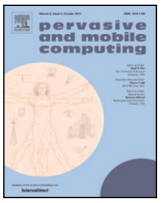
Pervasive and Mobile Computing 103 (2024) 101946

Contents lists available at ScienceDirect

Pervasive and Mobile Computing

journal homepage: www.elsevier.com/locate/pmc



A toolkit for localisation queries

Gabriele Marini [a,*], Jorge Goncalves [a], Eduardo Velloso [b], Raja Jurdak [c], Vassilis Kostakos [a]

[a] The University of Melbourne, Grattan Street, Parkville, Melbourne, 3010, Victoria, Australia
[b] University of Sydney, City Road, Sydney, 2006, New South Wales, Australia
[c] Queensland University of Technology & CSIRO Data61, 2 George St, Brisbane, 4000, Queensland, Australia

ARTICLE INFO

Keywords:
Semantic localisation
Indoor localisation
Patterns
Movements
Time-series
Pattern syntax
Mobility

ABSTRACT

While UbiComp research has steadily improved the performance of localisation systems, the analysis of such datasets remains largely unaddressed. In this paper, we present a tool to facilitate querying and analysis of localisation time-series with a focus on semantic localisation. Drawing on well-established models to represent movement and mobility, we first develop a query language for localisation datasets. We then develop a software library in R that implements this querying. We use case studies to demonstrate how our programming tool can be used to query localisation datasets. Our work addresses an important gap in localisation research, by providing a flexible tool that can model and analyse localisation data programmatically and in real time.

1. Introduction

Establishing and tracking the location of people and objects has been one of the defining aims of ubiquitous computing. Localisation techniques have evolved to be more accessible, scalable and accurate than ever. As systems such as Indoor Positioning Systems (IPS) and Real-Time Localisation Services (RTLS) make their way into our daily lives, the amount data collected grows exponentially and so does the complexity of exploring these datasets. Multiple types of localisation technologies have been developed over the years, both for indoor and outdoor settings. Global Positioning System (GPS) has become the de-facto standard for outdoor localisation, but despite the plethora of proposed options there is not yet a single general purpose technology for indoor localisation [1].

Researchers and practitioners from fields such as Ubiquitous Computing (UbiComp) [2] and Geographic Information Systems (GIS) [3] face the same recurring challenge: they set up a system to accurately localise entities across space and time, and set up a structured way to capture and store this data. From that point on, any analysis they want to conduct most likely needs to be developed from scratch. Turning localisation data to actionable information is still a relatively poorly structured or supported activity and it is not standardised in practice. The heterogeneity of indoor localisation datasets often requires researchers and data analysts to either make use of the available general-purpose programming languages or to develop their own ad-hoc tools for their analyses.

This is in fact a challenge that we faced during a longitudinal deployment of an indoor localisation: to the best of our knowledge, even in the most recent years, there are no tools to facilitate the analysis of localisation data in a systematic and rigorous manner. Despite tools such as R or Python packages such as dplyr [4], data.table [5] and pandas [6], SQL engines and spatio-temporal variants [7,8], the analysis often lacks a tool-agnostic explanation of the data processing pipeline. There is no standard shared by

* Corresponding author.
E-mail address: g.marini@unimelb.edu.au (G. Marini).

https://doi.org/10.1016/j.pmcj.2024.101946
Received 6 December 2023; Received in revised form 18 March 2024; Accepted 19 May 2024
Available online 28 May 2024

analysts and researchers to deal with spatio-temporal data-sets, neither at the coding level, at the analysis level, nor at the more abstract formalism level.

There is a distinct lack of literature that "closes the loop": systems that analyse localisation data in real-time, and respond to these changes dynamically. We lack replication, reproducibility, and extension of localisation work. Specifically, we come across papers or projects that involve a substantial localisation component to mainly describe the technology, accuracy, and deployment; yet, when it comes to interpreting the localisation data the analysis can be intricate, ad hoc, monolithic, and very often limited. These do not seem to be isolated cases, ranging from deployments in public and open spaces such as museums [9–12], shopping malls [13,14], to professional and academic environments such as workplaces and universities [15,16], schools [17,18] and even clinical settings [19].

We argue that this is the result of a distinct lack of tools for analysing localisation data. Visualisations of spatio-temporal data are not a novel approach [20,21], but they lack versatility and are not easy to implement in a data analysis pipeline. Further developments are needed to support more complex analyses. A more concise and flexible tool carries the potential to fully capture the richness of localisation data and extract unique and interesting patterns of human behaviour [22]. Hayward et al. [23] highlighted a literature gap in the academic research around IPS and ILBS and proposed that future work "will focus on developing a unified framework [...] to bridge the current literature gap of processing, storing and accessing data in addition to presenting this to the user through the use of an interface".

In this paper we present a tool we have developed to assist and guide the analysis of localisation data in a systematic and formal manner, while at the same time being versatile and integrating closely with a programming environment. We developed a rigorous formalism that lets us theorise about the analysis queries. We then present our tool built as an R package that facilitates pipeline-style analysis of localisation data, much in the spirit of libraries such as Dplyr. Our tool aims to be agnostic of the underlying technology used to generate the localisation data. For this reason, its focus is on semantic localisation, meaning that our tool is designed to analyse tuples of $\langle agent, location, time \rangle$.

2. Background

2.1. Localisation technologies

The field of indoor localisation has been an active field of research for many years with a large amount of data being collected by all sorts of localisation systems. The research has eventually converged into the usage of a few technologies [1], especially since smartphones have started including a wide array of sensors [24]. Nevertheless, the proliferation of devices connected to the Internet of Things made localisation a distributed task where devices of any kind, not just smartphones, contribute to the spatial information of common areas of interest. This phenomena highlights the heterogeneity of devices and datasets carrying spatio-temporal information. While the underlying technology differs according to the needs and the constraints posed by the physical spaces, the localisation methods used can be categorised in four main categories: Triangulation, proximity, fingerprinting, dead-reckoning and combinations of these methods also called hybrid localisation [24]. In all cases, the end goal is the extraction of an accurate location of the people or objects being tracked, and the classification of this position in relation to the area and its purpose.

Localisation systems have shown their potential in settings such as offices [25], hospitals [26,27], shopping centres and malls [28,29] and even underground mining [30]. Crowdsensing-based localisation systems have also become more common, with deployments appearing in museums [9] and being employed in disaster management [31]. However, a lot of the research in indoor localisation and its applications often falls short of a practical approach to analysing longitudinal localisation data which spans over multiple days and potentially up to multiple years. A common occurrence in localisation applications and mobility analysis is the need for future work to address the analysis of large amount of movement data to detect patterns and deviation from them to detect anomalies [14,32]. Even in the most recent years, localisation systems hint and the potential usefulness, for instance in the context of infectious diseases tracking and epidemics management such as COVID-19 [33].

2.2. Definition of movement

On a more theoretical level, the field of Geographic Information System (GIS) has been a parallel field of research dealing with movements and traces of both outdoor and indoor localisation systems [34,35]. Most of the work that deals with movement patterns and spatio-temporal data relies on framing the problem in a more familiar way, or at least a way that can be easier to read, understand and analyse. Andrienko G. & Andrienko S. have published a series of milestone works aimed at defining a formal and conceptual approach to analytics for localisation and movement data. One of their most successful works [21] lays down the basis of space, time and objects as the main components of movement data, taking inspiration from an earlier work by Peuquet [36]. In their conceptual model, Andrienko et al. [37] categorised the set of possible queries for dealing with movement datasets. In their model, time is the main focus and depending on the *a priori* knowledge and the expected result from the query, the authors identify two main types of questions:

- Given some definition of time within the data, the other entities are to be discovered.
- Given information of other type, the time component needs to be discovered.

One interesting application of these questions lies in the comparison of two or more time moments or instances, which can be especially useful when tracking the location of an object (or person) over time.

Another milestone work presents a conceptual framework to describe the information encoded in movement data [37]. The framework builds on top of the previous work and uses the same three fundamental components to describe movement: space, time and objects. Movement in particular is defined as "The change of the spatial position(s) of one or more movers over time". An important addition to the model is the classification of movement analysis tasks based on their respective focus:

- *Focus on objects*: Objects are defined in terms of their location at any given time and in terms their relation to locations, times and other objects
- *Focus on space*: Locations are defined in terms of the objects they contain at a given time and in terms of their relations to objects, times and other locations.
- *Focus on time*: Time units are defined in terms of objects and space and in terms of their relations to objects, places and other time units.

It is also important to consider how the data should be represented and the authors go on to list the different known methods of position recording [38]:

- *Time-based*: Time units are regularly spaced e.g. the location of objects changes every 10 s so records in the dataset are evenly spaced in time.
- *Change-based*: Whenever an object moves and its position changes, a new record is added to the dataset e.g. A person moves enough for their phone's GPS coordinates to show a meaningful change in position.
- *Location-based*: Whenever an object enters or leaves an area, a new record is added to the dataset e.g. A person enters or leaves the range of a scanner installed in a room.
- *Event-based*: Records get added as a particular event occur e.g. specific activities done by the tracked individual such as logging into a system or swiping a badge.

2.3. Analysis of movements and behaviour

A formal and conceptual definition of movement allows for a more methodical approach to the analysis of movement traces. Applications of the framework defined by Andrienko G. & N. range from a more visual approach [14,20] to more ways of encoding patterns through text and strings. Yaeli et al. [14] have proposed a more visual approach to analyse mobility data from customers and analyse their behaviour. Their approach is based on an analogy between user behaviour on the web and the movement of customers in a shopping centre. In both cases, a graph-based representation seems to clearly convey important insights in terms of customer (or web user) behaviour and the authors point out that although visualisation is an important exploratory first step, there might be more to discover through analytical results and their visualisation. Another interesting approach in gathering insights from indoor localisation is based on a symbolic, textual representation of customer shopping paths [13]. The customer's movements are encoded in such a way that a single character represents a variable amount of time the customer spent in a section of a supermarket. However, as the authors point out, the approach can be useful to represent a customer's movement pattern but it faces the fundamental problem of losing information regarding the actual amount of time spent in each section. Furthermore, the approach does not mention a way to extend this encoding to include multiple customers moving throughout multiple areas.

The evolution and the proliferation of indoor localisation technologies has brought an incredible amount of data, and thanks to advances in Machine Learning (ML) and Artificial Intelligence (AI) the location data can be more easily classified according to spatio-temporal information. Andrienko [39] have developed a new approach to analysing movement, using an analysis that is based on the concept of movements seen as behaviours and defined as an ordered sequence of states or activities. Movement (or behaviour) can then be represented as a State Transition Graph (STG) where an individual or a collection of individuals [40] move from one state to another, regardless of whether the state represents an activity or a space. In the case of indoor localisation we can talk about semantic localisation [41] where the "semantic" aspect of the analysis is given by the purpose of a room or space.

This sort of semantic abstraction is not directly related to the spatial information of the location. While this is effectively a departure from the definition of location as a set of spatial coordinates, "The absence of specific spatial information is not a weakness but a strength of an STG representation" [39]. One of the main drawbacks of a graph-based representations is the lack of a time dimension. While an STG representation works well to visually show movement and patterns in relation to their ordered sequence, we cannot encode the actual time duration nor define the movements of multiple individuals across spaces, in time.

2.4. Time-series encoding and analysis

The detection of spatio-temporal patterns in a time-series is not a recently addressed challenge and fields such as biology or sports science have developed ways to detect patterns in their particular datasets [42]. Encoding the data is usually the first step in analysing time-series and it is often the most important one for classification problems. Depending on the type of information and the classification problem, a time-series can be represented in a continuous and numerical way, or in a discrete and symbolic one. The choice of the time-series encoding heavily affects the rest of the data mining process, including the detection of patterns and their definition. Most prior work deals with continuous, numerical real-valued data which is often represented using well known transforms such as Discrete Fourier Transform (DFT) [43] and Discrete Wavelet Transform (DWT) [44]. However, these transforms

have the limitation of only working on continuous data, only allowing for a set encoding "resolution" or, in the case of wavelets, they only work on time-series with a length equal to a power of two.

We can narrow down the set of choices for the encoding of the time-series. Lin et al. [45] have developed a symbolic representation of time-series called SAX (Symbolic Aggregate approXimation) to overcome the limitation of a continuous dataset and to take advantage of existing time-series data mining algorithms such as Markov Models, Suffix Trees and Hierarchical Clustering. The representation first applies a dimensionality reduction through Piecewise Aggregate Approximation (PAA) [46], dividing the time-series into equally sized time-"frames". The PAA coefficient of each time-frame is then mapped to a symbol, reducing the time-series to a unique string of symbols. There have been extensions to the SAX representations [47] such as ESAX [48] and TrSAX [49] which addressed the limitation of only representing the mean value in each time-frame. More recently, Yu et al. have expanded the SAX symbolic representation and implemented regular expressions to overcome previous limitations [50]. Even in recent years new methods are being researched to represent indoor trajectories through a symbolic representation [51].

In the special case of spatio-temporal datasets and especially in the case of indoor localisation datasets, a symbolic representation works especially well with the concept of semantic localisation previously mentioned. Farina et al. [51] recently presented a compressing algorithm based on a symbolic representation specifically tailored for indoor trajectories.

2.5. Pattern matching in time-series

Pattern recognition and pattern matching are processes that are not just pervasive in today's technology, but essential to many of the tools we use today. Generally speaking, to extract patterns from a dataset, a common approach is to train a machine learning model on a training data set through supervised or unsupervised learning [52]. In most cases, the goal is to find a pattern, or a function, that links input data to its output in a training dataset and using this newfound pattern to make sense of new, previously unseen data. Machine Learning approaches work well for real-time data where the human input is not needed at all times or for automated classification problems. In our scenario however, the human input starts with the definition of the pattern of interest. We can call this a special case of semi-supervised learning where the pattern serves as a labeller for certain portions of the dataset that match the pattern.

Most of the pattern recognition algorithms, especially the ones based on Machine Learning or Deep Learning, work extremely well on continuous two or three dimensional signals such as images or sounds and sensor data including Bluetooth, WiFI or RFID signal strength. However, symbolic and discrete representations require a different kind of analysis due to the "signal" value at each interval being only meaningful in its own context. For instance, an object moving from a region labelled as 1 to a region labelled as 2 would need to be interpreted in the context of the space semantics and the purpose or location or each region means.

One of the earliest works in pattern recognition by Bunke [53] gives a good overview of analysing symbolic data structures such as strings or graphs. Despite being relatively old work, the theoretical foundation is still very much applicable to the present day, including algorithms for pattern string matching, hidden markov models and formal grammars. For instance, a string representation of indoor locations such as the one used for shopping mall visits or hospitals [13,26] could be analysed through the string matching algorithms proposed by Bunke such as their modified NN-classification algorithm. Lin et al. [54] have also proposed a technique for sequence matching and anomaly detection as a direct application to their SAX representation. An interesting perspective provided in this approach is the idea of anomalies as a deviation from the "normal" behaviour, which could be defined as a pattern itself, referring to the similar notion of anomaly in the field of biology and DNA sequencing. Even in recent years it is clear that a simpler and updated method to explore these type of dataset might be needed [55].

2.5.1. Querying movement data

The concept of mobility patterns encompasses the ideas of movement and time-series pattern matching to make sense of localisation data. Representations such as mobility Graphs [40], and STG [39] offer an intuitive way to visualise mobility data. However, to gather insights from movement data we also need a way to query the dataset in a structured and rigorous manner. Mamoulis et al. [56] used symbolic encoding to define an object's so called periodic pattern as an ordered sequence of symbols. However, a limitation of this approaches and similar ones is the inability to represent multiple objects or individuals and how they move across spaces.

Up to this point we have only considered a single time-series which focused on a single object. As Shurkhovetskyy et al. [57] point out: It is important to understand the difference between a multi-variate (or multi-dimensional) time-series and multiple uni-variate time-series. An example of a multi-variate time-series would be a large dataset that includes data from sensors such as temperature, pressure or humidity in a room. Deploying different temperature sensors in different rooms across one building would instead result in multiple uni-variate time-series. If we consider the example of an indoor localisation system in a building, then each room in the dataset would be a uni-variate time-series.

Mouza & Rigaux [8] developed an interesting approach to define and use mobility patterns through regular expressions also providing a visual representation of the patterns as a Non-Deterministic Finite state Automaton (NFA). Hadjieleftheriou et al. built upon the approach from Mouza & Rigaux and introduced a novel type of query called Spatio-Temporal Pattern Queries (STP) [58] where a pattern is represented by an ordered sequence of predicates. STP could be defined as *With Time* queries where time is expressed in terms of instants or intervals: "Find objects that crossed through region A at time $T1$, came as close as possible to point B at a later time $T2$ and then stopped inside circle C some time during interval $(T3, T4)$" or *With Order* queries where time is relative to each predicate: "Find objects that first crossed through region A, then passed as close as possible from point B and finally stopped inside circle C".

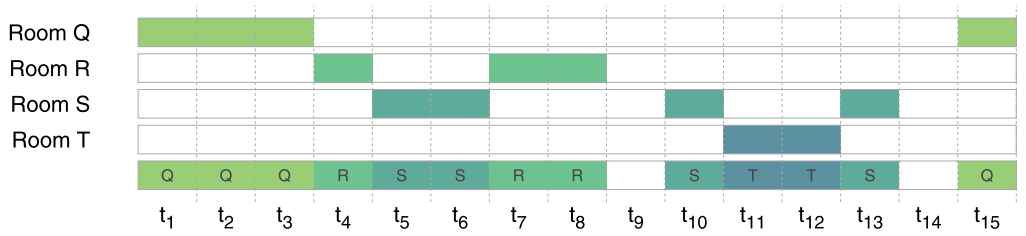


Fig. 1. A graphical representation of the journey of an object X.

2.6. Research gap

While within the Ubicomp community the topic of indoor localisation has mostly revolved around the development of technologies and sensors to further increase the accuracy of localisation, there has been substantial work on the analysis of movement in the field of GIS. There seems to be a disconnect between the two fields: As the field of ubiquitous computing grows, so does the amount of data collected as well as the need for tools to analyse such data. At the same time, the field of GIS has brought a plethora of solutions to the problem, but they are often confined to an outdoor environment or they mostly only work on special kind of datasets and with tools that computer scientists in other fields rarely use. As a reference: ArcGIS Indoors [59] is an extension to the widely used software ArcGIS which is meant to "Build an indoor geographic information system (GIS) and put the power of indoor mapping, wayfinding, and space management software into everyone's hands". This tool was only just released in 2019. A further extension that slightly closes the gap was released in 2022 [60] showing a growing interest in indoor localisation technologies and applications.

Our main objective is to bridge the gap between the theoretical findings and formulations in the field of GIS, and the more practical solutions in terms of implementation and data representation. We address four main research challenges:

1. Develop an appropriate encoding for movement time-series, and provide a choice of which component to focus on: objects, spaces or time.
2. Represent the movement of objects through regions with a sufficient control over the time component i.e. retain time and duration as well as ordering.
3. Develop a syntax that enables us to query movement time-series. The syntax should allow for flexible groups of objects, a flexible duration of states, and a flexible selection of regions.
4. Develop a tool that uses these patterns to query and analyse the dataset programmatically, in manner that is suitable for real-time and interactive systems.

Most existing approaches define the foundation and the theoretical model for the representation of spaces [37,39], movement [13,14,51] or even patterns [58] but usually lack a practical implementation or they are not appropriate for localisation datasets. On the other hand, the available programming tools for time-series analysis often mostly focus on continuous sensor data, leaving little to no room to adjust these tools to a different time-series representation such a symbolic one. More practical solutions with an example of potential implementation such as SSTS [55] or mobility patterns [8], are either limited in one of the three components of movement (space, time and objects) or their implementation uses languages and tools that are not common anymore among data analysts and programmers.

We argue that our approach is at intersection of the two fields of research, providing a more relevant implementation and a streamlined pipeline. Although it is possible to analyse large time-series focusing on movement patterns, a lot of the tools either require a deep technical knowledge and understanding of data analysis tools and languages, or they were not originally intended for the purpose of defining patterns of movements and extracts meaningful information through these patterns.

2.7. Limitations of available techniques

Before we delve into the definition of our model's syntax, it can be useful to see how some of the approaches we mentioned earlier have dealt with the obstacles we identified. We provide a simple basic example that uses a symbolic representation of a localisation time-series and how we can represent movement patterns using regular expressions.

Consider a generic example of an object moving through a discrete set of regions in time. Using a symbolic representation we can track the movement of an object X through regions Q, R, S, T. The length of each time interval is not important here but for the sake of simplicity let us define the time as evenly spaced 20-s intervals.

STGs [39] can be a useful visual representation of X movements but as the authors themselves state, spatial and time information is lost when focusing on the sequence of semantic locations. However, a state-based approach is a common way to represent movements not just in a visual manner. We can assume a *time-based* sequence of states [37] with regular intervals:

Description: Given the journey represented in Fig. 1 we can describe in natural language as:

- Object X was in Q for three time slots

- Moved to R for one time slots then to S for two time slots, then to R for two time slots
- The localisation system then lost track of X for one time slot
- X reappeared in S for one time slot
- X Moved to T for two time slots and then moved back to S for one time slot
- The system lost track of S again for one time slot
- Finally, X reappeared in Q

Symbolic Representation: $QQQRSSRR\_STTS\_Q$

Each symbol can be read as "Object X was in region R for t seconds (or any other time unit)". The special character '_' represents a time intervals where X was not found within the tracked area.

Query: We are seeking instances in which "*X started in either S or R and eventually arrived in Q*" which we can divide in three ordered sub-queries:

1. The object X started in either region S or R
2. Then went to any other region
3. Finally settled in region Q

One relatively straightforward approach is to use string matching tools such as *regular expressions* or *regexp*. Regular expressions have been especially useful in dealing with time-series and symbolic representations [53] but they work well on semantic localisation datasets too [13,26]. This powerful pattern definition tool has been widely used in the past and it is still prevalent today [50,55]. Recent tools such as SAXRegEx [50] have shown that, with some adjustments, a symbolic representation can substantially outperform numerical methods such as Dynamic Time Warping (DTW), while also noting that "Regex is meant for exact search instead of 'fuzzy' search like pattern search in time series". If we consider the case of localisation where location can be noisy and fuzzy, supporting this kind of noisiness in the system is not a trivial problem to address.

Regexps do allow for some flexibility in their definition: Using regexp on a symbolic time-series we can now answer simple questions such as "Times when X was in Q" using a pattern such as $/Q^{+}/$. But we can also answer more complex queries such as "Times when X appeared at least once in any region and then left the tracked area" $/(?:[QRST]^{+})(\_)/$. To answer our initial query, we can find instances where X arrived in Q starting from either S or R by using the following expressions $/[RS]^{+}.^{+}Q/$. Regular expressions are notoriously hard to work with and they were not originally meant to be used on time-series. Representing a very large dataset as a single string is not only inefficient but it can make the exploration and wrangling process very time consuming. This becomes especially clear when the time-series includes multiple entities, each with their own symbolic time-series.

Mamoulis et al. [56] borrow some concepts from regular expressions but provide a taxonomy adapted for the mining of localisation datasets. The authors defined the idea of representing a localisation time-series as a *sequence* of spatial locations (each encoded by a unique character), and the concept of *periodic segments* as smaller sub-sequences that appear multiple times within the larger sequence. The mining process starts with the definition of a *periodic pattern*, a sequence of spatial region identifiers or the special identifier $*$ representing the whole spatial domain (i.e. "anywhere"). Following their syntax, the first state would need two different periodic patterns such as $(S * Q)$ and $(R * Q)$. However, the pattern only matches sequences of the same length so both patterns would only return (S_Q). The lack of flexible time is a major drawback: a pattern such as $(S ** S)$ would only match the sequence $(STTS)$ but not (SRR_S) due to the different string length.

Another way to represent movement is the syntax introduced by Mouza and Rigaux [8] to support their mobility patterns:

$$Q\{3\}R\{1\}S\{2\}R\{2\}\_\{1\}S\{1\}R\{2\}S\{1\}\_\{1\}Q\{1\}$$

Using mobility patterns we can now define a more flexible pattern in a concise way and expand it as pseudo-code:

```
({(S|R)+.@x.(Q)+}, {@x != a}) # Compact version
start_at {S,R}, follow @x, follow Q; @x != a # Expanded version
```

Such approaches cannot represent a group of objects, and cannot define states where time and objects are variable. Mobility patterns have hinted at a similar solution using SQL:

```
SELECT * FROM Mob WHERE trajmatches({(S|R)+.@x.(Q)+}) AND @x != 'a'
```

Even in this scenario we are left with the question of how to define states that include one more or more specific objects and not just the whole set of objects. As most of other methods, mobility patterns too focus on one aggregated dataset that either follows a single object or all of the objects in the area. Even recent tools like SAXRegEx [50] require a costly preparation step to merge multiple entities' time-series into one. Intertwining the entities' symbols to merge multiple time-series into one sequence can enable regexp to work on datasets with more than one entity, however it also introduces extra complexity in the definition of queries, requiring users to specify each of the entities within the query. The available techniques fail to tackle all challenges, or manage to overcome challenge (1) and (2) but fail to provide a flexible enough pattern definition and implementation to solve (3) and (4).

Table 1 gives an overview of the existing tools for localisation queries and the features they currently provide. Stemming from the challenges we identified in the research gap section, we compare how the three main components (entities, time and space) are represented in the time-series and how they can be utilised in the query pattern definition. The combination of a robust time-series representation and a flexible pattern definition directly relate to the versatility of the tool:

Table 1
Comparison of existing querying tools for localisation datasets.

Tool	Entities		Space		Time	
	Repr.	Grouping	Repr.	Grouping	Repr.	Fuzzy
RegExp	Single	No	Simple	No	Relative	Yes
SQL	Multiple	No	Simple	Yes	Both	No
STP [58]	Single	Yes	Simple	Yes	Relative	No
STG [39]	Single	No	Simple	No	Relative	Yes
Mob. Patterns [8]	Single	No	Simple	Yes	Relative	No
SAX [47]	Single	No	Simple	No	Relative	Yes
SAXRegEx [50]	Multiple	Yes	Complex	No	Absolute	No
Mamoulis et al. [56]	Single	No	Simple	No	Relative	No
Ours	Multiple	Yes	Complex	Yes	Both	Yes

- *Entities*: The pattern syntax and the time-series representation can either support a single entity (e.g. RegExp on a string of symbolic locations) or multiple entities at once (e.g. SAXRegEx, where multiple entities' strings get merged by intertwining each entity's symbols in one large string). We further classify the tools by the possibility of searching for groups of entities.
- *Space*: We assume the space representation to be discrete and symbolic. Some tools only allow or work best with symbols represented by a *single* character such as Q, R or S. More advanced tools have been expanded to deal with *complex* identifiers such as "OR", "Office" or "Surgery".
- *Time*: Time can be represented as *relative*, with the first time slot representing the beginning of the time domain, or *absolute* where each time slot carries information about time beyond the time-series domain - i.e. Time expressed in UTC or unix time or even according to the ISO standard. In terms of pattern definition, we divide tools by the possibility of defining fuzziness within the pattern and to be able to query segments of the time-series with gaps in it.

For instance, regular expressions natively only support a single sequence of characters (i.e. a single entity's symbolic time-series) and therefore does not support grouping of entities. SAX [47] originally did not support a complex spatial identifier so each symbol was a simple single character, it also did not support variable-length patterns. SQL however allows for running one single query across multiple time-series (i.e. columns), but the temporal grouping of entities and spaces is non-trivial and not natively supported. There is also no native support for fuzziness and addressing gaps of variable length within the dataset.

3. System

Our system consists of two parts that mirror each other: a formal grammar that is used to articulate queries, and a library for R that enables analysts to queries on data. These two parts are independent: the formal grammar acts as a baseline reference, while the R library is a sample implementation. It is possible for other implementations to be created for different languages. Our code is open-source and available as a GitHub repository.[1]

To address the four challenges we identified, we applied the well-founded theoretical models from Andrienko et al. [37,39] as well as the idea of mobility patterns introduced by Mouza & Rigaux [8]. The main requirement for our model is for the dataset to be a discrete and regular time-series that can be represented through a symbolic representation. The fundamental idea of the model is to see movements as a change in the system state. A pattern can then be described in terms of an ordered sequence of system states where each state can focus on one or more objects and regions as well as one or more time intervals (duration).

In the following sections we follow a bottom-up approach to present the conceptual framework and formal syntax definition: starting from the most basic definitions of each component to the definition of a movements and patterns. We start with an overview of the model and its components. We then show some examples of applications in an hypothetical scenario as well as a real-world scenario. In the examples we also provide a sample implementation of the syntax as R code.

3.1. Basic components

Our model is based on the idea of the three components proposed by Peuquet [36]: time (*when*), space (*where*) and objects or agents (*who*). We consider time to be discrete and regular and we assume space and regions to be immutable in time. For the purpose of semantic localisation and mobility we will refer to objects as agents to highlight the idea of people having a unique role in their workplace or team. We will also use the terms space, region or room interchangeably to avoid confusions with the symbolic representation of the *where* component. We can therefore explain the system in terms of any of the three components or a their combination: who was involved in the event, where the event happened, and when did it start & end.

Agent (*who*) $\{a_0, a_1, \ldots, a_n\}$**:** We call agents the entities that move across regions and time within our localisation system. An agent can be seen as anything we are able to track: people wearing a badge, objects, animals carrying a GPS tracker, vehicles, smartphones. We thus define a set A of agents a_i where $1 \leq i \leq n$ such as $\{a1, a2, a3\}$

[1] https://github.com/Gabryxx7/semantic-localisation-r.

Table 2
Descriptions and examples of all possible queries.

Query	Input	Output	Example
who → *where* + *when*	One or more agents	Time and location the agent(s) visited any location at any time.	Given a COVID positive patient, we are interested in finding all the rooms they visited, when and for how long.
where → *when* + *who*	One or more locations	Time and agent(s) that have visited the location.	Given a room visited by a COVID positive patient, we are interested in finding who visited the room, when and how long.
when → *where* + *who*	A time slot or a time range.	A list of locations visited and the agents tracked in the time range.	Knowing when a COVID positive patient visited the ward, we are interested in finding who was in any of the ward's rooms at the given time.
who + *when* → *where*	One or more agents and time slots	A list of locations visited by the agent at the given time slot(s)	Knowing when a COVID positive patient visited the ward, we are interested in which room they have visited.
who + *where* → *when*	One or more agents and rooms	A list of time slots when the agent(s) visited the location(s)	Knowing where a COVID positive patient has been, we are interested in knowing the exact time.
where + *when* → *who*	One or more rooms and time slots	A list of agents that been in the location(s) at the given time slot(s)	Knowing which room a COVID positive patient has been in and when, we are interested in knowing which other agents have visited the room at the given time.

Region (*where*) $\{r_0, r_1, \ldots, r_n\}$**:** The region domain is assumed to be discrete. We model the region to reflect the heterogeneity of localisation systems currently used. The granularity varies depending on the used technology: down to a meter, room level, building level or down to predefined areas. We define our *region-domain* as a set S of locations s, be it a region, a room or an area expressed in terms of coordinates. When formulating a query it is possible to define an area or a set of areas of interest as $\{r1, r2, r3\}$

Time (*when*) $\{t_0, t_1, \ldots, t_n\}$**:** Time in our case is considered to be discrete. Because we focus on movement resulting from sensing, it is often the case that the dataset resulting from filtering is discrete with respect to time. We define our *time-domain* as a set T of all the possible time chunks in the dataset t. A time interval can be expressed as $[a, b]$. The granularity of the time-series is not constrained. A time instance can shorter than a millisecond or longer than a day depending on the granularity required by the analysis. We can thus refer to either a specific instance in time or to a time range between two instances:

- Time **Instance**: $t \in \{t_0, t_1, \ldots, t_n\}$ where n is the number of discrete observations in the dataset
- Time **Range** or **Duration**: $\equiv [t_i, t_j]$ where t_i t_j are instances and $t_i < t_j$

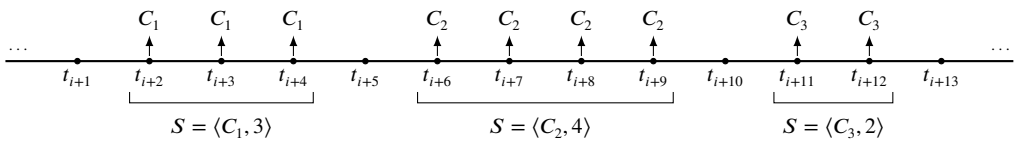
Movement analysis tasks differ depending on which component they are focused on. For instance, we might be interested in either studying the usage of rooms, tracking the movements of a single agent or finding out what happened at any given time. Depending on what information we already have and therefore which of these components are fixed or variable, the type of query and its output can be vary. Based on the notion of "question types" and "reading levels" introduced by Bertin [61], we identify six types of queries where one or more available components can be used to discover the missing ones (see Table 2). As an example we will consider COVID contact tracing, where the agents include a COVID positive patient as well as other patients, nurses and doctors moving through a hospital ward.

3.2. States

Andrienko [39] introduced the idea of State Transition Graphs (STG) for semantic analysis as a representation of movement. STGs representation is a "lossy" abstraction in that specific spatial or temporal information is lost in favour of a semantic representation. As the authors note, this is not necessarily a weakness and that is especially true for semantic localisation. However, STGs leverage the innate human capability of identifying visual patterns and do not deal with the practical implementation of the representation or the challenge of matching pattern of movements. Hadjieleftheriou, et al. [58] introduced Spatio-Temporal Pattern (STP) queries to support queries such as "Find all objects that crossed a region and then stayed in other region for a certain amount of time". Our goal is to support such queries while retaining the semantic representation of STGs. An example of a query we are interested in enabling is: "Identify all the instances of a pattern defined as such: Person $P1$ and $P2$ were together in either room $R1$ or $R2$ and then moved together to room $R3$ where they stayed for at least 20 time slots, before leaving the tracked area".

We differentiate between the idea of a system *configuration* a system *state*. A configuration is simply a description of the system at a time instance whereas a state can last for a flexible amount of time. As an analogy we can consider the configuration to be a snapshot or a single frame of a video. Given the set of agents and regions, the configuration is meant to describe how the selected agents are positioned in the selected rooms at one time instance.

$C = |A, R|$ $\quad S = \langle C, D \rangle \quad \equiv \quad S = \langle A, R, D \rangle$ (Configuration vs State)



In the example above, a configuration is defined by the set A of one or more agents distributed across the set of regions R. A configuration only exists in a single time instance and, on its own, it can be rather limiting given the absence of the time component. A state is essentially the extension of a configuration which lasts for a duration D. The duration D can represent a fixed time-length such as $D = 3$ which means that the system should persist in the specified configuration for exactly 3 time slots. In the special case where $D = 1$, the state becomes semantically identical to its configuration. The duration D can also represent a time range which can be bounded or semi-bounded. Depending on the definition of D we can therefore identify three different kinds of state:

- *Fixed state*: A state with a fixed duration where $D = d$ and $d \neq \infty$.
- *Semi-Flexible States*: States with a bounded duration $D = [d_1 + d_2]$ where $d_1 < d_2 \wedge d_2 \neq \infty$
- *Fully-Flexible States*: The duration D contains ∞, such as $D = [d_1 + \infty]$. When the boundary itself is *omitted* from the state definition, we can assume it to be implicitly defined as $[1 + \infty]$.

$$S = \langle A, R \rangle \iff S = \langle A, R, [1 + \infty] \rangle$$

The + sign in the duration definition serves as a reminder, within a state, time is not stationary (as opposed to a configuration) and it keeps moving forward. These basic concepts transfer nicely to the code syntax we implemented. As optional parameters in R can be ambiguous, we purposely require both duration boundaries to be explicitly defined to differentiate between a fixed-state duration or a left-bounded one.

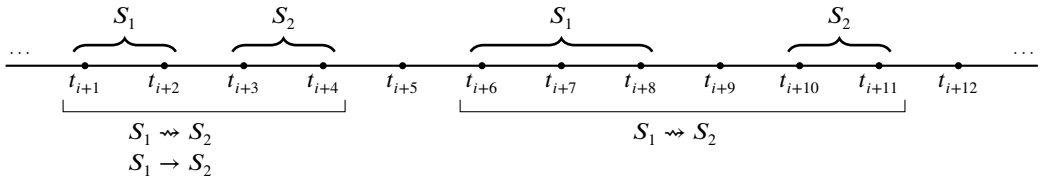
```
fixed_state       <- add_state(agents(...), rooms(...), duration(1,1))
semi_flexible_state <- add_state(agents(...), rooms(...), duration(2,5))
fully_flexible_state <- add_state(agents(...), rooms(...))
fully_flexible_state <- add_state(agents(...), rooms(...), duration(2,Inf))
```

3.3. Transitions

A single state can only represent contiguous time instances in which the system configuration has not changed. However, a movement, in its most basic definition, represents a change in one domain, usually space. Andrienko et al. defined movement as "A change in spatial position over time" [37]. This definition of movement can be limiting when considering the idea of states as a variation of any of the three system components. We refer to State Transition Graphs and the idea of a transition between states. For a transition in the system to happen, its state should change over time. Given two states S_1 and S_2, we can define the transition between them to be either direct or fuzzy:

$S_1 = \langle A, R_1 \rangle \quad S_2 = \langle A, R_2 \rangle$ (States Definitions)

$S_1 \rightarrow S_2 \quad S_1 \rightsquigarrow S_2$ (Direct Transition vs. Fuzzy Transition)



We distinguish between direct and fuzzy transitions depending on what can happen between the two states. In a direct transition the new state needs to follow the previous state directly without any time instance that does not belong to either state. A fuzzy transition however can include time intervals between the two states that do not belong to either state.

3.4. Semantic mobility patterns

In our model, patterns are a way to query the time-series and analyse it from one of the three components' perspective. As such, the definition of a pattern follows the concept of states and transitions: One or more transitions between states or even a single state can be used to define a pattern to match in the series. Defining more than one state allows us to segment the time series and place some conceptual fixed points. On a more practical level this translates to sectioning the datasets in chunks: Each chunk starts with the first state of the pattern and ends with its final state. A chunk is considered a pattern match only if the order of the intermediate states between the first and the last state is preserved. It is important to note that states that cross-fade into each other are still considered as a potential pattern match as long as none of the preceding state happens after the following one.

So far we tackled the research challenges (1) and (2) and only partially dealt with challenge (3). We can now encode a time-series in a space-focused manner and represent movements through states and transitions. We can use configurations to describe the system at a single time interval and states as a configuration lasting for more than one contiguous time interval. The introduction of transitions opens up the possibility of describing the evolution of a system at different ordered time intervals. Fuzzy transitions even allow for two sequential states to happen at non contiguous time intervals.

However, we would like to query the time-series focusing on agents and spaces as well as time. A variable number of agents or rooms would not fit the context of a time-series representation. At each time interval, the system only exists in a certain configuration. For instance, at any given moment, a person or multiple people can be in any room of a building, and even considering a system transitioning to another state, there is still only one way to describe the system. We introduce some concepts here that are unique to patterns to greatly increase the power and flexibility of our model.

3.4.1. Selection and grouping

In this section we propose a way to manipulate agents, regions and time to define more complex and flexible pattern states. Taking inspiration from the widely used regular expressions [13,14,55] we added a series of wildcards and quantifiers to the states definition to help analysts tailor their query to their needs.

Because our framework is system-agnostic, we might not know each of the components a priori. Especially when dealing with "noisy" localisation systems we might want to select a variable set of agents or rooms or define them in a way that accounts for this uncertainty. This is especially true for systems in which adjacent rooms might cause the signal to be picked up by multiple trackers, possibly changing an agent's location in different rooms in a short time span. States already include a flexible definition of time in terms of duration. Other models provide a way to clearly define the time interval in which the pattern could happen [58], and we argue that on a large scale dataset this feature could be of little to no use. In the context of data analysts using a language such as R to analyse a large time-series, we assume a pre-filtering step to restrict the time-series to a shorter time frame.

Agents. Agents can be individually selected or grouped together. For a state to be present in the data, all of its constraints as well as the set of agents must be met at each time interval:

- *Generic Selection* $*$: Any one or more agents can be in the selected rooms.
- *Set* $\{a_1, a_2, \dots, a_n\}$: At least one of the agents in the set must be in the room(s).
- *Group* $(a_1, a_2, \dots, a_n)$: All of the agents in the group must be in the room(s).

Agents set or group selection can be adjusted to the query's needs through the following modifiers:

- *Complement* $\neg$: Selects the complement of the set or group of agents. E.g $\neg\{a_1, a_2\}$ At least one agent that is not a_1 or a_2 must be in the room(s).
- *Negation* !: The agents in the set or group must *not* be in the room(s). For example $!\{a_1, a_2\}$ requires that a_1 or a_2 must not be in the room(s).
- *Numerical selection* $[n]$: Exactly n agents must be in the room(s).
- *Bounded Selection* $[n_{min} + n_{max}]$: At least n_{min} and at most n_{max} agents must be in the room. Either upper or lower bound can be omitted.
- *Partially Bounded Selection* $[n_{min}+]$ or $[+n_{max}]$: At least n_{min} agents or at most n_{max} agents, respectively.

All wildcards and modifiers can be combined together but some combinations are more useful than others. The complement and negation operations are especially useful when defining a variable set or group of agents. For instance, we might want to look for states in which agents a_1 and a_2 were in room r_1 with 3 other agents, but we want to make sure that agents a_3 was not part of the other three agents:

$$\langle \{a_1, a_2\}\ [5]\ !a_3, r_1 \rangle$$

We can obviously transition to other states defined in a similar way. For instance we could be looking for transitions between the state we just defined and a state in which a_1 was left alone in room r_1.

$$\langle \{a_1, a_2\}\ !a_3\ [5], r_1 \rangle \rightsquigarrow \langle a_1\ [1], r_1 \rangle$$

Rooms. Unlike agents and time, rooms or regions are fixed in time and space which makes their selection naturally less flexible:

- *Generic Selection* $*$: Selects any room.
- *Set* $\{r_1, r_2, \dots, r_n\}$: Selects a set of rooms. Agents can be in any of the rooms in the set but moving between them will trigger a change of state.
- *Group* $(r_1, r_2, \dots, r_n)$ Defines an abstract compound room made up of individual rooms. Agents can freely move between each room in the group and these movements would still be considered as part of the same state.

Similarly to the agents' selection, we provide modifiers for a higher level of flexibility:

- *Complement* $\neg$: Selects the complement of the set of rooms. E.g $\neg\{r_1, r_2\}$ selects any room that is not r_1 or r_2.
- *Negation* !: Turns the set or group into a negative selection. E.g $!\{r_1, r_2\}$ requires that the agents must not be in either r_1 or r_2.

Although agents can individually only be in a single room at any given time, there is no reason for a numerical selection of rooms. Statements such as "Agents a_1 in [2] rooms" or "a_1 and a_2 in $[2 + 3]$" do not bring any extra flexibility to the model, in fact they might negatively impact the syntax readability.

Duration. As we previously mentioned, configuration and states only differ in how long the system stays unchanged: a configuration can therefore be defined as a state with a duration of exactly 1 time interval.

- *Exact duration* $[d]$: The state should last for exactly d time intervals.
- *Bounded duration* $[d_{min} + d_{max}]$: The state should last for at least d_{min} time intervals and at most d_{max} time intervals
  - *Minimum duration* $[d_{min}+]$ The state should last for at least d_{min} time intervals
  - *Maximum duration* : $[+d_{max}]$ The state should last for at most d_{max} time intervals

3.5. Pattern definition

With the additional features such as quantifiers or complements, we can now craft patterns to find sections of the time-series we are interested in. A pattern can be defined as a sequence of states and the transitions between them. A single state can also represent a pattern, in fact we might be interested in extracting instances from the dataset where an agent stayed in one room without moving, without any spatial transition happening. One might assume that a multi-state pattern where the two states are identical could be interpreted the same way as a single-state pattern with that same state. However, this is not the case in our model.

$$\langle A, R, [1]\rangle \iff \langle A, R, [1]\rangle \rightarrow \langle A, R, [1]\rangle \qquad \text{(Single-State vs Multi-State patterns)}$$
$$\langle A, R, [1+2]\rangle \iff \langle A, R, [1+2]\rangle \rightarrow \langle A, R, [1+2]\rangle$$

A single-state pattern only looks for states in the datasets which means that the results only include instances of contiguous time intervals where the selected agent(s) were in the selected room(s) for the specified duration. A multi-state pattern however could represent a return-trip where an agent a left room r_1 and eventually came back to that same room r_1.

3.5.1. Strict vs. flexible patterns

States can be defined with an exact duration but also with an bounded or partially bounded duration. Moreover, transitions between states can be direct or fuzzy, adding another degree of freedom to the pattern definition. Depending on the combination of states and transitions, we can see patterns as strict or flexible with respect to time. This is easy to see for single-state patterns but it can be tricky for multi-state patterns.

A multi-state pattern is considered as strict when all of its states have an exact duration and when all the transitions are direct. Such a definition translates to a strict pattern description such as "Instances where agent a_1 was in room r_1 for [20] time slots and then immediately moved to room r_2 and stayed there for [10] time slots".

This type of pattern is arguably very limited especially in the context of semantic and indoor localisation where data can be noisy and so can be the location of agents at any given time. An agent might appear in room r_1 at a time and then appear in r_2 for a single time slot before reappearing in r_1. While it is possible for this situation to be meaningful in particular scenarios, it is not the case for datasets with a high time granularity or large spaces.

A multi-state pattern is considered as flexible when at least one of its states has a variable duration (either partially bounded, fully bounded or even unbounded) and/or when at least one of its transitions is fuzzy. A flexible multi-state pattern can last for a flexible amount of time. If all the states are variable but bounded and all of its transitions are direct, then its maximum duration is simply the sum of maximum durations of each state. If even one state is unbounded and/or one of the transitions is fuzzy, the pattern could potentially include the whole dataset, especially if the state definition is loose.

$$\langle A, R, [3]\rangle \qquad \text{(Strict Single-State Pattern)}$$
$$\langle A, R, [2+10]\rangle \qquad \text{(Flexible Single-State Pattern)}$$
$$\langle A, R_1, [2]\rangle \rightarrow \langle A, R_2, [4]\rangle \rightarrow \langle A, R_3, [1]\rangle \qquad \text{(Strict Multi-State Pattern)}$$
$$\langle A, R_1\rangle \rightarrow \langle A, R_2, [3+10]\rangle \qquad \text{(Flexible Multi-State Pattern (direct))}$$
$$\langle A, R_1\rangle \rightsquigarrow \langle A, R_2, [10]\rangle \qquad \text{(Flexible Multi-State Pattern (fuzzy))}$$

3.5.2. Combinations and implementation

While strict single-state patterns might not be very useful, there are situations in which they might be needed such as looking for instances in which an agent was in a room for exactly twenty minutes. However, in practice the concept of time is relative even in the context of movements or staying in one space. As the time granularity increases, the power of duration expressed as a set amount of time intervals diminishes. As a simple example: an agent staying in a room for 15 s instead of 10 s would not be considered as a matching pattern.

On the other end of the spectrum, flexible multi-state patterns can be extremely flexible and are arguably the most useful. These types of patterns can cover a wide range of movements in the time-series. The definition of a pattern begins with the definition of at least one state in a trace:

```
strict_single_state <- start(time_series) %>%
   add_state(agents(...), rooms(...), duration(2,2)) %>% end()
strict_multi_state <- start(time_series) %>%
   add_state(agents(...), rooms(...), duration(2,2)) %->%
   add_state(agents(...), rooms(...), duration(1,1)) %>% end()
variable_single_state <- start(time_series) %>%
   add_state(agents(...), rooms(...), duration(1,7)) %>% end()
variable_multi_state_direct <- start(time_series) %>%
   add_state(agents(...), rooms(...)) %->%
   add_state(agents(...), rooms(...), duration(1,5)) %>% end()
variable_multi_state_fuzzy <- start(time_series) %>%
   add_state(agents(...), rooms(...), duration(2,2) %~>%
   add_state(agents(...), rooms(...), duration(1,1) %>% end()
```

Table 3
An example of a regularised and aggregated indoor localisation dataset (a) and the same dataset converted into the required structure (b).

(a)

timeSlot	Id	Room	timeElapsed
2023-01-01 10:51:19	B1	Office1	10 s
2023-01-01 10:51:19	B3	Room2	10 s
2023-01-01 10:51:29	B1	Office1	10 s
2023-01-01 10:51:29	B3	Office1	10 s
2023-01-01 10:51:29	B2	Room2	10 s
2023-01-01 10:51:39	B1	Office1	10 s
2023-01-01 10:51:39	B2	Office1	10 s
2023-01-01 10:51:39	B3	Office1	10 s
2023-01-01 10:51:49	B2	Office1	10 s
2023-01-01 10:51:49	B3	Room2	10 s
2023-01-01 10:51:49	B1	Room2	10 s

(b)

timeSlot	Office1	Room2
2023-09-01 10:51:19	B1	B3
2023-09-01 10:51:29	B1, B3	B2
2023-09-01 10:51:39	B1, B2, B3	
2023-09-01 10:51:49	B2	B1, B3

Thanks to features like quasi-quotation and lazy evaluation of the R language, the code syntax closely resembles the formal definition of the model. However, for optimisation purposes that we will discuss later we use functions such as `start(...)` and `end(...)` to initialise the dataset and store the necessary metadata.

The implemented package works under the assumption that the data is structured in a way that follows the symbolic representation we presented. Generally speaking this means that the dataset should be in a *location-focused* structure as opposed to an *agent-focused* structure. In practice this translates to a fixed set of spaces, or labels. At any given time, zero or more agents can be in only one of these rooms that is, a single cell in the dataset contains a list of zero or more agents identifiers. This way of structuring the data allows a more flexible selection of agents since it is not necessary to know the identifiers of agents a priori.

While the set of symbols or labels assigned to each space is assumed to be immutable, the actual physical location they identify is not required to be fixed. This is an innate property of semantic localisation and symbolic representations that abstract over spatial coordinates. This choice of structure is mainly justified by indoor localisation datasets often being deployed in buildings or rooms that are unlikely to change over time. For instance, a room labelled as *Office* could initially reference a room on the ground floor but as the team moves into a new office on the fifth floor, the same label can now point to a different physical location while maintaining the same semantic label.

Furthermore, as a query effectively returns a list of sections or chunks of the time-series where the pattern happened, it is indeed possible to run more granular pattern queries on each of the resulting chunks.

4. Case studies

We now present a number of case studies to demonstrate how our tool can be used to query localisation datasets programmatically. Before we jump into the examples, we note that a requirement of our tool is for the time series to be in a *location-focused* and for the time to be regularised. While we understand that real-world data is not regular with respect to time, an irregular time series could produce misleading results. We propose a simple but effective solution to regularise the dataset by using a set of heuristics and some simple filtering. In the table below we provide an example of the data format the tool is designed to work with.

Once our tool loads the data such as the one shown in Table 3a, it will then transform it into the format shown in Table 3b. The is the format which the tool uses internally. In practice, the main purpose of the tool is to find regions of this table which match the given queries.

4.1. Basic examples

To demonstrate the capabilities and flexibility of our framework and implementation we provide examples of common queries we expect data analysts to use. In these examples we assumes a list of undefined length of agents $\{A_0, A_1, \dots, A_n\}$ and a list of fixed rooms $\{R_0, R_1, \dots, R_n\}$.

Agent tracking. Let us suppose we are interested in following the agent A_1 and simply getting the list of rooms they have visited throughout the time series. As the scenario does not include any specific transition, we can just express this as a single flexible state:

$$\langle A_1, * \rangle \qquad \text{(Agent Tracking Pattern)}$$

```
agent_journey <- start(data) %>%
  add_state(agents(set(a1)), rooms(set(*))) %>% end()
```

Simple transition. The next step is to find all the instances in which the agent A_1 moved from room R_1 to room R_2. We can express the transition as fuzzy or direct depending on the purpose of the query. A fuzzy transition would also consider sections of the time-series in which the agent visited other rooms before ending in R_2, while the direct transition would omit these cases.

$$\langle A_1, R_1\rangle \rightarrow \langle A_1, R_2\rangle \qquad \langle A_1, R_1\rangle \rightsquigarrow \langle A_1, R_2\rangle \qquad \text{(Direct vs. Fuzzy Transition)}$$

```
fuzzy_transition <- start(data) %>%
   add_state(agents(set(a1)), rooms(set(r1))) %~>%
   add_state(agents(set(a1)), rooms(set(r2))) %>% end()
direct_transition <- start(data) %>%
   add_state(agents(set(a1)), rooms(set(r1))) %->%
   add_state(agents(set(a1)), rooms(set(r2))) %>% end()
```

Finding gaps. As previously noted, indoor localisation datasets are especially prone to noisy data. While fuzzy transitions on their own help in dealing with an agent's location changing rapidly and unexpectedly throughout the time-series, we might be more interested in the gaps themselves, perhaps to identify issues in the system's deployment. We can look for gaps in the dataset with a simple query looking at rooms with no agents in them. We can also specify the time duration to specifically look for larger gaps - e.g. A room being empty for more than 10 time slots:

$$\langle [0], * \rangle \qquad \text{(Any empty room at any point)}$$

$$\langle [1+], R_1\rangle \rightarrow \langle [0], R_1, [10+]\rangle \rightarrow \langle [1+], R_1\rangle \qquad \text{(Find gaps in room } R_1 \text{ specifically)}$$

```
dataset_gaps <- start(data) %>%
   add_state(agents(range(1,Inf)), rooms(set(r1))) %->%
   add_state(agents(range(0,0)), rooms(set(r1)), duration(10,Inf)) %->%
   add_state(agents(range(1,Inf)), rooms(set(r1))) %>% end()
```

Groups and transitions. A more complex example is to include a group of agents and a transition to a state with different agents or rooms. A real-world scenario could be instances in which a classroom initially only contains the lecturer, eventually fills up with students, and finally it becomes empty. It should be noted that the last transition can be either fuzzy or direct: Since the set of rooms of interest remains the same, and the penultimate transition includes a variable amount of agents in the room and at least one agent, there would not be a case in which the room becomes empty in between. An empty room in between would not be included in neither the first nor the second state, effectively breaking up the trace.

$$\langle A_1[1], R_1\rangle \rightsquigarrow \langle A_1[1+], R_1\rangle \rightarrow \langle [0], R_1\rangle$$

```
group_transition <- start(data) %>%
   add_state(agents(set(a1), range(1,1)), rooms(set(r1))) %~>%
   add_state(agents(set(a1), range(1,Inf)), rooms(set(r1))) %->%
   add_state(agents(range(0,0)), rooms(set(r1))) %>% end()
```

Multiple complex transitions. Combining the previous examples we could now try to define a more complex pattern that includes some wildcards as well. For instance, we might be interested in following agent A_1 and maybe check when they joined another group of agents. The pattern we are interested in can be explained as:

1. Agent A_1 was alone in one of the rooms $\{R_1, R_2\}$
2. Agent A_1 left the room and, eventually, a new group of at least two agents came in
3. Agent A_1 eventually joined the group
4. Agent A_1 eventually left and moved to another room on their own

$$\langle A_1[1], \{R_1, R_2\}\rangle \rightsquigarrow \langle !\{A_1\}[2+], \{R_1, R_2\}\rangle \rightarrow \langle A_1[3+], \{R_1, R_2\}\rangle \rightarrow \langle A_1[1], !\{R_1, R_2\}\rangle$$

It is important to note the last two transitions being direct transitions. The direct transition would only allow for a situation in which a group of at least 2 agents without A_1 was in the set of rooms and A_1 immediately joined the room in the next time interval. In simpler terms, it does not matter whether the group of agents is different, as long as the group contains at least two agents and none of them is A_1.

```
group_transition <- start(data) %>%
   add_state(agents(set(a1), range(1,1)), rooms(set(r1, r2))) %~>%
   add_state(agents(!set(a1), range(2,Inf)), rooms(set(r1, r2))) %->%
   add_state(agents(set(a1), range(3,Inf)), rooms(set(r1, r2))) %->%
   add_state(agents(set(a1), range(1,1)), rooms(!set(r1, r2))) %>% end()
```

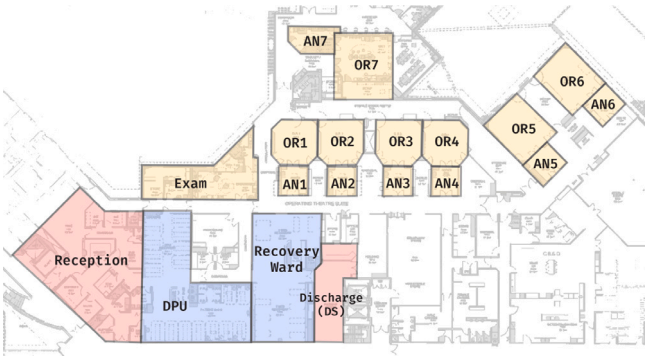


Fig. 2. Map of a hospital ward with operating rooms, anaesthetic rooms, reception, DPU and recovery ward.

4.1.1. Hospital scenario

Hospitals are known to be time-critical workplaces where inefficiencies and delays can potentially have serious consequences. Indoor tracking is not new in health care [62,63] but the real-life applications often do not go beyond the data collection phase or real-time location tracking. With our model's syntax and concise approach it is possible to leverage the richness of information hidden in large datasets of indoor traces.

The following examples are based on real-world data collected by a Bluetooth Low Energy (BLE) indoor localisation system [26]. The floor plan in Fig. 2 shows the general structure of the hospital. We will focus on three main scenarios with some variants: room occupancy, contact tracing and patients flow tracking.

Operating rooms occupancy and utilisation. Although room occupancy does not inherently include movement traces, the flexibility of the tool allows us to use the flow of people, and the absence of them, through rooms to calculate their occupancy rate. One instance in which this could be useful is to extract rooms utilisation data for operating rooms as well as patient wards, assuming that all staff and patients are tracked. To show the flexibility of our tool, we present four different ways of looking at room occupancy and utilisation. The simplest way to calculate the occupancy rate looks at a binary empty or non-empty status of the rooms. However we could also define some threshold value n_u such that a number of agent in a room above the threshold translates to the room effectively being utilised. Given a set of operating rooms sequentially numbered (e.g OR_1, OR_2, OR_3, etc.) and an arbitrary number of people present in the room, we can then find the time ranges in which an operating room was being utilised. For the sake of simplicity we define the list of operating rooms as the result of a selection based off regular expressions $ORs = /OR*/$.

$$\langle [0], \{ORs\}\rangle \qquad \text{(Empty ORs)}$$

$$\langle [1+], \{ORs\}\rangle \qquad \text{(Non-empty ORs)}$$

$$\langle [0], \{ORs\}\rangle \rightsquigarrow \langle [n_u+], \{ORs\}\rangle \qquad \text{(Start of ORs utilisation)}$$

$$\langle [n_u+], \{ORs\}\rangle \rightsquigarrow \langle [0], \{ORs\}\rangle \qquad \text{(End of ORs utilisation)}$$

The two patterns to detect simple room occupancy only need one state, in fact a room can either be empty or non-empty at any given time, and occupancy does not include any transition by definition. The two states that focus on utilisation instead do require a transition, more specifically they require a *fuzzy* transition. The requirement for a fuzzy transition comes from the fact that in a real-world scenario, a room does not suddenly become empty but rather people leave the room one after another. To capture the exact scenario in which agents left the room one by one, we would need a much longer chain of direct transitions:

$$\langle [0], \{ORs\}\rangle \rightarrow \langle [1], \{ORs\}\rangle \rightarrow \cdots \rightarrow \langle [n_u], \{ORs\}\rangle \qquad \text{(ORs gradually filling up)}$$

$$\langle [n_u], \{ORs\}\rangle \rightarrow \langle [(n_u - 1)], \{ORs\}\rangle \rightarrow \cdots \rightarrow \langle [0], \{ORs\}\rangle \qquad \text{(ORs gradually getting empty)}$$

The R code to extract the instances in which ORs were not empty and then calculate the occupancy rate of all ORs is as follows:

```
ORs_non_empty <- start(data) %>%
  add_state(agents(range(1,Inf)), rooms(set(ORs))) %>% end()
ORs_occ_rate <- nrow(ORs_non_empty[[-1]]) / nrow(data)
```

The code above will first run the query and obtain a list of results containing the instances in which the state happened continuously. The short hand `OR_Occupancy[[-1]]` returns a single data-set merging the list of results. Taking the ratio between the time slots in which the operating rooms were not empty and the total amount of time slots returns the occupancy rate for all of the operating rooms. It should be noted that this particular type of query returns the occupancy rate for all of the ORs. Defining the list of operating rooms as either a set or a group does not affect the result due to the lack of state transitions.

Table 4
A tabular view of the output of a query looking at encounters of two agents for contact tracing. Each row represents a different possible encounter of A_x with other agents. Each encounter can happen multiple times and last a variable amount of time intervals.

Contact	Encounters	max_duration	total_duration
$A_x + A_1$	10	4	34
$A_x + A_2$	1	12	12
$A_x + A_3$	4	2	56
$A_x + A_4$	21	10	76
$A_x + A_5$	13	7	41

Contact tracing 1. The contact tracing scenarios are largely based on the need of hospitals of tracking the spreading of viruses and potential contaminating agents across the ward. This type of query became especially relevant in light of the COVID-19 pandemic presenting a serious threat to the global population since 2019. We provide three different examples of possible contact tracing queries.

While in the previous example the selection of agents was only numerically bounded, in this example we assume that at least one agent A_x has to be in the selected room(s). We are interested in extracting situations in which the fixed A_x agent has come in contact with any other agent, i.e. we are looking for any other agent being present in the same room as A_x at any given time.

$$\langle A_x[2+], * \rangle$$

The R code translation of the above expression is as follows:

```
contact_tracing1 <- start(data) %>%
   add_state(agents(set(ax), range(2,Inf)), rooms(set(.any))) %>% end()
```

While it would be optimal to use the same character $*$ as the general selector wildcard in the R code, we note that this could cause confusion when used together with the multiplication operation in R, which uses the same character. We opted instead for the special keyword `.any` to represent the generic selector. While we could assume a generic selection when the function `set()` is called without any parameter, we chose not to do so for optimisation purposes and to avoid ambiguity in the code. Executing a query on a very large dataset might take a very long time to complete, it is not uncommon for any programmer or data analysts to forget a variable and more commonly, to forget an unbalanced bracket. Therefore we opted for an explicit way to select *all* rooms, in an attempt to ensure that the data analysts are indeed defining a pattern over the whole dataset. We also show here an example of R code to further process the output from the previous step and extract a summary of all encounters of A_x with other agents. Table 4 shows an example of how a structured summary of contact tracing could look like:

```
# Code to tabulate the results
found_agents <- contact_tracing1$result$agents
summary <- lapply(found_agents[found_agents != "a1"], function(x){
   contacts <- contact_tracing1[c("ax", x)]
   res <- list(contact=paste0("$A_x + ", x))
   contact_times <- !is.na(contacts$rooms.with.agents)
   consecutive <- rle(contact_times)
   c.true <- consecutive$lengths[which(consecutive$values == TRUE)]
   res["encounters"] <- length(c.true)
   res["max_duration"] <- max(c.true)
   res["total_duration"] <- nrow(contacts[contact_times])
   return(res)
 })
summary_table <- do.call(rbind, lapply(summary, data.frame))
```

Contact tracing 2. The decision of grouping rooms comes from the need for more complex queries where agents can be in any room in a set and move to a different set of rooms. Let us take the example above and only look for transitions in which the agents moved from an operating room to any other room.

$$\langle A_x, \{OR1, OR2, OR3\} \rangle \rightsquigarrow \langle A_x, !\{OR1, OR2, OR3\} \rangle$$

In this case, given the selected combination of agents, we are looking for any transition between any of the three ORs and any another room that is not one of the three ORs. Note that in this case the curly brackets make more sense as we do not want to consider the three rooms as a single compound space. Instead, we would like and individual result for each of the rooms. However, it should be noted that this query does not cover transitions within the set such as $OR_1 \rightsquigarrow OR2$. The output in this case would be a list of results, one for each combination of agents together with A_x such as $(A_1 + A_x)$ or $(A_2 + A_x)$. Each result covers the transition of the two agents to another room that was not one of the three ORs.

```
ContactTracing2 <- start(data) %>%
   add_state(agents(set(a1), range(2,Inf)), rooms(set(OR1, OR2, OR3))) %>%
   add_state(agents(set(a1), range(2,Inf)), rooms(~set(OR1, OR2, OR3))) %>% end()
```

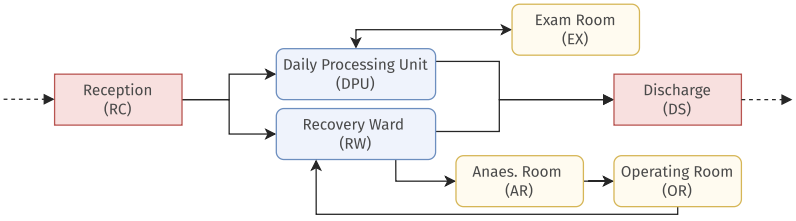


Fig. 3. Flow graph representing the rooms that a patient might go through after passing through the reception and before being discharged.

Contact tracing 3. Finally, as one last contact tracing example, we would like to extract every instance in which an agent was initially in a room either alone or with other agents, then moved to another room where A_x was present and subsequently left to another room where at least another agent was present and they were not A_x.

$$\langle !A_x, \{*\}\rangle \rightsquigarrow \langle A_x[2+], \{*\}\rangle \rightsquigarrow \langle !A_x[2+], \{*\}\rangle$$

In this scenario, the initial state selects any agent who is in any room either alone or with other agents as long as none of these other agents are A_x. It is not necessary to specify the amount of agents in this case but one could also rewrite the state as $\langle !A_x[1+], *\rangle$ for the sake of clarity. If, for instance, an agent A_1 moved to a room where A_x was present and subsequently moved to another room where A_x was not present. On a side note, we would like to point out that the parentheses {} serve a specific purpose. Including the generic selection wildcard $\{*\}$ in a set ensures that each room in the set is treated individually as opposed to defining a compound space by using the group square brackets [].

```
ContactTracing3 <- start(data) %>%
   add_state(agents(!set(ax)), rooms(set(.any))) %~>%
   add_state(agents(set(ax), range(2,Inf)), rooms(set(.any))) %>%
   add_state(agents(!set(ax), range(1,Inf)), rooms(!set(.any))) %>% end()
```

Patient tracking patterns. Hospitals are critical environments where time management is essential and can have serious consequences. Most hospitals have a well defined flow for each type of patient. Tracking the patient's progress is a task often performed manually by people either with the help of a specialised software or even by hand writing time-sheets. The dataset we are focusing on [26] included the movement traces of patients in the operating ward. Let us consider two different journey that a patient might go through: A daily case where the patient is scheduled for an exam and gets discharged on the same day, and a longer case where the patient is scheduled for a surgery and recovery before discharge. Fig. 3 and Fig. 4) show the possible flows of a patient through the hospital word and an example of how two patients might go throgh these steps, visiting different rooms. We can summarise the flow and the rooms visited in the following steps:

1. Patient gets admitted through Reception (RC) at the Emergency Department (ER)
2. Patients can go to either the DPU (Daily Processing Unit) or to the Recovery Ward (RW)
    - Patients in the Recovery Ward and scheduled for surgery will go through an Anaesthetic Room (AR) before entering the Operating Room (OR)
    - Patients in the DPU can be scheduled for a specific exam such as an endoscopy and will move to the Exam Room (EX)
3. Patients go back to either DPU or RW
4. Once they are approved for Discharge (DS), patients can leave the wards and the hospital

Both journeys are of interest to the hospital which wants to study particular scenarios for quality improvements, resource management, or surgery and staff scheduling. The pattern of a day case can be expressed as such:

$$\langle P, RC\rangle \rightsquigarrow \langle P, DPU\rangle \rightsquigarrow \langle P, EX\rangle \rightsquigarrow \langle P, DPU\rangle \rightsquigarrow \langle P, DS\rangle \qquad \text{(Day exams)}$$

```
day_cases <- start(data) %>%
   add_state(agents(set(P)), rooms(set(RC))) %~>%
   add_state(agents(set(P)), rooms(set(DPU))) %~>%
   add_state(agents(set(P)), rooms(set(EX))) %~>%
   add_state(agents(set(P)), rooms(set(DPU))) %~>%
   add_state(agents(set(P)), rooms(set(DS))) %>% end()
```

Let us suppose that the hospital is looking at cases where the patient was left alone in the DPU before the exam. We can use a combination of complement and negation. In this case we are looking for a situation in which the DPU only had patients (including the patient being tracked) and only patients. We can thus select the complement of all the patients, which we denote as P_{all} and negate the resulting set.

$$\langle P, RC\rangle \rightsquigarrow \langle P\ !\neg\{P_{all}\}, DPU\rangle \qquad \text{(Patients alone in DPU)}$$

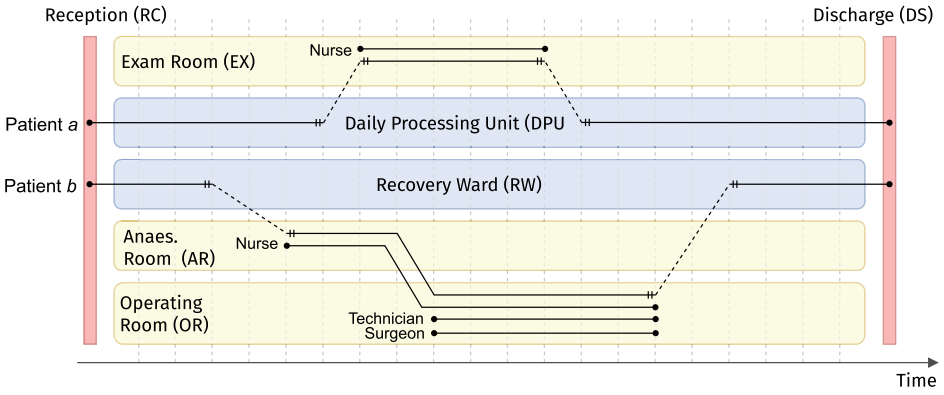


Fig. 4. A graphical visualisation of the journey of two patients. Patient *a* is scheduled for a daily exam, a nurse appears in the exam room together with the patient for the duration of the exam. Patient **b** is scheduled for a surgery and therefore goes through the anaesthetic room, then to the surgery room and finally to the recovery ward before discharge.

```
patients_alone <- start(data) %>%
   add_state(agents(set(P)), rooms(set(RC))) %~>%
   add_state(agents(set(P), !~set(patients)), rooms(set(DPU))) %>% end()
```

Surgery cases tend to be naturally more complex. The surgery itself can last for hours and the patient (P) stay spans over multiple days. The transition from AR to OR should also be considered as a direct, non-fuzzy transition. This is due to the fact that usually ARs are usually located just besides ORs. More importantly, anaesthesia only lasts for a certain amount of time so the patient should immediately move to the OR after going under anaesthesia, if that is not the case then the surgery most likely did not take place and that should be treated as a special case.

$$\langle P, RC\rangle \rightsquigarrow \langle P, RW\rangle \rightsquigarrow \langle P, \{ARs\}\rangle \rightarrow \langle P, \{ORs\}\rangle \rightsquigarrow \langle P, DS\rangle \qquad \text{(Surgeries)}$$

We point out that ARs and ORs are symbolic placeholders referring to all Anaesthetic Rooms and all Operating Rooms respectively. Therefore $\{ARs\}$ and $\{ORs\}$ denote the rooms selected all together as a set rather than a group. In practice, this is a shorthand for a label selection through regular expressions, similar to what we have previously shown as {/OR*/}.

We can augment the pattern by also tracking the length of the surgery and only looking for surgeries which went overtime. We can assume that a surgery is taking place only when all required staff is present in the room together with the patient. For this example let us require the presence of at least a surgeon (S), anaesthetist (A) and a nurse (N) together with the patient (P). It is important to note the use of a group (with square brackets) rather than a set: All of the listed agents must be in the room. We denote the expected duration of the surgery as d_{exp}:

$$\langle P, \{ARs\}\rangle \rightarrow \langle [P, S, A, N][4{+}6], \{ORs\}, [d_{exp}{+}]\rangle \qquad \text{(Overtime surgeries)}$$

Finally, we could try and extract the instances in which more people enter the operating room possibly due to a machine's technical fault requiring another technician or in the worst case scenario due to a surgery taking an unfortunate turn of events and requiring more personnel.

$$\langle P, \{ARs\}\rangle \rightarrow \langle [P, S, A, N][4{+}6], \{ORs\}\rangle \rightsquigarrow \langle [P, S, A, N][7{+}], \{ORs\}\rangle \qquad \text{(Emergency during surgeries)}$$

We can easily translate the syntax to R code and even reuse some sections of the code. For instance, given the same initial dataset, we know that the first three state where the patient moves from RC to any AR are common to all surgeries, while the special cases of surgery different from this point in time onwards.

```
to_AR <- start(data) %>%
   add_state(agents(set(P)), rooms(set(RC))) %~>%
   add_state(agents(set(P)), rooms(set(RW))) %~>%
   add_state(agents(set(P)), rooms(set(ARs)))

extended_OR <- to_AR %->%
   add_state(agents(set(P,S,A,N), range(4,6)), rooms(set(ORs)), duration(expected+1, Inf) %>% end()

emergency_OR <- to_AR %->%
   add_state(agents(set(P,S,A,N), range(4,6)), rooms(set(ORs))) %~>%
   add_state(agents(set(P,S,A,N), range(7,Inf)), rooms(set(ORs))) %>% end()
```

5. Discussion

In this paper we present a tool to analyse localisation traces in a systematic and formal manner. While there has been work in the past that looked at a more formal way to analyse time series in the field of localisation [37,51], we argue that many of these approaches are either not easy to use for non experts or they lack an implementation or at least examples of real-world applications. Finally, many of them are actually outdated and use tools or languages that are not as common anymore at the time of writing or were not developed for the purpose. Languages such as C or C++ were not originally developed with data analysis in mind or with the data structures we commonly find in more apt tools such as data-frames. Languages such as SQL can be extended to better deal with time-series, but as a database management and access language it works better when paired with more versatile tools such as R or Python and their packages or libraries.

Our tool is based on the idea of semantic representation where each space is not represented in terms of their geometrical shape or coordinates but rather in terms of their functionality and purpose. This tool is especially suited for dealing with buildings or workspaces where the purpose of each area is well defined and unlikely to change over time. A semantic representation closely resembles a symbolic representation [49] or a so called Bags-Of-Patterns-Representation (BOTS) [64] where any discrete time interval can be represented by a symbol, and multiple symbols grouped together represent a pattern. Representing a time series as a series of symbols can allow analysts to exploit the power of regular expressions [55] to recognise a list of patterns in the time series.

However, regular expressions have crucial limitations both conceptually and in terms of implementation and usability. First and foremost, regular expressions are known to be hard to read and especially hard to formulate, it is often not immediately clear what the pattern is describing and whether the resulting matches are correct or not. Another conceptual limitation comes from the uncertainty of localisation datasets: Localisation systems often have to adapt to their environment which often means not knowing a priori which agents appear in the datasets. This limitation is especially problematic when the number of agents exceeds the available characters and would therefore require more than one character to represent different agents.

On a more practical level, regular expressions are known to be quite slow for large strings of text especially ones with repetitions. Finally, disregarding the aforementioned issues, regular expressions would work well on a single time series, which in this scenario would be a single room's time-series. The list of matches returned by applying a regular expression pattern on a time series contains the start and end position of the matched tokens, however, comparing the result to the resulting matches from other room's time series quickly grows in complexity.

5.1. Contributions

Our work set out to address 4 research challenges that we identified in our survey of the related work. We started by symbolically representing the time-series as a semantic localisation dataset. The choice of focusing on spaces and more specifically on rooms allowed us to address the first challenge (1) while also setting the foundations for a flexible representation of movements.

We then proposed a way to describe the semantic configuration of the system in terms of the location of agents at any given time. We extended the concept of an instant configuration to allow for a fine-tuned control over time. By introducing direct and fuzzy transitions we provided a wide variety of ways to describe the semantic state of the system and how it evolves over time. The evolution of the system defined as a sequence of states ensures the ordering of events while keeping the flexibility of variable states duration as well as non-direct transitions (2).

By seeing patterns as an extension to the system's definition we provided, we leveraged the power and flexibility of the state-transition model. As patterns serve a wider purpose than a simple representation, we set off to expand states and transitions to account for the variability and uncertainty of real world scenarios and indoor localisation datasets.

In summary, a pattern is an extension of the flexible system representation of states and transitions (both direct and fuzzy) with the addition of variable agents and rooms selection. With these extensions, our pattern definition effectively also allows for the tracking of groups of variable size of known or unknown agents or a combination of them (3). We have showed the power of our model by applying the syntax to both hypothetical and real-world scenarios. We also provided practical examples on how the implementation of our tool in the R language can be used to directly convert the syntax into code. Some of the examples go beyond the mere syntax conversion and also show how the output of our pattern detection system could be used to calculate meaningful insights and summaries (3).

5.2. Implementation

In regards to our tool's implementation, the R language is widely used to analyse any kind of data and the open-source community offers plethora of packages to analyse time-series specifically. We argue that even with packages such as `data.table`, `dplyr`, `zoo`, `xts`, analysing a multi-dimensional time-series can prove to be a difficult task even for experienced data analysts.

Thanks to the flexibility of R and mechanisms such as quasi-quotation and lazy evaluation, the resulting implemented syntax closely the formal definition of our model. The simplicity of our code's syntax hides many important implementation details, some of which are worth explaining. Running complex queries on large datasets often involves, at the lowest level, iterating over the whole series over and over, at least once per room. Furthermore, most localisation datasets from workplaces are sparse by nature as they follow the typical working hours of the workplace, which would inevitably include large spans of time with few to no workers in the workplace.

We exploit the sparsity in localisation datasets to optimise multi-room queries: By first calculating the occupancy rate for each room across the time-series, we can run queries starting from the room with the highest occupancy rate, reducing the dimensionality of the data to be checked at each steps. From our tests, it is very likely that when checking all rooms in a dataset, by the time the last one is to be checked, all of the time-slots will have been excluded due to their occupancy rate being very low and due to the state already capturing most of the remaining time-slots.

Another optimisation we applied is a pre-processing for multi-state queries. When two or more states are defined in the pattern, we exploit the order of states to reduce the datasets to only consider the time-slots contained between the initial and the final state. Furthermore, under the same assumption, we can seek for intermediate transitional states only in the time-slots appearing between pairs of consequent initial and final states.

5.3. Formalism and flexibility

It is true that the results achieved by our model and its implementation are also attainable through libraries or packages in different languages and environments. Other languages such as Python also offer a variety of packages for analysing time-series. However, we argue that none of these packages have been developed for the concepts of movements, patterns and semantic representation of entities. Similar results can be achieved through a convoluted and lengthy process which often includes using tools not meant or even fit for the purpose. It is not uncommon for data analysts and programmers to lose track of the main goal of their analysis while delving into the technicalities of the implementation. Programmers and analysts alike are also often faced with the challenge of preserving a high level description of the analysis to share with their non-technical peers.

While data analysts, localisation experts or experienced GIS scientists are more likely to solely use our implemented package, having a formal definition of the model can set the foundation for a more structured reasoning as well as a consistent implementation in other languages. We argue that the power of the tool is in the structure of the reasoning and its flexibility. Furthermore, a language-agnostic representation would make code portability and documentation writing a less cumbersome endeavor. Formally defining a pattern and thinking of a localisation dataset in terms of semantically meaningful spaces, agents moving between them and a sequence of states would ensure a consistent input to output relationship in any implementation. On a logistic level, patterns defined through natural language can easily be expressed through the syntax we provide, potentially improving the communication between teams with different expertise. The real-world examples in the hospital scenario we provided clearly show how a fairly complex query can be represented in a simple and concise way and how the implementation follows the pattern definition.

Defining spaces and regions according to their meaning and purpose means that this tool could work on different domains and on systems where the spatial configuration is not fixed. One such example could be the analysis of eye-tracking movements or generally user behaviour with a User Interface. A UI is often made up of parts that serve a specific purpose such as buttons, input fields, titles, menu options. However, a UI is very likely to change over time: buttons can change location or buttons in the same location can have a different purpose, titles and menu options change too according to the context. Identifying a region of the screen through a meaningful label adds a layer of abstraction that does not deal with coordinates, which effectively means that the region it represents can change without affecting the syntax or the pattern definition. As previously mentioned in regards to an STG representation, the lack of spatial information is not necessarily a weakness. As explain by Andrienko “A comprehensive analysis of movement requires that the abstract STG representation is used together with a representation containing specific spatial information, so that links between the semantic and spatial aspects can be established” [39].

As an example, one could measure the ambiguity of a UI by searching eye-tracking and use logs for instances where the user performed the wrong action such as clicking the wrong button or searching through different menus. We can refer to the user gaze simply as U and label the UI components as such:

- OK: An “Ok” confirmation button.
- NO: A no/cancel button.
- SB: A search button.
- IN: An input field.

We can then define the expected user behaviour as:

$$\langle U, SB\rangle \rightsquigarrow \langle U, IN\rangle \rightsquigarrow \langle U,\ \{OK,\ NO\}\rangle \qquad \text{(Expected user behaviour)}$$

We can now look for deviations from these patterns, and even specify where the deviation could happen:

$$\langle U, SB\rangle \rightsquigarrow \langle U,\ !\{IN, OK, NO\}\rangle \qquad \text{(Unexpected user behaviour: Popup could not be found)}$$

$$\langle U, SB\rangle \rightsquigarrow \langle U, IN\rangle \rightsquigarrow \langle U,\ !\{OK, NO\}\rangle \qquad \text{(Unexpected user behaviour: Buttons could not be found)}$$

5.4. Limitations

Our approach however, still presents some limitations. First, the tool has only been tested on discrete regular time-series. Applying this syntax to irregular time-series could result in an unexpected output. Expanding the tool to work with such irregularities can be challenging, a state only existing in a single time-slot might last for an unknown amount of time, effectively making it harder to include other states in the pattern or to even read the resulting traces.

Another limitation comes from the flexibility of the tool itself: There are instance in which it could be more appropriate to define a variable state including a group or set of rooms, and defining the following states according to the intermediate result. For instance, sometimes we do not know where the agent has been, but after finding out their whereabouts in one state, we would like to have the following states focusing on that space instead.

Similarly, the toolkit lacks the possibility to refer to other states or compare them. In the example regarding surgeries taking place, it could be useful to refer to other states and compare them. For instance we might want to only select instances in which more people entered a room without having to clearly define how many people are expected in the preceding state. This powerful feature would allow expressions such as the comparison between two groups of agents e.g. "This state should have a number N of arbitrary agents, the next state should have the same amount but the arbitrary agents are all different from the previous state"

6. Conclusions

In this paper we presented a novel representation of indoor localisation systems based on the idea of semantic localisation. By representing spaces as their purpose in the context of their environment such a workplace or a hospital, we shifted the focus of localisation to the movement of people and their behaviour. We define movements through three main components: agents, time and space. With these concepts we symbolically encoded localisation datasets and provided a flexible syntax to describe the semantic configuration of the system at any given time. We applied the well established concept of states and transitions to describe the evolution of the system over time by keeping the duration of states and the time between them as flexible as possible. Finally, we expanded the syntax to allow the definition of patterns that use the same system description in terms of states and transitions.

Additionally, the pattern definition we provide is flexible in all three components of space, time and objects and does not require any *a priori* knowledge of either of them. The theoretical definition of the model is easily translated to code and we show that by providing an implementation of the model in the R language. The case scenarios we analysed include the query definitions as a patterns but they also include a practical usage example of our implementation. In conclusion, we set off to attempt at bridging the gap between the technical aspect of localisation and the more qualitative part of it.

We believe that such a theoretical model, accompanied by an actual implementation in a widely used programming language will prove useful in helping data scientists and programmers alike in dealing with large localisation datasets. The real-world examples clearly show how a localisation datasets can be directly shaped into a semantic representation of spaces and how these transformed datasets can be easily queried and explored through a flexible pattern definition that easily translates into usable efficient code.

CRediT authorship contribution statement

Gabriele Marini: Validation, Software, Methodology, Investigation, Conceptualization, Visualization, Writing – original draft, Writing – review & editing. **Jorge Goncalves:** Supervision, Writing – review & editing. **Eduardo Velloso:** Supervision, Writing – review & editing. **Raja Jurdak:** Writing – review & editing, Supervision. **Vassilis Kostakos:** Writing – original draft, Validation, Supervision, Conceptualization, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

References

[1] G. Deak, K. Curran, J. Condell, A survey of active and passive indoor localisation systems, Comput. Commun. 35 (16) (2012) 1939–1954, http://dx.doi.org/10.1016/j.comcom.2012.06.004.

[2] F. Zafari, A. Gkelias, K.K. Leung, A survey of indoor localization systems and technologies, IEEE Commun. Surv. Tutor. 21 (3) (2019) 2568–2599, http://dx.doi.org/10.1109/COMST.2019.2911558.

[3] H. Teixeira, A. Magalhães, A. Ramalho, M.d.F. Pina, H. Gonçalves, Indoor environments and geographical information systems: A systematic literature review, SAGE Open 11 (4) (2021) http://dx.doi.org/10.1177/21582440211050379.

[4] H. Wickham, R. François, L. Henry, K. Müller, D. Vaughan, Dplyr: A grammar of data manipulation, 2023, R package version 1.1.4, https://github.com/tidyverse/dplyr.

[5] M. Dowle, A. Srinivasan, data.table: Extension of 'data.frame', 2023, https://r-datatable.com.

[6] Wes McKinney, Data structures for statistical computing in python, in: S. van der Walt, J. Millman (Eds.), Proceedings of the 9th Python in Science Conference, 2010, pp. 56–61, http://dx.doi.org/10.25080/Majora-92bf1922-00a.

[7] K. Kulkarni, J.-E. Michels, Temporal features in SQL:2011, ACM SIGMOD Rec. 41 (3) (2012) 34–43, http://dx.doi.org/10.1145/2380776.2380786.

[8] C. du Mouza, P. Rigaux, Mobility patterns, GeoInformatica 9 (4) (2005) 297–319, http://dx.doi.org/10.1007/s10707-005-4574-9.

[9] M. Girolami, D. La Rosa, P. Barsocchi, A CrowdSensing-based approach for proximity detection in indoor museums with bluetooth tags, Ad Hoc Netw. 154 (2024) 103367, http://dx.doi.org/10.1016/j.adhoc.2023.103367.

[10] Y. Yoshimura, R. Sinatra, A. Krebs, C. Ratti, Analysis of visitors' mobility patterns through random walk in the louvre museum, J. Ambient Intell. Humaniz. Comput. (2019) http://dx.doi.org/10.1007/s12652-019-01428-6.

[11] M. Zancanaro, T. Kuflik, Z. Boger, D. Goren-Bar, D. Goldwasser, Analyzing museum visitors' behavior patterns, in: Lecture Notes in Computer Science, Springer Berlin Heidelberg, pp. 238–246, http://dx.doi.org/10.1007/978-3-540-73078-1_27.

[12] G. Elmamooz, B. Finzel, D. Nicklas, Towards understanding mobility in museums, 2017.

[13] K. Yada, String analysis technique for shopping path in a supermarket, J. Intell. Inf. Syst. 36 (3) (2009) 385–402, http://dx.doi.org/10.1007/s10844-009-0113-8.

[14] A. Yaeli, P. Bak, G. Feigenblat, S. Nadler, H. Roitman, G. Saadoun, H.J. Ship, D. Cohen, O. Fuchs, S. Ofek-Koifman, T. Sandbank, Understanding customer behavior using indoor location analysis and visualization, IBM J. Res. Dev. 58 (5/6) (2014) 3:1–3:12, http://dx.doi.org/10.1147/jrd.2014.2337552.

[15] F. Qi, F. Du, Trajectory data analyses for pedestrian space-time activity study, J. Vis. Exp. (72) (2013) http://dx.doi.org/10.3791/50130.

[16] K. Jayarajah, A. Misra, Predicting episodes of non-conformant mobility in indoor environments, Proc. ACM Interact. Mob. Wearable Ubiquitous Technol. 2 (4) (2018) 1–24, http://dx.doi.org/10.1145/3287050.

[17] R. Martinez-Maldonado, K. Mangaroska, J. Schulte, D. Elliott, C. Axisa, S.B. Shum, Teacher tracking with integrity, Proc. ACM Interact. Mob. Wearable Ubiquitous Technol. 4 (1) (2020) 1–27, http://dx.doi.org/10.1145/3381017.

[18] R. Martinez-Maldonado, V. Echeverria, J. Schulte, A. Shibani, K. Mangaroska, S. Buckingham Shum, Moodoo: Indoor positioning analytics for characterising classroom teaching, in: Lecture Notes in Computer Science, Springer International Publishing, 2020, pp. 360–373, http://dx.doi.org/10.1007/978-3-030-52237-7_29.

[19] A. Kho, K. Johnston, J. Wilson, S.J. Wilson, Implementing an animated geographic information system to investigate factors associated with nosocomial infections: A novel approach, Am. J. Infect. Control 34 (9) (2006) 578–582, http://dx.doi.org/10.1016/j.ajic.2006.02.007.

[20] N. Andrienko, G. Andrienko, Visual analytics of movement: An overview of methods, tools and procedures, Inf. Vis. 12 (1) (2012) 3–24, http://dx.doi.org/10.1177/1473871612457601.

[21] N. Andrienko, G. Andrienko, P. Gatalsky, Exploratory spatio-temporal visualization: an analytical review, J. Vis. Lang. Comput. 14 (6) (2003) 503–541, http://dx.doi.org/10.1016/s1045-926x(03)00046-6.

[22] L.C. Shum, R. Faieghi, T. Borsook, T. Faruk, S. Kassam, H. Nabavi, S. Spasojevic, J. Tung, S.S. Khan, A. Iaboni, Indoor location data for tracking human behaviours: A scoping review, Sensors 22 (3) (2022) 1220, http://dx.doi.org/10.3390/s22031220.

[23] S. Hayward, K. van Lopik, C. Hinde, A. West, A survey of indoor location technologies, techniques and applications in industry, Internet of Things 20 (2022) 100608, http://dx.doi.org/10.1016/j.iot.2022.100608.

[24] F. Liu, J. Liu, Y. Yin, W. Wang, D. Hu, P. Chen, Q. Niu, Survey on WiFi-based indoor positioning techniques, IET Commun. 14 (9) (2020) 1372–1383, http://dx.doi.org/10.1049/iet-com.2019.1059.

[25] Z.D. Tekler, R. Low, B. Gunay, R.K. Andersen, L. Blessing, A scalable bluetooth low energy approach to identify occupancy patterns and profiles in office spaces, Build. Environ. 171 (2020) 106681, http://dx.doi.org/10.1016/j.buildenv.2020.106681.

[26] G. Marini, B. Tag, J. Goncalves, E. Velloso, R. Jurdak, D. Capurro, C. McCarthy, W. Shearer, V. Kostakos, Measuring mobility and room occupancy in clinical settings: System development and implementation, JMIR mHealth uHealth 8 (10) (2020) e19874, http://dx.doi.org/10.2196/19874.

[27] A. Filippoupolitis, W. Oliff, G. Loukas, Bluetooth low energy based occupancy detection for emergency management, in: 2016 IUCC-CSS, IEEE, 2016, pp. 31–38, http://dx.doi.org/10.1109/iucc-css.2016.013.

[28] S.L. Lau, C. Toh, Y. Saleem, Wi-fi fingerprint localisation using density-based clustering for public spaces: A case study in a shopping mall, in: 6th International Conference - Cloud System and Big Data Engineering (Confluence), IEEE, 2016, pp. 356–360, http://dx.doi.org/10.1109/confluence.2016.7508143.

[29] Y. Liu, T. Pei, C. Song, H. Shu, S. Guo, X. Wang, Indoor mobility interaction model: Insights into the customer flow in shopping malls, IEEE Access 7 (2019) 138353–138363, http://dx.doi.org/10.1109/access.2019.2942428.

[30] J. Jung, Y. Choi, Measuring transport time of mine equipment in an underground mine using a bluetooth beacon system, Minerals 7 (1) (2016) 1, http://dx.doi.org/10.3390/min7010001.

[31] D. Cicek, B. Kantarci, Use of mobile crowdsensing in disaster management: A systematic review, challenges, and open issues, Sensors 23 (3) (2023) 1699, http://dx.doi.org/10.3390/s23031699.

[32] I. Omer, R. Goldblatt, Using space syntax and Q-analysis for investigating movement patterns in buildings: The case of shopping malls, Environ. Plan. B: Urban Anal. City Sci. 44 (3) (2016) 504–530, http://dx.doi.org/10.1177/0265813516647061.

[33] Y. Ding, D. Jiang, Y. Liu, D. Zhang, T. He, SmartLOC, Proc. ACM Interact. Mob. Wearable Ubiquitous Technol. 5 (4) (2021) 1–24, http://dx.doi.org/10.1145/3494972.

[34] G. Lau, B. McKercher, Understanding tourist movement patterns in a destination: A GIS approach, Tour. Hosp. Res. 7 (1) (2006) 39–49, http://dx.doi.org/10.1057/palgrave.thr.6050027.

[35] F. Valdés, R.H. Güting, A framework for efficient multi-attribute movement data analysis, VLDB J. 28 (4) (2018) 427–449, http://dx.doi.org/10.1007/s00778-018-0525-6.

[36] D.J. Peuquet, It's about time: A conceptual framework for the representation of temporal dynamics in geographic information systems, Ann. Assoc. Am. Geogr. 84 (3) (1994) 441–461, http://dx.doi.org/10.1111/j.1467-8306.1994.tb01869.x.

[37] G. Andrienko, N. Andrienko, P. Bak, D. Keim, S. Kisilevich, S. Wrobel, A conceptual framework and taxonomy of techniques for analyzing movement, J. Vis. Lang. Comput. 22 (3) (2011) 213–232, http://dx.doi.org/10.1016/j.jvlc.2011.02.003.

[38] N. Andrienko, G. Andrienko, N. Pelekis, S. Spaccapietra, Basic concepts of movement data, in: F. Giannotti, D. Pedreschi (Eds.), Mobility, Data Mining and Privacy: Geographic Knowledge Discovery, Springer Berlin Heidelberg, Berlin, Heidelberg, 2008, pp. 15–38, http://dx.doi.org/10.1007/978-3-540-75177-9_2.

[39] N. Andrienko, G. Andrienko, State transition graphs for semantic analysis of movement behaviours, Inf. Vis. 17 (1) (2017) 41–65, http://dx.doi.org/10.1177/1473871617692841.

[40] T. von Landesberger, F. Brodkorb, P. Roskosch, N. Andrienko, G. Andrienko, A. Kerren, MobilityGraphs: Visual analysis of mass mobility dynamics via spatio-temporal graphs and clustering, IEEE Trans. Vis. Comput. Graph. 22 (1) (2016) 11–20, http://dx.doi.org/10.1109/tvcg.2015.2468111.

[41] G. Marini, Towards indoor localisation analytics for modelling flows of movements, in: ACM UbiComp/ISWC '19 Adjunct, ACM, 2019, pp. 377–382, http://dx.doi.org/10.1145/3341162.3349306.

[42] B.J. Narang, G. Atkinson, J.T. Gonzalez, J.A. Betts, A tool to explore discrete-time data: The time series response analyser, Int. J. Sport Nutr. Exerc. Metab. 30 (5) (2020) 374–381, http://dx.doi.org/10.1123/ijsnem.2020-0150.

[43] C. Faloutsos, M. Ranganathan, Y. Manolopoulos, Fast subsequence matching in time-series databases, ACM SIGMOD Rec. 23 (2) (1994) 419–429, http://dx.doi.org/10.1145/191843.191925.

[44] K.-P. Chan, A.W.-C. Fu, Efficient time series matching by wavelets, in: Proceedings 15th International Conference on Data Engineering (Cat. No.99CB36337), IEEE, 1999, http://dx.doi.org/10.1109/icde.1999.754915.

[45] J. Lin, E. Keogh, S. Lonardi, B. Chiu, A symbolic representation of time series, with implications for streaming algorithms, in: ACM DMKD 2003, Association for Computing Machinery, 2003, pp. 2–11, http://dx.doi.org/10.1145/882082.882086.

[46] E. Keogh, K. Chakrabarti, M. Pazzani, S. Mehrotra, Locally adaptive dimensionality reduction for indexing large time series databases, in: ACM SIGMOD 2001, ACM, 2001, http://dx.doi.org/10.1145/375663.375680.

[47] J. Lin, E. Keogh, L. Wei, S. Lonardi, Experiencing SAX: a novel symbolic representation of time series, 2007, http://dx.doi.org/10.1007/s10618-007-0064-z, Issue: 2 Pages: 107–144 Publication Title: Data Mining and Knowledge Discovery Volume: 15.
[48] B. Lkhagva, Y. Suzuki, K. Kawagoe, New time series data representation ESAX for financial applications, in: 22nd International Conference on Data Engineering Workshops (ICDEW'06), IEEE, 2006, http://dx.doi.org/10.1109/icdew.2006.99.
[49] H. Ruan, X. Hu, J. Xiao, G. Zhang, TrSAX—An improved time series symbolic representation for classification, ISA Trans. 100 (2020) 387–395, http://dx.doi.org/10.1016/j.isatra.2019.11.018.
[50] Y. Yu, T. Becker, L.M. Trinh, M. Behrisch, SAXRegEx: Multivariate time series pattern search with symbolic representation, regular expression, and query expansion, Comput. Graph. 112 (2023) 13–21, http://dx.doi.org/10.1016/j.cag.2023.03.002.
[51] A. Farina, P. Gutierrez-Asorey, S. Ladra, M.R. Penabad, T.V. Rodeiro, A compact representation of indoor trajectories, IEEE Perv. Comput. 21 (1) (2022) 57–64, http://dx.doi.org/10.1109/mprv.2021.3120801.
[52] G. James, D. Witten, T. Hastie, R. Tibshirani, An Introduction to Statistical Learning, Springer New York, 2013, http://dx.doi.org/10.1007/978-1-4614-7138-7.
[53] H. Bunke, Structural and syntacic pattern recognition, in: Handbook of Pattern Recognition and Computer Vision, World Scientific, 1993, pp. 163–209, http://dx.doi.org/10.1142/9789814343138_0008.
[54] J. Lin, E. Keogh, S. Lonardi, J.P. Lankford, D.M. Nystrom, Visually mining and monitoring massive time series, in: SIGKDD, Association for Computing Machinery, New York, NY, USA, 2004, pp. 460–469, http://dx.doi.org/10.1145/1014052.1014104.
[55] J. Rodrigues, D. Folgado, D. Belo, H. Gamboa, SSTS: A syntactic tool for pattern search on time series, Inf. Process. Manage. 56 (1) (2019) 61–76, http://dx.doi.org/10.1016/j.ipm.2018.09.001.
[56] N. Mamoulis, H. Cao, G. Kollios, M. Hadjieleftheriou, Y. Tao, D.W. Cheung, Mining, indexing, and querying historical spatiotemporal data, in: SIGKDD 2004, ACM, 2004, http://dx.doi.org/10.1145/1014052.1014080.
[57] G. Shurkhovetskyy, N. Andrienko, G. Andrienko, G. Fuchs, Data abstraction for visualizing large time series, Comput. Graph. Forum 37 (1) (2017) 125–144, http://dx.doi.org/10.1111/cgf.13237.
[58] M. Hadjieleftheriou, G. Kollios, P. Bakalov, V.J. Tsotras, Complex spatio-temporal pattern queries, in: VLDB, Vol. 5, 2005, pp. 877–888.
[59] ArcGIS indoors, in: Domestic Interior, University of Pittsburgh Press, pp. 67–67, http://dx.doi.org/10.2307/j.ctt5hjr4r.43.
[60] S. Pichler, Introducing ArcGIS IPS - esri's new indoor positioning system, 2022.
[61] J.R. Eastman, J. Bertin, Semiology of graphics, Econ. Geogr. 62 (1) (1986) 104, http://dx.doi.org/10.2307/143508.
[62] J. Wichmann, Indoor positioning systems in hospitals: A scoping review, Digit. Health 8 (2022) 20552076221081696, http://dx.doi.org/10.1177/20552076221081696.
[63] G. Shipkovenski, T. Kalushkov, E. Petkov, V. Angelov, A beacon-based indoor positioning system for location tracking of patients in a hospital, in: HORA 2020, IEEE, 2020, pp. 1–6, http://dx.doi.org/10.1109/hora49412.2020.9152857.
[64] J. Lin, R. Khade, Y. Li, Rotation-invariant similarity in time series using bag-of-patterns representation, 2012, http://dx.doi.org/10.1007/s10844-012-0196-5.