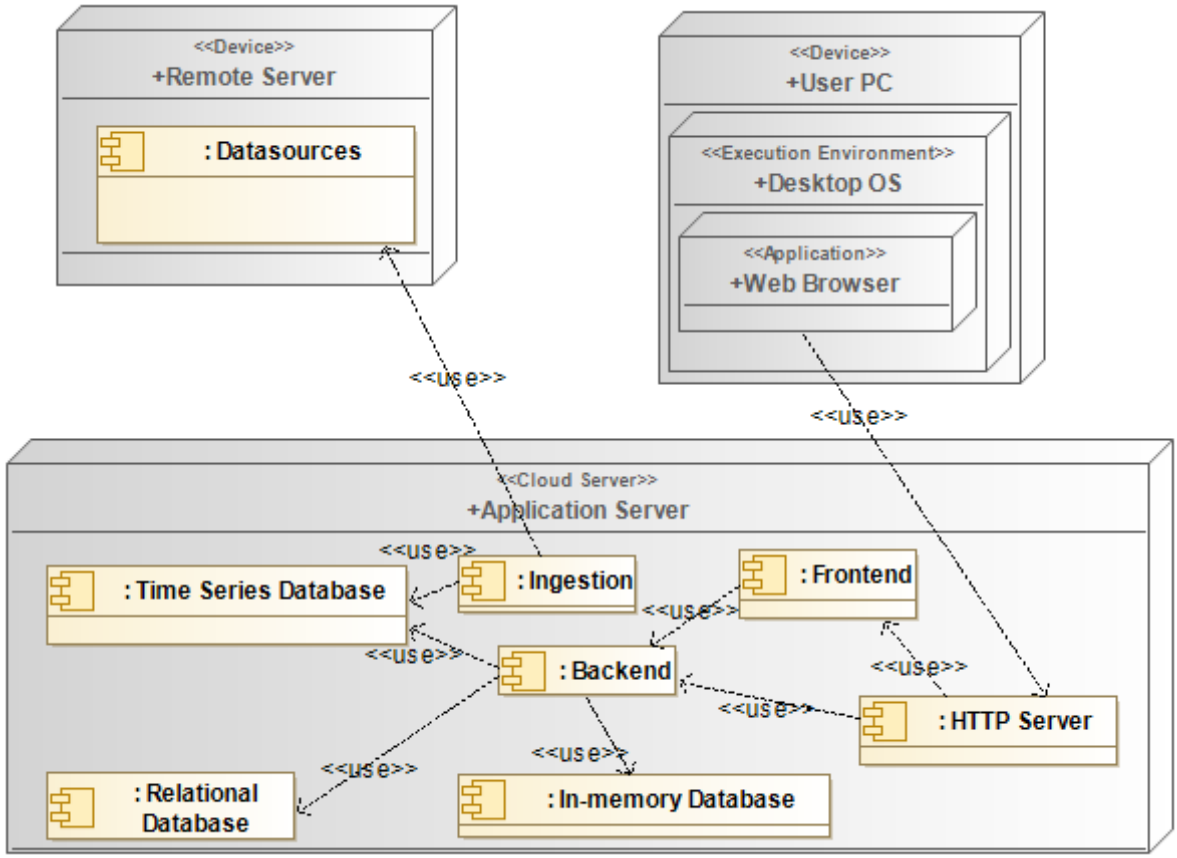


Figure 4.6: UML deployment diagram for the platform

Figure 4.6 shows the UML deployment diagram for the platform. It consists of three nodes corresponding to different devices, with subnodes and components indicating their internal structures. Interaction between components is indicated with arrows, with the arrowheads pointing at the components that receive the requests.

The components shown in 'Remote Server' and 'Application Server' can be in different machines or all in one server, as long as there is TCP connectivity on the ports used for their respective protocols. With this in mind, what is shown in the figure is the most likely scenario for an SME, with the crowding data being available in one server and all the software components required for the platform itself on a single cloud server.

The main components that are implemented by the platform's code have corresponding Dockerfiles for building Docker images, and a Docker Compose configuration file is also available for orchestrating the various containers. However, Docker is not required to deploy the platform, and the GitHub repository includes example shell scripts for running the components without containerization.

The 'HTTP Server' component is optional and is used as a reverse proxy forwarding incoming requests onto the backend and frontend. It is used for two purposes: avoiding cross-origin requests between frontend and backend by presenting a single TCP port to the end-user client, and allowing for Transport Layer Security (TLS) configuration in a single point, instead of having to configure TLS for both backend and frontend [1]. The default configuration in the platform's GitHub repository uses nginx as the HTTP server, although other solutions may be used.

More information about the platform's deployment can be provided for completeness. As mentioned previously, the backend is a Flask application. Flask applications should be served by a dedicated Web Server Gateway Interface (WSGI) server when deploying to production. The platform's default configuration uses uWSGI, which has good interoperability with nginx, but other solutions may be used.

[1] TLS is crucial for encrypting the data that is shared between server and client.

Chapter 5.

Demonstration & Evaluation

Contents

This chapter demonstrates the platform's usage in two different scenarios, each in different instances of the platform, one using crowding data for Melbourne and another using crowding data for Lisbon. The multiple contexts in which the demonstrations were reified are then described.

The platform evaluation along four dimensions is then presented: usability, performance, forecasting ability, and peer review.

[This page has been intentionally left blank]

Chapter 5
Demonstration & Evaluation

5.1 Demonstration

In the Demonstration step of the DSR approach, the goal is to showcase the artifact in action, illustrating how it can solve the problem or achieve its purpose. Demonstrations are often interactive, involving stakeholders or users to test the artifact's practical application in real-world scenarios. We start by describing the scenarios used and then report on the instances where the latter were used to highlight the platform's functionalities through a multichannel online site, several events (general public, professional, and academic), or hands-on experiences with students.

5.1.1 Demonstration scenarios

The proposed platform explored two scenarios with high levels of crowding: New Year's Eve in Melbourne and World Youth Day in Lisbon. The two subsections below correspond to these scenarios. They begin with a description of the datasets used and then analyze the scenarios.

5.1.1.1 New Year's Eve in Melbourne

We used the crowding data publicly available from Melbourne's Pedestrian Counting System. It currently includes movement sensors at 93 locations in the city, selected based on three criteria: commercial and event activity, regular use by pedestrians, and the flow of entering and exiting these areas. The sensors are installed at high points, such as on top of lampposts, to cover counting zones on the pavement below. Pedestrian counts are stored locally and transferred to a central server every 10 minutes via a 3G communication box.

There are several ways to obtain the data recorded by this system. An 'export' functionality in its web page allows us to obtain the data for a given month in CSV format. Another way to obtain data is through the API endpoints in Opendatasoft. Three related datasets are available for querying and exporting through the API:

1. a dataset containing sensor information, including names and geographic coordinates;
2. a dataset containing pedestrian counts per hour, updated every month;
3. a dataset containing pedestrian counts per minute, updated every 15 minutes.

The two last datasets can be suitable for different purposes. The second dataset can be used as an alternative to manually loading the available CSV files, while the third is real-time data. Concerning the platform described in this dissertation, one dataset can be used for viewing historical data, while the other can be used for real-time visualization. Recall that, as was said in subsection 4.4.1, there exists a connector in the platform for importing data from Opendatasoft's APIs, which can be used for ingesting data from these two datasets into an InfluxDB instance.

New Year's Eve is one of the busiest events in Melbourne in the area covered by sensors. So we used the platform to visualize and analyze data from New Year's Eve 2023 to 2024. For this purpose, the CSV files in the Pedestrian Counting System's web page for December 2023 and January 2024 were manually loaded into an InfluxDB instance.

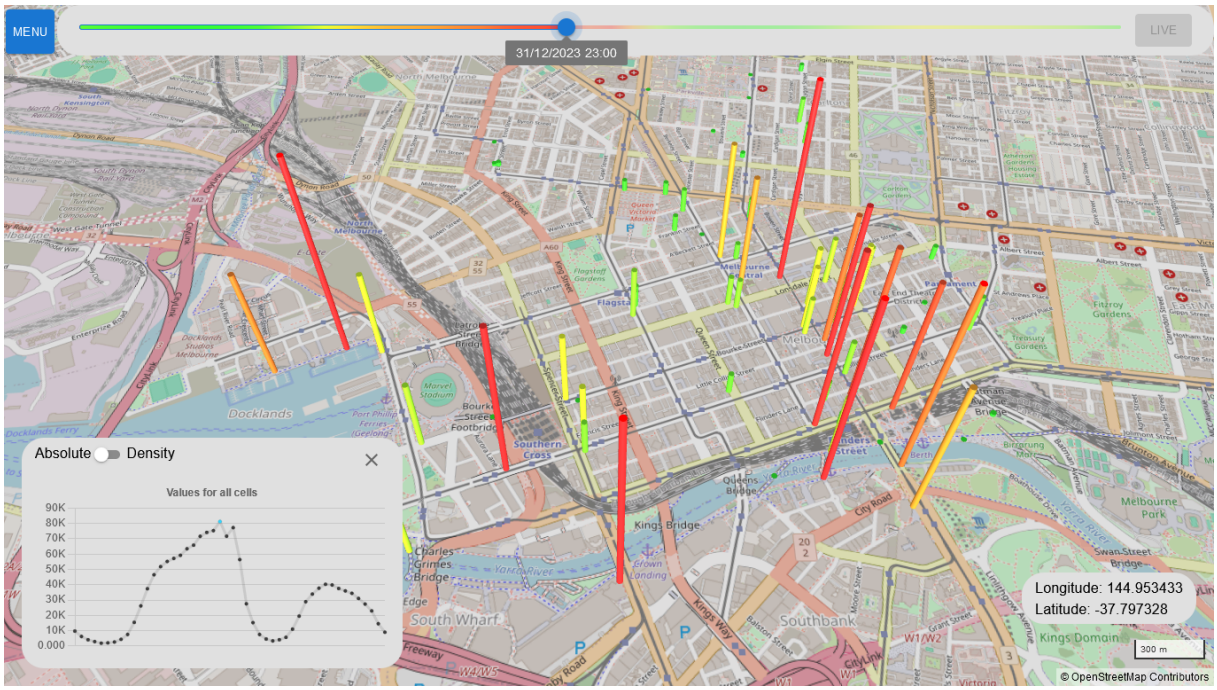


Figure 5.1: Melbourne, during the peak hour of New Year's Eve (31 December 2023)

We found that there was a peak in crowding on December 31st at 11 p.m., with around 81,000 pedestrians detected (Figure 5.1). The peak on the following day was around 40,000 pedestrians. The platform user can obtain these values by observing the line graph and then hovering over the points corresponding to local maxima, giving them the date/time and total number of pedestrians detected by all sensors.

As discussed in subsection 4.3.2, the platform considers each sensor's carrying capacities. The color of each cylinder in the figure represents a 'density' of pedestrians relative to their carrying capacities. The values obtained by hovering points in the line graph can also be shown as densities when the user toggles on the switch in the line graph's pane. Doing this tells us that the global maximum for this New Year's Eve is around 91 pedestrians per hectare, while the maximum for January 1st is around 45 pedestrians per hectare. Note that this is the result of summing the pedestrians detected by all sensors, divided by the total walkable area of the zones of influence of all sensors. The values for individual sensors vary widely, with some having one pedestrian per hectare while others surpass 500 pedestrians per hectare.

5.1.1.2 World Youth Day 2023

The instance of the platform configured for Lisbon uses data from the LxDataLab initiative. This results from partnerships between the Municipality of Lisbon and several organizations. LxDataLab provides datasets in the context of several 'challenges' that aim to improve the city's

management. The author of this dissertation was given access to one of these datasets, hereafter referred to as the "LxDataLab dataset".

This dataset divides the city into 3,743 square cells, each containing values for several metrics. These values are tagged with specific date and time information, and the records are spaced at 5-minute intervals. The metrics are derived from mobile devices detected by a mobile network operator within the city. For example, one metric, labeled 'C1', represents the total number of devices in a given cell, while another metric, 'C2', tracks the number of devices in roaming mode.

World Youth Day (WYD) is an event for young people organized by the Catholic Church. The 2023 edition was held in Lisbon from 1 to 6 August. The event consisted of several activities in different places in the city.

After loading the data from the LxDataLab dataset for a period that covered the event, we have concluded that the peak of the total number of devices detected in the city occurred on 4 August at 6 p.m. At that time, the WYD activity was taking place in the location *Parque Eduardo VII*. The crowd's peak in this area was at 7 p.m., which we can verify by selecting this area (see Figure 5.2). The line graph shows three peaks for this location, corresponding to the three WYD activities there.

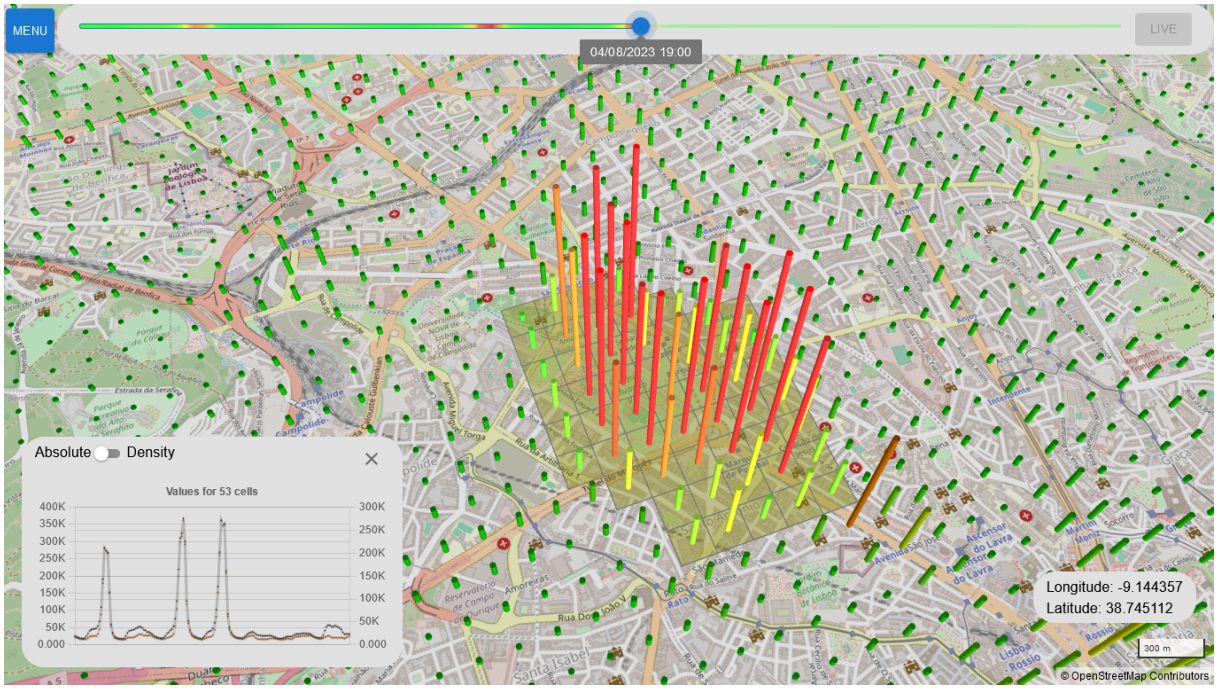
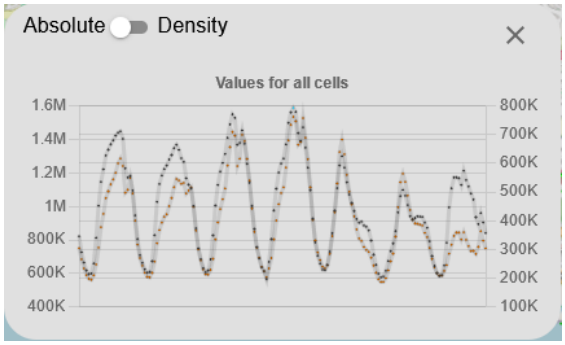


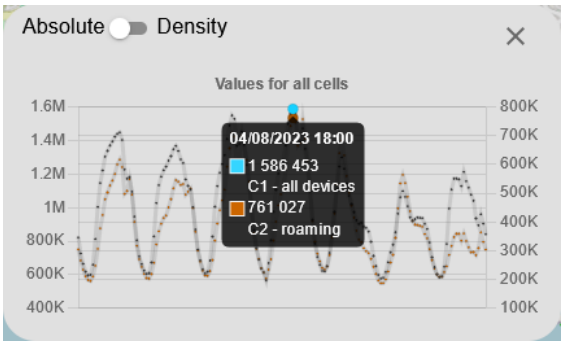
Figure 5.2: Crowd at *Parque Eduardo VII* during the WYD Lisbon 2023 event

The August 5th event occurred at the location *Parque Tejo-Trancão*. Moving the slider through the different hours of that day, we can see a high density of detected devices near that location.

The offered user experience is rich. For example, selecting two metrics at the same time, namely, C1 (total number of devices detected) and C2 (total number of devices detected that were roaming), draws two lines on the line chart, each with its color (Figure 5.3). This visualization allowed us to see the fluctuation in the number of tourists in Lisbon for WYD, assuming



Line graph for WYD week



Line graph for WYD week with the global maximum being hovered

Figure 5.3: Line graph for WYD week showing C1 and C2 metrics

that tourists were roaming with their mobile devices. The last peak in the graph for the C1 metric corresponds to August 8th, two days after the end of WYD. Perhaps unsurprisingly, the relative number of tourists was much lower that day, which is immediately apparent to the user by the gap between the two lines.

Another observation can be made regarding these two metrics. When *Parque Eduardo VII* is selected, the peak at 7 p.m. that was previously mentioned had registered about 354,000 total devices and 264,000 devices in roaming. We can infer from this that around 75% of this activity's participants were tourists.

5.1.2 Demonstration instances

RESETTING project site

The RESETTING project under which this work was developed has a homepage for the contributions made by Iscte-IUL to the project. It contains a page for the platform developed in the context of this dissertation, with links and subpages providing YouTube video demos, a user manual and a technical report, and updated information on the dissemination actions that will now be described.

1st International Posters & Demos Workshop on Smart Tourism demo

This event occurred at the Iscte-IUL II building patio on January 12, 2023. All attending RESETTING project members, excluding the Portuguese ones (since they coauthored the presented materials), voted on the demos and posters. The demo on the platform described in this dissertation obtained the "Best Demo Award". The poster presented was titled "Geo-temporal crowding visualization in Lisbon". Further details can be found here.

Demo at the inauguration day of the Iscte - Knowledge and Innovation building

A running demonstration of the platform was offered at the ISTAR-IUL demo room to all participants of the inauguration day of the "Iscte - Knowledge and Innovation" building, that occurred on November 20, 2023. This demo allowed showcasing the platform's capabilities in a formal, real-world setting. The prime minister and several ministers of his cabinet attended the

demo, while two national TV channels covered the global visit. Individual demonstrations were given to several people, who then experimented with the platform and answered a task load questionnaire, the results of which were used for a usability evaluation (see subsection 5.2.1). Further details can be found here.

Demo and hands-on experience to the delegation of Lafayette College

A demonstration of the visualization tool, followed by hands-on experience with professors and students from the US Lafayette College, occurred at the Iscte-IUL computer labs on January 17, 2024. Delegation participants could interact with the artifact and provide insight into its practical utility and usability by answering a usability questionnaire (see subsection 5.2.1). Further details can be found here.

2nd International Posters & Demos Workshop on Smart Tourism

This event occurred at two locations in Spain on consecutive days – Tecnoparc (Reus) and Auditori Diputació (Tarragona) – on May 22-23, 2024. A poster titled "Geo-temporal crowding visualization platform" was shown and demos were performed on demand for participants in both locations. Further details can be found here.

LxDataLab Meeting demo

A presentation titled "Platform for visualizing spatiotemporal tourism crowding" (in Portuguese) was offered at the 2nd Meeting of the Lisbon Urban Data Laboratory (LxDataLab), that occurred at the CML Social Services Auditorium, on May 29, 2024. This was an initiative of the Lisbon Center for Urban Management and Intelligence (CGIUL), a department of the Lisbon Municipal Council (CML) that promotes Smart City initiatives within the framework of collaboration protocols celebrated with 19 universities and higher education and scientific research institutions that have seized the opportunity to respond to concrete city challenges using real data. At the same time, CML has the chance to test analytical solutions capable of promoting innovation, efficiency, and proactivity in the services provided to citizens.

5.2 Evaluation

As expected in the DSR methodology, carefully designed methods are essential to assessing the effectiveness and efficiency of the produced artifact. To this end, the platform's usability and performance were evaluated. The predictive performance of the forecasting models included in the platform was also evaluated. In parallel, several papers were submitted for peer evaluation in three conferences and one journal.

5.2.1 Usability evaluation

Two assessments were conducted, each using different usability surveys and with different populations, to assess the platform's usability.

In both assessments, participants were given a task that required them to use some of the platform's features, namely selecting a region of interest, changing the metric displayed, and interpreting the results on the platform's line graph. To ensure that all participants had access to the same crowding data, in the first assessment, the data was loaded before the participant interacted with the platform, and in the second assessment, the button for loading historical data (as shown in subsection 4.4.2) resulted in always loading the same crowding data, regardless of the parameters that might be present in the historical data loading pane.

For the first assessment, a standardized three-step individual experience was designed where the participant was first shown the different functionalities of the platform and then had to perform a task using the platform to answer a question. The event was the Lisbon music festival *Rock in Rio Lisboa 2022*. Finally, the participant had to complete a widely used questionnaire, the NASA-Task Load Index (NASA-TLX) [19], to measure the mental workload the subject attributes to perform the abovementioned task.

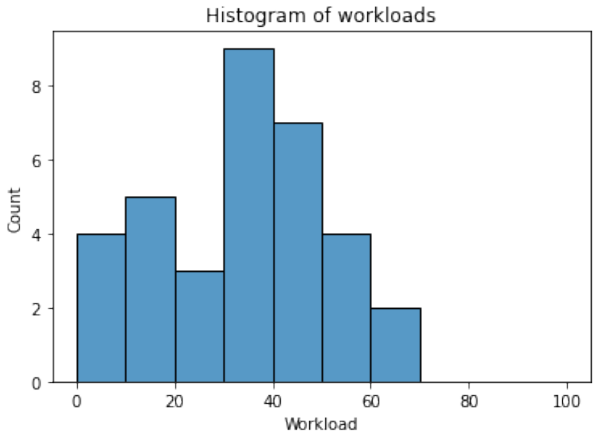


Figure 5.4: Histogram of NASA-TLX results

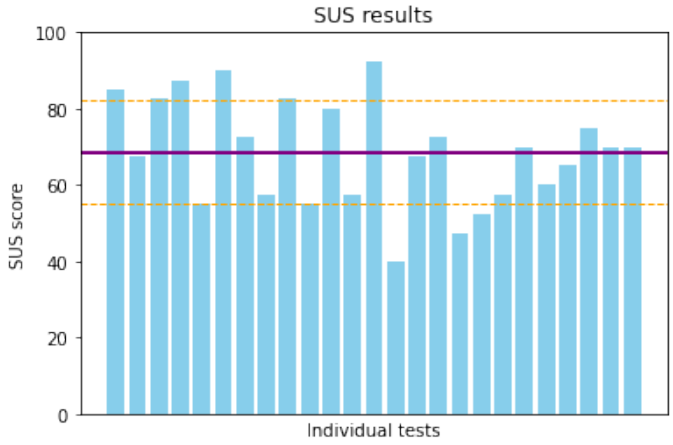


Figure 5.5: Individual results of SUS (purple line: average; orange lines: average ± standard deviation)

Thirty-four volunteers with varying levels of technological literacy participated in this experiment. Figure 5.4 shows the resulting histogram. The average score for the NASA-TLX,

on a scale of 0 to 100, was 33.45. According to the analysis presented in [18], this score is just above the 25th quartile for tasks performed on a computer. In other words, the platform was considered low-to-medium difficult to use.

Participants' errors and questions while using the platform were also recorded. This led to minor changes to the platform's user interface to improve usability.

A second usability evaluation was then carried out. This time, the demonstrations were given to groups of several people simultaneously. The platform was deployed on a server that was publicly accessible on the Internet so the participants could interact with the platform on their computers. They were asked to use the platform to answer three questions about crowding at the *World Youth Day Lisbon* event. Participants then completed two surveys, the System Usability Scale (SUS) and the User Experience Questionnaire (UEQ).

Thirty-two volunteers, again with varying levels of technological literacy, participated in this experiment. The results are shown in Figure 5.5. On a scale of 0 to 100, the average SUS score was 68.36. This value is close to the average SUS score in [10]. As for the UEQ, an official tool available on the UEQ website was used to analyze the results, which calculates scores for different dimensions: attractiveness, perspicuity, efficiency, dependability, stimulation, and novelty. The tool provides benchmarks for these dimensions based on 468 studies of different products. The mean scores for these dimensions in our results were almost all around the average, except for novelty, which had a score that was considered "Good".

The data collected during these two usability experiments is available at https://doi.org/10.5281/zenodo.11389582.

The results of the two usability assessments are not directly comparable, as different questionnaires were used in each. However, when taking into account the meta-studies on the scores obtained by these questionnaires, one can see that the results of the first assessment were better than the second one. The fact that the initial assessment was through individual demonstrations, while the second was made with a shared demonstration for a wider audience, who performed the tasks simultaneously on different PCs, may contribute to the difference.

5.2.2 Performance evaluation

The platform's performance was measured by looking at the request's processing time in the backend after fetching the data and the processing time in the frontend after receiving the data from the backend. These times were measured on increasing numbers of records, from around 1 million records to 5 million records, where a record is a unique combination of $(date/time, location, metric)$. The number of date/time records increased by 12 in each test, while the number of metrics was always 20, and the number of locations was always equal to the 3743 locations in the LxDataLab dataset.

Overall, the processing time in the frontend is negligible, taking about 1.3 seconds in the test with the largest amount of data. For the same scenario, the processing in the backend (after the query) reaches 134 seconds. While the bottleneck is the query, the post-query processing time increases more than linearly, as seen in Figure 5.6.

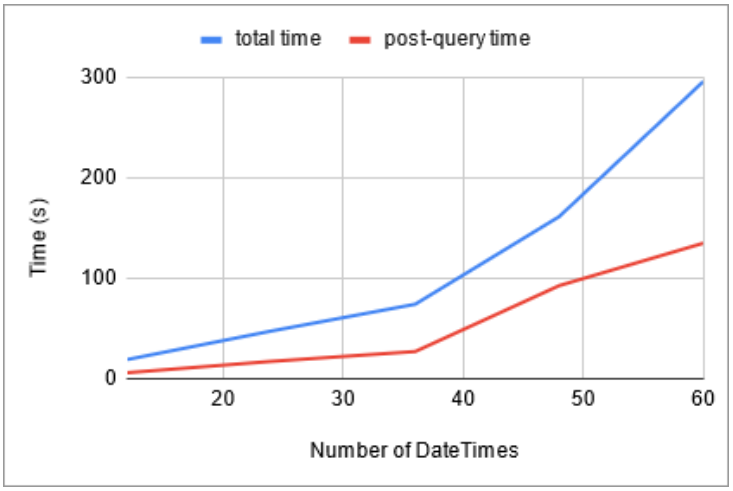


Figure 5.6: Time spent in backend for historical data loading

The relative amount of time spent in the backend on post-query processing was also analyzed (Figure 5.7); for 1 million records, it took about 29% of the total time to process the data, while for 5 million records, it took about 45%.

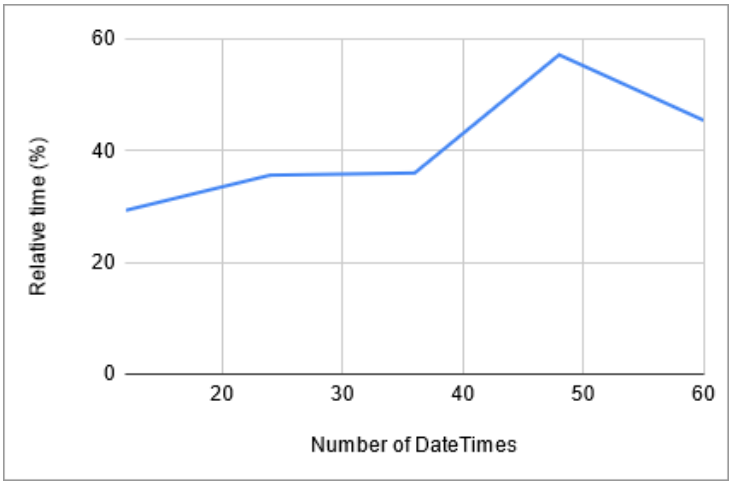


Figure 5.7: Relative time spent in backend post-query processing for historical data loading

These results show that the platform was not built with high performance in mind. Indeed, the list of non-functional requirements in section 3.4 contains a single requirement regarding performance, namely that the application's user interface should be responsive, which was met. Loading large volumes of crowding data, on the other hand, can take a considerable amount of time, which is not only related to the query made to the time-series database where the crowding data is stored, as the relative time of post-query processing also increases with the number of records being fetched. Recall that the platform's backend was implemented in Python, one of the least performant languages out of the most popular ones[1].

[1] As an example, one benchmark comparing many programming languages shows that Python is one of the slower languages

5.2.3 Forecasting evaluation

The Melbourne dataset containing pedestrian counts for every minute, mentioned in subsubsection 5.1.1.1, was used to measure the predictive power of the models available on the platform. The data was ingested into an InfluxDB instance, and a query aggregated the values into 1-hour groups. The aggregation step was necessary because the time-series for the different locations had different timestamps and needed to be homogenized. The time range for the values that were used for evaluation purposes started at 2024 October 13th at 5 AM (Melbourne time) and ended at 2024 October 18th also at 5 AM. This resulted in exactly five days' worth of data or 120 timestamps. This data was split into a training set and a test set, following a typical 80/20 split, meaning 96 timestamps for the training set and 24 timestamps for the test set. When forecasting time-series, the data must be ordered chronologically prior to the division into training set and test set [17]. For this scenario, this means that the training set contains data for the first four days, while the test set contains data for the last day.

Note that many of the locations had missing values. As described in section 4.5, the platform discards locations with missing data when forecasting. To ensure consistency with that behavior, locations with missing values were also discarded in the evaluation process. This resulted in a total of 55 locations being considered.

Both the LSTM and SARIMA implementations in the platform, described in subsection 4.5.1, were evaluated. Concerning the parameters of the models for this evaluation, SARIMA had a seasonality of 24 timestamps, corresponding to a daily seasonality in this context. At the same time, for LSTM, the learning rate was set to 0.02, with 100 training epochs, and the hidden layer of the neural network had a size of 24 neurons. Recall that the platform's LSTM implementation makes use of a validation set; this set is obtained from the training set defined earlier, thereby effectively shortening its actual training set.

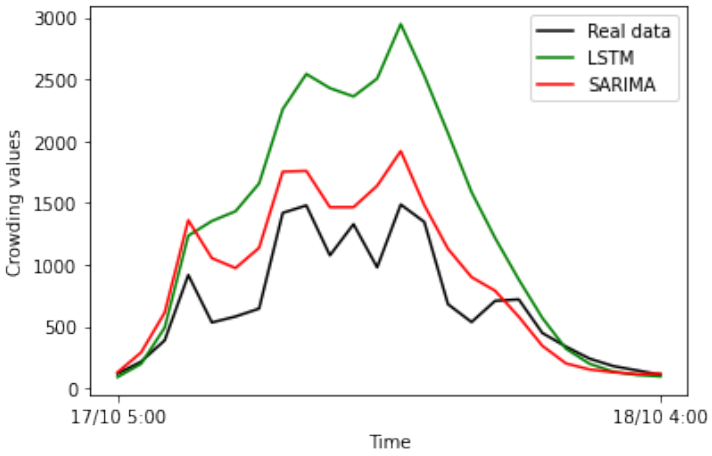


Figure 5.8: Comparison of real crowding values and forecasted values

A comparison of the values predicted by the models with the real data in the test set is shown in Figure 5.8. This graph shows the sums of crowding values for all 55 locations. As described in subsection 4.5.3, these models output several values for a given date/time, corresponding to the quartiles of the forecasting simulation results. Recall Figure 4.5, in which the platform's line graph visualization becomes a band graph, displaying the uncertainty of the

predictions based on those quartiles. For the purposes of this evaluation, 50 simulations were made for each prediction, and only their Q2 (median) values were considered.

Table 5.1: Error metrics for forecasting models

Model	RMSE	MAE	WMAPE
LSTM	26.81	12.52	0.99
SARIMA	10.19	5.32	0.42

Based on these results, several metrics for measuring predictive power were calculated, shown in Table 5.1. These metrics are Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), and Weighted Mean Absolute Percentage Error (WMAPE). Taking into account the previous graph (Figure 5.8), it is unsurprising that SARIMA has lower errors across all metrics compared to LSTM, as the graph clearly shows the SARIMA line to be closer to the real values.

There is a caveat to these results. Having two forecasting models in the platform, LSTM and SARIMA, allows a direct comparison of their predictive power. SARIMA outperforms LSTM, but this could change depending on several factors, such as the amount of training data fed into the models. Indeed, one study [28] compares ARIMA, SARIMA and LSTM for energy consumption forecasting and found LSTM to be prominent over ARIMA and SARIMA. LSTM is often described in the literature as a deep learning model. This term is related to the amount of training data, implying that the predictive power of the model increases when massive amounts of data are available, which is far from being the case in our evaluation. Another relevant aspect is the model parameters. In this evaluation, no attempt was made to fine-tune them, although doing so could increase the predictive power of the model. In addition, the LSTM implementation in the platform does not use the sliding window technique recommended in [17], which is another factor contributing to the model's poor performance compared to SARIMA.

5.2.4 Peer-review evaluation

Peer review serves as a validation activity because it involves expert scrutiny to assess the work's quality, rigor, and relevance, ensuring it aligns with established standards in the field. This process acts as a quality control mechanism, offering independent feedback that helps refine the research and strengthens its credibility and potential impact. The impartial evaluation by field experts is an external endorsement, confirming that the artifact or research meets accepted scientific standards and contributes meaningfully to the discipline, thus validating its applicability and value. We sought peer-review evaluation on four occasions during the lifetime of this research work. Further details and updates on the following peer-reviewed papers can be found at the corresponding dissemination page.

A digital transformation approach to scaffold tourism crowding management: pre-factum, on-factum, and post-factum

A full-fledged section about the visualization platform was included in this paper on tourism crowding management presented at the ECTI DAMT & NCON 2024 IEEE conference [9] in

Thailand. This paper was instrumental in launching a cooperation action between the Iscte-IUL RESETTING team and the team of Professor Pradorn Sureephong[2].

Plataforma para visualização geo-temporal de apinhamento turístico

This conference paper in Portuguese was presented at the XIV Congresso da Geografia Portuguesa (XIV CGP) at IGOT, Lisbon, on 14-17 November 2023 [44].

A Carrying Capacity Calculator for Pedestrians Using OpenStreetMap Data: Application to Urban Tourism and Public Spaces

This conference paper was accepted at the ENTER 2025 International eTourism Conference, to occur in Wroclaw, Poland, on 17-21 Feb., 2025 [5].

A Spatiotemporal Crowding Visualization Platform

Finally, we recently submitted this paper to a Q1 journal focusing on the intersection of technology and tourism, publishing research that advances theoretical and applied knowledge in this field [45].

[2]Vice Director at College of Arts, Media and Technology at Chiang Mai University

[This page has been intentionally left blank]

Chapter 6

Conclusions

Contents

This chapter concludes the dissertation by summarizing and discussing various aspects of the platform, drawing conclusions from the various steps of the adopted Design Science Research (DSR) methodology, and answering the research questions. Finally, topics for future work are listed.

[This page has been intentionally left blank]

Chapter 6
Conclusions

6.1 Discussion

In this dissertation, a spatiotemporal crowding visualization platform was described. The Design Science Research (DSR) methodology was adopted to construct this artifact. After the problem statement was formulated, requirements were elicited, defining the various capabilities that the platform would support, described in chapter 3. Key among these requirements is the platform's ability to support visualizing historical, live, and predictive data, as well as the ability to present crowding data relative to the carrying capacities of the considered locations. These features set this platform apart from commercial GIS applications such as ArcGIS Pro and QGIS.

Another important aspect of the platform is its extensible architecture regarding two different aspects. For the ingestion of data into the time-series database, new modules can be developed for integrating alternative data sources without the need to modify the existing source code. Regarding the platform's forecasting capabilities, besides the LSTM and SARIMA models that are included out of the box, other implementations can be included in the platform, again without needing to modify the existing source code.

Case studies in which the platform is used were presented in section 5.1, in two different scenarios: Melbourne and Lisbon. Events in which there was high levels of congestion were analyzed, and the usefulness of various features of the platform was demonstrated.

In that same chapter, in section 5.2, evaluations of the platform in four different domains were presented. These domains are usability, performance, forecasting ability and peer-review. Two usability assessments were conducted, using different questionnaires in each. Using meta studies regarding results obtained from these questionnaires, it can be seen that the platform's usability is considered average. Regarding the performance evaluation, although the client-side of the platform is responsive even on low-end PCs, the backend is quite slow when loading large volumes of data. For the forecasting evaluation, the predictive power of the two forecasting models in the platform, SARIMA and LSTM, was compared by using the same dataset. SARIMA clearly outperforms LSTM in the evaluation, but several limitations in this evaluation precludes us from concluding that SARIMA is a better model than LSTM. Lastly, work carried out in the context of the development of our platform has been accepted in peer-reviewed publications, which validates the quality and relevance of this work.

We will now answer the Research Questions (RQs) formulated in section 1.3. Recall RQ1: *Which visualization paradigm is best suited for exploring crowding data in cities, both in time and space?* As mentioned previously, many different idioms have been proposed for spatiotemporal visualizations, and many such examples have been collected in [3]. Regarding the visualizations used in the related work presented in chapter 2, several papers use visualizations based on cartographic maps, and line graphs have been used to display crowding data over time. Another

relevant input for generating an answer to this RQ is the availability of visualization idioms in existing software libraries, such as deck.gl.

We have chosen the three linked visualization idioms described in section 4.2. As seen in the demonstrations in section 5.1, the line graph that our platform's visualization paradigm has is sometimes more important than the main visualization itself. Overall, the platform's visualization paradigm follows the rule of thumb known as Shneiderman's Mantra [42]. That is, zooming out of the map provides a *overview* visualization, as does the line graph itself. The *zoom and filter* in the concept is provided by the ability to zoom and pan the map and the line graph. Finally, the *details on demand* is provided by the tooltips on the map, the line graph, and the timeline slider. By following this guideline, the platform allows the user to get an overview of the congestion of a region both in space and time, while at the same time getting details on demand.

RQ2 is: *How to deal with irregular geographical locations of crowding data collection points?* Regarding the presentation of the data, the visualization idioms used by the platform are usually not affected by the regularity of the geographic locations. The main 3D visualization uses cylinders on top of the locations, which does not require spatial interpolation, as would be the case for a heatmap. However, one scenario in which the irregularity of the locations may be relevant is for locations defined by points (sensors), where two points are so close together that the superimposed cylinders may overlap. The radius of the cylinders is configurable, but this configuration affects all cylinders, not just the problematic ones. However, if two points are much closer together than others, this is likely to indicate poor sensor placement.

There is also the issue of processing and transforming the data before displaying it. Again, geographic irregularity is not a problem. The factor that makes the most significant difference in this data processing is whether the sites are defined by polygons or points, which affects how the carrying capacity of the site is determined. For a polygon site, the carrying capacity is determined using an algorithm that subtracts all non-walkable areas from the total area of the polygon. For a point location, its isochrone area is calculated and then fed into the same algorithm.

As for RQ3, it is: *How to quantify the perception of crowding in a given location?* Part of the answer was given in the motivation presented previously for including this RQ: the concept of carrying capacity. Once determined for a location, crowd density can be quantified by dividing the raw number of people by the carrying capacity. In the related work of this dissertation (chapter 2), two papers claim the ability to determine the carrying capacity of locations. One of these [40] used a GIS platform to do so, but did not specify which GIS it was and how it determined carrying capacities. The other [55] used a 3D model, which is a method that is potentially more accurate than determining carrying capacities based on 2D cartographic data because it considers vertical space. However, the need for a 3D model is a serious drawback.

In the end, our approach to determining carrying capacity, described in subsection 4.3.2, was to apply the algorithm described in [5], which calculates the walkable area of locations based on data from OpenStreetMap. Recall from the same section that our platform has the ability to determine the carrying capacity of locations defined by points rather than polygons by first calculating an isochrone area and then applying the same algorithm.

6.2 Future work

The presented spatiotemporal crowding visualization platform is, like most software products, an incomplete work. This section lists several improvements that could be made. The first two subsections describe visualization and forecasting improvements, respectively. The third and last subsection mentions the estimation and visualization of crowd flows, which is a more ambitious proposal, as it suggests a different research direction, suitable for a Ph.D. project plan.

6.2.1 Visualization improvements

Adapting the three basic types of questions that can be answered by exploratory spatiotemporal visualization [6] to the domain of crowd visualization, we get the following specific questions:

- Given a time and a set of locations, what are the crowding levels (when + where → what);
- Given a time and a crowding level threshold, which locations cross the threshold (when + what → where);
- Given a set of locations and a specified threshold for crowding levels, at what times does the crowding at these locations exceed the threshold (where + what → when).

The presented platform is designed to answer the first question directly, while a feature that supports the other two use cases ('finding notable points') is currently missing and is part of planned future work.

Another important feature to be implemented later is the ability to use custom maps, such as indoor plans, instead of OpenStreetMap. This feature could be important for scenarios such as museums and other touristic offerings that take place inside a private, delimited space for which the spatial data is not available in OpenStreetMap, as described in the *Crowded Indoor Scenario* in section 1.2.

Possible visualization techniques are still being explored. We are currently experimenting with a three-dimensional surface covering the entire region of interest, using linear interpolation to estimate location values when no available data exists. KDE could be another approach, possibly more appropriate for irregular location topologies. This visualization can be seen as the three-dimensional counterpart to a traditional heatmap. The latter is a helpful metaphor for analyzing spatiotemporal data [27], which justifies interest in implementing the former. Note that such a visualization idiom would require a reassessment of RQ2 (*How to deal with irregular geographical location of crowding data collection points?*), as the answer given in this chapter is specific to the 3D visualization employed in the platform.

6.2.2 Forecasting improvements

As mentioned in subsection 4.5.3, the crowding phenomenon is too complex to be modeled by a single variable (past crowding counts). Although there will always be uncertainty for the reasons described in that subsection, it is possible to improve the prediction models by including additional data that affect crowding. Some examples of this are weather, as rain reduces pedestrian counts, and very high temperatures.

The urban fabric at each location is typically another variable that we should consider in our predictive model. A group of one hundred tourists walking through the open space of Praça do Comércio in Lisbon will be much faster than the same group walking the same distance but through the old neighborhood of narrow streets in Alfama, also in Lisbon.

Another variable in the context of pedestrian flow and urban design is the effect of attractive retail displays that slow pedestrian movement, often referred to as "friction"or "side friction". This term encompasses any element that diverts or slows the flow of pedestrians, such as window displays, street vendors, or even benches.

However, before enriching existing models with additional variables, we could rigorously evaluate existing model implementations that use spatial information, such as STGCN, mentioned in subsection 4.5.1.

Another planned improvement to the platform's forecasting capabilities is the estimation of missing values. Recall that in certain circumstances, the platform may discard locations with missing values when making predictions, which would be unacceptable in many real-world scenarios. Spatiotemporal estimation of missing values is a research topic relevant to overcoming this limitation of the platform.

6.2.3 Crowd flow estimation & visualization

In addition to visualizing crowd density in space and time, it would be helpful to visualize the flow of the crowd. This can be achieved by visualizing the direction of pedestrian movement. Ideally, this visualization should have street-level granularity, as this level will be most helpful to potential users of the presented platform, given the motivation outlined earlier.

Many traditional geo-visualizations are available for visualizing flows, like contour maps, shading maps, and hypsometric maps [20]. However, other techniques can use additional data, such as trajectory orientation: trajectory-based visualizations, vector fields, and network analysis.

The main challenge is inferring mobility flows from crowd density values without access to individual pedestrians' position or speed. Some work has been done in this area. One study addresses this problem in the context of traffic estimation of unmonitored roads, using fluid dynamics models [7]. Adaptations are required for the proposed method to be suitable for pedestrian flow estimation (e.g., car traffic may be constrained on one-way roads, while pedestrians have no such restrictions). Other approaches for constructing flows exist, such as using Agent-Based Modeling (ABM) to model possible tourist flows [30, 52].

Bibliography

[1] M. Abdar, F. Pourpanah, S. Hussain, D. Rezazadegan, L. Liu, M. Ghavamzadeh, P. Fieguth, X. Cao, A. Khosravi, U. R. Acharya, V. Makarenkov, and S. Nahavandi. "A review of uncertainty quantification in deep learning: Techniques, applications and challenges." In: *Information Fusion* 76 (2021), pp. 243–297. ISSN: 1566-2535. DOI: 10.1016/j.inffus.2021.05.008.

[2] B. A. Adie, M. Falk, and M. Savioli. "Overtourism as a perceived threat to cultural heritage in Europe." In: *Current Issues in Tourism* 23.14 (2020), pp. 1737–1741. DOI: 10.1080/13683500.2019.1687661.

[3] W. Aigner, S. Miksch, H. Schumann, and C. Tominski. *Visualization of time-oriented data*. Vol. 4. Springer, 2011.

[4] M. M. Albattah and S. E. Youssef. "The potential of space geomatics engineering applications in transportation analysis and planning." In: *Egyptian Journal of Remote Sensing and Space Science* 24.1 (2021), pp. 29–40. ISSN: 11109823. DOI: 10.1016/j.ejrs.2019.10.001.

[5] D. S. de Almeida, R. Simões, F. B. e Abreu, A. Lopes, and I. Boavida-Portugal. "A Carrying Capacity Calculator for Pedestrians Using OpenStreetMap Data: Application to Urban Tourism and Public Spaces." In: *Proceedings of the ENTER 2025 International eTourism Conference* (Feb. 17–21, 2025). Preprint available at arXiv. Wroclaw, Poland: Springer Proceedings in Business and Economics, 2025, pp. 1–11. DOI: 10.48550/arXiv.2406.16781.

[6] N. Andrienko, G. Andrienko, and P. Gatalsky. "Exploratory Spatio-Temporal Visualization: An Analytical Review." In: *Journal of Visual Languages & Computing* 14.6 (Dec. 2003), pp. 503–541. DOI: 10.1016/s1045-926x(03)00046-6.

[7] P. Bellini, S. Bilotta, p. Nesi, M. Paolucci, and M. Soderi. "Real-Time Traffic Estimation of Unmonitored Roads." In: *Proceedings of the 16th Intl Conf on Dependable, Autonomic and Secure Computing, 16th Intl Conf on Pervasive Intelligence and Computing, 4th Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress (DASC/PiCom/DataCom/CyberSciTech)*. 2018, pp. 935–942. DOI: 10.1109/DASC/PiCom/DataCom/CyberSciTec.2018.000-6.

[8] G. Booch, J. Rumbaugh, and I. Jacobson. *The Unified Modeling Language User Guide*. Addison-Wesley Professional, 2005.

[9] F. Brito e Abreu, R. Neto Marinheiro, I. Boavida-Portugal, A. Lopes, T. Mestre Santos, D. Sampaio de Almeida, and R. Simões. "A digital transformation approach to scaffold tourism crowding management: pre-factum, on-factum, and post-factum." In: *Proceedings of the 9th International Conference on Digital Arts, Media and Technology and 7th ECTI Northern Section Conference on Electrical, Electronics, Computer and Telecommunications Engineering (ECTI DAMT & NCON 2024)*. IEEE, Feb. 2024, pp. 625–630. DOI: 10.1109/ECTIDAMTNCON60518.2024.10480056.

[10] J. Brooke. "SUS: A Retrospective." In: *JUX - Journal of User Experience* 8.2 (2013), pp. 29–40. URL: https://uxpajournal.org/sus-a-retrospective/.

[11] M. Cifuentes. *Determinación de Capacidad de Carga Turística en Áreas Protegidas*. 1992.

[12] H. Coccossis and A. Mexa. *The challenge of tourism carrying capacity assessment: Theory and practice*. Routledge, 2017.

[13] M. Fowler. *UML distilled: a brief guide to the standard object modeling language*. Addison-Wesley Professional, 2003.

[14] Y. Gal and Z. Ghahramani. "Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning." In: *Proceedings of The 33rd International Conference on Machine Learning*. Ed. by M. F. Balcan and K. Q. Weinberger. Vol. 48. Proceedings of Machine Learning Research. New York, New York, USA: PMLR, June 2016, pp. 1050–1059. URL: https://proceedings.mlr.press/v48/gal16.pdf.

[15] F. A. Gers, J. Schmidhuber, and F. Cummins. "Learning to forget: Continual prediction with LSTM." In: *Neural computation* 12.10 (2000), pp. 2451–2471.

[16] U. Gretzel, M. Sigala, Z. Xiang, and C. Koo. "Smart tourism: foundations and developments." In: *Electronic Markets* 25.3 (2015), pp. 179–188. ISSN: 1422-8890. DOI: 10.1007/s12525-015-0196-8.

[17] I. Gridin. *Time Series Forecasting using Deep Learning: Combining PyTorch, RNN, TCN, and Deep Neural Network Models to Provide Production-Ready Prediction Solutions (English Edition)*. BPB Publications, 2021.

[18] R. A. Grier. "How high is high? A meta-analysis of NASA-TLX global workload scores." In: *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*. Vol. 1. SAGE Publications, 2015, pp. 1727–1731. ISBN: 9780945289470. DOI: 10.1177/1541931215591373.

[19] S. G. Hart. "NASA-task load index (NASA-TLX); 20 years later." In: *Proceedings of the Human Factors and Ergonomics Society*. 2006, pp. 904–908. ISBN: 9780945289296. DOI: 10.1177/154193120605000909.

[20] J. He, H. Chen, Y. Chen, X. Tang, and Y. Zou. "Diverse visualization techniques and methods of moving-object-trajectory data: A review." In: *ISPRS International Journal of Geo-Information* 8.2 (2019), p. 63. DOI: 10.3390/ijgi8020063.

[21] A. Hevner and S. Chatterjee, eds. *Design Research in Information Systems: Theory and Practice*. Integrated Series in Information Systems. Springer US, 2010. ISBN: 9781441956538. DOI: 10.1007/978-1-4419-5653-8.

[22] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov. *Improving neural networks by preventing co-adaptation of feature detectors*. 2012. DOI: 10.48550/arXiv.1207.0580. arXiv: 1207.0580 [cs.NE].

[23] E. Hüllermeier and W. Waegeman. "Aleatoric and epistemic uncertainty in machine learning: an introduction to concepts and methods." In: *Machine Learning* 110.3 (2021), pp. 457–506. ISSN: 1573-0565. DOI: 10.1007/s10994-021-05946-3.

[24] R. J. Hyndman and G. Athanasopoulos. *Forecasting: Principles and Practice*. 3rd. Accessed on August 26, 2024. Melbourne, Australia: OTexts, 2021. URL: https://OTexts.com/fpp3.

[25] M. Ji, W. Guo, Z. Chen, and N. Morgan. "Managing tourist congestion: insights from Chinese package tours to the UK and Ireland." In: *Current Issues in Tourism* 26.5 (2023), pp. 752–771. ISSN: 13683500. DOI: 10.1080/13683500.2022.2037525.

[26] D. P. Kingma and J. Ba. *Adam: A Method for Stochastic Optimization*. 2017. DOI: 10.48550/arXiv.1412.6980.

[27] A. Komninos, J. Besharat, D. Ferreira, J. Garofalakis, and V. Kostakos. "Where's everybody? Comparing the use of heatmaps to uncover cities' tacit social context in smartphones and pervasive displays." In: *Information Technology and Tourism* 17.4 (2017). Cited by: 8, pp. 399–427. ISSN: 10983058. DOI: 10.1007/s40558-017-0092-5.

[28] A. Kumar Dubey, A. Kumar, V. García-Díaz, A. Kumar Sharma, and K. Kanhaiya. "Study and analysis of SARIMA and LSTM in forecasting time series data." In: *Sustainable Energy Technologies and Assessments* 47 (2021), p. 101474. ISSN: 2213-1388. DOI: 10.1016/j.seta.2021.101474.

[29] A. Kundu, S. Ghosh, and S. Chakraborty. "A long short-term memory based deep learning algorithm for seismic response uncertainty quantification." In: *Probabilistic Engineering Mechanics* 67 (2022), p. 103189. ISSN: 0266-8920. DOI: 10.1016/j.probengmech.2021.103189.

[30] I. Majic, J. Scholz, R. Bulbul, and S. Wallinger. "Tourist Flow Simulation in GAMA Using Historical Data Parameters." In: *Proceedings of the ENTER23 e-Tourism Conference*. Springer Nature Switzerland Cham. 2023, pp. 255–260. DOI: 10.1007/978-3-031-25752-0_27.

[31] C.-Y. Man, O. F. Shyr, Y.-Y. Hsu, S. Shepherd, H.-L. Lin, and C.-H. Tu. "Tourism, transport, and land use: a dynamic impact assessment for Kaohsiung's Asia New Bay Area." In: *Journal of Simulation* 14.4 (2020), pp. 304–315. ISSN: 17477778. DOI: 10.1080/17477778.2020.1806748.

[32] T. Mestre Santos, R. Neto Marinheiro, and F. Brito and Abreu. "Wireless Crowd Detection for Smart Overtourism Mitigation." In: *Smart Life and Smart Life Engineering: Current State and Future Vision*. Ed. by E. Kornyshova, R. Deneckere, and S. Brinkkemper. Lecture Notes in Business Information Processing (LNBIP). Springer Nature, 2024.

[33] T. Munzner. *Visualization analysis and design*. CRC press, 2014.

[34] P. Peeters, S. Gössling, J. Klijs, C. Milano, M. Novelli, C. Dijkmans, E. Eijgelaar, S. Hartman, D. J. Heslinga, R. Isaac, O. Mitas, S. Moretti, J. Nawijn, B. Papp, and A. Postma. *Overtourism: Impact and Possible Policy Responses*. Tech. rep. PE 629.184. Policy Department for Structural and Cohesion Policies, Directorate-General for Internal Policies, European Parliament, Oct. 2018. URL: https://www.europarl.europa.eu/RegData/etudes/STUD/2018/629184/IPOL_STU(2018)629184_EN.pdf.

[35] K. Peffers, T. Tuunanen, M. Rothenberger, and S. Chatterjee. "A design science research methodology for information systems research." In: *Journal of Management Information Systems* 24 (Jan. 2007), pp. 45–77.

[36] J. Perktold, S. Seabold, J. Taylor, and statsmodels-developers. *SARIMAX: Model selection, missing data*. Accessed: 2024-08-27. 2023. URL: https://www.statsmodels.org/stable//examples/notebooks/generated/statespace_sarimax_internet.html.

[37] M. Pietilä and N. Fagerholm. "Visitors' place-based evaluations of unacceptable tourism impacts in Oulanka National Park, Finland." In: *Tourism Geographies* 18.3 (2016), pp. 258–279. ISSN: 14616688. DOI: 10.1080/14616688.2016.1169313.

[38] N. Provos and D. Mazieres. "A future-adaptable password scheme." In: *USENIX Annual Technical Conference, FREENIX Track*. Vol. 1999. 1999, pp. 81–91.

[39] K. Prueksakorn, J. C. Gonzalez, J. Keson, S. Wongsai, N. Wongsai, and P. Akkajit. "A GIS-based tool to estimate carbon stock related to changes in land use due to tourism in Phuket Island, Thailand." In: *Clean Technologies and Environmental Policy* 20.3 (2018), pp. 561–571. ISSN: 1618954X. DOI: 10.1007/s10098-017-1455-5.

[40] M. Ruiz-Pérez, V. Ramos, and B. Alorda-Ladaria. "Integrating high-frequency data in a GIS environment for pedestrian congestion monitoring." In: *Information Processing and Management* 60.2 (2023). ISSN: 03064573. DOI: 10.1016/j.ipm.2022.103236.

[41] D. Sampaio de Almeida, F. Brito e Abreu, and I. Boavida-Portugal. "Agent-based simulation of non-urgent egress from mass events in open public spaces." In: *Simulation Modelling Practice and Theory* (2024). ISSN: 1569-190X/1878-1462.

[42] B. Shneiderman. "The eyes have it: a task by data type taxonomy for information visualizations." In: *Proceedings 1996 IEEE Symposium on Visual Languages*. 1996, pp. 336–343. DOI: 10.1109/VL.1996.545307.

[43] A. Siddhant and Z. C. Lipton. *Deep Bayesian Active Learning for Natural Language Processing: Results of a Large-Scale Empirical Study*. 2018. DOI: 10.48550/arXiv.1808.05697. arXiv: 1808.05697 [cs.CL].

[44] R. Simões, F. Brito e Abreu, and A. Lopes. "Plataforma para visualização geo-temporal de apinhamento turístico." In: *Proceedings of the XIV Congresso da Geografia Portuguesa (XIV CGP)*. Ed. by E. Reis. Lisbon, Portugal, Nov. 2023, pp. 1–6.

[45] R. Simões, F. Brito e Abreu, and A. Lopes. "A Spatiotemporal Crowding Visualization Platform." In: *Information Technology & Tourism* (2025). submitted, under evaluation, pp. 1–20.

[46] I. Sommerville. *Software engineering*. International computer science series. Addison-Wesley, 2007. ISBN: 978-0-321-31379-9.

[47] T. Song, W. Ding, H. Liu, J. Wu, H. Zhou, and J. Chu. "Uncertainty Quantification in Machine Learning Modeling for Multi-Step Time Series Forecasting: Example of Recurrent Neural Networks in Discharge Simulations." In: *Water* 12.3 (2020). ISSN: 2073-4441. DOI: 10.3390/w12030912.

[48] R. C. Staudemeyer and E. R. Morris. *Understanding LSTM – a tutorial into Long Short-Term Memory Recurrent Neural Networks*. 2019. DOI: 10.48550/arXiv.1909.09586. arXiv: 1909.09586 [cs.NE].

[49] A. C. Tricco, J. Antony, W. Zarin, L. Strifler, M. Ghassemi, J. Ivory, L. Perrier, B. Hutton, D. Moher, and S. E. Straus. "A scoping review of rapid review methods." In: *BMC Medicine* 13.1 (Sept. 2015). ISSN: 1741-7015. DOI: 10.1186/s12916-015-0465-6.

[50] V. K. Vaishnavi and W. L. Kuechler. "Design Science Research in Information Systems." In: *Ais* (2004), pp. 1–45. DOI: 10.1007/978-1-4419-5653-8.

[51] J. Vías, J. Rolland, M. L. Gómez, C. Ocaña, and A. Luque. "Recommendation system to determine suitable and viable hiking routes: a prototype application in Sierra de las Nieves Nature Reserve (southern Spain)." In: *Journal of Geographical Systems* 20.3 (2018), pp. 275–294. ISSN: 14355930. DOI: 10.1007/s10109-018-0271-8.

[52] S. Wallinger, L. Grundner, I. Majic, and T. J. Lampoltshammer. "Agent-Based Modelling for Sustainable Tourism." In: *Proceedings of the ENTER23 e-Tourism Conference*. Springer Nature Switzerland Cham. 2023, pp. 355–360. DOI: 10.1007/978-3-031-25752-0_40.

[53] P. Yan and J. Guo. "Analysis of public traffic information system based on WebGIS." In: vol. 1. 2011, pp. 448–450. ISBN: 978-142448623-6. DOI: 10.1109/ITAIC.2011.6030244.

[54] A. Zeghina, A. Leborgne, F. Le Ber, and A. Vacavant. "Deep learning on spatiotemporal graphs: a systematic review, methodological landscape, and research opportunities." In: *Neurocomputing* (2024), p. 127861.

[55] M. Zubiaga, J. L. Izkara, A. Gandini, I. Alonso, and U. Saralegui. "Towards smarter management of overtourism in historic centres through visitor-flow monitoring." In: *Sustainability (Switzerland)* 11.24 (2019). ISSN: 20711050. DOI: 10.3390/SU11247254.

[This page has been intentionally left blank]