| | |
|---|---|
| Project Title | Extreme-scale Urban Mobility Data Analytics as a Service |
| Project Acronym | EMERALDS |
| Grant Agreement No. | 101093051 |
| Start Date of Project | 2023-01-01 |
| Duration of Project | 36 months |
| Project Website | [https://emeralds-horizon.eu/](https://emeralds-horizon.eu/) |

# D3.1 – Mobility Data Processing at the Computing Continuum

| | |
|---|---|
| Work Package | **WP3, Mobility Data Processing at the Computing Continuum** |
| Lead Author (Org) | **George Theodoropoulos (UPRC)** |
| Contributing Author(s) (Org) | **Nikos Koutroumanis (UPRC), Christos Doulkeridis (UPRC), Ioannis Athanasopoulos (UPRC), Nikolas Doulos (UPRC), Yannis Kontoulis (UPRC), Yannis Theodoridis (UPRC), Mahmoud Sakr (ULB), Bahare Salehi (ULB), Daniel Calvo (ATOS), Ignacio Elicegui (ATOS), Yerhard Lalangui (ATOS), Melitta Dragaschnig (AIT)** |
| Due Date | **31.03.2024** |
| Date | **29.03.2024** |
| Version | **V1.0** |

**Dissemination Level**

| | |
|---|---|
| X | PU: Public |
| | SEN: Sensitive, only for members of the consortium (including the Commission) |

This project has received funding from the Horizon Europe R&I programme under the GA No. 101093051

## Versioning and contribution history

| Version | Date | Author | Notes &/or Reason |
|---|---|---|---|
| 0.1 | 02/06/2023 | George Theodoropoulos, Mahmoud Sakr, Christos Doulkeridis | Initial Deliverable Structure - TOC and adherence to D2.1 |
| 0.2 | 26/02/2024 | George Theodoropoulos, Nikos Koutroumanis, Christos Doulkeridis, Ioannis Athanasopoulos, Nikolas Doulos, Melitta Dragaschnig | Chapters 2 and 3 |
| 0.3 | 01/03/2024 | Mahmoud Sakr, Bahare Salehi, George Theodoropoulos, Daniel Calvo, Ignacio Elicegui, Yerhard Lalangui | Chapters 1, 4, 5 |
| 0.4 | 12/03/2024 | George Theodoropoulos, Bahare Salehi, Nikos Koutroumanis | Addressed first round internal review comments |
| 0.5 | 22/03/2024 | George Theodoropoulos, Ignacio Elicegui, Christos Doulkeridis, Mahmoud Sakr | Addressed second round internal review comments |
| 0.6 | 27/03/2024 | George Theodoropoulos, Yannis Kontoulis | Addressed Scientific and Technical Manager comments |

## Quality Control (includes peer & quality reviewing)

| Version | Date | Name (Organisation) | Role & Scope |
|---|---|---|---|
| 0.3 | 04/03/2024 | Anita Graser (AIT), Mattia Pretti (Sistema) | Internal review 1st round |
| 0.4 | 14/03/2024 | Yannis Theodoridis (UPRC) | Review by Scientific and Technical Manager |
| 0.4 | 21/03/2024 | Anita Graser (AIT), Mattia Pretti (Sistema) | Internal review 2nd round |
| 0.5 | 26/03/2024 | Yannis Theodoridis (UPRC) | Final Scientific and Technical Manager revisions |
| 1.0 | 29/03/2024 | Foivos Galatoulas (INLE) | Final review by Coordinator |

Funded by the European Union

*This project has received funding from the European Union's Horizon Europe research and innovation programme under Grant Agreement No 101093051*

# Table of Contents

## List of Figures

EMERALDS

# List of Tables

## Terminology

| Acronym | Description |
|---|---|
| **2D/3D** | **2 D**imensions / **3 D**imensions |
| **AI/ XAI** | **A**rtificial **I**ntelligence/ **E**xplainable **AI** |
| **CC** | **C**omputing **C**ontinuum |
| **GCNN** | **G**raph **C**onvolutional **N**eural **N**etworks |
| **GPS** | **G**lobal **P**ositioning **S**ystem |
| **GPU** | **G**raphics **P**rocessing **U**nit |
| **KPI** | **K**ey **P**erformance **I**ndicator |
| **MDA** | **M**obility **D**ata **A**nalytics |
| **ML/AL** | **M**achine **L**earning/ **A**ctive **L**earning |
| **MLOps** | **M**achine **L**earning **Op**eration**s** |
| **WP** | **W**ork **P**ackage |
| **(RR)MSE** | (**R**elative **R**oot) **M**ean **S**quare **E**rror |
| **NN/ GNN** | **N**eural **N**etworks/ **G**raph **N**eural **N**etworks |
| **SotA** | **S**tate-**o**f-**t**he-**A**rt |
| **TRL** | **T**echnology **R**eadiness **L**evel |
| **UC** | **U**se-**c**ase |

**Table 1 – Terminology**

## Matrix of Alignment

Table 2 outlines the outputs of D3.1 mapped to the GA commitments as stated in the Description of Action (DoA) Annex 1 and Annex 2.

| GA Components Title (and type) | GA Component Outline | Document Chapter(s) | Justification |
|---|---|---|---|
| **Deliverable** | | | |
| **D4.1 Mobility Data Processing Services v1** | *Software library containing early prototypes of the extreme mobility data processing services, methods and tools which are scalable and support the ingestion of heterogeneous data types from each tier of the computing continuum. The deliverable will include a report and manual defining the data preparation activities, that include data modelling (ensuring standardisation thus interoperability), data semantics, annotation, multilingual processing aspects.* | Chapters 1-5 | This document presents the early versions of the processing components that have been developed under WP3. These components perform mobility data analytics across the CC. |

| | | | |
|---|---|---|---|
| | | | Chapter 1 presents the rationale of the WP, whereas Chapter 5 summarises and declares next steps towards v2 (to be documented in D3.2).. Chapters 2, 3, and 4 present the technical work of the WP per task, including well documented source code repositories that are reproducible and interpretable. |
| **Tasks** | | | |
| **T3.1 Privacy-aware In Situ Data Harvesting** | *This task focuses on the ways data can be acquired and processed. With large amounts of connected devices communicating constantly and producing large volumes of data, novel approaches that can handle such extreme-scale data need to be proposed. Exploiting the processing power of multiple edge devices that can be deployed in-situ will allow the platform to scale vertically and dynamically, since such a system can be highly scalable by nature, with its overall computational capability tied to the number of available devices as well as the underlying software and its distributed capabilities. Additionally, evaluating the software with respect to the resources it needs can make the system adaptive, since each piece of software can be tied to a specific level of the computing continuum. This way, the system stands to make the most of its available resources. Greater exploitation of the large number of data streams that the system is able to produce can also be achieved by producing multiple data-streams per device with each one being processed differently and aimed towards different use-cases. This way, each use-case specific part of the system can receive meaningful data and focus on the accompanying data analytics and mining methods instead of re-processing the data that has already been processed. Furthermore, edge computing can be used to enforce privacy policies by enabling edge devices to assess and manage privacy risks, for example regarding location probability or home-work attacks. Lastly, the nature of such a system makes it prone to errors related with network availability. Naturally, when large numbers of devices are part of a widespread network, failures can be common. Designing a network topology* | Chapter 2 | Chapter 2 presents the work of Task 3.1, including the advancements that have been made with respect to the in-situ data harvesting approached that EMERALDS promises, focusing on ensuring privacy and making the deployment and communication between services simpler and more efficient. |

| | | | |
|---|---|---|---|
| | *that addresses these issues, thus making the system more resilient and safer is a priority.* | | |
| **T3.2 Extreme-scale Cloud/Fog Data Processing** | *This task is responsible for the design and implementation of processing algorithms for extreme-scale data, by capitalizing on scalable and data-parallel frameworks. The toolset will be tailored for urban mobility data, to support ultra-scalable query processing algorithms over road networks, from similarity search of spatio-temporal data (range and k-nearest neighbour queries) to more complex processing tasks for trajectory joins, discovery of hot-spots, as well as for supporting mobility analytics. To cope with the extreme scale of data, adaptive and query-aware partitioning techniques will be proposed to ensure fair work allocation and load balancing, to fully exploit the available resources and minimize the processing time. Also, the toolset will provide a distributed indexing framework for urban mobility data along with an appropriate bounding scheme, aiming at the design of algorithms that drastically prune the search space, to facilitate the development of an efficient and scalable data processing solution.* | Chapter 3 | Chapter 3 presents the work done by Task 3.2, introducing mobility data processing methods that focus on being highly scalable when the more computational power is available (at the Fog/Cloud). Highly efficient and scalable algorithms are implemented, allowing for extreme-scale analytics over data harvested by multiple sources and over large time spans. |
| **T3.3 Mobility Data Fusion and Management** | *The focus of this task is to develop theory and prototypes for data summarization. In particular, it will investigate trajectory simplification/smoothing methods to reduce the data size, lossless data compression, depending on the sampling rate and the change in temporal properties. In Lossy compression in the form of task-aware simplification will also be explored. Task-awareness is about providing correctness guarantees for the analysis which will be performed on the summarized trajectories. Since the scope of EMERALDS is urban mobility, we will exploit the fact that the trajectories are restricted to the road network to achieve even higher compression. MobilityDB, an open-source geospatial trajectory database system developed by ULB will be used as the implementation platform and testbed.* | Chapter 4 | Chapter 4 presents the work of Task 3.3, a task that is responsible for managing diverse and heterogenous data sources while performing data summarization and compression to reduce the overall size and footprint of data. |

**Table 2 - Matrix of Alignment**

## Executive Summary

The goal of this deliverable is to present the 1st version of the Mobility Data Processing at the Computing Continuum services, hereafter called 'emeralds', that are being developed under Work Package 3. As its title suggests, the objective of this WP is to develop mobility processing modules that are built for the Computing continuum (CC). The CC spans a wide range of devices that can be deployed in various conditions, with some tailored for centralized compute and some for decentralized in-situ processing. The distinction between the two mostly revolves around energy usage (which directly correlates with compute capability, size etc.), because devices that are highly capable require much more energy and space, meaning that they are not easily deployable and horizontally scalable. However, compute nodes that tend to use less power are often cheaper and smaller, meaning that they are easily configurable to accommodate a wide range of scenarios through their multi location installation. This allows for in-situ data processing, greatly reducing latency and response times when the underlying computation can be effectively performed by such nodes meeting the advanced needs of modern mobility data analytics applications.

This deliverable presents seven (7) emeralds that are developed under the three WP3 tasks. The work covered in this deliverable is organized according to the designated computing tier supported by each component, with emeralds from T3.1 focusing on Edge/Fog applications and emeralds from T3.2 and T3.3 primarily utilizing the Fog/Cloud. The emeralds code releases are provided as software repositories with release notes and instructions on executing the developed methods. This report accompanies the emeralds code releases. The key aspects covered in this report include thorough descriptions of the emeralds and how they advance the state of the art, both with respect to the mobility data science field as well as the achieved and targeted performance and overall improvements related to the state-of-the-art, reflected by the scientific/technical KPIs. Validation and demonstration of the WP3 tools will be performed and reported within the respective WP5 use cases.

WP3 emeralds contribute to and align with the scope of the EMERALDS project, alas to design, develop and create an urban data-oriented Mobility Analytics as a Service (MAaaS) toolset, consisting of the reusable software modules (EMERALDS services), compiled in a proof-of- concept prototype, capable of exploiting the untapped potential of extreme urban mobility data. The developments presented herein are offered in combination with WP2, WP4 implementations as integral parts of the EMERALDS toolset, a one-stop solution that will enable the stakeholders of the urban mobility ecosystem to collect and manage ubiquitous spatio-temporal data of high-volume, high-velocity and of high-variety, analyse them both in online and offline settings, import them to real-time responsive AI/ML algorithms, and visualise results in interactive dashboards, whilst implementing privacy preservation techniques at all data modalities and at all levels of a data workflow architecture. The envisaged toolset will offer advanced capabilities in data mining (searching and processing) of large amounts and varieties of urban mobility data.

# 1 Introduction

## 1.1 Purpose and scope of the document

The purpose of D3.1 "Mobility Data Processing Services 1st Version" is to introduce the tools/emeralds that are being developed in WP3, as listed in Table 3. This table also shows where each emerald's codebase is hosted. This deliverable provides a thorough description of the emeralds, puts them in the context of the state of the art, and presents preliminary evaluation results, before laying out the next steps of development.

| Task / emerald | Maturity | Repository Link |
|---|---|---|
| **Privacy-aware in situ Data Harvesting (T3.1)** | | |
| **Privacy aware data ingestion** | 1st version | Project's GitHub |
| **Extreme-scale stream processing orchestrator** | 1st version | Project's GitHub |
| **Extreme-scale Cloud/Fog Data Processing (T3.2)** | | |
| **Extreme-scale map-matching** | 1st version | Project's GitHub |
| **Weather enrichment** | 1st version | Project's GitHub |
| **Spatio-temporal querying** | 1st version | Project's GitHub |
| **Hot-spot analysis** | Work in Progress (To be fully reported in D3.2) | Under Construction |
| **Mobility Data Fusion and Management (T3.3)** | | |
| **Mobility/trajectory data compression** | 1st version | Project's GitHub |
| **Sensor (GPS, GTFS, radar, etc.) data fusion** | 1st version | Project's GitHub |

**Table 3: Overview of emeralds under development in WP3**

This document provides insights regarding the state of each proposed emerald, focusing on the technical challenges that each service has faced and addressed. All emeralds are accompanied by a code repository that includes the corresponding codebase, installation, usage and demonstration/evaluation instructions.

## 1.2 Relation to Work Packages, Deliverables and Activities

The emeralds that are presented in this deliverable adhere to the EMERALDS reference architecture introduced in D2.1 "Reference Architecture" (Figure 1-1-1) and are validated and demonstrated through real-world data from selected representative use cases whose main context is outlined in D5.1 "Use Cases Scoping Document".

To facilitate the validation and demonstration during the course of the project as well as re-usability beyond its scope, the emeralds are designed to support end-to-end pipelines as the ones reported in the WP5 use case deliverables (D5.2-D5.7).

The emeralds reported in this deliverable link to those developed under WP2 and WP4 as they facilitate data ingestion and curation across the Computing continuum (Edge/Fog/Cloud).

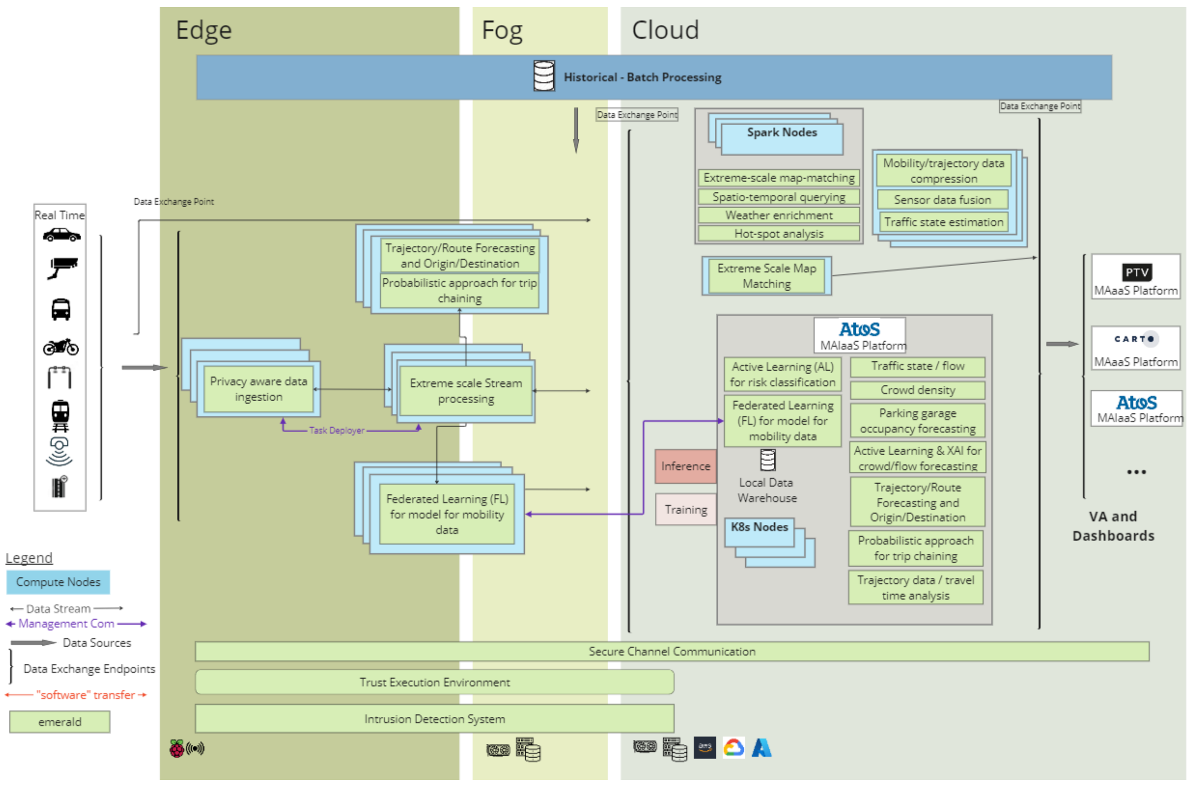

Edge
Fog
Cloud
Historical - Batch Processing
Data Exchange Point
Data Exchange Point
Data Exchange Point
Spark Nodes
Extreme-scale map-matching
Spatio-temporal querying
Weather enrichment
Hot-spot analysis
Mobility/trajectory data compression
Sensor data fusion
Traffic state estimation
Extreme Scale Map Matching
Real Time
Trajectory/Route Forecasting and Origin/Destination
Probabilistic approach for trip chaining
Privacy aware data ingestion
Extreme scale Stream processing
Task Deployer
Federated Learning (FL) for model for mobility data
Atos MAIaaS Platform
Active Learning (AL) for risk classification
Federated Learning (FL) for model for mobility data
Traffic state / flow
Crowd density
Parking garage occupancy forecasting
Active Learning & XAI for crowd/flow forecasting
Trajectory/Route Forecasting and Origin/Destination
Probabilistic approach for trip chaining
Trajectory data / travel time analysis
Inference
Training
Local Data Warehouse
K8s Nodes
PTV MAaaS Platform
CARTO MAaaS Platform
Atos MAIaaS Platform
...
VA and Dashboards
Legend
Compute Nodes
Data Stream
Management Com
Data Sources
Data Exchange Endpoints
"software" transfer
emerald
Secure Channel Communication
Trust Execution Environment
Intrusion Detection System

**Figure 1-1 - EMERALDS services as part of the project's Reference Architecture (D2.1)**

The status of the emeralds reported in this deliverable is the result of the 1st implementation cycle until M15. Therefore, the results of the 1st integration, 1st assessment cycle, and the 2nd set of cycles will be presented in forthcoming project deliverables, D5.2-D5.7, D2.2-D2.6 and D3.3. Throughout all upcoming cycles, the **collaboration framework** established within the project (with technical meetings and workshops conducted between technical partners and use case leaders) will be further intensified to enable iterative development and targeted innovations.

## 1.3 Contribution to Project Objectives

This document is the key output of all tasks in WP3 in project year 1. The objective tackled by WP3 is **Project Objective 2 (O2) "*Develop Extreme Scale acquisition and processing methods and tools for urban mobility data*"**. WP3 proposes emeralds that perform mobility data processing across the CC, including privacy-aware data ingestion at the Edge/Fog using low-powered IoT devices, Extreme-scale data processing at the Fog/Cloud utilizing more capable hardware and Data Fusion and Management across the whole CC. WP3 tasks also directly contribute the respective expected results of O3, minimizing latency and performance by offloading computation to the edge and developing data processing methods and summarization procedures for heterogeneous mobility data.

D3.1 also contributes towards the attainment of Project Objective 1 (O1) **Design a service-oriented reference architecture of a palette of services ('emeralds') for extreme scale urban mobility data analytics,** underpinned by a distributed computing environment that includes edge/fog nodes and cloud nodes, that ensure that both edge and cloud processing contribute towards establishing a robust processing pipeline, by developing and delivering advancements in software services tailored to the needs of mobility data analytics and the execution of extreme scale data workflows across diverse computing environments. Components described within Chapters 2-4 are integral parts of the

EMERALDS toolset architecture and the underlying methods, algorithms and tools materialize respective layers depending on the intended deployment environment. Interoperability, multiplatform-awareness and modularity of the services presented in this document is achieved through containerization, further enhancing their overall reusability and facilitating their incorporation into the design, development and deployment of new systems capable of utilizing resources in a manner that consistently improves accuracy, processing times and easiness of use.

Finally, D3.1 provides the means to achieve Project Objective 3 (O3) **Develop mobility data analytics and AI/ML tools and services – MAaaS**, appropriately designed to perform along the edge/fog/cloud continuum to achieve substantial speedups for analytics jobs. In this regard, services reported in D3.1 seamlessly interconnect or/and function as processing steps feeding the services presented in D4.1 and developed within the frame of WP4, forming pipelines generating fast and accurate information critical to urban mobility stakeholders and decision-making systems as showcased in the EMERALDS use cases. The latter highlights the importance of D3.1 for reaching Project Objective 4 (O4) to demonstrate, measure and validate the efficiencies of WP3 and WP4 services in extreme data workflows, as addressed in WP5.

## 1.4 Structure of the Document

This deliverable is of type OTHER, therefore, it focuses on source code for the emeralds developed in WP3. The source code is provided in code repositories (i.e. GitHub repositories with well-documented README files) that are grouped under the Project GitHub organization (link). This document accompanies the source code and provides an overview and the context for the emeralds development in WP3.

This deliverable is structured according to the WP3 tasks T3.1-T3.3. Each task is described in a dedicated chapter. Task 3.1 results are described in Chapter 2, Task 3.2 results are described in Chapter 3, and Task 3.3 results are described in Chapter 4. The task chapters include the conclusions and next steps, which are summarized in the final section of each chapter.

# 2 Privacy-aware In Situ Data Harvesting

Task 3.1 includes the development of services (emeralds) that perform state-of-the-art privacy-aware data harvesting utilizing the ample compute that is deployed near the sources of data in the form of Edge/Fog IoT devices. By leveraging these compute nodes, the emeralds presented in this chapter aim to make data ingestion more secure and efficient, offloading computation that is either sensitive or optimised well enough to the Edge. This way, sensitive data is never compromised while making the job of the centralized data centre easier and more scalable. The emeralds that will be reported by this task are the following:

1. Privacy-Aware Data Ingestion: This emerald experiments with methods that can identify privacy-compromising records of moving objects and provide ways of mitigating risk without undermining the quality of the underlying dataset. These methods are easily tailored to run in-situ, providing extra guarantees by keeping data away from prying eyes.
2. Extreme-Scale Stream Processing Orchestrator: This emerald develops a deployment/orchestration toolset that can make the deployment of Computing continuum (CC) modules easier and more streamlined. By providing two components, a Deployer and a Data Broker, the services that utilise this tool are able to be remotely deployed to a wide array of compute nodes under different scenarios while being able to communicate and share data with other services, making possible the definition and execution of comprehensive end-to-end pipelines.

## 2.1 Privacy-Aware Data Ingestion

In this section, we present the advancements of Task 3.1 during the reporting period in relation to the proposed emerald named “Privacy-Aware Data Ingestion”. Our goal is to design and implement a privacy-aware pipeline that would be responsible for the preprocessing of data and the execution of Mobility Privacy Attacks for real-time scenarios. This toolset will enable the effective evaluation of the privacy risk of a dataset’s records and furthermore allow a network of compute nodes to recognize and subsequently delete records which contain sensitive information and can possibly lead to the identification of an individual.

### *2.1.1 Brief Survey of the State-of-the-Art*

**Mobility Analytics:** In a world that depends on data more than ever, the analysis and preprocessing of them and especially mobility data is more important than ever. In the past few years, there has been a gradual increase in applying spatial and spatio-temporal data analysis to mobility data[1].For this purpose, of course, new structures and management systems are needed to better organize, aggregate and analyse mobility data, such as the MobilityDB tool [2], a moving object database for representing moving object data and the Dragoon[3], a big trajectory management system for both offline and online analytics. Finally, concerning mobility analytics, there is a need for tools that cover the whole procedure of the analysis of data, an example being TransBigData[4], a Python package used for processing, analysing and visualizing transportation spatio-temporal big data.

**Preprocessing:** One of the most important stages of data analytics is the preprocessing stage, during which the dataset is cleansed by detecting and removing outliers and missing values, thus allowing a better-quality dataset and subsequently a better interpretation of results. Some examples on the preprocessing stage are: i) CloudTP[5], a cloud-based flexible trajectory data preprocessing framework, which transforms raw GPS logs into organized trajectories, by following the stages of noise filtering, trajectory segmentation, map matching and index building and ii) PTRAIL[6], a Python package offering preprocessing steps such as filtering and feature extraction, while also giving the opportunity of parallel computations and trajectory data processing.

**Privacy:** Most datasets often hide patterns and sensitive information, and mobility data is not an exception to that, proving the necessity of the privacy aspect of the procedure of data harvesting, processing, and identification risk evaluation. Existing privacy-preserving techniques and risk evaluation methods for urban mobility vary; examples being: i) generalization and k-anonymity[7], ii) execution of privacy attacks on mobile individuals[8] using the tool Scikit-mobility Privacy and iii) modelling of an adversarial behaviour as a mobility trajectory in order to discover the most impactful adversary path concerning the privacy risk posed to the individuals[9].

**Scikit Mobility Privacy:** With the increasing use of location-based services and the recent shift from spatio-temporal data to mobility data, tools that analyse human mobility become more and more relevant in multiple scenarios and environments. However, the privacy-aware mobility data analysis is also of high importance, which justifies the existence of tools and libraries such as the Scikit Mobility Privacy library[8]. With Scikit Mobility Privacy, the risk of identification of an individual can be calculated for a variety of attacks, showing how vulnerable a dataset is and if there is sensitive information that should be omitted. This existing library is implemented in Python, giving the opportunity for major runtime / performance improvements by switching to more performant languages.

### 2.1.2 Overview and Description

This section introduces the Privacy risk evaluation tool along with its initial demonstration. The primary objective is to comprehensively outline the implementation of the proposed tool. The tool is composed of two components commonly utilized with mobility/trajectory data. Each component encompasses multiple individual modules, ranging from straightforward tasks like data filtering to more intricate processes, such as privacy risk evaluation. This emerald provides the user with a rich set of processes that are needed for the assessment of the **re-identification risk** of an individual's information within a dataset based on sensitive location information.
More specifically, the current version of the pipeline consists of the following steps:

1. Filtering, general preprocessing and clustering of the data.
2. Evaluation of sensitive records.

In the next versions, a third step will be added to the pipeline that will be responsible for the mitigation of the aforementioned dangerous records.

In the current section (Section 2.1.2), a detailed exposition of each component and its constituent modules is presented, and the Section 2.1.3 is dedicated to athorough evaluation process for each scenario, emphasizing real-world metrics and infrastructure.
The privacy risk evaluation tool consists of two components, the **Preprocessing** and the **Risk Evaluation** component. The aforementioned components are themselves composed of individual modules, each tasked with a very specific processing role. This pipeline prioritizes efficiency, specifically designed for deployment in resource-constrained settings, such as edge devices, where a substantial volume of trajectory data is expected. The tool's structure consists of two directories, each containing one of the two components. Both components should be treated as separate tasks since their executions are independent, and they do not share any files or communicate with one another.

It is also important to note that both components communicate with the Orchestrator (to be presented in Section 3.2), in order to receive from and upload data to it. Thus, there is no need to create local datasets, as the two components communicate and share data through the Orchestrator. A brief description of how this works is the following: The preprocessing component receives data from the Orchestrator, processes it and uploads the resulting data to it. Subsequently, the privacy evaluation component communicates with the Orchestrator, receives the data that the preprocessing component uploaded, calculates the risks based on that and then uploads the resulting privacy risk values to the Orchestrator, completing the whole process of the pipeline. Subsequently, the modules of each component are presented below in detail.

**<u>Preprocessing component</u>**

The first component of the pipeline is the preprocessing component, which cleans, transforms and prepares the data for the risk evaluation component that follows (Figure 2-1). More specifically, this component comprises three stages:

### 1. Data Filtering/Cleaning

This particular module implements an outlier detection technique reliant on threshold parameters. Specifically, it assesses individual data records against a defined threshold, typically pertaining to features like speed, either retaining the record if it is within bounds, or removing it in case it is considered an outlier.

### 2. Stop Detection

In the Stop Detection stage, the module classifies the observed object as moving or stationary, based on a speed threshold. This is crucial for the next module of the component, the clustering stage.

### 3. Clustering

This module is responsible for the clustering of coordinates of stationary objects, as close proximity points should be considered the same for the purpose of privacy risk evaluation. The absence of this stage would result in unique coordinates for all records, leading to considerable challenges during the privacy risk assessment phase (will be explained thoroughly in the Privacy Risk Evaluation component presentation that will follow). Only the stationary objects' coordinates are clustered, which are also the targets of the privacy attacks.

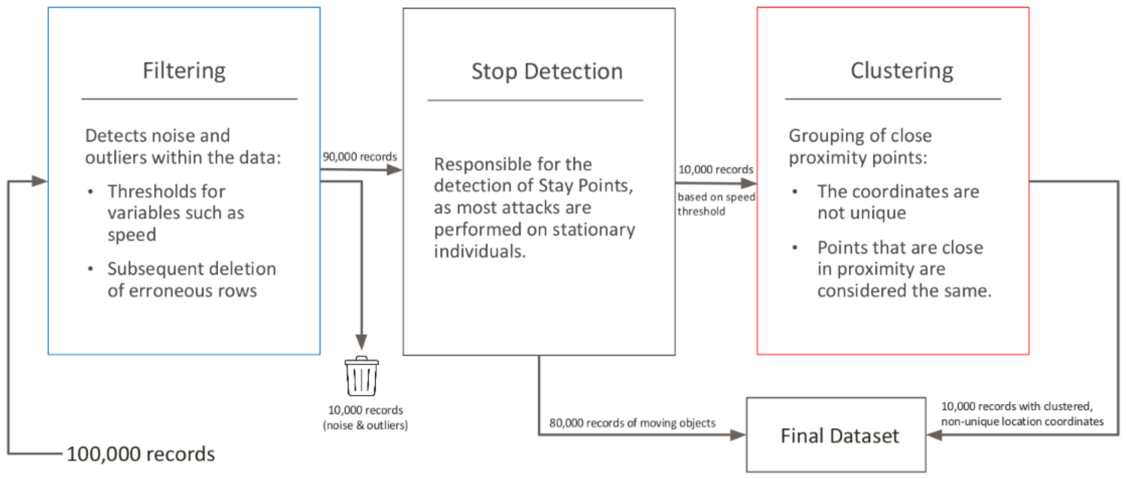
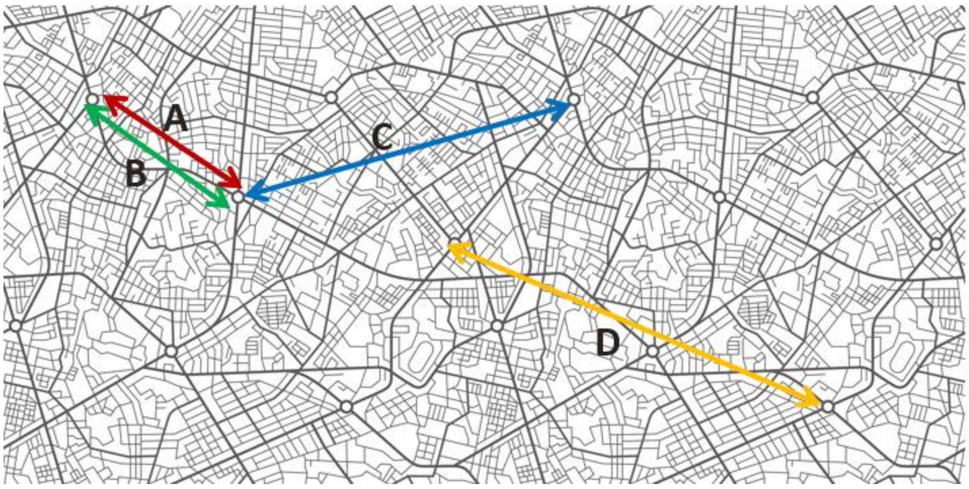

Filtering
Detects noise and outliers within the data:
• Thresholds for variables such as speed
• Subsequent deletion of erroneous rows
90,000 records
Stop Detection
Responsible for the detection of Stay Points, as most attacks are performed on stationary individuals.
10,000 records
based on speed threshold
Clustering
Grouping of close proximity points:
• The coordinates are not unique
• Points that are close in proximity are considered the same.
10,000 records (noise & outliers)
80,000 records of moving objects
Final Dataset
10,000 records with clustered, non-unique location coordinates
100,000 records

**Figure 2-1- Privacy-aware preprocessing component.**

**<u>Privacy Risk Evaluation component</u>**

The second component of the pipeline is the risk evaluation component, which calculates the probability of a successful execution of a privacy attack for each individual. The five implemented attacks are the following:

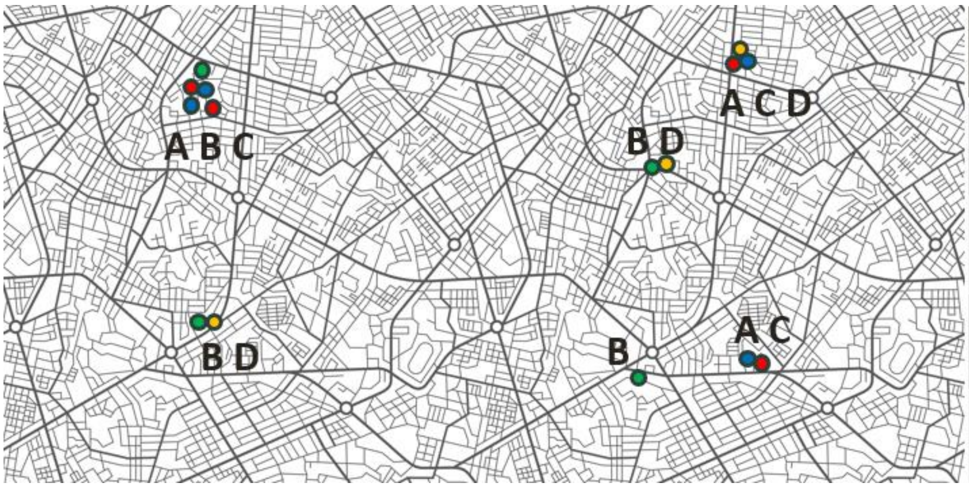
**Home and Work Privacy Attack:** In this attack, we assume that the adversary knows the work and home locations of the individual that he/she wants to identify and also, we make the assumption that the two most frequently visited locations of the majority of individuals correspond to their home and work locations. By finding the two most frequently visited locations of an individual in the dataset and matching those two locations against known work and home locations, the adversary can identify the target among the other individuals. Evidently, if multiple individuals share the same home and work locations, the risk is inversely proportional to the number of those individuals (Figure 2-2Figure 2-1).



| Identification Risk | |
|---|---|
| A | 0.5 |
| B | 0.5 |
| C | 1.0 |
| D | 1.0 |

**Figure 2-2- Home & Work Privacy Attack**

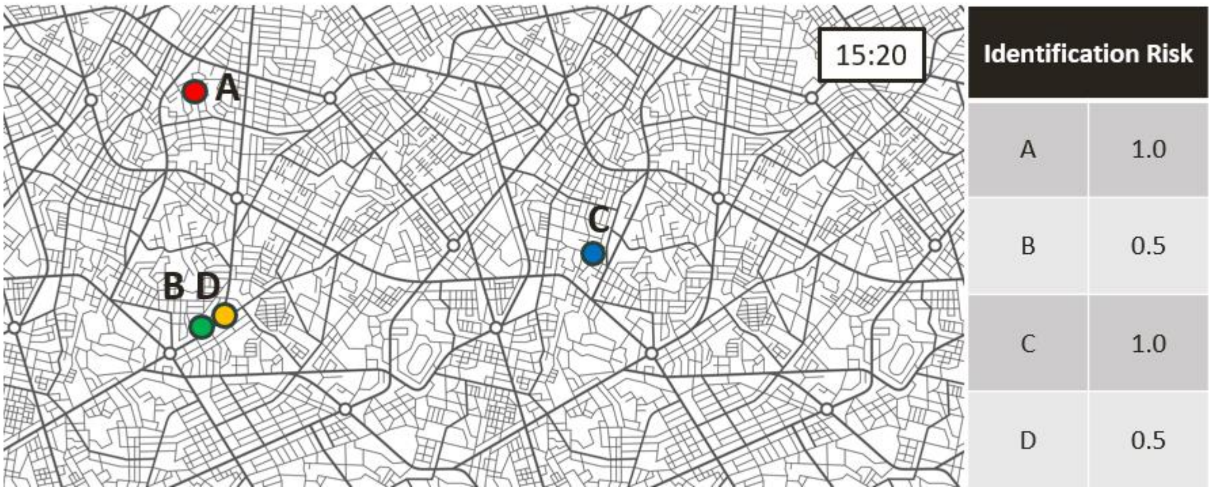
**Location Privacy Attack:** In this attack, we assume that the adversary knows a subset of the locations visited by an individual that he/she wants to identify (the temporal order of the visits is irrelevant, and each location can appear multiple times in the subset). Those locations are afterwards matched against the trajectories of all individuals, narrowing down the list of the possible matches. For the individuals to belong to the matches, they need to have at least N visits per location, where N is the number of appearances of the location in the subset of locations. Evidently, if more than one individuals belong to the list of possible matches, the risk is inversely proportional to the number of those individuals (Figure 2-3).



| Identification Risk | |
|---|---|
| A | 0.5 |
| B | 1.0 |
| C | 0.5 |
| D | 1.0 |

**Figure 2-3 - Location Privacy Attack**

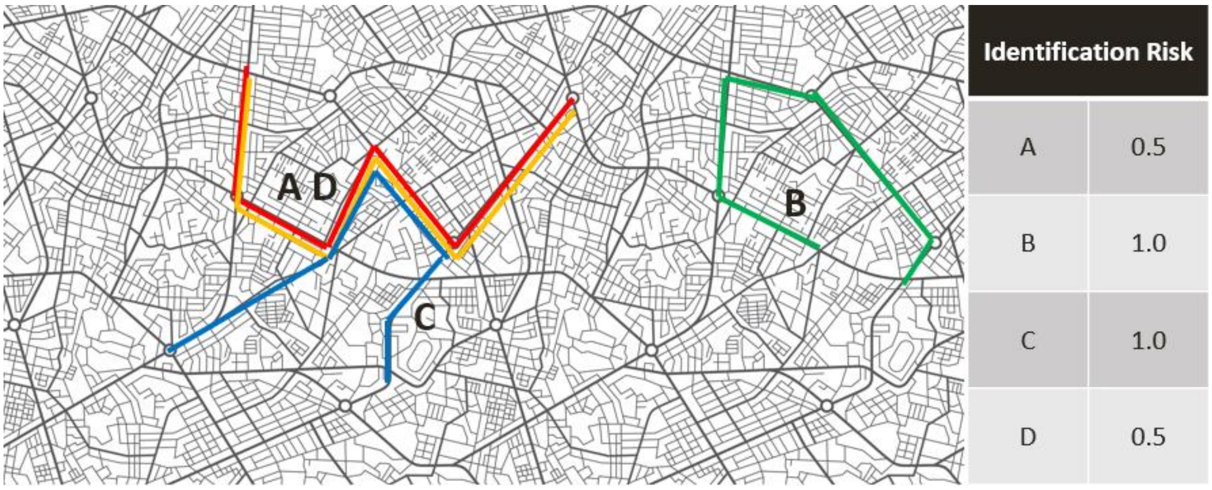
**Location Time Privacy Attack:** This attack is a variation of the Location Privacy Attack, with the additional feature of time precision. We assume that the adversary knows a subset of locations visited by an individual that he/she wants to identify and the respective timestamps of those visits. For the individuals to be considered as possible matches, they need to have visited the specified locations of the subset within a timeframe of the respective timestamp of the location (the resolution of the timeframe could be at a minute, hour, day, etc. scale, depending on the application). If more than one individuals belong to the list of possible matches, the risk is again inversely proportional to the number of those individuals (Figure 2-4).



| Identification Risk | |
|---|---|
| A | 1.0 |
| B | 0.5 |
| C | 1.0 |
| D | 0.5 |

Figure 2-4 - Location Time Privacy Attack

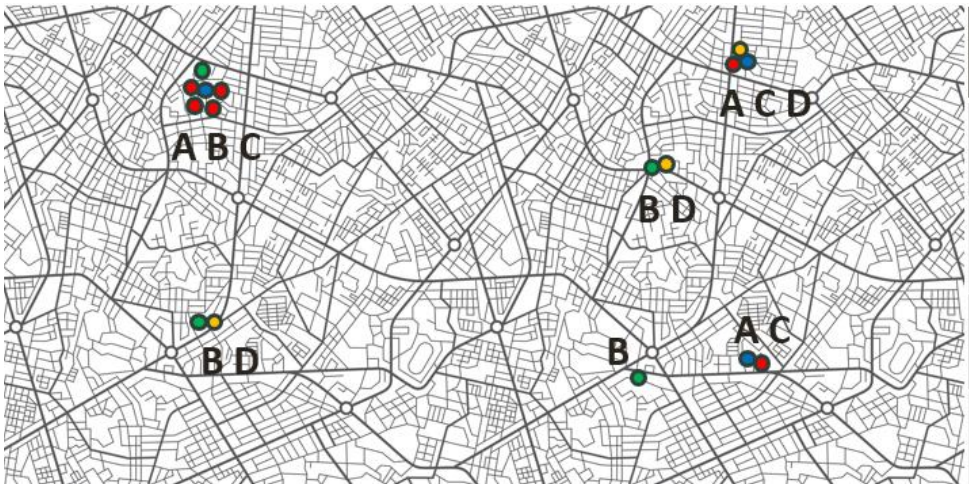
**Location Sequence Privacy Attack:** This attack is a variation of the Location Privacy Attack, where we assume that the adversary again knows a subset of the visited locations of the individual, he/she wants to identify, however now he/she also knows the temporal order of those visits. For the individuals to be considered as possible matches, they need to have visited the specified locations of the subset in the specified order. A location can appear multiple times in the subset. If more than one individual belongs to the list of possible matches, the risk is again inversely proportional to the number of those individuals (Figure 2-5).



| Identification Risk | |
|---|---|
| A | 0.5 |
| B | 1.0 |
| C | 1.0 |
| D | 0.5 |

Figure 2-5 - Location Sequence Privacy Attack

**Unique Location Privacy Attack:** This attack is a close variation of the Location Attack. The adversary knows a subset of the visited locations of the individual that they want to identify, however each location appears only once in the subset. For an individual to be considered a possible match, they need to have at least one visit per location of the subset of visited locations. If more than one individual belongs to the list of possible matches, the risk is again inversely proportional to the number of those individuals (Figure 2-6).



| Identification Risk | |
|---|---|
| A | 0.5 |
| B | 1.0 |
| C | 0.5 |
| D | 1.0 |

**Figure 2-6 - Unique Location Privacy Attack**

It is important to note that if the location coordinates are not clustered and the coordinates are unique, only one individual will be considered a possible match for an attack. This is due to the uniqueness of coordinates, as even minor variations are crucial, and the matching is performed based on locations. In case no clustering is performed, two close-proximity locations that should be considered the same location will be distinct and thus each individual will be easily distinguishable among the others. This of course will lead to the immediate identification of the individual jeopardizing their privacy and security.

### *2.1.3 Preliminary Evaluation*

The evaluation process consists of two parts. In the first part, we compare the implementation developed in this task with a SotA implementation (Scikit-mobility). Then, we execute all attacks and measure their execution time.

More precisely, Table 3 summarizes the KPIs relevant to this emerald:

**Table 3 - KPIs for Privacy Evaluator**

| Description | Baseline Value | Target Value | Method of Measurement | Achieved Value |
|---|---|---|---|---|
| **Better performance** | Scikit Mobility[8] | 2x speed up | # records processed | 5.31x average speed up |

In our experimental study, we utilize the Geolife GPS Trajectory Dataset[10] using either a subset of the first 10.000 unprocessed records or 1.000.000 records based on the timestamp of the record. The first (second) subset is used for the first (second, respectively) part of the process. All the attacks are executed using each respective dataset for both the implementations, producing the following results Figure 2-7.

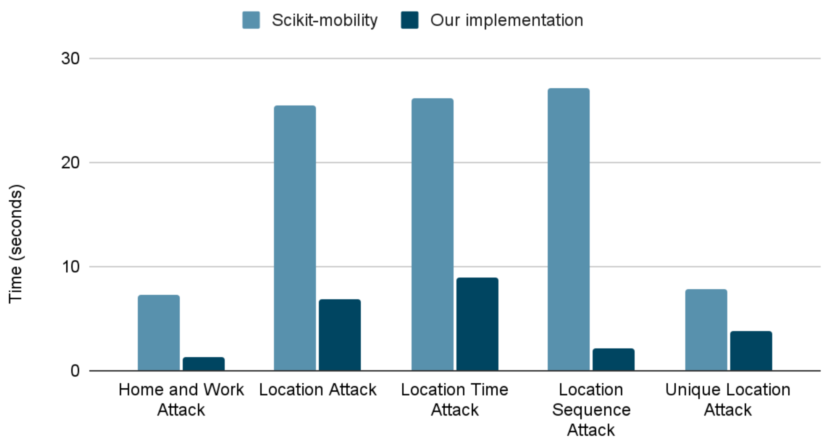


**Figure 2-7 - Comparison of the execution time of Privacy Attacks of both implementations**

On the other hand, for the evaluation part, we calculate benchmarks for both the preprocessing and privacy risk evaluation procedures.
For the pipeline's first component, which is the preprocessing component, the stages of filtering, stop detection and clustering produced the following results (Table 4). As we can see, clustering is far slower than filtering and stop detection, some that is to be expected since the first two components are simpler and thus faster. Clustering however, of 1M points into hundreds of clusters, is a far more computationally intensive process that takes up considerably more time.

| Preprocessing Stage | Time (seconds) |
|---|---|
| **Filtering** | 0.338 |
| **Stop Detection** | 0.161 |
| **Clustering** | 1915 |

**Table 4 - Preprocessing execution time**

The output of the preprocessing stage is, in this example, a dataset of approximately 100,000 records of stopped points, 875,000 moving points and 25,000 outliers/noise points. For the second component, the privacy risk evaluation component, we execute the privacy attacks on the DBSCAN clusters grouping the stopped points of the output dataset of the previous stage (for the Clustering stage, we used the popular DBSCAN clustering algorithm[11]).

The results for Home and Work and Unique Location privacy attacks are shown below (Figure 2-8).

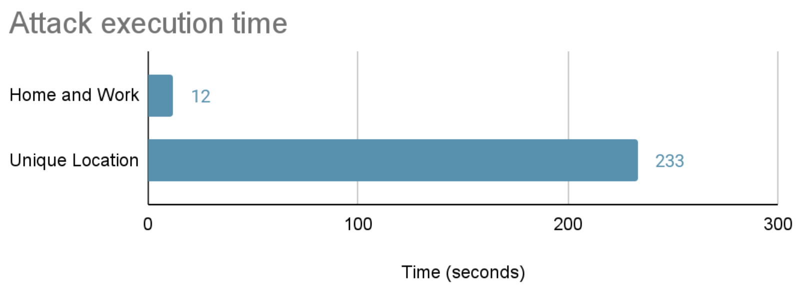


**Figure 2-8 - Home&Work and Unique location performance evaluation.**

On the other hand, the Location, Location Time, and Location Sequence privacy attacks require much higher computational power and time to be executed on 100.000 records because they create very large background knowledge memory structs related to the individual locations that each user visits. The background knowledge created is in the form of tables that store information about all the possible combinations of locations of the individuals, that are afterwards filtered to find possible matches of individuals. As such, a 1% sample (i.e., 1000 records) was used for the benchmarking of the computation of the three attacks. Figure 2-9 presents the performance results.

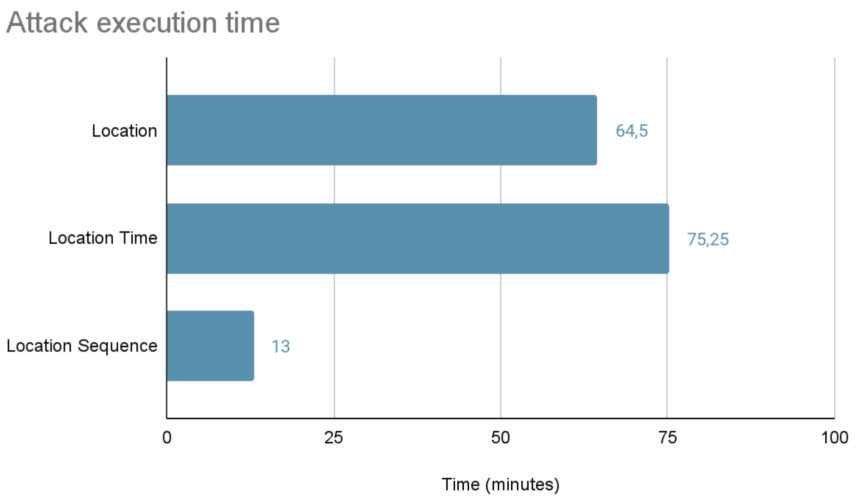


**Figure 2-9 - Location, Location Time & Location Sequence performance evaluation**

### *2.1.4 Next Steps*

The next steps for Task 3.1 belong to two categories, expediting the current slower implementations and implementing the real-time sensitive record detection and removal.

First of all, algorithm optimization is necessary, in order for the privacy attacks to be able to be executed in real time. This could be done with various methods, such as skipping records of close-proximity stoppage, grouping, etc.

Secondly, the record flagging stage needs to be implemented, so that sensitive records can be removed from the final dataset, thus ensuring the safety of sensitive information of individuals. For the record flagging, we assume that the records of moving individuals are not considered sensitive, with the only exception being if they lead to a stoppage record that is considered to contain sensitive information. (Of course, this assumption will also be validated through extensive experimentation during the next steps.) We assume that the target is a recent subset of the whole dataset, containing a set history of records of stopped individuals. The privacy attacks that were described earlier are executed on this subset and the indexes of records that are considered dangerous are kept in a list. This continues for all the individuals, until all the records are either considered safe or unsafe. Lastly, the unsafe records and the previous records that lead to them in a set time span (e.g. five minutes) are removed from the dataset, in order to ensure that an adversary cannot find nor extrapolate sensitive information.

## 2.2 Extreme-Scale Stream Processing Orchestrator

Today's world is more connected than ever, with millions of devices that harvest and process data being used everywhere, from mobile phones to smart cities and industrial/commercial sensors. The inflow of data from all these sources is tremendous, something that is to be expected when the sheer number of devices is that large. The Computing Continuum (CC) that consists of Cloud, Fog and Edge layers, offers a wide range of possibilities for building and deploying novel user-centric services, but also introduces challenges that make harnessing its capabilities challenging. Different layers serve different needs, with the Cloud being in charge of large scale computation over extreme-scale data, the Fog – an intermediary between Cloud and Edge – that can support more intensive computational needs while being closer to the sources, more efficient and horizontally scalable, and the Edge that consists of smaller, cheaper and more energy efficient devices that can process data in-situ.

Some of the main challenges that hyper-distributed systems like the CC face have to do with:
1) **Programmability**: Unlike the cloud where users can deploy their offerings to a centralized server that is very similar in terms of architecture and configuration to their prototyping workstation, the Edge/Fog layers are filled with heterogenous platforms with runtimes that differ highly, making code deployment less straight-forward.
2) **Data Handling:** A CC network can include hundreds/thousands of nodes, depending on its use-case, meaning that it should be capable of handling and curating large amounts of heterogenous data flows. Pairing that with the varying degree of processing/storing that needs to be applied to each individual node/use-case and the underlying architecture because a multivariate problem that also becomes increasingly difficult to solve.

To tackle these challenges and effectively exploit its ample yet dispersed resources, designing an effective orchestration and communication protocol that is built for the CC is a top priority. This emerald, called "Extreme-Scale Stream Processing Orchestrator" aims to be just that, a tool that can effectively and seamlessly deploy and integrate various analytics and processing modules to any CC layer.

The goals of this emerald are:
1. Propose a quick and easy way to deploy modules on any type of hardware, given that a simple SSH connection is possible.
2. Propose deployment protocols that are lightweight and can be consistently transferable even in low bandwidth networks (as it is often the case for Edge/Fog networks).
3. Pair the aforementioned deployer with a lightweight data broker that can make data sharing between modules that are based on different toolsets seamless.
4. Focus on the ease of use and implementation speed of both modules (deployer and broker) and make rapid prototyping of tools and services at the CC possible.

Ultimately, this emerald will act as an accelerator for the adoption of CC horizontal deployments that will in turn allow for much of the computation that currently happens at the Cloud level to be moved to the Edge/Fog, a much more energy efficient CC that can also provide better response times and reduced latency due to its geographical proximity to the user.

### *2.2.1 Brief Survey of the State-of-the-Art*

Regarding deployment at the Edge/Fog, the current state of the art is based on cloud-native methods like containers, microservices etc. These concepts can boost portability, scalability etc. making them great candidates for such use-cases. In recent years, many commercial platforms have been created to facilitate deployment at the Edge. IBM[i] has introduced its edge application manager, an autonomous management solution that enables enterprises to remotely deploy and manage containerized and AI-enabled applications on thousands of edge devices, using an open-source framework called Open Horizon. Microsoft introduced a similar service as part of its Azure ecosystem

[i] https://www.ibm.com/products/edge-application-manager

called Azure IoT Edge[ii] with the goal of Azure cloud capabilities to edge devices, allowing developers to build and deploy modules using Azure services, tools, and languages, such as Visual Studio Code. Many of these platforms are based on well-established tools like Docker[iii] that offer isolated environments called containers that are transferable and interoperable, paired with services like Kubernetes[iv] and some of its variants like KubeEdge, k3s and k0s that automate the deployment of such containers under different use-cases and deployment scenarios.

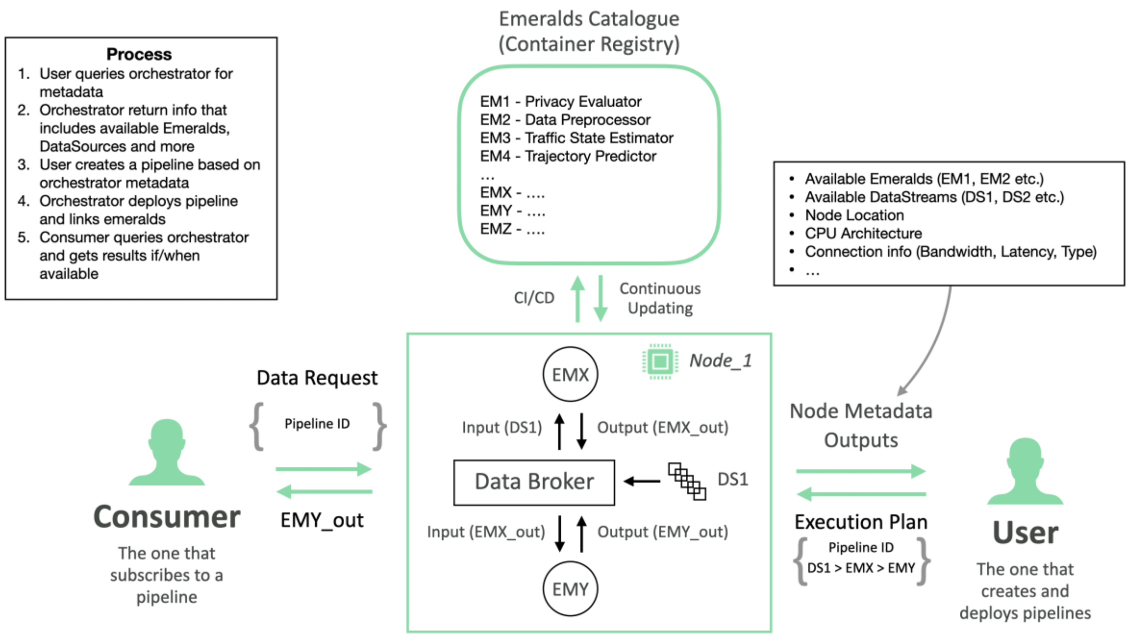
Regarding data transfers and messaging, multiple technologies are currently being used based on the individual needs of each user/use-case. Apache Kafka[v] for example, is one of the most well-known platforms that is used by thousands of companies for high-performance data pipelines, streaming analytics, data integration, and mission-critical applications. Kafka consists of brokers that store and manage data in topics and partitions, producers that publish data to topics, and consumers that subscribe to topics and consume data. It also offers APIs and tools for data transformation, analysis, and integration. Kafka uses Zookeeper to coordinate and configure the cluster. It is scalable, fault-tolerant while providing APIs and tools for data transformation, aggregation, analysis, and integration. Additionally, messaging protocols that are lightweight and thus suitable for CC/IoT related activities also exist, with some examples being MQTT[vi], zeroMQ[vii] and others. Those pub-sub protocols, however, are designed to transmit small messages between multiple nodes rather than transferring large amounts of streaming data using unstable connections like it is often the case for large CC deployments.

### *2.2.2 Architecture Overview*

In this section, we present an overview of the "Extreme-Scale Stream Processing Orchestrator" emerald that will facilitate the deployment and communication of other emeralds at various CC levels.

Figure 2-10 presents the architecture that the orchestrator will adopt in its finalized state, including both of its components, the CC Deployer that is responsible for the deployment of services at the CC, and the Data Broker that is in charge of connecting the inputs and outputs of various heterogenous deployed services. Starting from the bottom-right, the interaction begins with the user querying the orchestrator for metadata. This metadata will include information regarding inputs and emeralds. More specifically, the user will have access to data samples from all the nodes of the network, paired with information about the data location, size etc. Regarding processing/analytics modules, the user will also be supplied with information regarding the emeralds that are available to the system. This interaction will be backed by either built-in emeralds that will be shipped alongside the orchestrator, or by querying a centralized repository where many implemented emeralds are stored and maintained. Additional metadata will include the location of each CC node, its architecture (x86, ARM etc), the state of the connection etc. Based on the aforementioned information, the user will have a complete view of what is currently supported by the orchestrator. Then, the user will define a list of modules that need to be executed. These modules will all read and write data through the Data Broker, enabling interactions between different components and services and thus, the creation of job sequencies or pipelines. Finally, the consumer can pool the system in order to get results as soon as they are ready by specifying the emerald/pipeline that will be consumed.

[ii] https://azure.microsoft.com/en-us/products/iot-edge
[iii] https://www.docker.com
[iv] https://kubernetes.io
[v] https://kafka.apache.org
[vi] https://mqtt.org
[vii] https://zeromq.org



Process
1. User queries orchestrator for metadata
2. Orchestrator return info that includes available Emeralds, DataSources and more
3. User creates a pipeline based on orchestrator metadata
4. Orchestrator deploys pipeline and links emeralds
5. Consumer queries orchestrator and gets results if/when available
Emeralds Catalogue (Container Registry)
EM1 - Privacy Evaluator
EM2 - Data Preprocessor
EM3 - Traffic State Estimator
EM4 - Trajectory Predictor
...
EMX - ....
EMY - ....
EMZ - ....
Available Emeralds (EM1, EM2 etc.)
Available DataStreams (DS1, DS2 etc.)
Node Location
CPU Architecture
Connection info (Bandwidth, Latency, Type)
...
CI/CD
Continuous Updating
Node_1
EMX
Input (DS1)
Output (EMX_out)
Data Broker
DS1
Input (EMX_out)
Output (EMY_out)
EMY
Data Request
Pipeline ID
EMY_out
Consumer
The one that subscribes to a pipeline
Node Metadata Outputs
Execution Plan
Pipeline ID
DS1 > EMX > EMY
User
The one that creates and deploys pipelines

**Figure 2-10 - Orchestrator Architecture**

### 2.2.3 *Prototype - Deployer*

Based on our plans for the final state of this tool, we have implemented prototypes of both modules (CC Deployer and Data Broker) that include an initial set of features need for the overall prototype to be validated.

The "Deployer" module is based on Python 3 and is part of the EMERALDS GitHub organisation. As shown in Figure 2-11, it consists of a Read, Evaluate, Print, and Loop (REPL) interface that manages the CC devices that are part of the prespecified network. SSH and SFTP connections are established between the host machine and all the network nodes in order to facilitate remote command execution and data transfers respectively. Through the REPL, the user can execute the following commands:

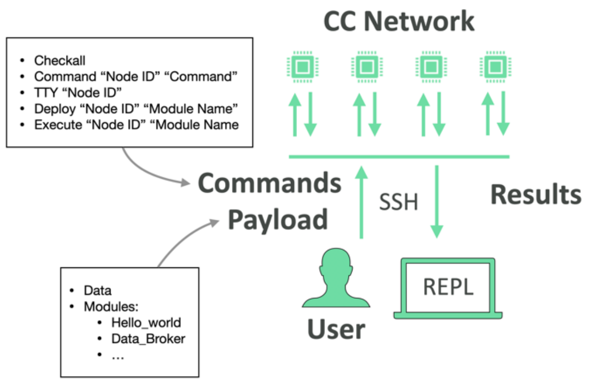


**Figure 2-11 - Deployer Architecture**

- **Checkall**: Reads the contents of **hosts.json** file that includes the connection information for each of the specified hosts in the form of a dictionary. Each individual host needs to include information regarding SSH (hostname, ssh_key, port etc). This format has been chosen because it is expandable, allowing for the addition of fields that might be useful in the future (important paths, network info etc.).

- **Command <NodeID> <Command>**: The user can execute this command if there is a need for some **unix** command to be executed at as specific remote client. This is useful during prototyping as it can support the quick execution of a debugging command for example without the need for a TTY (Teletypewriter) session.
- **TTY <NodeID>**: This command creates a TTY session (terminal emulation) at the specified host. This is useful when multiple commands need to be executed as it allows the user to have direct access to the host without the need of creating a separate **SSH** connection.
- **Deploy/Execute <NodeID> <Module name>**: These commands (deploy and execute) are the ones responsible for the initialization and execution of any of the available modules (emeralds). "Deploy" transfers the module's payload to the dedicated host/node and "Execute" executes the execution scripts that the module developer has specified. More information regarding the parameters of these commands and the whole deployment scheme follows.

**Deployment**: The following is an outline of the steps taken when a module is deployed at the CC using our "Deployer" module:

1. The user specifies the module that will be deployed & executed as well as the target host using the "Deploy" command. The module name needs to be identical to the name of the respective modules folder under "modules". Deployer already includes a simple "hello_world" module as standard, but other emeralds can also be copied to that folder or "cloned" using **git**.
2. The Deployer transfers the contents of the module directory to the target host and stores them under a directory called "modules" at the host's user's home directory.
3. The Deployer looks for a file named "init.sh" inside the remote module's directory. This bash script is responsible for initializing the module. The script can include various commands, with examples being commands that download and install compilers or create Docker containers.
4. The user can then execute the "Execute" command. This command looks for a file named "run.sh" inside the remote module's directory. This script should include execution commands like code execution for example (python main.py for Python, cargo run for Rust etc.) or Docker execution commands. The output of the remote execution will be redirected to local stdout.

### 2.2.4 Prototype – Data Broker

The second module that is part of the Orchestrator emerald is the Data Broker that is also a part of the emeralds organisation. This module is also implemented on Python 3 and its goal is to facilitate seamless data exchanges between other emeralds. This way, the project will be able to develop and demonstrate insightful pipelines that transform the raw inputs into actionable and meaningful results.

In more detail, the Data Broker utilizes an HTTP server that is capable of ingesting and hosting data. The server receives and transmits data through POST and GET requests respectively, allowing the user to utilize its capabilities through a simple client that only needs to implement those two types of requests. This simplicity and ease of development is at the core of emeralds Data Broker since its goal is to facilitate quick prototyping and testing of software across multiple devices and CC layers. This is a clear differentiating factor between our data broker and other commercial offerings that need substantially more effort to spin up and maintain, while often needing language specific drivers/packages that need to be implemented and maintained by third parties. Our data broker is based on a well-known and easy to implement protocol (HTTP), allowing every programming language/toolset to be able to communicate with it through simple requests that are almost always part of the standard library of all the widely used languages and toolsets.

Figure 2-12 presents the flow of data between a user, the broker, two emeralds (X and Y) and an input data stream. As we can see, in this example, the broker receives DSX as input. emerald X requests this data stream as input and provides a new data stream called EMX as output. Then, emerald Y requests EMX as input and transmits EMY as output. As a result, the user can request data from any of the aforementioned streams, including the raw DSX stream. Essentially, the broker links all these services by allowing each one to use the outputs of others as inputs, regardless of toolset, programming language etc., greatly increasing productivity while making integration between components much easier.

In its current state, the broker includes the following set of features:

- **In-Memory Persistence**: The current implementation supports both consuming and non-consuming GET requests. In a CC environment where network availability is not always guaranteed, finding ways of safeguarding data is very important, so our broker will be expanded to include Disk persistence that considers both device capabilities and use-case characteristics.
- **Varying Size of Request**: The user can specify the amount of data that will be transmitted by the broker by including the related headers to the GET request, allowing each service/emerald to consume data based on its needs and capabilities.
- **Multiple Formats**: The broker primarily distributes text-based data, as that type is the one that is most often used in data related activities (since it is the one that is most easily parsed). However, changing the format of the input or output data stream is as easy as changing a single internal parameter.

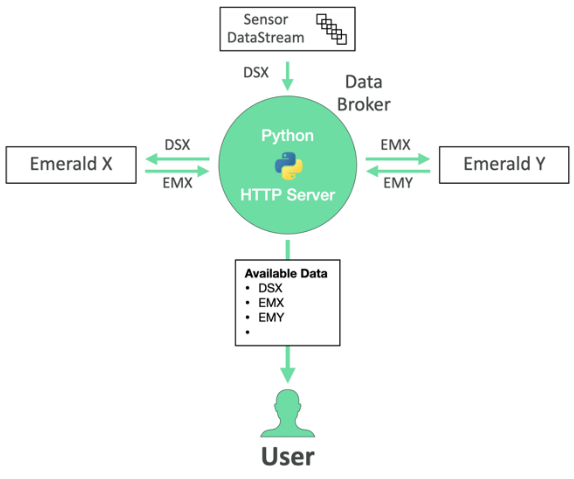


**Figure 2-12 - Data Broker Architecture**

Table 5 presents examples of one GET and one POST request that read and write data to the broker respectively, alongside their individual parameters.

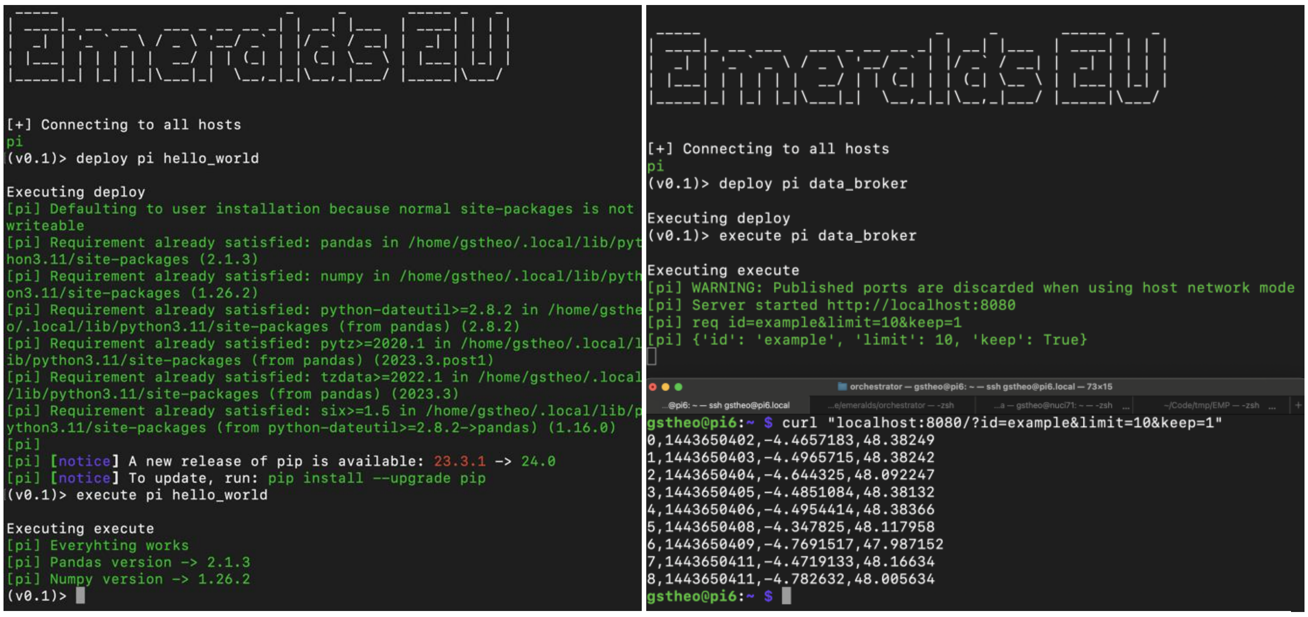
**Table 5 - Data Broker requests example**

| Request | DataStream Name | Parameters | Request text |
|---|---|---|---|
| **GET** | Example1 | Records received=10, Persist data=False | GET http://HOSTNAME:8080/name=Example1&limit=10&keep=0 |
| **GET** | Example2 | Records received=10, Persist data=True | GET http://HOSTNAME:8080/name=Example2&limit=10&keep=1 |
| **POST** | Example2 | N/A | POST http://HOSTNAME:8080/Example2 “test data” |

It is important to note that, in a CC network, it is not obligatory for each node to host a data broker session. Some very-low powered Edge nodes, for example, can choose to not deploy a session to save resources and instead, instantly transmit data to a Fog node, for example, that can more easily host a broker session. However, if the node can execute multiple processing jobs, a broker session can make communication between modules easier than a manual data exchange approach.

### 2.2.5 *Preliminary Evaluation*

In this section, we present two execution examples of the Orchestrator module. The first one is a demonstration of the Deployer component only, by using the included "hello_world" module, and the second one is an example of the remote execution of the Data Broker module, again utilizing the Deployer module. Figure 2-13 includes screenshots of both examples.



```
[+] Connecting to all hosts
pi
(v0.1)> deploy pi hello_world

Executing deploy
[pi] Defaulting to user installation because normal site-packages is not
writeable
[pi] Requirement already satisfied: pandas in /home/gstheo/.local/lib/pyt
hon3.11/site-packages (2.1.3)
[pi] Requirement already satisfied: numpy in /home/gstheo/.local/lib/pyth
on3.11/site-packages (1.26.2)
[pi] Requirement already satisfied: python-dateutil>=2.8.2 in /home/gsthe
o/.local/lib/python3.11/site-packages (from pandas) (2.8.2)
[pi] Requirement already satisfied: pytz>=2020.1 in /home/gstheo/.local/l
ib/python3.11/site-packages (from pandas) (2023.3.post1)
[pi] Requirement already satisfied: tzdata>=2022.1 in /home/gstheo/.local
/lib/python3.11/site-packages (from pandas) (2023.3)
[pi] Requirement already satisfied: six>=1.5 in /home/gstheo/.local/lib/p
ython3.11/site-packages (from python-dateutil>=2.8.2->pandas) (1.16.0)
[pi]
[pi] [notice] A new release of pip is available: 23.3.1 -> 24.0
[pi] [notice] To update, run: pip install --upgrade pip
(v0.1)> execute pi hello_world

Executing execute
[pi] Everyhting works
[pi] Pandas version -> 2.1.3
[pi] Numpy version -> 1.26.2
(v0.1)>
```

```
[+] Connecting to all hosts
pi
(v0.1)> deploy pi data_broker

Executing deploy
(v0.1)> execute pi data_broker

Executing execute
[pi] WARNING: Published ports are discarded when using host network mode
[pi] Server started http://localhost:8080
[pi] req id=example&limit=10&keep=1
[pi] {'id': 'example', 'limit': 10, 'keep': True}
```

orchestrator — gstheo@pi6: ~ — ssh gstheo@pi6.local — 73×15
...@pi6: ~ — ssh gstheo@pi6.local
...e/emeralds/orchestrator — -zsh
...a — gstheo@nuci71: ~ — -zsh
~/Code/tmp/EMP — -zsh

```
gstheo@pi6:~ $ curl "localhost:8080/?id=example&limit=10&keep=1"
0,1443650402,-4.4657183,48.38249
1,1443650403,-4.4965715,48.38242
2,1443650404,-4.644325,48.092247
3,1443650405,-4.4851084,48.38132
4,1443650406,-4.4954414,48.38366
5,1443650408,-4.347825,48.117958
6,1443650409,-4.7691517,47.987152
7,1443650411,-4.4719133,48.16634
8,1443650411,-4.782632,48.005634
gstheo@pi6:~ $
```

**Figure 2-13 – Orchestrator examples: "hello_world" execution (left) and data broker with GET example (right)**

The first example is a simple demonstration of the Deployer's workflow using the built-in "hello_world" module. As we can see at the left screenshot of the figure, we start by calling the orchestrator module that, in turn, connects to the specified CC nodes; in both examples, we use a single Raspberry Pi 4 for testing. As we can see, the "pi" alias is green, meaning that the node is online and ready to be used. The first command that we execute is the "deploy" command, namely "deploy pi hello_world". This command transfers the contents of the "hello_world" module to the Raspberry Pi and then executes the init.sh script (if there is one inside the module's directory). In this case, the "hello_world" module includes an initialization script that uses PIP to install two python packages, namely **Pandas** and **Numpy**. We then run the "execute" command ("execute pi hello_world"). This command remotely executes the run.sh file that must be included in all modules of the orchestrator/deployer. In this case, this file includes a python execution command for the main.py file. As we can see, all steps appear to be executed correctly, as we get output from the host's stdout (that is why the colour is green, if the colour was red the output would be redirected from the host's stderr) showing the versions of the modules that were installed during the deployment phase.

The second example is more detailed since it also includes the Data Broker component. As we can see at the top-right side of the figure, we again start with the deploy command ("deploy pi data_broker"). The Data Broker's init.sh script includes a **docker build** command that utilized the Dockerfile that is included inside the module. Then, we run the execute command and see that server has started and can be accessed through port 8080. In a separate tab (bottom-right), we SSH into the Raspberry Pi and execute a simple curl GET command that fetches 10 records from the example dataset. As we can see, we successfully obtain the records that we asked for, with the server also displaying the request it has received and the dictionary format of the request.

Regarding KPIs, Table 6 summarizes the ones relevant to this emerald:

**Table 6 - KPIs for Extreme Scale Orchestrator**

| Description | Baseline Value | Target Value | Method of Measurement | Achieved Value |
|---|---|---|---|---|
| **Facilitate Deployment of Jobs @ the Edge** | N/A | >5 | No. of emeralds deployed | To be evaluated after 1st integration cycle (D3.2) |

### 2.2.6 *Next Steps*

As mentioned earlier, the Orchestrator emerald is a module that bridges other modules (i.e., emeralds) and makes integration easier and more streamlined. Our planned next steps for this service mainly revolve around challenging features of the CC. Specifically:

1. Disk persistence: This feature will allow nodes with restricted memory capabilities to be able to support larger workloads that include multiple emeralds. It is also important that multiple storage formats are introduced in order for multiple scenarios to be supported (high/low throughout, high/low memory, etc.)
2. Built-in compression: It is important for the Orchestrator to include compression methods that can reduce the size of both stored data and transmitted data streams, allowing for better management of scenarios where network bandwidth is restricted or unstable.
3. Performance / parallelism: The main goal of the Orchestrator module is to be fast and lightweight. To achieve this, both the Deployer and the Data Broker submodules need to be studied and optimized, with parallelism being implemented both during deployment (multiple concurrent deployments) and during data transfers (multi-threaded HTTP etc.).
4. Pipeline deployment: Supporting the formulation and deployment of pipelines based on text-based deployment plans will allow for a streamlined, interoperable, and transferable deployment scheme to be used across the project.

# 3 Extreme-Scale Cloud/Fog Data Processing

The work carried out under Task 3.2 includes the development of algorithms (emeralds) for various task related to data processing, always in relation with the project's use-cases and datasets. In particular, our algorithms pertain to the following categories:

1. **Extreme Scale Map-Matching** (Section 3.1), for handling urban GPS datasets of moving objects that need to be transformed into meaningful trajectories,
2. **Weather Enrichment** (Section 3.2), for combining movement data with weather information in order to enable upstream AI/ML tasks to perform model training using rich datasets,
3. **Spatio-Temporal Querying** (Section 3.3), for highly efficient and scalable query processing over spatial data that may also contain a temporal or a textual dimension, and
4. **Hot-spot Analysis**, for analysing large urban datasets and discovering statistically significant hot spots.

At the time of this writing (M15 of the project) we report our progress in three of the four categories, namely (1) - (3). In the next phase of the project, we will report on our progress on hot-spot analysis as well.

## 3.1 Extreme Scale Map-matching

Map-matching is essential in spatial data analysis for aligning GPS coordinates with road networks to reconstruct vehicle trajectories accurately, playing a vital role in applications ranging from traffic management to urban planning. This process utilizes probabilistic algorithms that integrate the vehicle's GPS data, assessing potential paths by their proximity to GPS points, conformity to road topology, and vehicular dynamics. Map-matching aligns GPS points onto the road network, using advanced algorithms such as Valhalla, and then we enhance these results with our *trajectory reconstruction algorithm*, which uses these aligned points to generate a more accurate vehicle path representation. This reconstruction is particularly challenging with low sampling rate GPS data, requiring sophisticated methods to interpolate and deduce vehicular movements between recorded points, thus enhancing the understanding of vehicular trajectory. In our solution, we compute this exact two-step approach to generate the most accurate results.

Low sampling rates in GPS data make it challenging to accurately reconstruct vehicular trajectories. Sparse datasets resulting from infrequent GPS recordings fail to capture the vehicle's path adequately, especially in critical points like intersections, leading to ambiguities and potential misalignments in the deduced routes. The sparseness of data points also limits the algorithms' ability to counteract GPS noise and multipath effects, common in areas with obstructions like urban canyons, further diminishing map-matching quality. Additionally, low sampling compromises the detection of transient behaviours such as lane changes, essential for applications demanding high-precision data. These limitations highlight the necessity for advanced map-matching algorithms that can effectively interpolate sparse data and counter environmental and noise-related inaccuracies, ensuring the reliability and precision of trajectory information for critical decision-making and analytical purposes.

### *3.1.1 Brief Survey of the State-of-the-Art*

Map-matching is a challenge that appears for decades in urban trajectory analysis, thus there are different approaches that try to resolve this problem. Among the spectrum of strategies employed, the Geometric, Topological and Probabilistic algorithms stand out. Although all three of the strategies are valid, recent trends indicate that most of the solutions apply Probabilistic algorithms to produce more accurate results.

The most widely adopted Probabilistic algorithm used for Map-Matching is the Hidden Markov Model (HMM) and Viterbi Algorithm that Paul Newson and John Krumm proposed[12]. The most popular implementations leveraging this algorithm include Valhalla's Meili[viii], Open-Source Routing Machine (OSRM)[ix] , Graphhopper[x] and more. This universal use of the HMM algorithm is due to its very high accuracy metrics, and typically modest computational resource requirements. Valhalla[xi] is an advanced open-source routing engine that utilizes OpenStreetMap. It is distinguished by its efficient use of a tiled hierarchical data structure that reduces memory usage and supports offline routing and incremental updates, addressing key spatial data processing challenges. A significant component of Valhalla is Meili, its map-matching engine that incorporates algorithms like the HMM and the Viterbi algorithm for accurate alignment of GPS trajectories with road networks, thereby improving the accuracy of spatial data analysis and routing. Figure 3-1 shows the application of Valhalla's Meili map-matching service to a trajectory of Riga's dataset (UC #3).

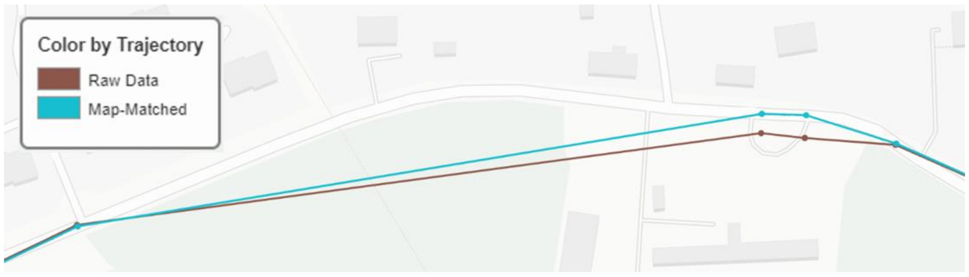


**Figure 3-1 - Map Matching using Valhalla**

Other notable probabilistic algorithms are STMatch[13] that is mostly used for map-matching low sampling rate data, and Fast Map-Matching (FMM)[14] that is highly efficient. Both algorithms can be used via the Fast Map-matching (FMM) framework[xii].

Currently, the Hidden Markov Model (HMM) is used broadly in the most popular map-matching tools and provides very accurate results. Nevertheless, there is a recent trend towards employing Deep Learning (DL) and AI to do the map-matching, aiming to overcome the limitations associated with HMM, such as its inability to utilize the potential of enormous trajectory big data and its susceptibility to noise data. Despite their potential, DL solutions are still immature and still demand high resources and computing power, while facing scalability challenges. Instead, we focus on improving the limitations of HMM by means of post-processing.

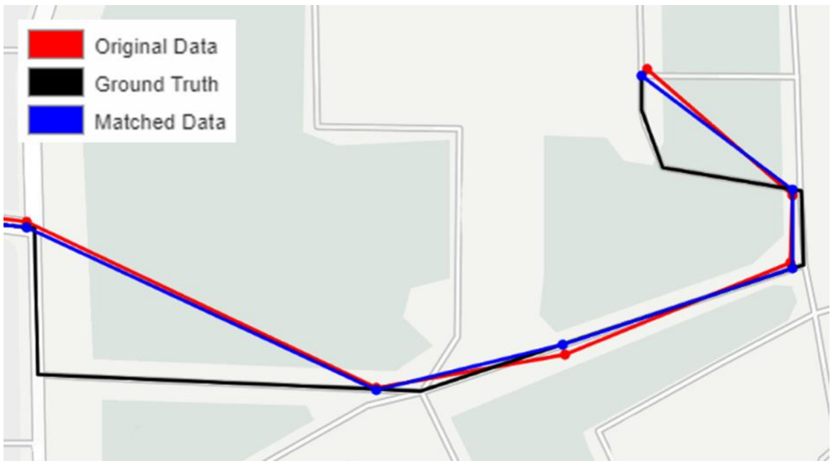
### 3.1.2 Overview and Description

In Figure 3-2, an example of map-matching of a sparse trajectory and the actual trajectory is provided. It shows the application of Valhalla's Meili map-matching service to a low sampling rate trajectory, and how that "matched" trajectory is far from the actual ground truth trajectory. The trajectory used in

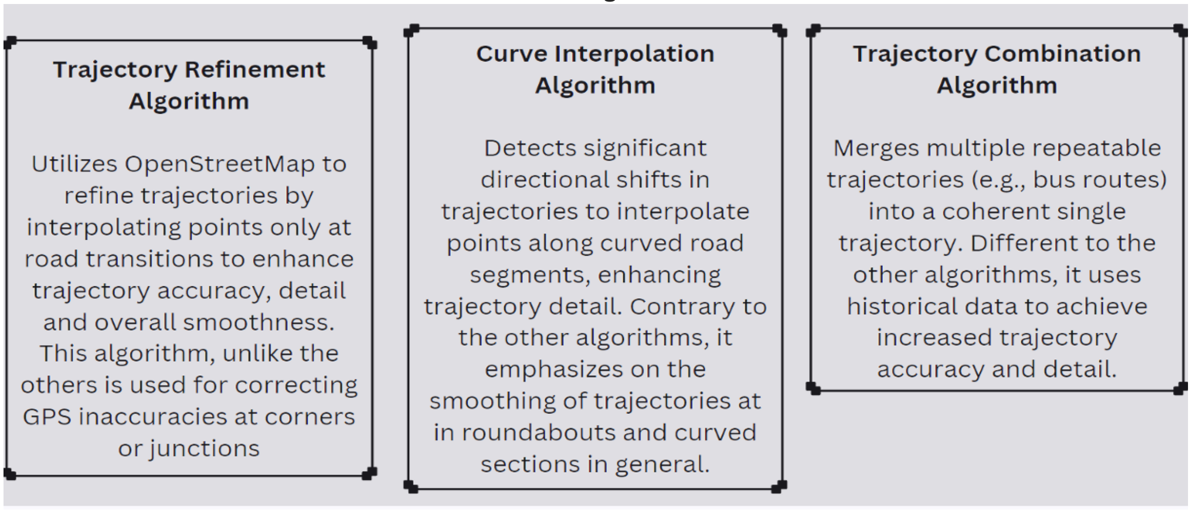
**Figure 3-2 - Map Matching a low sampling rate trajectory**

[viii] https://valhalla.github.io/valhalla/meili/overview/
[ix] https://github.com/Project-OSRM/osrm-backend
[x] https://github.com/graphhopper/graphhopper#map-matching
[xi] https://valhalla.github.io/valhalla/
[xii] https://github.com/cyang-kth/fmm

this figure origins from an open dataset (https://zenodo.org/records/57731) that also provides the ground truth.



To address the accuracy challenges in map-matching due to low GPS sampling rates, we have been developing an emerald that contains sophisticated spatial analysis algorithms, such as ***Curve Interpolation, Trajectory Refinement, and Trajectory Combination*** that enhance the map matched trajectory that Valhalla's Meili and other state of the art map-matching algorithms return. It should be noted that plain solutions, such as computing the shortest path between map-matched points, do not work well for sparse GPS data[15]. This is because merely computing the shortest path between map-matched points does not guarantee accurate trajectory representation, especially in complex urban road network, as moving vehicles do not always follow the shortest path. In this way, we obtain a much more precise and realistic trajectory that can be exploited by WP4 components for AI/ML model training. Each of the three algorithms mentioned above can be assigned to one of two distinct categories: (a) The first category encompasses algorithms that operate in an online fashion using the (streaming) feed of GPS data and the underlying road network, independent of supplementary data or specific information. Instead, (b) the second category uses historical data pertaining to analogous vehicle trajectories for operation and exploits this historical data for more accurate trajectory reconstruction. The Trajectory Refinement algorithm and the Curve Interpolation algorithm belong to the first category, whereas the Trajectory Combination algorithm belongs to the second category, i.e., it exploits information from historical trajectories for the same itinerary. In the current version (as of M15 of the Project), we have developed preliminary versions of these algorithms. In Figure 3-3 we can observe the main ideas and differences of these algorithms.



**Trajectory Refinement Algorithm**

Utilizes OpenStreetMap to refine trajectories by interpolating points only at road transitions to enhance trajectory accuracy, detail and overall smoothness. This algorithm, unlike the others is used for correcting GPS inaccuracies at corners or junctions

**Curve Interpolation Algorithm**

Detects significant directional shifts in trajectories to interpolate points along curved road segments, enhancing trajectory detail. Contrary to the other algorithms, it emphasizes on the smoothing of trajectories at in roundabouts and curved sections in general.

**Trajectory Combination Algorithm**

Merges multiple repeatable trajectories (e.g., bus routes) into a coherent single trajectory. Different to the other algorithms, it uses historical data to achieve increased trajectory accuracy and detail.

**Figure 3-3 - The Main Ideas of Trajectory Refinement, Curve Interpolation and Trajectory Combination algorithms.**

The ***Trajectory Refinement algorithm*** significantly improves trajectory accuracy by integrating GPS data with road network information from OpenStreetMap. It precisely matches each GPS point to the road segment directly beneath it, utilizing the 'osmid' attribute to identify the segment. This matching occurs specifically when there is a change in the 'osmid' from one point to the next, signalling a transition between road segments. At such transitions, typically at the apex of turns or where roads change, the algorithm interpolates a point to the intersection of the two segments, thereby enhancing the trajectory's alignment with the physical road network. This process not only refines the trajectory's smoothness but also safeguards its spatio-temporal coherence. The initial step involves converting GPS coordinates into Shapely Point objects, preparing them for spatial operations. Following this, the algorithm retrieves a road network graph from OpenStreetMap, aligning the GPS trajectory with the actual road layout, represented as nodes and edges in the graph. By examining each GPS point and its associated road segment, the algorithm identifies transitions between segments and interpolates points at these junctures to smooth the trajectory, a critical step for correcting GPS inaccuracies, especially at corners or junctions. It also adjusts timestamps for new points to ensure temporal alignment with the spatial refinements. The algorithm culminates in compiling both the original and interpolated points, providing a more accurate representation of the path. Optional post-processing can further refine the trajectory, removing points that stray from the road network, based on a set tolerance, to ensure the trajectory's fidelity to the actual route. This algorithm performs efficiently on its own, delivering significant improvements in trajectory alignment with the road network, and serves as a foundational step for Curve Interpolation.

The ***Curve Interpolation algorithm*** constructs an accurate path representation by calculating the bearing between consecutive points to identify significant directional shifts, using a variable threshold to detect curves. It also introduces 'granularity', a variable denoting the number of points to interpolate along each identified curve, enabling adjustable detail enhancement. This algorithm employs geodesic interpolation, considering the Earth's curvature for a truer geographical path depiction, over simple linear methods. It meticulously aligns interpolated points with corresponding temporal information, ensuring the trajectory's spatio-temporal coherence. Thus, by leveraging a bearing threshold and granularity, the algorithm effectively smooths trajectories and maintains their integrity, proving useful for accurate spatial analysis in paths with frequent directional changes. While capable of operating independently, Curve Interpolation is recommended to follow Trajectory Refinement, as it benefits from refining the trajectory with essential details on turns and road changes.

Finally, the ***Trajectory Combination algorithm*** refines the accuracy of repeatable trajectories, such as bus routes, by a comprehensive process that initially merges multiple sets of trajectory data, thereby amalgamating diverse trip instances into a single dataset. This integration leverages latitude and longitude coordinates, which are subsequently extracted and structured for precise geodesic distance calculations using the *WGS 84 ellipsoidal model*, ensuring an accurate representation of distances on the Earth's surface. The algorithm then employs a sorting mechanism that iteratively identifies the nearest unvisited point from the current location based on geodesic proximity, thereby systematically organizing the trajectory points. This sorting process relies on a 'visited' set to track the inclusion of points, ensuring a unique and logical sequence. The culmination of this algorithm is the construction of a reordered dataset that presents a spatially coherent and logically sequenced trajectory, reflecting an optimized aggregation of the individual paths. This enhanced trajectory not only improves the fidelity of the representation of the route but also ensures the preservation of its spatial and temporal integrity, providing a reliable foundation for subsequent analyses and applications requiring accurate path data.

### 3.1.3 Preliminary Evaluation

Our preliminary evaluation results are based on visual inspection of the map-matched trajectories in comparison with the reconstructed trajectories using our algorithms. We used a dataset from Use-case #3 Public Transport Trip Characteristics Inference and Traffic Flow Data Analytics in Riga, filtering one specific day: August 30th, 2019. Also, the OpenStreetMap data for each trajectory is downloaded while the Trajectory Refinement component is executed via the **osmnx** python library.

Figure 3-4 provides an illustrative depiction exemplifying the operational execution of two algorithms applied to augment a sparsely sampled map-matched vehicle trajectory. It shows how a map-matched trajectory from Use-case #3 dataset gets enhanced by using the Trajectory Refinement and Curve Interpolation algorithms. In this example, Trajectory Refinement has interpolated points at crucial junctures in the path, such as turns or road segment transitions, as seen in the lower section of the figure. Additionally, the Curve Interpolation has included many points to reflect the trajectory's curvature, as apparent in the figure’s upper right segment when the vehicle passes through a roundabout and generally in every curved segment of the given trajectory.

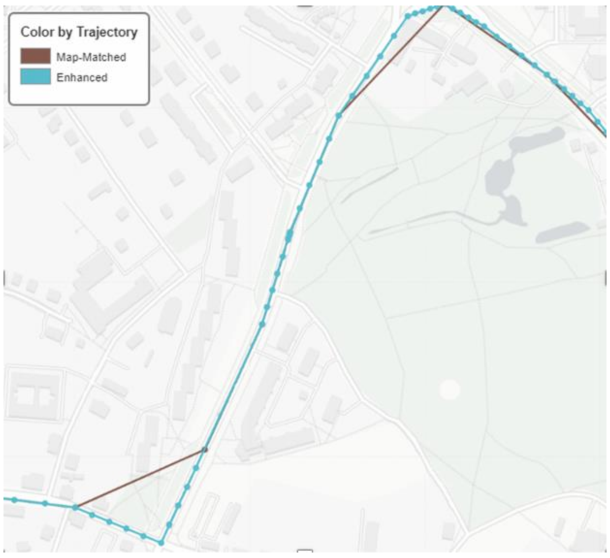


**Figure 3-4 - Enhancing the map matched trajectory with the Curve Interpolation and Trajectory Refinement algorithms**

Figure 3-5 is a visual representation of the Trajectory Combination algorithm in operation, which combines two sparsely sampled map matched trajectories to produce a detailed trajectory. It shows

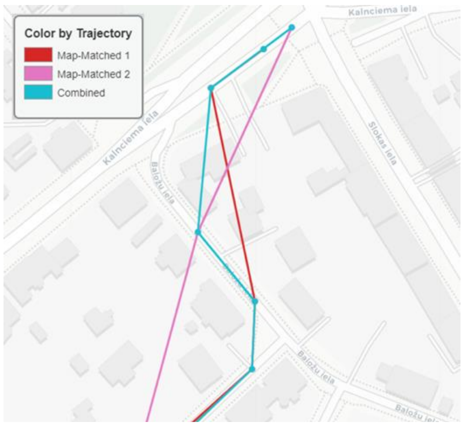


**Figure 3-5 - Enhancing the Map Matched trajectory with the Trajectory Combination algorithm.**

how two map-matched trajectories from Use-case 3 dataset get enhanced by using the Trajectory Combination algorithm.

For a preliminary quantitative evaluation, we assessed the performance of the Key Performance Indicators (KPIs) for map-matching. The experiment was run in a system equipped with AMD Ryzen 7 7800X3D CPU, Samsung 990 pro 1TB SSD, 32GB DDR5 of RAM. The metric used was the throughput, measured in number of processed records per second (records/sec). The throughput for the Trajectory Refinement (includes the time to look up the graph) was about 42 records/sec for trajectories of a small size (116 points), and 132 records/sec for larger trajectories (1169 points). This enhancement highlights the algorithm's capacity to effectively manage larger input data. On the other hand, when handling larger trajectories, the Curve Interpolation's processing performance decreased from 9,404 records/sec to 7,395 records/sec for smaller inputs. On the other hand, the Trajectory Combination has an overall throughput of 17,253 records/sec. The average throughput of the Trajectory Refinement and Curve Interpolation when coupled was further calculated, producing 111 and 7,540 records/sec, respectively. These numbers are promising but we believe that they can be improved further in the next phase of the project. In particular, we aim to investigate improvements in the performance of the Trajectory Refinement by improving the access cost to the underlying graph representing the road network.

More precisely, Table 7 summarizes the KPIs relevant to this emerald:

**Table 7 - KPIs for Map Matching**

| Description | Baseline Value | Target Value | Method of Measurement | Achieved Value |
|---|---|---|---|---|
| **Trajectory Combination Performance Improvement** | N/A | Thousands of records/sec | Throughput (records/sec) | hundreds of records per sec for large trajectories |
| **Curve Interpolation Performance Improvement** | N/A | Thousands of records/sec | Throughput (records/sec) | hundreds of records per sec for large trajectories |
| **Trajectory Refinement Performance Improvement** | N/A | Thousands of records/sec | Throughput (records/sec) | hundreds of records per sec for large trajectories |

### 3.1.4 *Next Steps*

In the next phase of the project, we plan to work on the following aspects:

- Evaluate the algorithms for larger datasets and using quantifiable measures rather than only by visual inspection.
- Refine the algorithms to increase result quality and accuracy.
- Focus on performance and scalability aspects, including data-parallel processing.
- Apply our results in more datasets; both public and from the use-cases of the project.

## 3.2 Weather Enrichment

The weather enrichment subtask is a data integration task, aiming at enriching mobility data sets with weather-related information. Weather plays a vital role in the analysis of moving objects' trajectories, affecting human mobility. To discover weather-related mobility patterns, it is necessary to perform exploratory data analysis. This requires the availability of weather conditions together with positional information for each moving object. Moreover, having together the weather information with mobility information, complex reasoning about trajectory data is enabled, with prominent examples trajectory prediction and clustering. The weather information is fetched from gridded-binary (GRIB) files. This kind of format is used for the storage and transport of gridded meteorological data. The format is maintained by the World Meteorological Organization, and data (numeric values) are stored on a 3D grid of points. The grid is formed by the splitting of the longitude and latitude axes per a specified value of degrees. The methods introduced are useful in establishing a robust framework for linking with Copernicus data, enabling the enrichment of mobility datasets with accurate and relevant weather information. Standardized data access protocols such as OGC web services, specifically the Web Coverage Service (WCS) for retrieving Copernicus data in geospatial formats are leveraged to access gridded meteorological data stored in Copernicus repositories and fetching weather-related information like temperature, humidity, wind speed, and precipitation.

### *3.2.1 Brief Survey of the State-of-the-Art*

Few works study the concept of weather data integration, focusing on real-time applications. Gooch and Chandrasekar[16] present the concept of integrating weather radar data with ground sensor data which will respond to emergent weather phenomena such as tornadoes, hailstorms, etc. For the integration procedure a special technique is used to address the high dimensionality of weather radar data. Kolokoloc[17] applies open-access weather-climate data integration on local urban areas. Specifically, by using open-access data by meteo-services, integration of weather data is applied on locations stored in MySQL database. The proposed mode of operation is record-by-record, accessing at the same time weather information in GRIB (binary) format files, used widely for storing meteorological data efficiently. Their work has been also extended for operating in parallel, extreme-scale mode[18]. Weather data integration has also been studied to link mobility data to external data sources. This aims at producing semantic trajectories, and one external source that has been considered is weather. FAIMUSS[19] is a system that generates links between mobility data and other geographical data sources, including static areas of interest, points of interest, and weather. However, this also poses an overhead to the application since it dictates the use of an ontology and the representation of domain concepts.

### *3.2.2 Overview and Description*

The weather enrichment component[5] is a service that integrates weather information to data that contain spatial and temporal information. The component's structure is composed of two parts, each one having a specific role concerning the enrichment process (Figure 3-6). The first part is the spatio-temporal parser, which parses the given input data set. The parser supports local file reading of CSV, TXT and JSON formats. It also supports reading from streaming data (topics) on the Kafka streaming platform[6] for extreme scale weather data enrichment. The parser forwards the spatial and temporal information to the weather data obtainer which constitutes the second part. The weather data obtainer does take the spatio-temporal information of the record and finds for it the respective weather information. The weather information is fetched as several attributes, each associated with a numeric value describing the prevailing weather conditions (e.g. temperature).

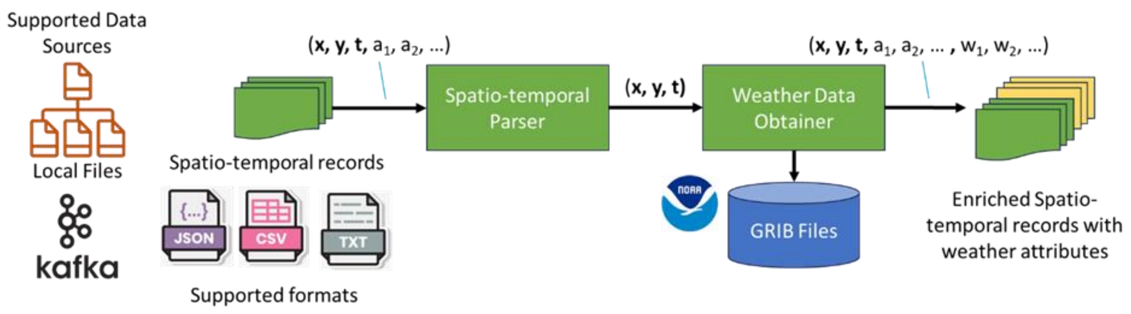


**Figure 3-6 - Weather enrichment architecture**

In the context of the EMERALDS project, we utilize the GRIB files that are offered by the National Oceanic and Atmospheric Administration (NOAA). The weather information is stored in a grid, split into the two axes per 0.5° degree. This means that there exists a point per 0.5° degrees on each axe, that contains the values of weather attributes for the prevailing weather conditions for its location. The provided data are computer-generated, based on the Numerical Weather Prediction models. The utilized GRIB files contain dozens of attributes that cover the time span of 6 hours. One day is composed of 4 GRIB files. The enrichment process includes the integration of rain, snow, ice, wind, humidity-related fields, which are the following ones:

- Per_cent_frozen_precipitation_surface
- Precipitable_water_entire_atmosphere_single_layer
- Precipitation_rate_surface_3_Hour_Average
- Storm_relative_helicity_height_above_ground_layer
- Total_precipitation_surface_3_Hour_Accumulation
- Categorical_Rain_surface_3_Hour_Average
- Categorical_Freezing_Rain_surface_3_Hour_Average
- Categorical_Ice_Pellets_surface_3_Hour_Average
- Categorical_Snow_surface_3_Hour_Average
- Convective_Precipitation_Rate_surface_3_Hour_Average
- Convective_precipitation_surface_3_Hour_Accumulation
- U-Component_Storm_Motion_height_above_ground_layer
- V-Component_Storm_Motion_height_above_ground_layer
- Geopotential_height_highest_tropospheric_freezing
- Relative_humidity_highest_tropospheric_freezing
- Ice_cover_surface
- Snow_depth_surface
- Water_equivalent_of_accumulated_snow_depth_surface
- Wind_speed_gust_surface
- u-component_of_wind_maximum_wind
- u-component_of_wind_height_above_ground
- v-component_of_wind_maximum_wind
- v-component_of_wind_height_above_ground
- Total_cloud_cover_low_cloud_3_Hour_Average

Given a spatio-temporal position, the respective accessed GRIB file is the one whose timespan covers the position's time. Then, based on its location, the values from the 4 nearest grid points are fetched for the weather attributes we want to integrate to the spatio-temporal data. The value for a specific weather attribute for the given spatio-temporal position, is calculated by interpolating the four values of the nearest grid points considering their distance to the given position. For this purpose, the inverse distance weight formula is used:

$$V = \frac{v1\frac{1}{d1} + v2\frac{1}{d2} + v3\frac{1}{d3} + v4\frac{1}{d4}}{\frac{1}{d1} + \frac{1}{d2} + \frac{1}{d3} + \frac{1}{d4}}$$

Previous implementation of the weather integrator mechanism considered for a weather attribute, the value of the nearest grid point to a given spatio-temporal point. By interpolating the four values of the nearest grid points we achieve the integration of more accurate values to spatio-temporal points. Figure 3-7 shows an instance of the previous (left) and the current (right) implementation for determining the value of a field to a spatio-temporal point (red dot).

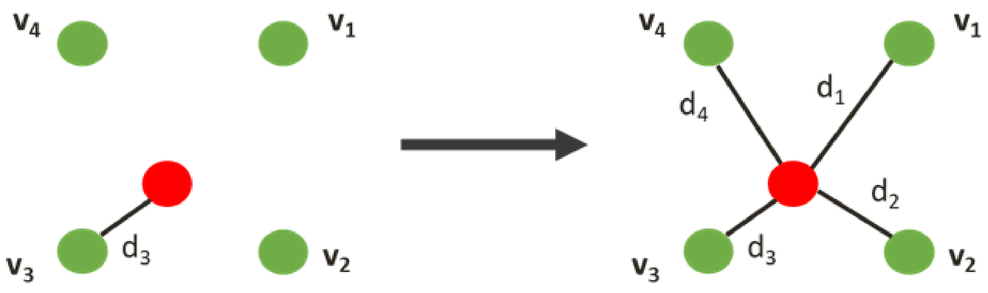


**Figure 3-7 - Previous and current approach for determining the value of a weather attribute.**

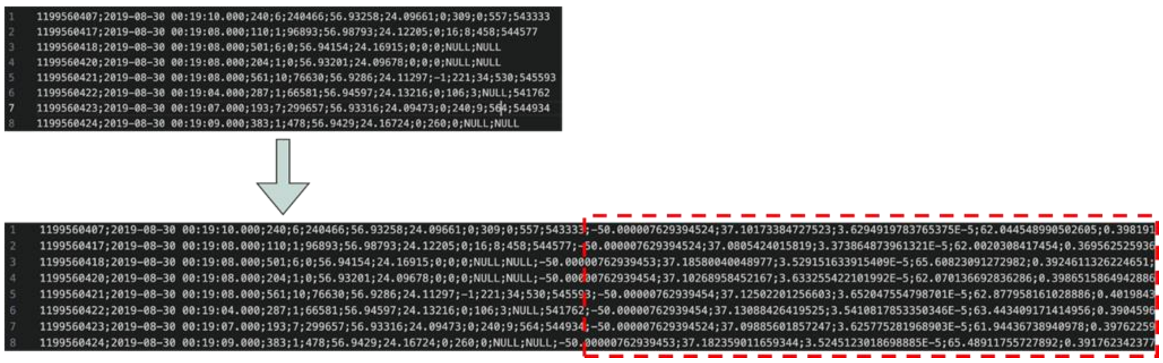
As aforementioned, the weather enrichment component service can support extreme scale data enrichment via the Apache Kafka Streaming platform. Specifically, the component operates in multiple consumers, which consume the data (messages) from a topic. The messages are enriched with weather attributes and written to another topic that contains the enriched data. GRIB files can be accessed by storing them either in each node where the consumers run or in a distributed file storage system such as Hadoop distributed file system (HDFS). The former approach has disk space requirements as GRIB files are large, while the latter requires large bandwidth as the GRIB files will be transferred to the nodes when requested over the network.

### *3.2.3 Preliminary Evaluation*

We evaluated the weather enrichment component in centralized environment by running the enrichment process on a sample of public transportation data (GPS data of buses) in the city of Riga (UC #3). The sample is in CSV file format (2 files) and covers the data of 1 day, composed of 1.2M spatio-temporal points. The initial data set consisted of 12 columns related to mobility and vehicle information, whose size was 96 MB. The enrichment process integrated 24 weather attributes to the data set, whose size was 479 MB. Table 8 shows information about the GRIB file size and performance, where the throughput is about 85 thousand records per second. This constitutes the 1st KPI "Performance measurement of the enrichment in terms of execution time". Figure 3-8 shows a sample of the initial and the enriched records in CSV file. The red dashed rectangle indicates the values of the weather attributes.

| GRIB Files size (1 day) | Throughput (rec/sec) | Runtime (sec) |
|---|---|---|
| **378 MB** | 85,068 | 14 |

**Table 8 - GRIB Files Size and Performance**



```
1199560407;2019-08-30 00:19:10.000;240;6;240466;56.93258;24.09661;0;309;0;557;543333
1199560417;2019-08-30 00:19:08.000;110;1;96893;56.98793;24.12205;0;16;8;458;544577
1199560418;2019-08-30 00:19:08.000;501;6;0;56.94154;24.16915;0;0;0;NULL;NULL
1199560420;2019-08-30 00:19:08.000;204;1;0;56.93201;24.09678;0;0;0;NULL;NULL
1199560421;2019-08-30 00:19:08.000;561;10;76630;56.9286;24.11297;-1;221;34;530;545593
1199560422;2019-08-30 00:19:04.000;287;1;66581;56.94597;24.13216;0;106;3;NULL;541762
1199560423;2019-08-30 00:19:07.000;193;7;299657;56.93316;24.09473;0;240;9;564;544934
1199560424;2019-08-30 00:19:09.000;383;1;478;56.9429;24.16724;0;260;0;NULL;NULL
```

```
1199560407;2019-08-30 00:19:10.000;240;6;240466;56.93258;24.09661;0;309;0;557;543333;-50.000007629394524;37.10173384727523;3.62949197837653 75E-5;62.044548990502605;0.398191
1199560417;2019-08-30 00:19:08.000;110;1;96893;56.98793;24.12205;0;16;8;458;544577;-50.000007629394524;37.0805424015819;3.373864873961321E-5;62.0020308417454;0.36956252593б
1199560418;2019-08-30 00:19:08.000;501;6;0;56.94154;24.16915;0;0;0;NULL;NULL;-50.00000762939453;37.18580040048977;3.529151633915409E-5;65.60823091272982;0.3924611326224651;
1199560420;2019-08-30 00:19:08.000;204;1;0;56.93201;24.09678;0;0;0;NULL;NULL;-50.00000762939454;37.10268958452167;3.633255422101992E-5;62.070136692836286;0.3986515864942886
1199560421;2019-08-30 00:19:08.000;561;10;76630;56.9286;24.11297;-1;221;34;530;545593;-50.00000762939454;37.12502201256603;3.652047554798701E-5;62.877958161028886;0.4019843
1199560422;2019-08-30 00:19:04.000;287;1;66581;56.94597;24.13216;0;106;3;NULL;541762;-50.00000762939454;37.13088426419525;3.541081785335034 6E-5;63.443409171414956;0.3904596
1199560423;2019-08-30 00:19:07.000;193;7;299657;56.93316;24.09473;0;240;9;564;544934;-50.000007629394524;37.09885601857247;3.625775281968903E-5;61.94436738940978;0.39762259
1199560424;2019-08-30 00:19:09.000;383;1;478;56.9429;24.16724;0;260;0;NULL;NULL;-50.00000762939453;37.182359011659344;3.524512301869888 5E-5;65.48911755727892;0.391762342377
```

**Figure 3-8 - Initial and Final (enriched) data of records existing in CSV format.**

We also proceeded to a qualitative evaluation regarding the values of the weather attributes that were integrated to the data set. This constitutes the 2nd KPI of the component "Measurement of the difference of the interpolated values with the non-interpolated".Table 9 shows per weather attribute the range of values (min, max) that result from the interpolation. Along with that, the average from the differences of the interpolated and non-interpolated values has been calculated per weather attribute. In a few words, this measures the average deviation between the interpolated and non-interpolated values, indicating the fixed error that is made by performing interpolation. Notice that for some values the average is zero. The reason behind that is because the values of the weather attributes are also zero as the covering weather conditions from these attributes were not in effect for the spatial and temporal bounds of the data set.

**Table 9 – Statistical Value of the Weather Attributes**

| Attribute | Min | Max | Avg of Diff |
|---|---|---|---|
| **Per_cent_frozen_precipitation_surface** | -18.91709 | 0.000006 | 5.15163 |
| **Precipitable_water_entire_atmosphere_single_layer** | 20.68947 | 47.6 | 1.04257 |
| **Precipitation_rate_surface_3_Hour_Average** | 0.0 | 0.0000009 | 0.000002 |
| **Storm_relative_helicity_height_above_ground_layer** | 10.84061 | 82.77035 | 9.47422 |
| **Total_precipitation_surface_3_Hour_Accumulation** | 0.0 | 0.61383 | 0.01733 |
| **Categorical_Rain_surface_3_Hour_Average** | 0.0 | 1.0 | 0.10891 |
| **Categorical_Freezing_Rain_surface_3_Hour_Average** | 0.0 | 0.0 | 0.0 |
| **Categorical_Ice_Pellets_surface_3_Hour_Average** | 0.0 | 0.0 | 0.0 |
| **Categorical_Snow_surface_3_Hour_Average** | 0.0 | 0.0 | 0.0 |
| **Convective_Precipitation_Rate_surface_3_Hour_Average** | 0.0 | 0.0000009 | 0.000001 |
| **Convective_precipitation_surface_3_Hour_Accumulation** | 0.0 | 0.61383 | 0.01690 |
| **U-Component_Storm_Motion_height_above_ground_layer** | -0.00123 | 0.00099 | 0.41304 |
| **V-Component_Storm_Motion_height_above_ground_layer** | -1.86747 | 7.41703 | 0.38406 |
| **Geopotential_height_highest_tropospheric_freezing** | 3644.90429 | 4968.32 | 106.76319 |
| **Relative_humidity_highest_tropospheric_freezing** | 21.0 | 80.69102 | 1.72259 |
| **Ice_cover_surface** | 0.0 | 0.0 | 0.0 |
| **Snow_depth_surface** | 0.0 | 0.0 | 0.0 |

| | | | |
|---|---|---|---|
| **Water_equivalent_of_accumulated_snow_depth_surface** | 0.0 | 0.0 | 0.0 |
| **Wind_speed_gust_surface** | 1.42879 | 8.90881 | 0.50519 |
| **u-component_of_wind_maximum_wind** | -15.13828 | 9.29803 | 0.08730 |
| **u-component_of_wind_height_above_ground** | -0.00100 | 0.00078 | 0.32974 |
| **v-component_of_wind_maximum_wind** | -1.67917 | 7.14257 | 0.19295 |
| **v-component_of_wind_height_above_ground** | -0.17080 | 8.27791 | 0.38252 |
| **Total_cloud_cover_low_cloud_3_Hour_Average** | 0.0 | 91.0 | 5.04277 |

More precisely, Table 10 summarizes the KPIs relevant to this emerald:

**Table 10 - KPIs for Weather Enrichment**

| Description | Baseline Value | Target Value | Method of Measurement | Achieved Value |
|---|---|---|---|---|
| **Accuracy of Interpolation** | Non Interpolated Values | >0 (Interpolation improved result) | Difference between interpolated and non-interpolated | >0 in 18/24 attributes |
| **Performance** | Thousands records per second | Thousands records per second | Throughput (recs/sec) | ~85,000 |

### *3.2.4 Next Steps*

In the next phase of the project, we are going to apply weather enrichment to datasets of the use-cases, aiming at providing upstream AI/ML algorithms with enriched training datasets. Also, we will apply our solution for bigger input datasets that span larger geographical areas and longer timespans.

## 3.3 Spatial-Temporal Querying

Efficient and scalable querying of large collections of complex multidimensional data is challenging for database systems. This is the case for example for variants of spatial queries, including spatio-temporal and spatio-textual queries. Spatio-temporal queries are typically encountered in applications that concern moving objects, whereas spatio-textual queries are used over geotagged collections of social data (e.g., tweets, check-ins, etc.) as the ones suggested to be used in Use Case #1 Risk-assessment, prediction & forecasting during events in the Hague. Both types of queries are of interest in EMERALDS, as indicated in Deliverable D1.4 “EMERALDS Data Management Plan”. In this work, we focus on *spatio-textual query processing* aiming at querying data collected from social networks and investigating how such external datasets can provide added value to the use-cases of the project.

### *3.3.1 Brief Survey of the State-of-the-Art*

Spatio-textual data have a spatial and a textual dimension. The spatial dimension is the location of an object, whereas the textual dimension refers to some associated text, which is typically represented as a set of keywords. Examples of such data include geotagged tweets, geotagged photos, check-ins in social networks, etc. Efficient processing of spatial-keyword queries is a challenging topic that has been studied recently and several novel indexing and partitioning algorithms have been proposed[20].

However, there still exist limitations of existing solutions in at least two cases: (a) scalable, data-parallel processing a large collection of spatio-textual objects, and (b) semantic querying of spatio-textual objects.

**Scalable processing**: Existing approaches for scalable batch processing of spatio-textual data largely rely on MapReduce and exploit big data frameworks, such as Apache Spark or Apache Flink. Ballesteros et al.[21] propose SpSJoin for parallel processing of spatio-textual joins. Given two datasets $R$ and $S$, the best match in $S$ for each object in $R$ is retrieved. Zhang et al.[22] propose a data-parallel method for processing the spatio-textual similarity join (STSJ) in MapReduce. Given two collections of spatio-textual objects with a spatial location and textual descriptions, STSJ finds out all similar object pairs that have similar textual descriptions and are spatially close to each other. For more details and relevant surveys on processing of spatial keyword queries[23,24].

**Semantic query processing**: Spatial keyword query processing still largely relies on exact matching of query keywords with object keywords. Only a handful of works use semantic representations of text (usually dense vector representations produced from GloVe, Word2Vec, etc.) that allow semantic matching. In this way, a keyword “king” has similarity with another keyword “queen”, even though they have no syntactic matching. S2R-tree[25] is an extension of the R-tree built on top of spatial coordinates and very low-dimensional semantic vectors that represent the text. The semantic vectors are obtained by projecting the high-dimensional word embeddings to an $m$-dimensional space using a pivot-based technique. The NIQ-tree[26] is a multi-level indexing structure where at the top level, a Quadtree is built on the spatial coordinates and then, at the second level, the objects of each leaf are indexed based on topic relevance. For this, Latent Dirichlet Allocation (LDA) is used to create a probabilistic topic model that captures semantics. Finally, at the last level, $n$-gram inverted lists are constructed. Later, the approach is extended by introducing another index, called LHQtree[27]. Both approaches belong to the “spatial-first” approach, meaning that they organize first the spatial information and then the textual. Instead, as will be discussed below, our work follows a hybrid approach where we jointly index the spatial and the textual domain, thus leading to improved pruning and higher efficiency.

### 3.3.2 Overview and Description

In the following, we describe two approaches related to spatio-textual processing. The first concerns scalable processing of spatial keyword range queries, whereas the second is about semantic spatio-textual query processing.

**Scalable Processing of Spatial Keyword Range Queries**

Consider a data set $D$ of spatio-textual data objects, where each object $p$ is associated with a spatial location $(p.x, p.y)$ as well as a set of keywords (tags) denoted with $P$. We use capital letters $(P,Q)$ to denote the keyword sets associated with an object $(p, q)$. Given a query $q$ that consists of a location $(q.x, q.y)$ and a set of keywords $Q$, the spatial-keyword range query retrieves all objects within distance $r$ and having keyword set similarity above $\tau$. More formally:

**Definition** (*Spatial-keyword range query*): Given a query point $q$, a set of query keywords $Q$, a distance threshold $r$, and a textual similarity threshold $\tau$, the spatial-keyword range query retrieves all spatio-textual objects $p \in D$, such that: $\text{dist}(p, q) \le r$ and $\text{sim}(P,Q) \ge \tau$.

Notice that in this work, we use the Euclidean distance function:

$$\text{dist}(p, q) = \sqrt{(p.x - q.x)^2 + (p.y - q.y)^2}$$

for the spatial domain, and we use the Jaccard similarity:

$$\text{sim}(P,Q) = |P \cap Q| / |P \cup Q|$$

for the textual set similarity that includes the complete textual message. In a big data setting, the data set $D$ is horizontally partitioned to a set of $m$ worker nodes. By horizontal partitioning, we refer to a setup where each worker $Wi$ contains a subset $Di$ of the objects of the data set, where the union of $Di = D$ and $Di \cap Dj = \emptyset$, for $i \neq j$. The worker nodes $\{W_1, W_2, \ldots, Wm\}$ can be virtual machines (VMs) in a cloud infrastructure. The problem addressed here is how to efficiently process the spatial-keyword range query in parallel, given such a hardware infrastructure of $m$ workers, aiming at minimizing the query execution time.

Our approach is based on mapping the spatio-textual objects in a 2D space, where one dimension is used to represent distance and the other textual similarity. For the distance, the basic idea is that instead of storing the spatial coordinates of spatio-textual objects, we can store the distance to specific locations. For this purpose, we cluster the locations of spatio-textual objects using a partitioning algorithm, such as K-means. Each cluster $Ci$ is represented by a reference object $K_i$ (its centroid) and a radius $r_i$ , which is the distance of the farthest point assigned to $Ci$ from $K_i$ . Then, for each spatio-textual object $p$, we compute its distance dist($K_i$ , $p$) from the centroid $Ki$ of its cluster. Then, we use the iDistance technique, which produces for object $p$ the 1D value:

$$\text{iDist}(p) = i * c + \text{dist}(K_i , p)$$

where $c$ is a constant used to separate the values produced from different clusters. By setting the value of $c$ large enough, all objects in cluster $Ci$ are mapped to the interval:

$$[i * c, (i + 1) * c]$$

Thus, each object $p$ is mapped to a 1D value iDist($p$). In addition, we maintain in memory the cluster centroid $Ki$ and the radius $ri$ of each cluster.

The next step is to map the textual information, i.e., the keywords of a spatio-textual object, to 1D values, in a way that supports efficient querying. Let $V$ denote the vocabulary of the data set, i.e., $V = \{t_1, t_2, \ldots, t_{|V|}\}$, where $ti$ represents a keyword of the vocabulary. Following a similar rationale as for the spatial dimension, our objective here is to partition $V$ in $k$ disjoint subsets of keywords. For this purpose, we build the keyword co-occurrence graph, an undirected weighted graph that contains as vertices the keywords from the vocabulary $V$. An edge between two keywords (vertices) is added if these keywords co-occur in at least one spatio-textual object. The weight of an edge is set equal to the number of objects in which the respective keywords co-occur. Then, we apply a graph partitioning algorithm to generate the desired subsets of keywords $\{V_1 ,V_2, \ldots ,V_k\}$. These can be considered as the textual partitions of the data set.

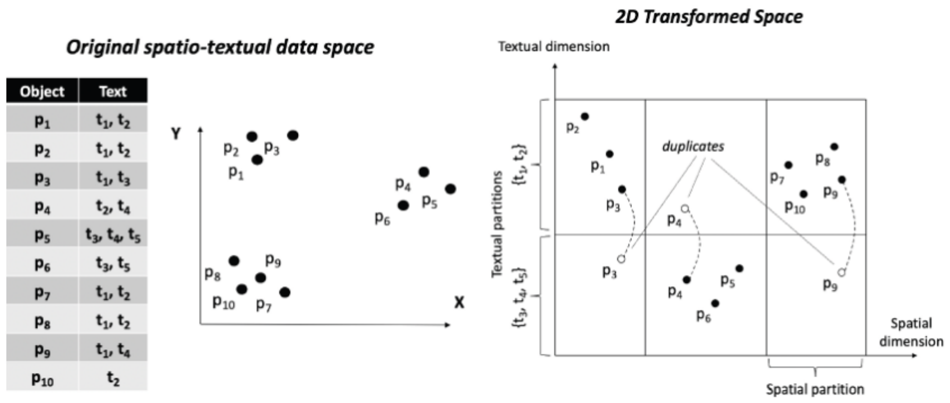
A spatio-textual data object $p$ may contain keywords from different (say $\ell \geq 1$) subsets $Vi$. Thus, we need to assign such a spatio-textual object $p$ to $\ell$ partitions. Therefore, $p$ is replicated $\ell$ times in the transformed data set. For each partition $Vi$ we eventually compute a 1D value val($P$,$Vi$ ) for object $p$ with keyword set $P$. This is necessary in order to ensure the correctness of the computed result set for any spatial-keyword range query. For a given spatio-textual data object $p$ (with keyword set $P$) and a partition $Vi$ for which it holds that $P \cap Vi \neq \emptyset$, we define a 1D similarity value val($P$,$Vi$ ) based on the following equation:

$$\text{val}(P,Vi ) = |P \cap Vi | / |P |$$

which practically "distributes" the size of the overlap between sets $P$ and the vocabulary $V$ to those partitions $Vi$ that have $P \cap Vi \neq \emptyset$. In fact, val($P$,$Vi$ ) represents the size of the overlap normalized over the size of $P$.

The next step is to assign similarity values of data points for different partitions $Vi$ ,$Vj$ to disjoint 1D intervals of the textual dimension. Thus, the textual similarity values val($P$,$Vi$ ) are mapped to 1D values in such a way that only data objects that have keywords that belong to the partition $Vi$ are mapped to the same interval. Expecting that $c'$ is large enough, all data objects $p$ with keywords belonging to the partition $Vi$ ($P \cap Vi \neq \emptyset$) are mapped to the 1D values:

$$\text{iSim}(p,Vi) = i * c' + \text{val}(P,Vi)$$



| Object | Text |
|---|---|
| $p_1$ | $t_1, t_2$ |
| $p_2$ | $t_1, t_2$ |
| $p_3$ | $t_1, t_3$ |
| $p_4$ | $t_2, t_4$ |
| $p_5$ | $t_3, t_4, t_5$ |
| $p_6$ | $t_3, t_5$ |
| $p_7$ | $t_1, t_2$ |
| $p_8$ | $t_1, t_2$ |
| $p_9$ | $t_1, t_4$ |
| $p_{10}$ | $t_2$ |

**Figure 3-9- Example of our mapping to 2D. Left: original space. Right: transformed space**

presents a graphical overview of the mapping approach. The data set $D$ depicted on the left, whereas the transformed 2D space is shown on the right. In summary, the location information is mapped in one dimension (horizontal axis), and the textual information in another dimension (vertical axis). Essentially, in the transformed 2D space, the objects form spatial partitions based on their pairwise distances, as well as textual partitions (in the example $\{t1, t2\}$ and $\{t3, t4, t5\}$) based on grouping together subsets of frequently co-occurring keywords. While each object belongs to a single spatial partition, it may be assigned to multiple textual partitions. For example, objects $p1$, $p2$ and $p3$ are assigned to the same spatial partition because they form a spatial cluster in the original space. On the other hand, objects $p3$, $p4$ and $p9$ are duplicated to both textual partitions depicted in Figure 3-9, because they contain keywords from both textual partitions.

At a pre-processing step, we generate data partitions in the 2D space that consist of the spatio-textual objects of the input data set $D$. Figure 3-10 provides a visual representation of the generated 2D partitions, where the x-axis corresponds to the spatial dimension and the y-axis is the textual dimension. In this example, 10 spatial clusters and 20 textual partitions are depicted. The visualization shows that distinct partitions are generated in the transformed 2D space, which intuitively explains how a query with a reasonably small distance threshold and relatively high textual similarity threshold will need to access very few partitions.

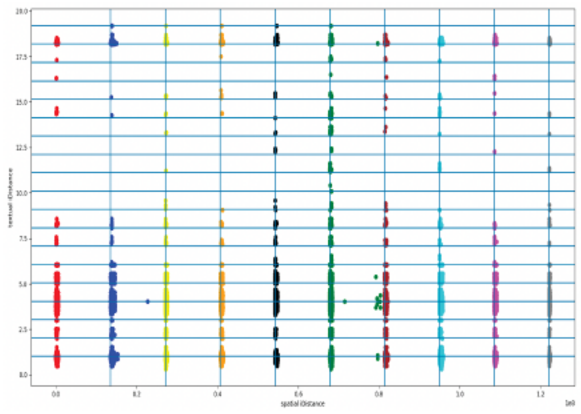


**Figure 3-10- Visualization of 2D partitions generated by our algorithm.**

We organize the obtained data partitions using Hive partitions. A Hive partition corresponds to a physical directory on disk which contains data based on a specific attribute value. We use the spatial cluster ID for this purpose, so each Hive partition corresponds to a spatial cluster. Within each Hive partition, we organize data based on the textual cluster ID. Furthermore, we use Apache Parquet as data format for the actual data storage. Parquet is a column-oriented storage format, which provides compressed storage as well as efficient retrieval by means of data skipping. Essentially, given a query predicate, Parquet maintains metadata (e.g., min-max values) on columns that allow skipping (i.e., not accessing) entire data blocks, if the metadata can ascertain that certain blocks do not match the query. Finally, during pre-processing, we generate a metadata table, which is maintained in memory for efficient access. It contains for each spatial cluster, its identifier, its centroid and radius, and then for each textual cluster, its identifier and the keywords that define it.

We designed and implemented an algorithm in Apache Spark for scalable query processing (Figure 3-11). As a first step, the algorithm uses the metadata table to find the spatial and textual partitions that contain candidate results for the query. The spatial partitions are those that overlap with the circle centered at $(q.x, q.y)$ and having radius $r$. These are the only spatial partitions that contain candidate results for the query. The identifiers of these spatial partitions are denoted with $Pspatial$ (line 4). We can show that only a limited set of textual partitions need to be accessed (for details we point to our published paper[28]). The identifiers of these textual partitions are denoted with $Ptextual$ (line 5).

```
Algorithm 1 Spark-based spatial-keyword range algorithm
1: Input: Query $q$ $(q.x, q.y, Q)$, radius $r$, similarity threshold $\tau$
2: Output: Result set $\mathcal{R}$
3: $\mathcal{R} \leftarrow \emptyset$
4: $P_{spatial} \leftarrow$ findSpatialPartitions$(q.x, q.y, r)$
5: $P_{textual} \leftarrow$ findTextualPartitions$(Q)$
6: $df \leftarrow$ spark.read
           .parquet(path)
           .filter(col("spatial_id").isin($P_{spatial}$))
           .filter(col("textual_id").isin($P_{textual}$))
7: $\mathcal{R} \leftarrow df$.filter(dist$(p, q) \leq r$).filter(sim$(P, Q) \geq \tau$)
8: return $\mathcal{R}$
```

**Figure 3-11 – Algorithm for scalable query processing in spatio-textual datasets.**

After having identified the partitions that contain candidate results, we use the read operation of Spark to load only the Hive partitions that contain the relevant candidate data objects in a Spark dataframe $df$ (line 6). In this step, two optimizations are performed based on the data organization. First, the spatial cluster identifiers are used as PartitionFilters, effectively limiting the directories that need to be accessed. Second, the textual cluster identifiers are used as PushedFilters, exploiting the Parquet data format and fetching in memory only the candidate data objects. As the algorithm adopts the filter-and-refine methodology, in this step, the filtering is performed, which finds a set of candidate data objects very fast. Then, in line 7, the refinement step takes place. In this step, the dataframe $df$ is processed and only the subset of candidate objects $p$ that fulfil the distance constraint (dist($p$, $q$) ≤ $r$) and the textual constraint (sim($P$,$Q$) ≥ $\tau$) are kept and returned as result R (line 8). In the next section, we present evaluation results of this algorithm in comparison with state-of-the-art algorithms.

**Semantic Spatio-textual Query Processing**

Given a collection of spatio-textual objects $\{o_i\} \in$ O consisting of a location ($o_i.x$, $o_i.y$) and a textual description $o_i.text$ and a query $q$ with a location ($q.x$, $q.y$) and a textual description $q.text$ , retrieve the $k$ objects in O that minimize the distance function $d$ ($q$, $o$) that takes into account both the semantic and the spatial distances of any object to the query. As such, the distance function $d$ ($q$, $o$) includes two components. The first component is the spatial distance $d_s(q, o)$ of the query location ($q.x$, $q.y$) to an object's location ($o.x$, $o.y$). We use a normalized variant of the Euclidean distance for $d_s(q, o)$, obtained by dividing with the maximum possible Euclidean distance of any two objects in the data set.

The second component is the semantic distance $d_t(q, o)$ between $q.text$ and $oi .text$ . To define this distance, we use word embeddings to represent the textual description of a spatio-textual object $o$ as a dense $n$-dimensional vector: $\{o[1], o[2], \ldots, o[n]\}$. This vector is called the semantic representation (or semantic vector) of the spatio-textual object. In principle, any method for word embeddings can be used, e.g., Word2Vec or Glove, as our algorithms do not make any assumption on the specific method. Then, the semantic distance $d_t(q, o)$ is defined as the normalized Euclidean distance applied on the semantic vectors of $q$ and $o$, divided by the maximum possible Euclidean distance.

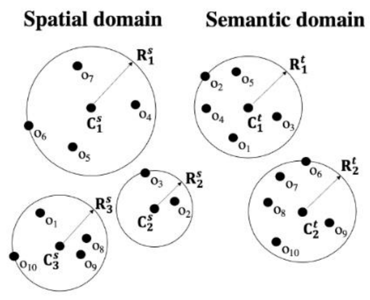
Having both constituent parts normalized in [0, 1], we consider the following equation for the distance function:

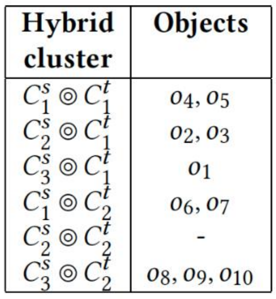
$$d\,(q, o) = \lambda * ds\,(q, o) + (1 - \lambda) * dt\,(q, o)$$

where $\lambda \in [0, 1]$ is a user-dependent parameter that balances the contribution of the two parts to the final distance function.

**Definition** (Semantic Spatio-textual Similarity Search). Given a collection of spatio-textual objects $\{oi\}$ ∈ O and a query object $q$, retrieve the $k$ objects $O_k = \{o_1, \ldots, o_k\}$, such that: $d(q, o_i) \leq d\,(q, o_j)$, $\forall o_i \in O_k$ and $\forall o\ j \in O - O_k$.

To design an efficient query processing algorithm, we follow a joint indexing approach that combines both domains into a single structure using the following steps. We perform two separate clustering algorithms to the spatio-textual dataset. First, regarding the spatial information, we group together objects based on their location by applying spatial clustering. Regarding the textual information, we make use of pre-trained models that produce high-quality word embeddings, obtaining high-dimensional dense semantic vectors that represent the aforementioned object. These semantic vectors are $n$-dimensional, with $n$ in the order of hundreds. Unfortunately, grouping such high-dimensional data by computing pairwise distances is known to be ineffective due to the curse of dimensionality. To tackle this problem, we apply Principal Component Analysis (PCA) in order to project the $n$-dimensional word embeddings to meaningful $m$-dimensional ($m << n$) vectors. The choice of PCA is due to its salient property of maximizing the total variance of the projection. Then, we apply K-Means on the projected vectors. Next, we initialize the set of hybrid clusters H by forming combinations of spatial with semantic clusters. Finally, each object $o$ is assigned to a single hybrid cluster, which is the one formed by $o$'s spatial and semantic clusters respectively.





| Hybrid cluster | Objects |
|---|---|
| $C^s_1 \circledcirc C^t_1$ | $o_4, o_5$ |
| $C^s_2 \circledcirc C^t_1$ | $o_2, o_3$ |
| $C^s_3 \circledcirc C^t_1$ | $o_1$ |
| $C^s_1 \circledcirc C^t_2$ | $o_6, o_7$ |
| $C^s_2 \circledcirc C^t_2$ | - |
| $C^s_3 \circledcirc C^t_2$ | $o_8, o_9, o_{10}$ |

**Figure 3-12 - An example of six (6) hybrid clusters**

Figure 3-12 presents an illustrative example of how the hybrid clusters are formed. In the example, we have $K_s$ = 3 spatial clusters {($C^s{}_1$, $R^s{}_1$), ($C^s{}_2$, $R^s{}_2$), ($C^s{}_3$, $R^s{}_3$)} and $K_t$ = 2 semantic clusters {($C^t{}_1$, $R^t{}_1$), ($C^t{}_2$, $R^t{}_2$)}. As a result, $K_s$ $K_t$ = 6 hybrid clusters can be formed. We make two observations: (a) the hybrid cluster ($C^s{}_2$ , $R^s{}_2$), ($C^t{}_2$, $R^t{}_2$) is not actually formed, since there exists no data object that belongs to both clusters, and (b) each spatio-textual data object is eventually assigned to a single hybrid cluster.

Based on this, we design and implement a query processing algorithm called *Cluster-based Semantic Spatio-textual Indexing* (*CSSI*) for $k$-nearest neighbor ($k$-NN) search over spatio-textual data. Given a query *q*, the algorithm sorts the hybrid clusters based on distance so that the clusters most promising to contain the k-nearest neighbors are accessed first. As the objects of a cluster are accessed, the algorithm maintains a threshold which allows (a) pruning some objects of the cluster (intra-cluster pruning), and (b) pruning entire clusters (inter-cluster pruning). The theory that supports these pruning properties and lead to a provably correct algorithm are described in our paper[29]. Moreover, we present an approximate algorithm CSSIA that is more efficient while it yields results of high accuracy.

### 3.3.3 *Preliminary Evaluation*

**Scalable Processing of Spatial Keyword Range Queries**

Our algorithm and the competitors have been implemented in Apache Spark using Python. For evaluating our algorithm, we used a small-sized cluster infrastructure provided by okeanos, a cloud service that offers virtual computing and storage services, supported by GRNET (https://grnet.gr) for research purposes. In total, the infrastructure consists of 20 cores, 32GB of RAM, and 80GB of secondary storage.

We evaluate the performance of our algorithm, denoted *SpatioTextual Index*. To compare its performance, we use two Spark-based baselines that we developed, as well as two existing systems for big spatial data:

- Spatial First: a Spark-based implementation that applies spatial constraint and then further filters the retrieved data based on the textual constraint. The input data is provided as a CSV file.
- Textual First: also, a Spark-based implementation, but it applies the textual constraint and then filters the retrieved data based on the spatial constraint. The input data is provided as a CSV file.
- Sedona: a spatial first algorithm implemented in Apache Sedona (formerly GeoSpark[30]), using the default spatial index (QuadTree).
- Geomesa: a spatial first algorithm implemented in Geomesa (https://www.geomesa.org/), which uses the Z2 index, a two-dimensional Z-order curve to index latitude and longitude for point data.

We used a real-life data set of 5M geotagged tweets that all contain the GPS location of the user and the text of the tweet, which has been pre-processed and represented as a set of keywords. After pre-processing, each spatio-textual object (i.e., tweet) is represented with a unique identifier, two spatial coordinates, and a set of keywords. Our main metric is the query execution time. Each query is repeated 20 times, and we report the average query execution time. Notice that the query execution time includes the time needed to read data from disk, i.e., we do not perform the queries on in-memory data.

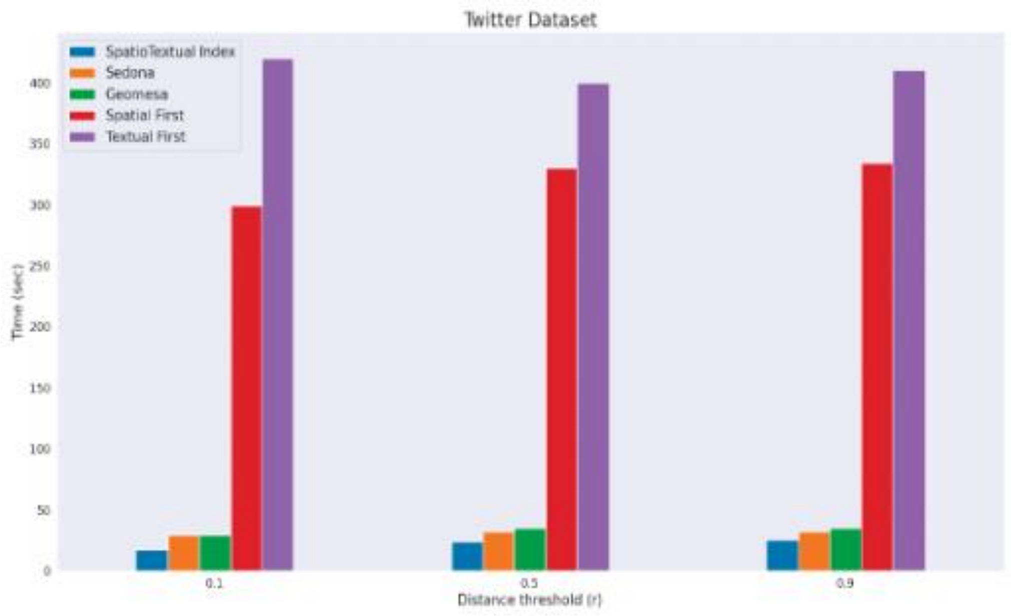


**Figure 3-13 - Effect of varying the distance threshold $r$**

In Figure 3-13 we vary the distance threshold (r) and measure the query execution time. Evidently, when we increase the distance threshold $r$ most algorithms need more time as more objects qualify as query results. Our algorithm (Spatio Textual Index) is faster than all competitors. This demonstrates the benefits of using spatio-textual partitions in our algorithm, as it can prune more objects based both on spatial and textual constraints, and thus improve the query execution time. One interesting observation is that our Spark-based baselines (spatial first and textual first) perform much worse than the other algorithms.

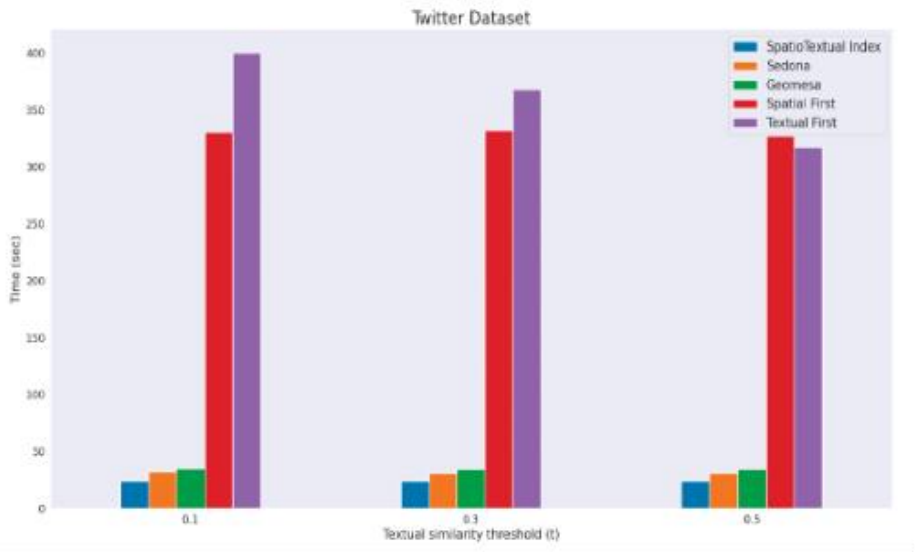


**Figure 3-14- Effect of varying the textual similarity threshold τ**

Figure 3-14 shows the effect of varying the textual similarity threshold $\tau$. It is expected that higher values of $\tau$ will produce fewer results, since it is harder to find spatio-textual objects with very high textual similarity to the query keywords. In turn, this is expected to have an effect on the execution time. The textual first algorithm becomes faster than the spatial first for the high value of $\tau$. The main difference is that Sedona and Geomesa are much faster than textual first for all values of $\tau$. However, out spatio textual index is consistently better than all competitors. For more experiments, we point to our publication in BigSpatial@SIGSPATIAL'23.

**Semantic Spatio-textual Query Processing**

Our algorithms are memory-resident and implemented in C++ using g++ ver. 11.4.0, while data set preprocessing is implemented in Python3, whereas index creation in Rust. Our code is publicly available (https://github.com/noervaag/CSSI). We use a Twitter dataset that consists of geo-tagged tweets, written in English, that were collected using the public Twitter API. The location is given by metadata attached to the tweet, provided by the API. We examine the efficiency and quality of five indexing methods, including the two methods that we propose (CSSI and CSSIA). The first one is a linear scan algorithm (denoted Scan) that calculates the distances between the query and all the objects in the data set, included because in high dimensions linear scan often outperforms index-based algorithms. The second one is an index-based algorithm (denoted R-tree) that builds an R-tree on the spatial coordinates of each object, with semantic vectors located in the leaves, practically a spatial-only index. It performs a best-first k-NN search, where mindist is calculated based on the assumption that in worst case semantic distance is zero, as there can always be a semantic vector with semantic distance equal to zero located in some non-visited leaf node. Moreover, we compare against the state-of-the-art algorithm S2R-tree (denoted S2R).

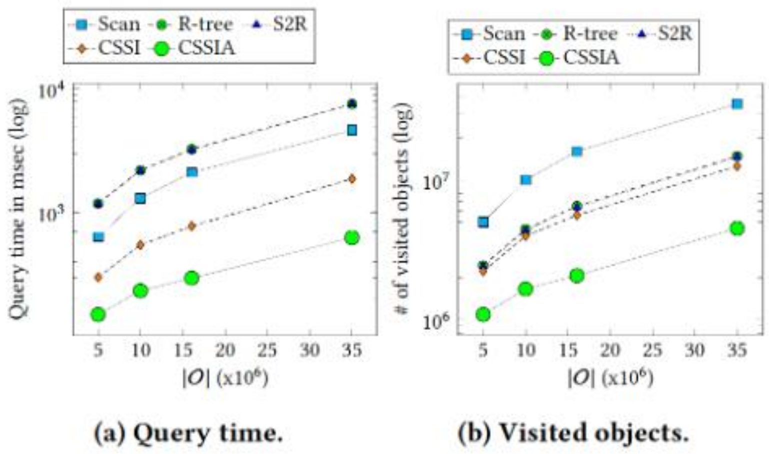


**Figure 3-15 Scalability with dataset size.**

Figure 3-15 shows the results for increasing number of objects in the dataset from 5M to 35M objects. The query time is depicted in log scale on the left chart, whereas the number of visited (accessed) objects by each algorithm is shown on the right chart. Overall, CSSIA is shown to be significantly faster than CSSI, typically x2–x3 times faster, and this gain increases for larger data sets (notice the log scale). CSSIA owes its superior performance to the better pruning, as it needs to access the fewest data objects. Scan accesses all data objects, followed by the index-based algorithms: R-tree and S2R, which also access a large number of objects. However, R-tree is slower than Scan, because of the overhead for traversing the index structure and less efficient memory access (the tree-based index structures do not benefit from efficient prefetching from memory to the extent that linear scan can do). Interestingly, S2R also performs as bad as R-tree, as it accesses almost the same number of objects. Despite its enhancement of index nodes with semantic bounding boxes, S2R cannot prune large portions of the index. The reason is that it follows a spatial-first approach, organizing index nodes based on spatial coordinates and then adds to them semantic bounding boxes which may cover the entire space, as there may exist no correlation of spatial location with semantic vectors. The difference between S2R and R-tree decreases with increasing cardinality, and for our significantly larger data sets, the difference in performance is only marginal. For more experiments, we point to our publication in EDBT'24.

More precisely, Table 11 summarizes the KPIs relevant to this emerald:

**Table 11 - KPIs for Spatio-Temporal Querying**

| Description | Baseline Value | Target Value | Method of Measurement | Achieved Value |
|---|---|---|---|---|
| **Performance (Time)** | SotA (Apache Sedona) | Improved performance | Query execution time | Up to 50% speed up |
| **Performance (Query efficiency)** | SotA ($S^2$Rtree) | Reduced query time | Query execution time | 10x improvement in extreme-scale scenarios |

### 3.3.4 *Next Steps*

In the next period of the project, we intend to evaluate how our algorithms can be applied to external datasets (gathered from the Web or from social networks) that are aligned with the input datasets of UC #1. However, it should be noted that this is subject to the availability of such datasets, as the task of data acquisition of a big, geotagged dataset is not straightforward. Our intention is to demonstrate how our algorithms may be used to retrieve relevant data in an efficient and scalable way, thus supporting upstream AI/ML applications in the project by providing access to useful training data.

Moreover, we are designing and evaluating a generic algorithm for spatial joins that can be applied for massive data. The algorithm is being implemented in Apache Spark to support scalable processing of large datasets. Our intention is to compare its performance against the state-of-the-art algorithm to demonstrate its performance gains.

# 4 Mobility Data Fusion and Management

This task includes the development of theoretical foundations and a framework for the transportation data fusion. Our efforts are concentrated on enhancing data collected from various sources, such as sensors, cameras, and GPS devices. A vital initial step is the preprocessing of raw data received from collectors, encompassing essential tasks like data compression and missing values estimation. The significance of this estimation task is underscored by the impact of the sparsity rate in transportation data on subsequent data analysis. The evaluation of this task is centred around the Rotterdam use-case: Multi-modal Integrated Traffic Management, which aligns with the data fusion EMERALD. Additionally, other data sources, including Brussels Mobility Data, are taken into consideration.
Exploring the transportation data reveals a significant challenge which is managing the extensive volume of data. This challenge is further complicated by the necessity to effectively reduce data size while preserving essential information. In response, this EMERALD addresses this challenge through two approaches: Firstly, we focus on applying summarization methods to trajectory data, recognizing its unique significance in the realm of transportation data. Secondly, efficient data compression methods are employed during storage within the database. This EMERALD aligns with all three use-cases of the project.
By accomplishing these subtasks, we aim to contribute toward the establishment of a robust data processing and management framework. This framework is designed to facilitate the effective analysis of transportation data. In the following sections, we will delve into the two emeralds considered in this task, named:

1. Sensor Data Fusion
2. Mobility/Trajectory Data Compression

## 4.1 Sensor Data Fusion

Data fusion in traffic management involves integrating information from diverse sources such as loop detectors, video sensors and GPS devices. However, these devices often encounter disruptions stemming from various factors, including hardware or software malfunctions, network communication issues, power supply fluctuations, maintenance activities, and adverse weather conditions. Consequently, some devices may intermittently fail to upload traffic data, resulting in what is commonly referred to as the "**missing value problem**."
The presence of missing data not only deteriorates the quality of data provided from intelligent transportation systems but also threatens the intelligent decision-making ability of participants such that they could make false decisions owing to incomplete information. Therefore, addressing the missing value problem requires incorporating data imputation as a crucial component of the data fusion process.
Most missing data imputation methods rely on neighbouring data, either spatially or temporally, to estimate the missing values. These methods are typically effective when dealing with random missing patterns, where missing values are distributed randomly in time and space. However, real-world disruptions to data collection devices often result from events such as power cuts, data losses, and adverse weather conditions. Consequently, more groups of missing values emerge in both spatial and temporal dimensions.
In this case, conventional imputation methods are restricted in such scenarios. Therefore, there is an urgent need to develop novel and more powerful techniques capable of handling these complex missing patterns effectively.

### 4.1.1 Brief Survey of the State-of-the-Art

The traffic flow estimation represents a significant challenge rooted in the spatial and temporal nature of traffic data. Observations gathered from neighbouring sensors and also time stamps, reveal intricate spatio-temporal correlations within the data. Therefore, the key to solve such problems is to effectively extract the nonlinear and complex spatial-temporal correlations of data.

Great efforts have been devoted to tackle this issue[31,32]. Some studies apply traditional machine learning methods, such as auto regressive and moving average model (ARMA)[33] and support vector regression (SVR)[34] mostly considering temporal data. However, most of these methods are linear and not suitable for handling volatile traffic data, the accuracy of results is often low.

In recent years, deep learning methods have gained attention for their potential in estimation tasks, including deep Recurrent Neural Networks (RNNs)[35,36] and Convolutional Neural Networks (CNNs)[37]. Yet, these methods are not suitable to apply to the data points with irregular graph relationship. Given the natural of graph structures in transportation networks, Graph Convolutional Neural Networks (GCNNs) emerge as an appealing solution which have been very efficient in extracting traffic patterns[38,39,40,41].

GCNNs extend from CNNs with consideration of graph structures. They heavily depend on the adjacency or Laplacian matrix of graph, representing spatial dependencies between road segments. GCNNs can be categorized into two classes according to the convolutional operator. The first one, named spatial-based approach, directly apply convolution filters on a graph's nodes and their neighbours. The core of these methods is how to select the neighbourhood of nodes. The other class, named spectral-based approach, considers the locality of graph convolution through spectral analysis.

Motivated by the studies mentioned above, plus considering the graph structure of the transportation network and the spatio-temporal patterns inherent in traffic data, we explore a range of GCNNs to address the missing value problem.

### 4.1.2 Overview and Description

GCNNs offer a promising approach for handling the spatio-temporal complexities inherent in traffic datasets. GCNNs utilize convolutional operations, akin to traditional CNNs, but adapted to process graph-structured data. A convolutional operation applies a set of learnable filters - called kernels- to the input data, enabling the network to extract relevant features. This input data represents spatial or temporal aspects of the traffic data.

Traditional convolutional filters are designed based on the assumption that nearby elements in the input data share local features (Figure 4-1 left). Consider an image as a 2D input or a time series as a 1D data where nearby elements are related to each other. However, the spatial input matrix may not always adhere to this assumption, where two adjacent rows in the matrix may represent two geometrically distant road segment. This renders classical 2D convolutional filters ineffective and calls instead for new filters that consider the topology of the road network utilizing matrices representing spatial relationships, such as the adjacency matrix (Figure 4-1 right).

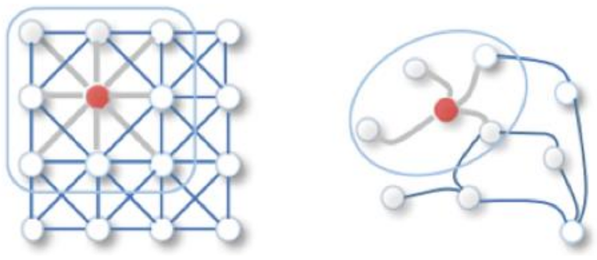


**Figure 4-1 - Traditional vs Graph convolution**

The convolutional filters analyse the connectivity patterns encoded in these matrices, allowing the network to capture spatial features between different road segments. Hence, information from input road segments propagates to adjacent, correlated edges via the road network topology. The GCNNs categorization as spatial-based or spectral- based depends on how kernels apply to the adjacency matrix. Spectral graph convolutions, defined in the spectral domain based on graph Fourier transform. Conversely, spatial-based graph convolutions aggregate node representations from the node's neighbourhood in the spatial domain.

Moving to the temporal aspect, traditional kernels are well-suited for capturing temporal dependencies in this type of data since time series representations inherently exhibit high dependence between recorded data in neighbouring time slots. Through this process, the network learns traffic evolution over time, identifying patterns and trends in the input data. By iteratively applying spatial and temporal layers described earlier, GCNNs can model the dynamics of traffic, ultimately providing an estimated results of missing values in the input.

Bringing together insights from the state-of-the-art, we have crafted a flexible GCNN framework as depicted in Figure 4-2. Within this framework, different settings for temporal and spatial layers in spatio-temporal blocks are available, based on user preferences. These settings encompass options such as spatial-based methods and spectral-based approaches for the spatial layer, and traditional convolution, dilated convolution, and attention-based convolution for the temporal layer.

The network requires two matrices as inputs:

- The symmetric adjacency matrix **A**, with a size of **n**×**n**, where n represents the number of road segments. Each pair of segments are neighbours if it is feasible to traverse from one to the other one by passing through exactly a single intersection.
- The historical matrix **S**, with a size of **n**×**t**, where t represents the number of time slots. Each row of this matrix records the historical information of a segment over time, potentially containing missing values. To enhance the exploration of temporal dependencies, three timeseries extract from matrix **S** in relation to the current time: hourly $\mathbf{S_h}$, daily $\mathbf{S_d}$, and weekly dependencies $\mathbf{S_w}$.

The output of the network is a matrix **S**' with the missing cells filled in by the estimated values.

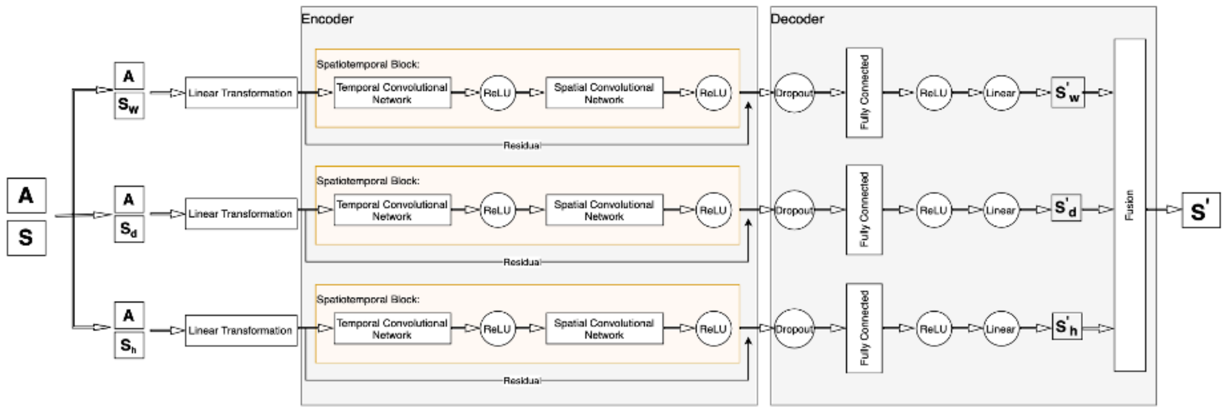


**Figure 4-2 - Architecture of the Baseline GCNN**

### *4.1.3 Preliminary Evaluation*

In this section, we verify the performance of the GCNNs on two datasets, PeMSD8 from California and Brussels mobility twin.

PeMSD4[xiii] is an open-source dataset encompassing traffic data aggregated into 5-minute intervals, covering the San Francisco Bay Area (Figure 4-3). The dataset provides geographic information for sensor stations and includes three key traffic measurements: total flow, average occupancy, and average speed. For our experiments, we focus on average speed as the target feature. Spanning from January to February 2018, the version of the dataset we selected includes 307 detectors, with each detector representing a segment.

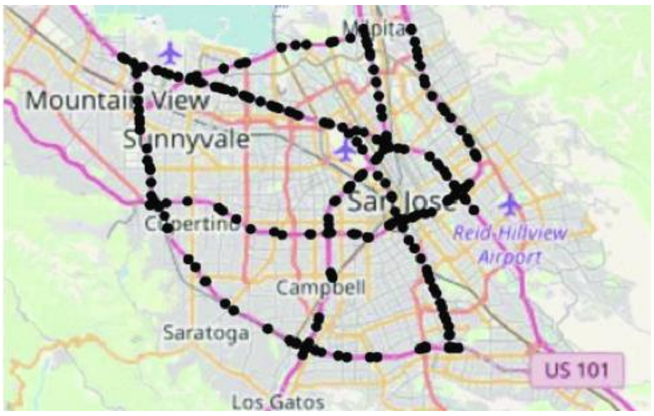


**Figure 4-3 - Sensor map of PeMS dataset**

**Brussels bus-line network (BBN)**: For the second dataset, we leverage the bus transportation data provided by STIB-MIVB, the urban transit service in Brussels[xiv]. STIB-MIVB, a publicly owned corporation, offers real-time operational data. We access this data via the Brussels Mobility Twin platform, which serves as a centralized hub for mobility-related information in Brussels. This platform collects, archives, enriches, and republishes open mobility data from various sources, including STIB. Specifically, we utilize bus trajectory data from STIB within this platform. Figure 4-4 illustrates the STIB bus network. By considering vehicle positions within given time intervals, we access traffic information such as average flow and speed per road segment. In line with the approach for the previous dataset, the average speed data selects as the target feature. This dataset covers 1146 road segments and spans the morning peak-hour period from September to November 2023, with data aggregated at 20-minute intervals.

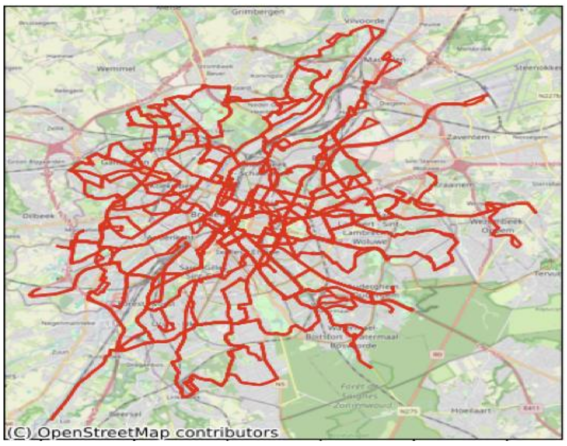


**Figure 4-4 - The Brussels Bus Network operated by STIB**

[xiii] https://github.com/Davidham3/ASTGCN-2019-mxnet/tree/master/data/PEMS04
[xiv] https://mobilitytwin.brussels

**Experimental Setup**

The missing values of node **v** at time interval **t** initials using all observed values of **v** at **t** from other days in the training set. To extract temporal patterns, the size of preceding time intervals for each dataset is set to cover one hour, 7 days, and 4 weeks before the current time. For training our model, we utilize the *Adam optimizer* with an initial learning rate of 0.001 and batch size of 32.

For each convolution layer, several approaches are considered, each requiring specific parameters:

- Dilated temporal convolution: This method requires a dilation factor **d**, which represents the size of gaps between filter elements. Initially set to 1, the dilation factor exponentially increases by a rate of $\mathbf{d}^2$ as the layers deepen to extract daily and hourly temporal dependencies. For weekly dependencies, **d** remains constant at 1.
- Spectral convolution: In spectral graph analysis, graph's properties are often inferred from its corresponding *Laplacian matrix* and eigenvalues. However, directly performing eigenvalue decomposition on the *Laplacian matrix* can be computationally expensive. To address this, *Chebyshev polynomials* are adopted as an efficient approximation method. Specifically, *Chebyshev polynomial* with 3 terms employ for both datasets.
- Attention mechanism: Two different attention mechanisms for temporal and spatial layers have been designed and applied to both datasets. The goal of these mechanisms is to select information that is relatively critical to the current task from all neighbours.

**Evaluation of algorithm accuracy**

To assess the accuracy of convolutional approaches, the baseline network with different selections of approaches is applied to the datasets. The mean square error (MSE) between the estimator and the ground truth is then used as the loss function and minimized through backpropagation.

Table 12 presents the MSE of applying GCNN with different selection of convolutional method over PEMSD4 and BBN respectively. In the table, the columns represent the GCN methods, while the rows depict the TCN methods used for temporal and spatial convolutional layers, respectively. For temporal convolution in TCN layer, dilated convolution and attention convolution are considered. For spatial consolation in the GCN layer, spatial-based and spectral-based approaches, both with and without attention mechanisms, have been examined.

| | | Spectral GCN | Spectral Attention GCN | Spatial GCN | Spatial Attention GCN |
|---|---|---|---|---|---|
| **Brussels Bus-line** | **Dilated TCN** | 33.03 | 30.82 | 28.18 | 26.14 |
| | **Attention TCN** | 31.73 | 27.32 | 26.56 | 23.47 |
| **PeMSD4** | **Dilated TCN** | 29.14 | 24.30 | 24.04 | 21.99 |
| | **Attention TCN** | 25.29 | 23.21 | 22.89 | 20.11 |

**Table 12 – Performance (MSE) of GCNN with various temporal and spatial configurations**

The table illustrates that spatial-based methods outperform spectral-based convolution in capturing spatial features. Notably, incorporating attention mechanisms enhances the performance of both spatial-based and spectral-based methods, indicating the effectiveness of multi-level attention mechanisms in capturing dynamic changes in traffic data.

Similarly, attention mechanisms enhance performance within the temporal layer. While attention mechanisms positively impact convolutional layers, their effect is more pronounced in spatial layers. This is evident when comparing the error rates between scenarios where attention is applied just to the GCN layer and where it is applied only to the temporal layers. This may stem from the baseline design, which inherently incorporates three temporal attentions on hourly, daily, and weekly bases.
Overall, the trends observed in both datasets align closely. However, the PeMSD4 dataset yields superior results compared to the Brussels Bus-line dataset, likely due to differences in networks' complexity. As the traffic network's scale expands, incorporating more time series into the model, results in higher error rates.
Figure 4-5 depicts the outcomes of applying GCNN with "Spatial Attention GCN" and "Attention TCN" layers to the Brussels Bus-line network in a snapshot. To facilitate readability, only 1% of road segments are labelled with average speed values in time **t**. The observed data is represented by green labels, while the estimated values are denoted by blue.

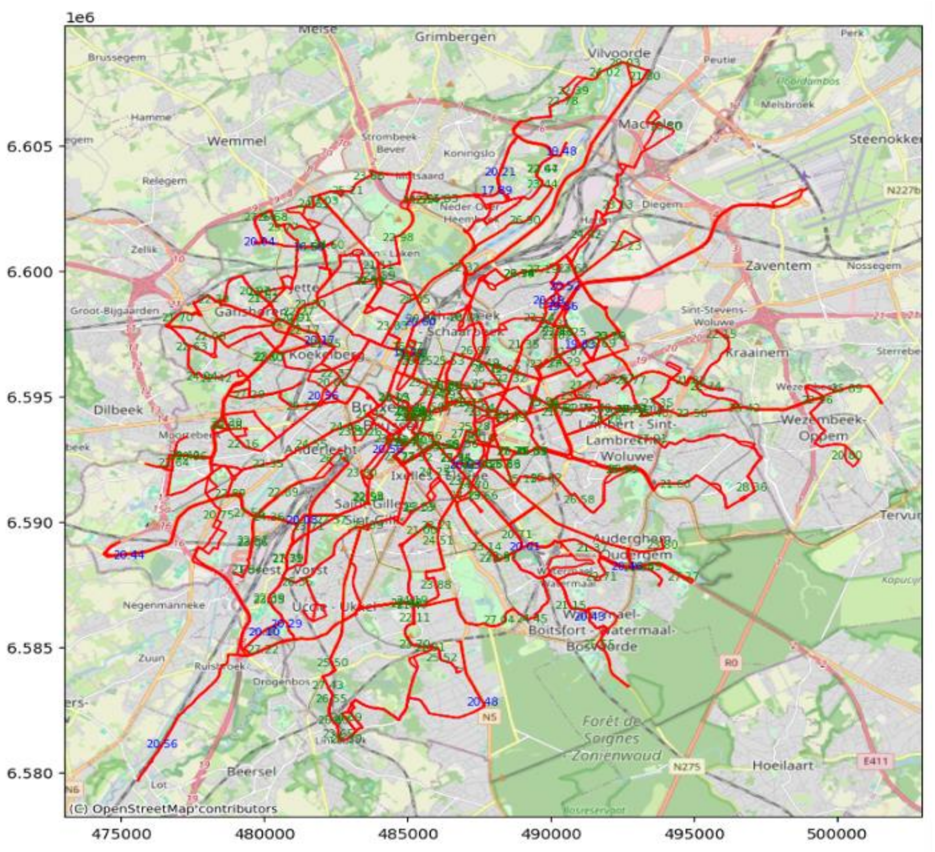


**Figure 4-5 - GCNN results on Brussels Bus-line network. The observed data is indicated by green labels, while the estimated values are depicted with blue.**

More precisely, Table 13 summarizes the KPIs relevant to this emerald:

**Table 13 - KPIs for Sensor Data Fusion**

| Description | Baseline Value | Target Value | Method of Measurement | Achieved Value |
|---|---|---|---|---|
| **Estimation accuracy** | SotA[38,39,40] | Reducing estimation error | Accuracy metrics (e.g. MSE) | BRU 28.40 PeMS 23.87 |

### 4.1.4 Next Steps

In the next step, we will prioritize exploring datasets from UC #2 Multi-modal integrated traffic management situated in Rotterdam, as it aligns with the overarching goals of the EMERALDS project. By leveraging the UC #2 dataset, we aim to deepen our understanding of transportation dynamics and develop robust methodologies capable of handling varying rates of missing data. We seek to ensure that our models are adaptable across a spectrum of scenarios, encompassing missing patterns such as random or clustered distributions in both time and space.

Moving forward, our exploration will extend to encompass a diverse range of deep learning and statistical models tailored to analyse traffic data. Furthermore, we plan to explore the integration of external factors, such as weather conditions, into transportation network models to enhance its real-world applicability.

## 4.2 Mobility/Trajectory Data Compression

### 4.2.1 Brief Survey of the State-of-the-Art

Democratization of mobile devices, such as smartphones and wearable technologies, and the spread of Global Positioning System (GPS) equipped vehicles or Automatic Identification System (AIS) equipped vessels are some examples of reasons for this data explosion[42]. While the spatio-temporal data offers many exploitation opportunities (both commercial and research), its increase also causes some new challenges. One of these challenges are to process this large amount of data. Douglas and Peucker have shown that 100Mb would be necessary to store the localization of a set of 400 moving objects, with a frequency of 10 Hz (typical frequency of GPS devices)[43]. Bruxelles Mobilite[xv] the public administration overseeing mobility-related infrastructure in the Brussels Capital Region, collects positional data specifically for heavy-goods vehicles in Brussels. This information is primarily utilized to calculate toll charges, represents, on average, 19 Gigabytes of data accumulated daily.

To overcome this difficulty, different compression or simplification algorithms have been proposed, with one of the most well-known simplification algorithms being the Douglas Peucker (DP) algorithm, an algorithm that was initially aimed at line simplification (without temporal feature). Later, Meratnia and de By[44] introduced some variations of the DP algorithm (including the Top Down Time Ratio algorithm (TD-TR)), taking into account the temporal feature of the locations. Since then, multiple algorithms such as Squish (and its variations)[45], STTrace[46] or Dead Reckoning (DR)[47] have been proposed. The main contribution of this emerald is to extend the Squish, STTrace and DR algorithms so that they could be used in contexts where bandwidth limitations apply.

[xv] http://www.bruxellesmobilite.irisnet.be/

### 4.2.2 Overview and Description

**Compression under bandwidth constraints motivation**

Existing techniques for simplification of trajectories have already largely been studied. These techniques are generally aimed at simplifying the trajectories to facilitate their exploitation by machine learning techniques. This is usually performed by trying to minimize the number of points (position of an object at a given timestamp) kept without deteriorating the trajectory significantly. In this emerald, a different approach is used. Instead of trying to minimize the number of points kept, the algorithms introduced in this work will consider some bandwidth constraints. These constraints are defined as follows.

For each period, bandwidth constrained algorithm must respect a predefined limit on the quantity of points that can be kept. Therefore, algorithms presented in this work are aimed at minimizing the deterioration of the trajectories during compression without exceeding this limit on the quantity of points kept, and this, for all time periods. The size of these periods as well as the number of points that can be kept are parameters of the compression algorithms. While bandwidth limitations are mentioned for different contexts (vessels tracking, animal tracking), the problem of simplifying trajectories under bandwidth limitation has, not yet attracted the attention of the research community. Existing algorithms (such as the already mentioned Squish and STTrace) provide some solutions to compress trajectories under memory limitations (the final number of points is a parameter of the methods) but these are not adapted for bandwidth constrained contexts.

**Bandwidth constrained variants of existing trajectory compression algorithms**

This task consists in the extensions of Squish, STTrace and DR under bandwidth constrains. These algorithms are based on the implementation of a sliding windows as well as other modifications of the existing algorithms.

The existing algorithms will be adapted to be iteratively applied on sequential time windows. During each time window, a defined maximal number of points of the uncompressed trajectories will be kept in the compressed trajectories. The bandwidth limitations will therefore be enforced by selecting appropriate values for the size of the time windows as well as the number of points kept in each of them. Indeed, after the processing of each individual time window, there is the guarantee that the points belonging to this time window which have not been simplified yet cannot be further simplified. These points can therefore be transferred from the edge device directly after the end of the concerned time window. It should therefore be noted that a delay of the size of the time window is introduced between the actual time associated to a position generated by a vessel and the transmission of this position to the coastal station. Therefore, in situations where time delays are a sensitive issue, it is recommended to use shorter time windows.

Four extensions of Squish, STTrace and DR under bandwidth constrains have been developed, respectively BWC-Squish, BWC-STTrace, BWC-STTrace-Imp and BWC-DR. These extensions are also referred to as BandWidth Constrained (BWC) methods. All these variations work by assigning a priority to each processed point in each time window. While the number of processed points do not exceed the limit, every point is kept in a temporary priority queue (sorted according to the points priorities). Once the limit is reached, for each additional point processed, the priority of this point will be computed. If this priority is higher than the lowest priority of a point in the priority queue of temporarily kept points, then the point with the lowest priority will be dropped from the queue while the currently processed point will be added to the queue.

The main difference between BWC-Squish, BWC-STTrace, BWC-STTrace-Imp and BWC-DRlies in the method employed to compute the priorities of the points which is inspired from the original Squish, STTrace and DR methods

### 4.2.3 Preliminary Evaluation

This section analyses the performances of the BWC algorithms as well as their classical equivalents and the classical TD-TR algorithm.
The comparison is performed on two datasets of different spatial and temporal ranges. These datasets mark the initial phase of our methodological testing. As we advance, our forthcoming objective is to extend these evaluations to urban mobility datasets.

**AIS:** The main dataset motivating compression of trajectories under bandwidth constraints concerns the extension of AIS signal coverage for maritime monitoring. Since 2004, all cargo vessels over 500 GT and all passenger vessels are required to be equipped with AIS transceivers. These transceivers allow automatic exchange of information in between ships and coastal stations by broadcasting positional messages using the Self-Organizing Time Division Multiple Access (SOTDMA) protocol. Initiating the development with unconstrained data, we gain a broader understanding of movement patterns and behaviors, which can then be refined and specified for constrained movements, as those in urban streets. This approach enables us to capture the complexities of both unconstrained and network-constrained movements, allowing for a more focused and specific analysis when dealing with urban mobility challenges.
Our first dataset in this task consists of 24h of AIS data in the region between the cities of Copenhagen and Malmo on first January 202. It is composed of 103 trips totalling 96819 points. The trips can be seen in Figure 4-6.

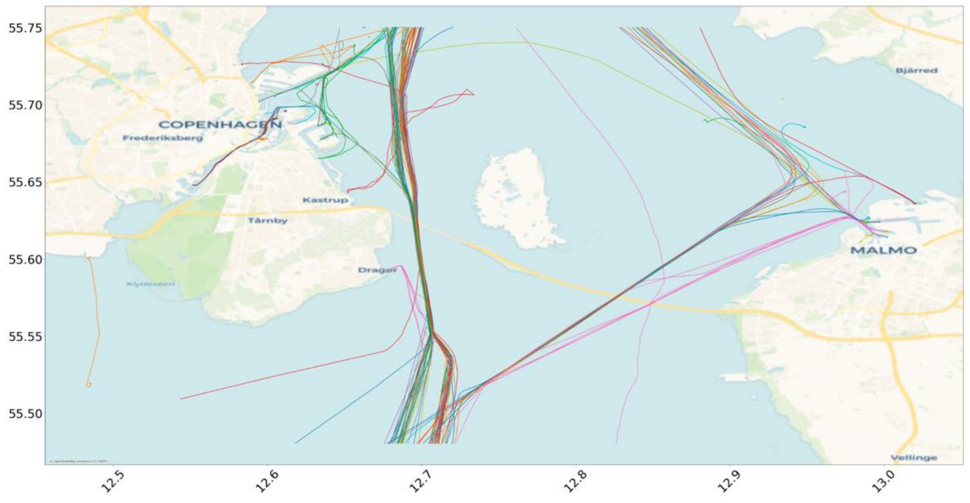


**Figure 4-6 - AIS trips around Copenhagen and Malmo**

**Birds**: The second dataset consists of three months of GPS of black-backed gulls between the 9th of July and the 9th of October 202. It is composed of 45 trips totalling 165244 points. While most of these trips originate from Belgium and North of France, some are spreading as far as the north of Spain. Few other trips are also entirely taking place in Spain and one in Algeria. These trips can be seen in Figure 4-7.

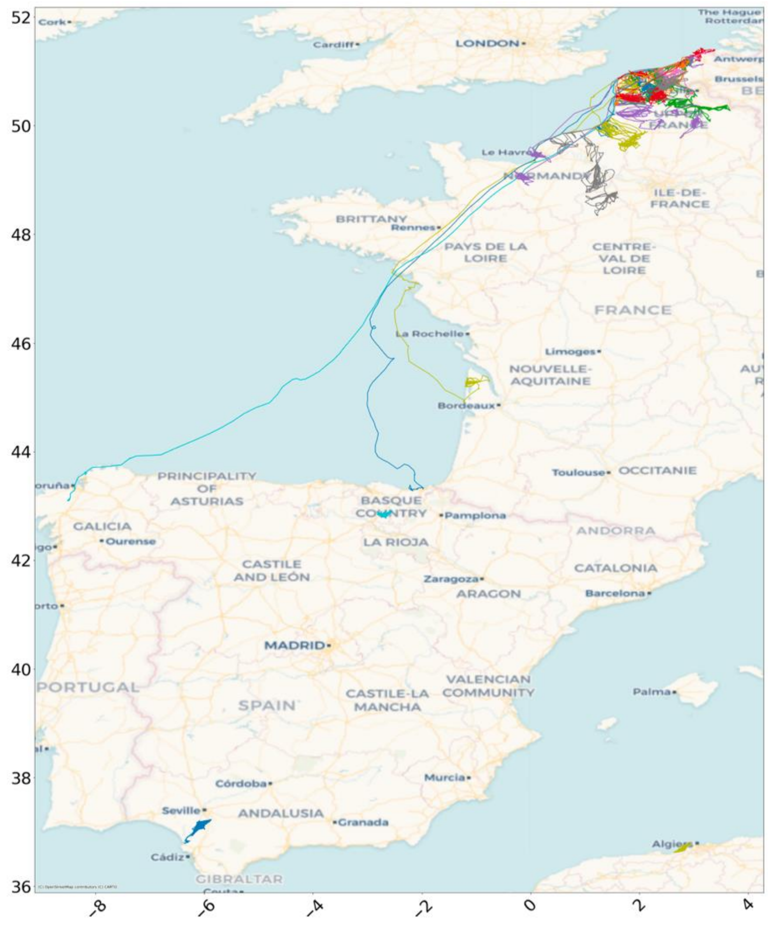


**Figure 4-7 - Birds trips**

**Evaluation of algorithm accuracy**

To assess the accuracy of the algorithms, each algorithm is applied to simplify all initial trajectories in both datasets. Once these simplifications are computed, the distance between synchronized projection on the initial and simplified trajectories are computed at a regular time interval.

It is important to note that this accuracy evaluation is not aimed at stating that some compression algorithms are better than others. Indeed, when selecting a compression algorithm, different factors must be considered. While accuracy is generally an important factor, other factors such as time and space complexity should be considered. Furthermore, the BWC algorithms are designed to be able to be used in situations with additional bandwidth constraints. It is not surprising that the fulfilment of these additional constraints may lead to a deterioration of the algorithm's accuracy.

While the different algorithms require different parameters, these were determined to produce a similar total number of points in the simplified trajectories produced by the different algorithms. For each dataset, the algorithms will be assessed with parameters such that both around 10% and around 30% of the original points are kept in the simplified trajectories. The exact value of the parameters for the classical algorithms are listed hereunder.

- **Squish** Squish requires the maximal number or points kept for each individual trajectory. This maximal number of points has been set to 10% and 30% of the initial points of each trajectory.
- **STTrace** STTrace requires the maximal number or points kept for all trajectories. This maximal number of points has been set to 10% and 30% of all initial points.
- **DR** DR requires a distance threshold. This threshold has been set to 425 and 115 meters for the AIS dataset and has been set to 2500 and 950 meters for the Birds dataset.

- **TD-TR** The TD-TR time algorithm requires a tolerance threshold. This threshold has been set to 0.15 and 0.051 in the AIS dataset as well as 16.7 and 1.5 for the Birds dataset.

For each of the classical algorithms, its accuracy (average distance in meters between the simplified and original trajectories) can be seen in Table 14. As can be seen, TD-TR is outperforming the other algorithms. This is because Squish, STTrace and DR are designed to be less computationally expensive.

| | **AIS** | | **Birds** | |
|---|---|---|---|---|
| | **10%** | **30%** | **10%** | **30%** |
| **SQUISH** | 20.87 | 4.83 | 585.34 | 44.95 |
| **STTrace** | 58.66 | 9.78 | 1823.10 | 431.65 |
| **DR** | 42.68 | 13.12 | 697.14 | 46.48 |
| **TD-TR** | 2.95 | 1.08 | 274.78 | 26.87 |

**Table 14 - Accuracy of the classical algorithms on the different datasets**

The performances of the BWC algorithms on the AIS dataset can be found in Table 15 and Table 16.

| Window size (min) | 120 | 60 | 15 | 5 | 0.5 |
|---|---|---|---|---|---|
| Points per window | 800 | 400 | 100 | 33 | 4 |
| BWC-SQUISH | 10.97 | 10.65 | 7.35 | 7.90 | 130.59 |
| BWC-STTrace | 17.23 | 12.49 | 6.25 | 5.09 | 81.54 |
| BWC-STTrace-IMP | 1.49 | 1.53 | 1.72 | 4.62 | 108.39 |
| BWC-TR | 13.77 | 15.82 | 14.91 | 13.07 | 11.16 |

**Table 15 - Accuracy of the different BWC algorithms when of the AIS dataset for different sizes of time windows**

| Window size (min) | 120 | 60 | 15 | 5 | 0.5 |
|---|---|---|---|---|---|
| Points per window | 2400 | 1200 | 300 | 100 | 12 |
| BWC-SQUISH | 1.82 | 1.67 | 1.51 | 1.32 | 21.57 |
| BWC-STTrace | 8.87 | 4.42 | 2.12 | 2.34 | 7.13 |
| BWC-STTrace-IMP | 0.55 | 0.55 | 0.56 | 0.57 | 24.55 |
| BWC-TR | 19.60 | 19.48 | 12.15 | 10.36 | 9.60 |

**Table 16 - Accuracy of the different BWC algorithms when of the AIS dataset for different sizes of time windows**

Furthermore, we can notice from these tables that for large enough windows (between 15 and 120 minutes), BWC-STTrace-Imp is outperforming the other BWC and classical algorithms. This is since the priority of the points is evaluated using the sample and the original trajectory. It can also be noticed

that for small time windows, the performances of BWC-Squish, BWC-STTrace and BWC-STTrace-Imp deteriorate. The deterioration is even drastic for 30 seconds time windows when keeping 10% of the points. This is since these three algorithms compute the priority of a point according to both the previous and the next point in the sample. Therefore, for small time windows, there will generally be less than 2 points per trajectory in the sample, making the removal of a point arbitrary and therefore leading to inaccurate simplifications. On the other hand, the performance of BWC-DR is more constant and even improves for smaller time windows. This is because BWC-DR only makes use of the previous one (or two) points to compute the priority of the currently processed point. Therefore, even with small time windows, it will be able to compute the priorities correctly using points kept during the previous time windows.
As expected, it can also be noted that the average error of the improved version of BWC-STTrace-Imp is indeed smaller than the one of BWC-STTrace. Surprisingly however, even BWC-STTrace outperforms the classical STTrace algorithm. One hypothesis is that this is due to STTrace both assessing the priority of points using current simplified trajectory only and simultaneously comparing different trajectories of different natures. Therefore, trajectories with different sampling frequencies could be compressed simultaneously. Trajectories with lower frequencies might fill up the priority queue as the priority of a point which is far apart in time from its neighbours in the sample will intuitively be higher than the one of a point close to its neighbours. Restarting with an empty priority queue at frequent time interval might help mitigate this phenomenon. Squish on the other hand, does not seem to suffer from this drawback. This might be due to their heuristic which counterbalances this effect by adding the priorities of points deleted from the sample.
The performances of the BWC algorithms on the Birds dataset can be found in Table 17 and Table 18.

| Window size (days) | 31 | 7 | 1 | 1/4 | 1/24 |
|---|---|---|---|---|---|
| Points per window | 5580 | 1260 | 180 | 45 | 8 |
| BWC-SQUISH | 777 | 939 | 884 | 1061 | 3615 |
| BWC-STTrace | 2780 | 2651 | 1144 | 1277 | 3096 |
| BWC-STTrace-IMP | 273 | 382 | 497 | 749 | 3437 |
| BWC-TR | 1997 | 1752 | 1677 | 1421 | 1314 |

**Table 17 - Accuracy of the different BWC algorithms when simplifying until 10% of the Birds dataset for different sizes of time windows**

| Window size (days) | 31 | 7 | 1 | 1/4 | 1/24 |
|---|---|---|---|---|---|
| Points per window | 16740 | 3780 | 540 | 135 | 22 |
| BWC-SQUISH | 77 | 104 | 108 | 126 | 4882 |
| BWC-STTrace | 1245 | 707 | 245 | 247 | 6828 |
| BWC-STTrace-IMP | 32 | 50 | 60 | 77 | 4706 |
| BWC-TR | 570 | 605 | 623 | 465 | 554 |

**Table 18 - Accuracy of the different BWC algorithms when simplifying until 30% of the Birds dataset for different sizes of time windows.**

Similar observations can be seen for the Birds dataset as for the AIS dataset. Surprisingly, increasing the bandwidth from 8 to 22 points for the 1-hour time window led to worse results for BWC-Squish, BWC-STTrace and BWC-STTrace-Imp. This confirms the arbitrary simplification performed by these algorithms if there are not enough points for each trip in each time window.

**Points distribution**

In this section, the time repartition of points conserved with classical compression algorithms is illustrated. This is done by compressing the AIS dataset to 10% of its original size and by analysing the time repartition of the points kept for each period of 15 minutes. It is shown that these classical algorithms do not produce a homogeneous time-partitioned results. In this configuration, 100 points should be kept in each period of 15 minutes to satisfy the bandwidth constrain. The time repartition of simplified points for the TD-TRis illustrated in Figure 4-8. Similar figures can be obtained for Squish, STTrace and DRbut will not be displayed here. The figure consists of a histogram representing the number of points remaining in all simplified trajectories during each period.

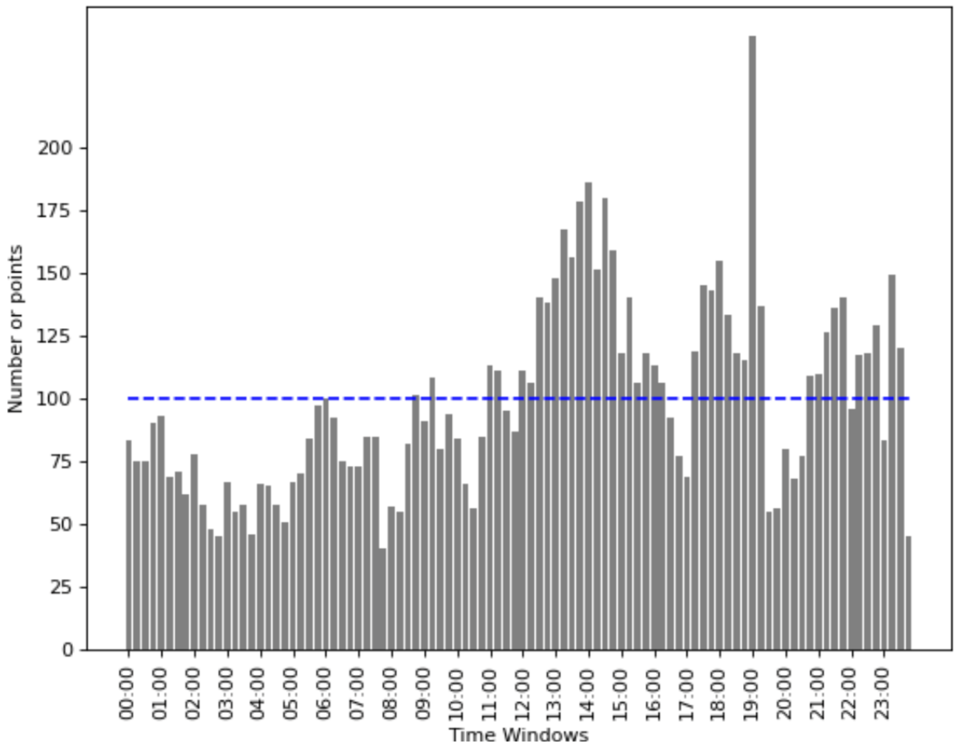


**Figure 4-8 - Histogram of the quantity of points in different time-windows in samples obtained with TD-TR**

The limit of 100 points is indicated with the blue dotted line. Figure 4-8 confirms the need of using different compression techniques in contexts with bandwidth constrains.

More precisely, Table 19 summarizes the KPIs relevant to this emerald. It's important to note that in this table, we have focused on the results obtained with the smallest window size and the highest compression ratio. The evaluation of the data quality is based on the error between the original and compressed trajectories, determined by the average distance in meters between the simplified and original trajectories.

**Table 19 - KPIs for Mobility/Trajectory Data Compression**

| Description | Baseline Value | Target Value | Method of Measurement | Achieved Value |
|---|---|---|---|---|
| **Data quality** | SotA[48, 49] | Reducing the error of the compressed trajectory, tend towards zero | Distance between the original and the simplified trajectories | ~1300 for Birds and ~11 for AIS. |
| **Data size** | SotA[50, 51] | 90% Compression for urban data | Compression rate based on window points/window time. | 90% compression for non-urban data. |

### 4.2.4 Next Steps

Our forthcoming step involves the application of the aforesaid compression algorithms to urban datasets. Following this application, several further improvements could still be considered to the algorithms. First, different algorithms might also be considered for such an extension. Furthermore, the presented algorithms could be further optimized. For instance, transition between time windows for the BWC-Squish as well as BWC-STTrace and BWC-STTrace-Imp could be improved. The DR algorithm could also be modified in a different manner to satisfy bandwidth constraints instead of using a time-windowed approach with a priority queue. For instance, the distance threshold could be modified in real time by the algorithm according to the number of points in the sample at a given time. Moreover, this emerald will be integrated with MobilityDB, a geospatial trajectory database system developed by ULB. Furthermore, we plan to extend this task by implementing data compression techniques during data storing process in database. This will involve several steps such as schematization and optimization of data types to streamline the storage process.

# 5 Conclusions and next steps

In this deliverable, we presented the current status of the developed mobility data processing tools for the CC, known as "emeralds" accompanied by their source/object codes collected under the EMERALDS github organization repository. In summary, the following emeralds have been presented:

| | emeralds presented in D3.1 | | Project Target |
|---|---|---|---|
| Task | # of emeralds | emerald name | # of emeralds |
| **Privacy-aware In situ Data Harvesting** | 2 | • Privacy-aware data Ingestion<br>• Extreme-Scale stream processing orchestrator | 2 |
| **Extreme-scale Cloud/Fog Data Processing** | 3 | • Extreme-Scale Map Matching<br>• Weather Enrichment<br>• Spatio-Temporal Querying | 4 |
| **Mobility Data Fusion and Management** | 2 | • Sensor Data Fusion and Mobility/trajectory Data Management<br>• Mobility/Trajectory Data Compression | 2 |

**Table 20 - Summary of emeralds presented in D3.1**

In line with the expected results and means of verification specified in the GA, the next steps are as follows:

In the **1st integration cycle**, we will:

- Implement a containerization strategy using technologies like Docker and Kubernetes. Each service or component of the EMERALDS toolset will be encapsulated into containers, ensuring portability, scalability, and efficient deployment across different environments.
- Set up communication channels, defining APIs, and orchestrating workflows for data processing and analysis, capitalizing on the advanced functionalities of D3.1 components such as the extreme-scale data stream processing orchestrator (Section 2.2).

In the **1st assessment cycle**, we will:

- Assess the KPIs of the developed emeralds along the compute (edge/fog/cloud) continuum, focusing on performance and quality through the WP5 Use Cases. Validate the attained improvement in UC data workflows and verify the technological maturity of the services.

In the **2nd implementation cycle**, we will:

- Implement the improvements outlined in the previous sections.
- Implement the new emerald (T3.2) that has not been included in the 1st implementation cycle.

In the **2nd integration cycle**, we will:

- Integrate the WP3 emeralds internally (utilizing and demonstrating the orchestrator as much as possible), as well as across WPs (particularly with WP4's analytics and learning emeralds in order to match WP5's business requirements).

In the **2nd assessment cycle**, we will:

- Assess the final KPIs of the developed emeralds along the compute (edge/fog/cloud) continuum, focusing on performance and quality.
- Assess the final status of integration of WP3's processing and WP4's analytics modules into the emeralds ecosystem.

# Annex

## Annex 1: README Template

```
# WPx Emerald: [Task] - [Service Name]

## Description
This repository contains the first version of the service [Service Name], developed as part of [Task] of WPx within EMERALDS project. [Service Name] is a service for [brief description, including info about input/output]. This README provides essential information for deploying, testing, and [+++].

## Table of Contents
[Include a table of contents in order to allow other people to quickly navigate especially long or detailed READMEs.]

## Requirements
requirements.txt

## Sample data input/output structures
[Describe the structure of sample input and output data for the service.]

## Input/Output interfaces & interactions
[Describe the input and output interfaces and interactions; ensure compliance with the EMERALDS Ref.Arch.]

## Deployment
[Along with the source code, include in this repository a Dockerfile containing the requirements to build the docker image and a docker-compose.yaml file defining the required configuration to execute each component in terms of ports, networks, and volumes.]

## Usage - Executing program
[Provide instructions on how to use the project or run the code. Include examples and code snippets if applicable. Include references to sample data if applicable.]

## Authors
[List the authors or contributors involved in the project.]

## +++
Feel free to customize this template further to suit your specific service requirements.
```

# Annex 2: Ethics Checklist and Questionnaire

EMERALDS

**Ethics Checklist and Questionnaire**

**<u>A. PERSONAL DATA</u>**

1. Has ***personal data*** going to be processed for the completion of this deliverable?

   **NO**

   - If "yes", do they refer only to individuals connected to project partners or to third parties as well?

2. Are "***special categories of personal data***" going to be processed for this deliverable? (whereby these include personal data revealing racial or ethnic origin, political opinions, religious or philosophical beliefs, and trade union membership, as well as, genetic data, biometric data, data concerning health or data concerning a natural person's sex life or sexual orientation)

   **NO**

3. Has the ***consent*** of the individuals concerned been acquired prior to the processing of their personal data?

   **N/A**

   - If "yes", is it based on the Project's Informed Consent Form, either on the provided Template or on other attached herein Template?
   - If "no" is it based on a different legal basis?

4. In the event of processing of personal data, is the processing:

   **N/A**

   - obviously "***Fair and lawful***", meaning executed in a fair manner and following consent of the individuals concerned or based on another - acknowledged as adequate and proportionate as per above - legal basis?
   - Performed for a ***specific*** (***project-related***) ***cause*** only?
   - Executed on the basis of the principle of ***proportionality and data minimisation*** (meaning that only data that are ***necessary*** for the processing purposes are being processed and such deductive reasoning is documented)?
   - Based on ***high-quality, updated and precise*** *personal data*?

5. Are there any provisions for a storage limitation period of the personal data-in case of storage- after which they must be erased?

   **N/A**

6. Are all ***other lawful requirements*** for the processing of the data (for example, ***notification of the competent Data Protection Authority(s)*** or undergoing a ***DPIA procedure*** and consulting with the competent DPA, if and where applicable) adhered to and on what legislative basis are such notifications justified as necessary or dismissed as unnecessary?

   **N/A**

7. Have individuals been ***made aware of their rights*** on the processing of the personal data as per the GDPR and the relevant and executive national legislation (particularly the rights to access, rectify and delete the personal data and their right to lodge a complaint with the relevant Competent Authority) and if yes, by what demonstrable means (e.g. the informed consent form as per above or as per other Templates, attached herein?)

   **N/A**

8. Even if anonymized or pseudonymized or aggregated data are referred to, does the dataset contain ***location data*** that could potentially (even via the combined use of other datasets) be ***traced back*** to ***individuals***? If yes, what specific measures are taken to ensure this data (i) is anonymized or pseudonymized and (ii) cannot be used to track individuals without their consent? If no, what is the scientific methodology used to collect and gather said data?

   **N/A**

*9.* In the context of risk assessment, prediction and forecasting, as foreseen in the scope of the EMERALDS project, during traffic, population movement monitoring or weather events, ***is there any risk*** that personal data could be inadvertently revealed in the event of an ***emergency*** or ***unusual event***, because of the dataset usage, either on its own or combined with other openly available datasets, triggering identification or unwanted disclosure of PII? What measures are in place to protect - still identifiable if the dataset allows such extraction - **personal data** in these circumstances?

   **NO**

10. For the use case of Trip Characteristics Inference as per the EMERALDS project scope, are there specific measures to ensure that **inferences made about trip characteristics** cannot be linked back to ***specific individuals*** or reveal ***sensitive information*** about their ***habits*** or ***routines*** i.e. by identifying specific individuals' absence or presence routines whether in the home or in a professional environment or in other premises?

    **N/A**

11. Are there any potentially personal identifiable information (PII) in the datasets, **disclosable by combination with other datasets**, either open data or proprietary (e.g., E-tickets validation data)? If yes, how is PII adequately anonymized or pseudonymized or how other datasets that by combination may result in unwanted or illegal disclosures or identification before any processing takes place?

    **NO**

**B. DATA SECURITY**

1. Have proportionate security measures been undertaken for protection of the data, taking into account project requirements and the nature of the data?
   - If yes, brief description of such measures (including physical-world measures, if any)
   - If yes, is there a data breach notification policy in place within your organization (including an Incident Response Plan to such a breach)?

**N/A**

2. Given the **large-scale nature** of some datasets, are there specific measures in place to protect **included personal data** at scale at the data source or in the possession of data processors?

**N/A**

*3.* Regardless of personal data*,* in the case of Multi-modal integrated traffic management as defined under the EMERALDS scope, are there specific measures in place to ensure **the availability and integrity of data spanning multiple modes of transport** from being disclosed in other manners than the ones intended and covered under an open data scheme?

**N/A**

4. Are there specific measures in place to secure ***sensitive infrastructure data***, if present?

**N/A**

## C. DATA TRANSFERS

1. Are personal data transfers beyond project partners going to take place for this deliverable?

**NO**

- If "yes", do these include transfers to third (non-EU) countries and if what policies apply?

2. Are personal data transfers to public authorities going to take place for this deliverable?

**NO**

3. Do any state authorities have direct or indirect access to personal data processed for this deliverable?

**NO**

3. Taking into account that the Project Coordinator is the "controller" of the processing and that all other project partners involved in this deliverable are "processors" within the same contexts, are there any other personal data processing roles further attributed to any third

parties for this deliverable? And if any, are they conformed to the GDPR provisions?

**NO**

4. Given the geographical diversity of the datasets, are there measures in place to ensure compliance with specific personal data protection regulations **in different jurisdictions** i.e. at the place of the data source establishment as well as at the place of the establishment of a Data processor?

**N/A**

5. Are there additional protocols for data transfers involving **sensitive infrastructure data**, if present?

**NO**

## D. ETHICS AND RELATED ISSUES

1. Are personal data of children going to be processed for this deliverable (ie. “underage” signified e-tickets)?

**NO**

2. Is ***profiling*** of identifiable individuals in any way enabled or facilitated for this deliverable?

**NO**

3. Are ***automated decisions*** for identifiable individuals made or enabled on the basis this deliverable?

**NO**

4. Have partners for this deliverable taken into consideration system architectures of ***privacy by design*** and/or ***privacy by default***, as appropriate?

**YES**

5. Have partners for this deliverable taken into consideration gender equality policies or is there an explicit reasoning that dismisses such risk as unsubstantiated or such need as irrelevant as per the methodology of work and production of the deliverable?

**YES**

6. Have partners for this deliverable taken into consideration means of protecting the confidentiality of the dataset if it is not signified as open data?

**N/A**

7. Are there additional considerations around the collection and processing of ***location data*** and data ***that could potentially be used to infer patterns about individuals' movements***?

**NO**

8. Have partners identified any ***additional ethical issues*** related to the processing of sensitive infrastructure data?

**NO**

9. Are shared economy (ie. “Uber” transfer services or “Lime” Scooters or other solution) or other shared mobility infrastructures used by the data sources? If yes, are there measures in place to ensure that the processing of ***shared mobility data*** respects privacy rights?

**NO**

10. In the context of Traffic Flow Data Analytics, are there specific considerations to ensure that the **analysis of traffic flow data** does not infringe on privacy rights or reveal sensitive information about individuals’ movements or routines?

**N/A**

11. Is the Project taking into account the need for an all people-inclusive policy in the future within its overall goals and not only the ‘’tech-savvy’’ (i.e. elderly people not familiar with some tech devices, poor people) and does it entail possible proposals for that?

**N/A**

## References

[1] M. Sakr, C. Ray, and C. Renso, “Big mobility data analytics: Recent advances and open problems,” *GeoInformatica*, vol. 26, no. 4, pp. 541–549, Oct. 2022, doi: 10.1007/s10707-022-00483-0.
[2] E. Zimányi, M. Sakr, and A. Lesuisse, “MobilityDB,” *ACM Transactions on Database Systems*, vol. 45, no. 4, pp. 1–42, Dec. 2020, doi: 10.1145/3406534.
[3] Z. Fang, L. Chen, Y. Gao, L. Pan, and C. S. Jensen, “Dragoon: a hybrid and efficient big trajectory management system for offline and online analytics,” *The VLDB Journal*, vol. 30, no. 2, pp. 287–310, Feb. 2021, doi: 10.1007/s00778-021-00652-x.
[4] Q. Yu and J. Yuan, “TransBigData: A Python package for transportation spatio-temporal big data processing, analysis and visualization,” *Journal of Open Source Software*, vol. 7, no. 71, p. 4021, Mar. 2022, doi: 10.21105/joss.04021.
[5] S. Ruan, R. Li, J. Bao, T. He, and Y. Zheng, “CloudTP: A Cloud-Based Flexible Trajectory Preprocessing Framework,” in *2018 IEEE 34th International Conference on Data Engineering (ICDE)*, Apr. 2018. http://dx.doi.org/10.1109/icde.2018.00186
[6] S. Haidri, Y. J. Haranwala, V. Bogorny, C. Renso, V. P. da Fonseca, and A. Soares, “PTRAIL — A python package for parallel trajectory data preprocessing,” *SoftwareX*, vol. 19, p. 101176, Jul. 2022, doi: 10.1016/j.softx.2022.101176.
[7] G. Andrienko, N. Andrienko, F. Giannotti, A. Monreale, and D. Pedreschi, “Movement data anonymity through generalization,” in Proceedings of the 2nd SIGSPATIAL ACM GIS 2009 International Workshop on Security and Privacy in GIS and LBS, Nov. 2009. http://dx.doi.org/10.1145/1667502.1667510
[8] L. Pappalardo, F. Simini, G. Barlacchi, and R. Pellungrini, “scikit-mobility: A Python Library for the Analysis, Generation, and Risk Assessment of Mobility Data,” *Journal of Statistical Software*, vol. 103, no. 4, 2022, doi: 10.18637/jss.v103.i04.
[9] R. Pellungrini, L. Pappalardo, F. Simini, and A. Monreale, “Modeling Adversarial Behavior Against Mobility Data Privacy,” IEEE Transactions on Intelligent Transportation Systems, vol. 23, no. 2, pp. 1145–1158, Feb. 2022, doi: 10.1109/tits.2020.3021911.
[10] V. W. Zheng, Y. Zheng, X. Xie, and Q. Yang, “Collaborative location and activity recommendations with GPS history data,” in Proceedings of the 19th international conference on World wide web, Apr. 2010. http://dx.doi.org/10.1145/1772690.1772795
[11] A. Ram, S. Jalal, A. S. Jalal, and M. Kumar, “A density based algorithm for discovering density varied clusters in large spatial databases,” International Journal of Computer Applications, vol. 3, no. 6, pp. 1–4, Jun. 2010, doi: 10.5120/739-1038.
[12] P. Newson and J. Krumm, “Hidden Markov map matching through noise and sparseness,” in Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, Nov. 2009. http://dx.doi.org/10.1145/1653771.1653818
[13] Y. Lou, C. Zhang, Y. Zheng, X. Xie, W. Wang, and Y. Huang, “Map-matching for low-sampling-rate GPS trajectories,” in *Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, Nov. 2009. http://dx.doi.org/10.1145/1653771.1653820
[14] C. Yang and G. Gidófalvi, “Fast map matching, an algorithm integrating hidden Markov model with precomputation,” *International Journal of Geographical Information Science*, vol. 32, no. 3, pp. 547–570, Nov. 2017, doi: 10.1080/13658816.2017.1400548.
[15] Y. Zhang and Y. He, “An advanced interactive-voting based map matching algorithm for low-sampling-rate GPS data,” in 2018 IEEE 15th International Conference on Networking, Sensing and Control (ICNSC), Mar. 2018. Accessed: Mar. 21, 2024. [Online]. Available: http://dx.doi.org/10.1109/icnsc.2018.8361315

[16] R. Gooch and V. Chandrasekar, “Integration of real-time weather radar data and Internet of Things with cloud-hosted real-time data services for the geosciences (CHORDS),” in *2017 IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*, Jul. 2017.
[17] Y. Kolokolov, A. Monovskaya, V. Volkov, and A. Frolov, “Intelligent integration of open-access weather-climate data on local urban areas,” in *2017 9th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS)*, Sep. 2017. http://dx.doi.org/10.1109/idaacs.2017.8095124
[18] N. Koutroumanis, G. M. Santipantakis, A. Glenis, C. Doulkeridis, and G. A. Vouros, “Scalable enrichment of mobility data with weather information,” *GeoInformatica*, vol. 25, no. 2, pp. 291–309, Sep. 2020, doi: 10.1007/s10707-020-00423-w.
[19] G. M. Santipantakis, A. Glenis, N. Kalaitzian, A. Vlachou, C. Doulkeridis, and G. A. Vouros, “FAIMUSS: Flexible Data Transformation to RDF from Multiple Streaming Sources,” in *International Conference on Extending Database Technology*, 2018. https://api.semanticscholar.org/CorpusID:3899826
[20] A. R. Mahmood and W. G. Aref, “Distributed Spatial-Keyword Processing,” in *Scalable Processing of Spatial-Keyword Queries*, Cham: Springer International Publishing, 2019, pp. 49–72. http://dx.doi.org/10.1007/978-3-031-01867-1_4
[21] J. Ballesteros, A. Cary, and N. Rishe, “SpSJoin,” in *Proceedings of the 19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, Nov. 2011. http://dx.doi.org/10.1145/2093973.2094054
[22] Y. Zhang, Y. Ma, and X. Meng, “Efficient Spatio-textual Similarity Join Using MapReduce,” in *2014 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT)*, Aug. 2014. http://dx.doi.org/10.1109/wi-iat.2014.16
[23] L. Chen, S. Shang, C. Yang, and J. Li, “Spatial keyword search: a survey,” *GeoInformatica*, vol. 24, no. 1, pp. 85–106, Jul. 2019, doi: 10.1007/s10707-019-00373-y.
[24] Z. Chen, L. Chen, G. Cong, and C. S. Jensen, “Location- and keyword-based querying of geo-textual data: a survey,” *The VLDB Journal*, vol. 30, no. 4, pp. 603–640, Mar. 2021, doi: 10.1007/s00778-021-00661-w.
[25] X. Chen *et al.*, “S2R-tree: a pivot-based indexing structure for semantic-aware spatial keyword search,” *GeoInformatica*, vol. 24, no. 1, pp. 3–25, Jul. 2019, doi: 10.1007/s10707-019-00372-z.
[26] Z. Qian, J. Xu, K. Zheng, W. Sun, Z. Li, and H. Guo, “On Efficient Spatial Keyword Querying with Semantics,” in *Database Systems for Advanced Applications*, Cham: Springer International Publishing, 2016, pp. 149–164. http://dx.doi.org/10.1007/978-3-319-32049-6_10
[27] Z. Qian, J. Xu, K. Zheng, P. Zhao, and X. Zhou, “Semantic-aware top-k spatial keyword queries,” *World Wide Web*, vol. 21, no. 3, pp. 573–594, Jun. 2017, doi: 10.1007/s11280-017-0472-y.
[28] A. Karabinos, P. Tampakis, C. Doulkeridis, and A. Vlachou, “Processing of Spatial-Keyword Range Queries in Apache Spark,” in Proceedings of the 11th ACM SIGSPATIAL International Workshop on Analytics for Big Geospatial Data, Nov. 2023. http://dx.doi.org/10.1145/3615833.3628592
[29] G.S. Theodoropoulos, K. Nørvåg, C. Doulkeridis, Efficient Semantic Similarity Search over Spatio-textual Data, in Proceedings of EDBT’24.
[30] J. Yu, Z. Zhang, and M. Sarwat, “Spatial data management in apache spark: the GeoSpark perspective and beyond,” *GeoInformatica*, vol. 23, no. 1, pp. 37–78, Oct. 2018, doi: 10.1007/s10707-018-0330-9.
[31] Y. Huang, Y. Weng, S. Yu, and X. Chen, “Diffusion Convolutional Recurrent Neural Network with Rank Influence Learning for Traffic Forecasting,” in *2019 18th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/13th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*, Aug. 2019.: http://dx.doi.org/10.1109/trustcom/bigdatase.2019.00096
[32] D. A. Tedjopurnomo, Z. Bao, B. Zheng, F. Choudhury, and A. K. Qin, “A Survey on Modern Deep Neural Network for Traffic Prediction: Trends, Methods and Challenges,” *IEEE Transactions on Knowledge and Data Engineering*, pp. 1–1, 2020, doi: 10.1109/tkde.2020.3001195.

[33] B. M. Williams and L. A. Hoel, “Modeling and Forecasting Vehicular Traffic Flow as a Seasonal ARIMA Process: Theoretical Basis and Empirical Results,” *Journal of Transportation Engineering*, vol. 129, no. 6, pp. 664–672, Nov. 2003, doi: 10.1061/(asce)0733-947x(2003)129:6(664).
[34] K.-R. Müller, A. J. Smola, G. Rätsch, B. Schölkopf, J. Kohlmorgen, and V. Vapnik, “Predicting time series with support vector machines,” in *Lecture Notes in Computer Science*, Berlin, Heidelberg: Springer Berlin Heidelberg, 1997, pp. 999–1004. http://dx.doi.org/10.1007/bfb0020283
[35] Z. Cui, K. Henrickson, R. Ke, and Y. Wang, “Traffic Graph Convolutional Recurrent Neural Network: A Deep Learning Framework for Network-Scale Traffic Learning and Forecasting,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 11, pp. 4883–4894, Nov. 2020, doi: 10.1109/tits.2019.2950416.
[36] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, “A Comprehensive Survey on Graph Neural Networks,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 1, pp. 4–24, Jan. 2021, doi: 10.1109/tnnls.2020.2978386.
[37] S. Guo, Y. Lin, N. Feng, C. Song, and H. Wan, “Attention Based Spatial-Temporal Graph Convolutional Networks for Traffic Flow Forecasting,” *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, pp. 922–929, Jul. 2019, doi: 10.1609/aaai.v33i01.3301922.
[38] C. E. M. Cuza, N. Ho, E. T. Zacharatou, T. B. Pedersen, and B. Yang, “Spatio-temporal graph convolutional network for stochastic traffic speed imputation,” in *Proceedings of the 30th International Conference on Advances in Geographic Information Systems*, Nov. 2022. http://dx.doi.org/10.1145/3557915.3560948
[39] Z. Wu, S. Pan, G. Long, J. Jiang, and C. Zhang, “Graph WaveNet for Deep Spatial-Temporal Graph Modeling,” in *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*, Aug. 2019.: http://dx.doi.org/10.24963/ijcai.2019/264
[40] Y. Huang, Y. Weng, S. Yu, and X. Chen, “Diffusion Convolutional Recurrent Neural Network with Rank Influence Learning for Traffic Forecasting,” in *2019 18th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/13th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*, Aug. 2019. http://dx.doi.org/10.1109/trustcom/bigdatase.2019.00096
[41] J. Zhou *et al.*, “Graph neural networks: A review of methods and applications,” *AI Open*, vol. 1, pp. 57–81, 2020, doi: 10.1016/j.aiopen.2021.01.001.
[42] S. Shekhar, V. Gunturi, M. R. Evans, and K. Yang, “Spatial big-data challenges intersecting mobility and cloud computing,” in *Proceedings of the Eleventh ACM International Workshop on Data Engineering for Wireless and Mobile Access*, May 2012. http://dx.doi.org/10.1145/2258056.2258058
[43] D. H. Douglas and T. K. Peucker, “ALGORITHMS FOR THE REDUCTION OF THE NUMBER OF POINTS REQUIRED TO REPRESENT A DIGITIZED LINE OR ITS CARICATURE,” *Cartographica: The International Journal for Geographic Information and Geovisualization*, vol. 10, no. 2, pp. 112–122, Dec. 1973, doi: 10.3138/fm57-6770-u75u-7727.
[44] N. Meratnia and R. A. de By, “Spatio-temporalCompression Techniques for Moving Point Objects,” in *Advances in Database Technology - EDBT 2004*, Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 765–782. http://dx.doi.org/10.1007/978-3-540-24741-8_44
[45] J. Muckell, J.-H. Hwang, V. Patil, C. T. Lawson, F. Ping, and S. S. Ravi, “SQUISH,” in *Proceedings of the 2nd International Conference on Computing for Geospatial Research &amp; Applications - COM.Geo ’11*, 2011. http://dx.doi.org/10.1145/1999320.1999333
[46] M. Potamias, K. Patroumpas, and T. Sellis, “Sampling Trajectory Streams with Spatio-temporalCriteria,” in *18th International Conference on Scientific and Statistical Database Management (SSDBM’06)*. http://dx.doi.org/10.1109/ssdbm.2006.45
[47] G. Trajcevski, H. Cao, P. Scheuermanny, O. Wolfsonz, and D. Vaccaro, “On-line data reduction and the quality of history in moving objects databases,” in Proceedings of the 5th ACM international

workshop on Data engineering for wireless and mobile access, Jun. 2006. http://dx.doi.org/10.1145/1140104.1140110

[48] Amigo, D., Pedroche, D. S., García, J., & Molina, J. M. (2021). Review and classification of trajectory summarisation algorithms: From compression to segmentation. International Journal of Distributed Sensor Networks, 17(10), 15501477211050729.

[49] Makris, A., Kontopoulos, I., Alimisis, P., & Tserpes, K. (2021). A comparison of trajectory compression algorithms over AIS data. IEEE Access, 9, 92516-92530.

[50] Amigo, D., Pedroche, D. S., García, J., & Molina, J. M. (2021). Review and classification of trajectory summarisation algorithms: From compression to segmentation. International Journal of Distributed Sensor Networks, 17(10), 15501477211050729.

[51] Makris, A., Kontopoulos, I., Alimisis, P., & Tserpes, K. (2021). A comparison of trajectory compression algorithms over AIS data. IEEE Access, 9, 92516-92530.