FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Representation and quantification of change on spatiotemporal phenomena

Edgar Filipe Amorim Gomes Carneiro

Mestrado Integrado em Engenharia Informática e Computação

Supervisor: Alexandre Miguel Barbosa Valle de Carvalho

Second Supervisor: Rui Pedro Amaral Rodrigues

July 20, 2020

Representation and quantification of change on spatiotemporal phenomena

Edgar Filipe Amorim Gomes Carneiro

Mestrado Integrado em Engenharia Informática e Computação

July 20, 2020

Abstract

A significant number of modern devices gather spatiotemporal data. This fact prompted the need for data mining frameworks capable of processing spatiotemporal information and extracting knowledge from it. In the more specific field of change detection, this knowledge comes from the ability to automatically identify changes characterising spatiotemporal phenomena. Considering that different users have different interests while consuming the retrieved information, it is essential to provide them with the means to express the degree of interest towards change detection. However, state-of-the-art approaches have not addressed both retrieving the changes detected in the phenomena and providing the users with the ability to express their interests. Therefore, in this work, we analyse how to detect, extract and quantify user-relevant changes affecting a spatiotemporal phenomenon.

To answer this question, a conceptual framework is proposed, incorporating computation methods for all of the change categories identified in the literature review. As input, and in order to retrieve pertinent information to the users, the framework accepts user-defined thresholds describing their interests. Subsequently, to validate the proposed framework, a prototype was devised and tested, in a variety of scenarios, that implements the base structure of the proposed conceptual framework and analysing a subset of the change features. To evaluate the results achieved, two novel complementary quantitative metrics were proposed. These metrics allow assessing the changes detected and their resemblance to the ground-truth. Supported by the prototype and the evaluation metrics, experiments were performed towards assessing the solution capability for detecting a singular change, sequences of changes and co-occurring changes. Additionally, tests also assessed the impact of the prototype component responsible for detecting spatial changes in the obtained performance.

Results showed that the proposed solution depends significantly on the accuracy of the components responsible by detecting changes. It was verified that when the component detects changes with high-precision, the prototype can successfully retrieve the changes of interest affecting the analysed phenomenon (the worst obtained score when using high-precision components was 92.5%). Therefore, the results obtained validate the proposed solution and provide the necessary information to answer the raised research question. Moreover, the obtained results and evaluation methods can also be employed as benchmarks for future frameworks with similar objectives.

Keywords: spatiotemporal data, data mining, change detection, features of interest

Resumo

Um número significativo de dispositivos atuais recolhem dados espaciotemporais. Este facto incita a necessidade de ferramentas de exploração de dados, capazes de processar informação espaciotemporal e extrair conhecimento desta. No campo mais específico de deteção de mudança, este conhecimento advém da capacidade de automaticamente identificar as mudanças que caracterizam os fenômenos espaciotemporais. Considerando que diferentes utilizadores têm diferentes interesses ao consumir a informação adquirida, é essencial providenciar os meios para que estes possam expressar o seu grau de interesse em relação à detecção de mudança. No entanto, as abordagens de vanguarda científica não analisam simultaneamente o retorno das mudanças detectadas em fenómenos e como providenciar os utilizadores com a capacidade de expressar os seus interesses. Assim, neste trabalho, nós analisamos como detectar, extrair e quantificar mudanças, de relevância para o utilizador, que afectam os fenômenos espaciotemporais.

Para responder a esta questão, propomos uma solução conceptual, incorporando métodos de computação para todas as categorias de mudança identificadas na revisão de literatura. Como *input*, e de forma a retornar informação pertinente para os utilizadores, a solução aceita limites, definidos pelo utilizador, que descrevem os seus interesses. Subsequentemente, para validar a solução proposta, foi criado e testado, numa variedade de cenários, um protótipo que implementa a estrutura base da solução conceptual proposta e processa um subconjunto das categorias de mudança. Para avaliar os resultados alcançados, duas novas métricas quantitativas complementares foram propostas. Estas métricas permitem aferir a mudança detectada e a sua semelhança com o valor de referência. Suportadas pelo protótipo e pelas métricas de avaliação, foram realizadas experiências com vista a aferir a capacidade da solução de detectar uma única mudança, uma sequência de mudanças e mudanças co-ocorrentes. Adicionalmente, os testes também aferiram o impacto no desempenho obtido do componente do protótipo responsável por detectar mudanças espaciais.

Os resultados demonstraram que a solução proposta depende significativamente da precisão dos componentes responsáveis por detectar mudanças. Foi verificado que quando o componente detecta mudanças com grande precisão, o protótipo consegue retornar com sucesso as mudanças de interesse que afectam o fenômeno analisado (o pior resultado obtido quando utilizados componentes de alta precisão foi 92.5%). Assim, os resultados obtidos validam a solução proposta e providenciam a informação necessária para responder à questão de investigação levantada. Ademais, os resultados obtidos e os métodos de avaliação podem também ser empregados como referências para futuras soluções com objetivos similares.

Keywords: dados espaciotemporais, exploração de data, deteção de mundaça, características de interesse

Acknowledgements

First, I thank my supervisor, Professor Alexandre Valle de Carvalho, for all the patience and guidance provided, not only in the development of this dissertation but also in the projects accomplished in the prior year. I also thank my second supervisor, Professor Rui Rodrigues, for all the experience and thought-provoking advice that significantly increased the value of this dissertation. A special thanks to Xavier Fontes, for always helping, even when the answer was not obvious.

I would also like to express my gratitude towards my family, for all the love and support. To my parents, for being able to educate me, guide me and make me the person I am today. To my little brother (not so little anymore), I thank you for always accompanying me, even through the long nights of work. To the rest of my family, for all the laughter shared and for always remembering me what we should truly cherish in this life.

To Sara, I am grateful for having you in my life.

Lastly, I thank my friends, for all the companionship and good moments spent and that we may never forget our motto: "um sargento nunca abandona os seus soldados".

Edgar Carneiro

“The man who moves a mountain begins by carrying away small stones.”

Confucius

Contents

List of Figures

List of Tables

List of Listings

Abbreviations

2D	Two-Dimensional
3D	Three-Dimensional
UML	Unified Modelling Language
ID	Identifier
REST	Representational State Transfer
API	Application Programming Interface
JSON	Javascript Object Notation
ORM	Object-relation mapping

Chapter 1

Introduction

Spatiotemporal data consist of data collected across space and time. Well-known spatiotemporal data examples encompass the tracking of animal species or human movement, the monitoring of the evolution of natural phenomena like tectonic movements, wildfires spreading or weather changes, the surveyance of epidemic outbreaks, among others. Considering the fast development and extensive usage of devices capable of gathering spatiotemporal data, like mobile phones, sensors, GPS gadgets, satellites, etc., large amounts of raw data are collected and therefore available for analysis [54]. The main challenges when handling spatiotemporal data dwell on the vast amounts of data available, the extraction of valuable information from raw data and the identification of patterns. To tackle these problems, the spatiotemporal data mining domain studies the identification of patterns and the attainment of knowledge from spatiotemporal data [41, 78]. The knowledge extracted using spatiotemporal data mining processes can lead to significant benefits, such as the prediction of hurricanes and volcano eruptions, the control of the dissemination of diseases [48], the prevention of wildfires [20], the analysis of wildfire fronts to control changes of direction, and so forth.

1.1 Motivation

Change detection represents a notable field of the spatiotemporal data mining domain [78]. This field focuses on identifying the sequences of changes that affect spatiotemporal phenomena. When a user utilises a visualisation tool to manually analyse simple sequences of raw spatiotemporal data, identifying the change affecting a given phenomenon can be a simple and intuitive process. However, when considering large amounts of data (e.g. historical data) identifying the changes and the underlying spatiotemporal patterns can become infeasible as the spatiotemporal patterns can be intricate and not intuitive to recognise, and the process of manually interpreting the data can be very time-consuming. Hence, there is a need for systems capable of automatically detecting changes on raw spatiotemporal data, with the intuit of tackling the issues associated with human-made analysis. Moreover, by using automated frameworks, the process is accelerated, thus permitting the detection of determinant changes in real-time [40], which, in extreme cases, can make a difference in the phenomenon outcome.

Therefore, the motivation behind this work consists of studying automated frameworks that allow the end-users to obtain valuable knowledge regarding the spatiotemporal changes affecting a phenomenon, thus facilitating the data mining process.

1.2 Problem

Moving-object data, meaning data describing objects having a position in space that changes over time [2], illustrated in Figure 1.1, are a notable kind of spatiotemporal data [41]. The most forward manner of modelling moving-object data consists of describing the spatial disposition of the objects, for example using shape descriptors (a simplified representation of a 2D or 3D shape in the form of a vector that geometrically describes the shape [47]), at different timestamps.

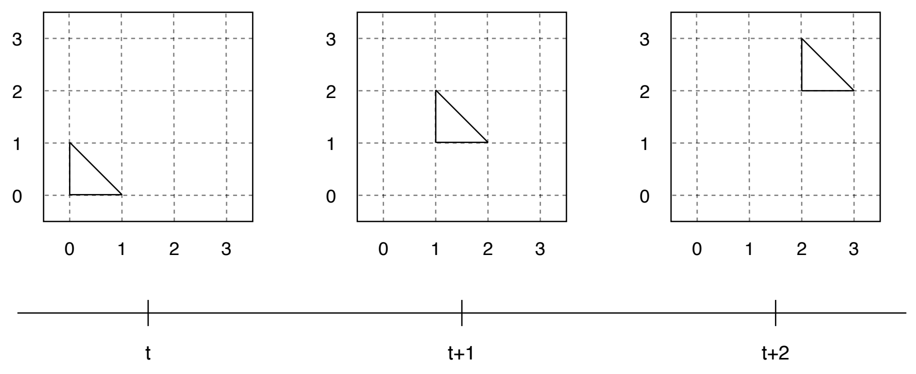


Figure 1.1: Spatial disposition, in three distinct timestamps, of a phenomenon being translated.

However, there are other ways of modelling spatiotemporal phenomena that do not require describing every single timestamp, as illustrated in Figure 1.2. This figure shows three possible manners of describing the phenomenon illustrated in Figure 1.1. The first approach is a spatiotemporal representation where the spatial configuration at each instant is described using shape descriptors. The second approach consists of the movement vectors of each of the phenomenon's vertices. Lastly, in a third approach, the phenomenon is modelled by the change wholly affecting it — the desired output on spatiotemporal data mining frameworks. The latter approach can be considered a change-based representation where not all timestamps are described.

Different users can also have different interests when it comes to the attainment of knowledge through the process of spatiotemporal data mining. Moreover, on vast amounts of data, there may occur the detection of various changes. However, from the end-users perspective, having many detected changes can lead to them having to filter the retrieved information manually. Hence, frameworks that take into account the user interest to retrieve the most appropriate changes prove to be of additional usefulness.

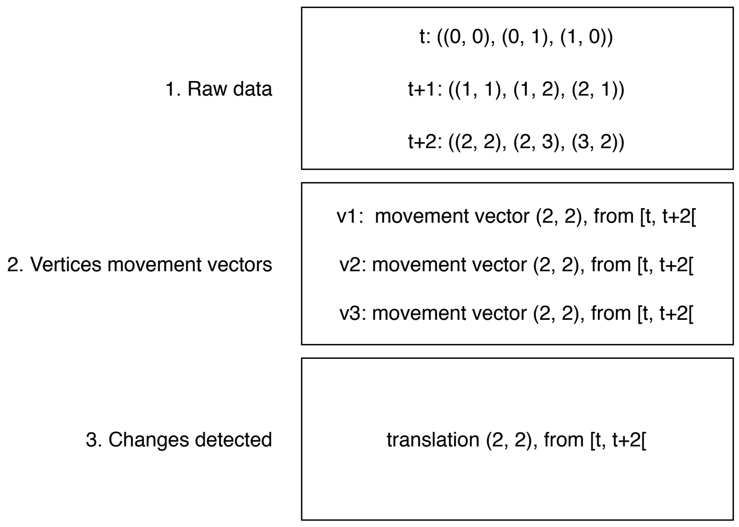

1. Raw data
t: ((0, 0), (0, 1), (1, 0))
t+1: ((1, 1), (1, 2), (2, 1))
t+2: ((2, 2), (2, 3), (3, 2))
2. Vertices movement vectors
v1: movement vector (2, 2), from [t, t+2[
v2: movement vector (2, 2), from [t, t+2[
v3: movement vector (2, 2), from [t, t+2[
3. Changes detected
translation (2, 2), from [t, t+2[

Figure 1.2: Possible representations of the moving-object presented in Figure 1.1.

Therefore, the main problem addressed in this dissertation is how to build a change-based representation, of interest to the end-user, by extracting the changes affecting the spatiotemporal phenomena. From the performed literature review, a single framework exists that focuses on extracting the changes affecting a spatiotemporal phenomenon from the original continuous representation [98]. However, this framework is domain-specific and can only identify a subset of the spatiotemporal changes affecting the phenomena.

1.3 Objectives

This work addresses the research of automated processes towards the identification, quantification and representation of change in a spatiotemporal phenomenon. The input data consists of a single spatiotemporal phenomenon, modelled by its 2D shape (shape descriptor) at different successive timestamps.

The objective of this dissertation is to study, develop, test and conclude about a proposed novel context-independent framework that processes the spatiotemporal data representing a phenomenon and extracts and quantifies the change of interest to the end-user occurring in it. Given the presented context, in this domain, the change occurring in a phenomenon refers to the spatiotemporal transformations affecting it. The main concerns focus on retrieving a change-based representation that is authentic to the original representation as well as filtering and retrieving segments that are of interest to the end-user.

1.4 Outline

This dissertation is structured as follows. Chapter 2 describes the performed literature review, focusing on the different forms of representing spatiotemporal data, the representation of change in the spatiotemporal domain, how to quantify that change and the analysis of approaches to similar problems. Chapter 3 presents a detailed description of the problem being solved, as well as the requirements a solution would have to solve the problem, and the associated research questions. Chapter 4 details the guidelines of a conceptual framework for the automated quantification of the change, of interest to the end-user, that affects a spatiotemporal phenomenon. Chapter 5 describes the performed implementation of the conceptual framework presented in chapter 4, while explaining the overall architecture and technical aspects of the solution. Chapter 6 explains the processes used for evaluating the developed solution, describes and discusses the results obtained, and analyses if the solution satisfies the requirements established in chapter 3 and how it answers the research questions. Finally, chapter 7 exposes the conclusions obtained in this dissertation and presents future work.

This work follows the "Manuscript Fundamentals" writing guideline, presented by Elsevier[1].

[1]Elsevier. Fundamentals of manuscript preparation. Available at https: `https://researcheracademy.elsevier.com/writing-research/fundamentals-manuscript-preparation`. Accessed in 2020-02-06.

Chapter 2

Change representation and quantification on spatiotemporal phenomena

This work is focused on the automatic detection of the change affecting spatiotemporal data. To that end, it is necessary to research existing literature on how to represent change, what changes can be detected, how to quantify the detected changes and what similar approaches exist that have similar purposes. Therefore, section 2.1 starts by describing the existing data models for representing spatiotemporal data. After, section 2.2 identifies and describes the different spatiotemporal change features that characterise a phenomenon's change. Subsequently, section 2.3 presents several methodologies for determining how to compute and quantify the identified features as well as classify them as interesting to the end-user. Section 2.4 explores other approaches to similar problems. Finally, a summary of this chapter's contents is presented.

Before proceeding, it becomes relevant to recall that, as initially presented in chapter 1, the scope of this work focuses on spatiotemporal data describing a single phenomenon, modelled by its 2D shape at different successive timestamps. This is a result of this thesis having as an underlying data provider the framework presented by Duarte et al. [25].

2.1 Spatiotemporal data representation

One of the primary challenges in dealing with spatiotemporal data is its representation. We start by introducing different models for representing spatiotemporal data. Next, we present different data decompositions, notably the different domains that constitute spatiotemporal data. We proceed with the analysis of possible data models for an inherent topic of spatiotemporal data: movement. Lastly, we focus on the differences between the continuous and discrete representations of spatiotemporal data.

2.1.1 Spatiotemporal data models

Worboys [97] identifies three interesting approaches when modelling dynamic geographic phenomena: *temporal snapshots*, *object change*, and *events and action*. The first approach, ***temporal snapshots***, models phenomena as a succession of temporal snapshots containing the object's spatial configurations. Hence, a temporal snapshot represents the object's status in a particular domain at a timestamp. Figure 1.1 illustrates this approach by presenting three distinct temporal snapshots, while Figure 1.2 shows a possible representation under the label 'raw data'. This approach samples 'dynamic phenomena at a sequence of temporal instants'. Thus, concepts such as timestamps and temporal granularity link directly to it. It is also the most widely-used approach from the ones being presented. Hornsby et al. first introduced the *object change* modelling approach [42]. In the ***object change*** model, the represented entities are the changes that can happen, be it to objects, attributes or relationships. The modelling focus shifts from the sequences of the phenomena's states to the changes occurring. Figure 1.2 shows a possible representation under the label 'changes detected'. The third and last approach, ***events and action***, models phenomena occurrences and the existing relationships between those and the objects participating in them. Events are from one of three main classes: events as occupations of spatiotemporal regions, events identified according to the cause-effect relation, and events as exemplifying a property or relationship at some time.

Shekhar et al. introduce a taxonomy of different spatiotemporal data types [78]. The representation used in their work shares principles with Worboy's work but focuses on the model representation over the different spatiotemporal data types. They categorise data into three possible models: temporal snapshot, temporal change and event or process model. In the first category, considering a set of points as the spatial layer, the **temporal snapshot** corresponds to the trajectories of these points. Likewise, snapshots can also represent trajectories of lines and polygons, among other types. The **temporal change** model describes a spatial layer at a given timestamp and the incremental changes occurring from that moment onward. This model facilitates the representation of displacement/motion (e.g. Brownian motion [46]), speed and acceleration on spatial points and of motion, extension, rotation, deformations and split/merges on lines and polygons. The last model represents spatiotemporal data using **events or processes** [99], where events represent entities whose properties do not change over time, while processes consist of entities that can suffer changes over time (e.g., process that is accelerating).

Lohfink et al. present a 'generic object-oriented model for representing evolving features' [53]. The authors name it *Feature Evolution Model (FEM)*. The authors identify the easiness to quickly escalate complexity, when trying to represent change, as the major implementation problem of spatiotemporal systems. For modelling change, FEM focuses on states (geographic equivalent of versions) and occurrents, which are modelled using a three-layer hierarchy (see Figure 2.1). States are then related to other states through transitions. Transitions aggregate occurrents. Thus, transitions between states are associated with occurrents (*object change* representation).

In contrast to Lohfink's work, Aydin et al. focus on the storage of spatiotemporal data using **non-relational** and distributed database systems [6]. The authors focus on the nuances of storing

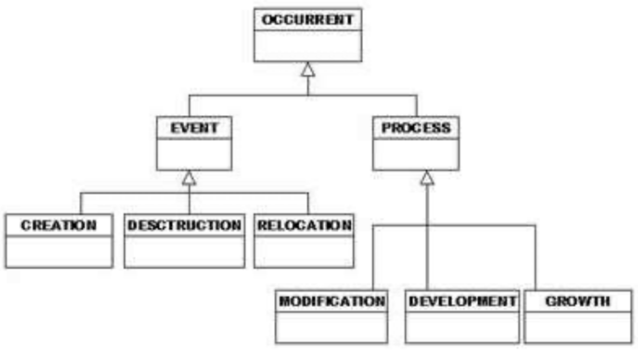


Figure 2.1: UML of occurrents in *FEM*, presented by [53]

large-scale trajectory data using *NoSQL*.

As shown in the beginning of this chapter, the input for the current work consists of sequences of temporally consecutive 2D representations of a polygon's shape. Therefore, for the remainder of this work, we will focus on spatiotemporal based on using polygonal data, over the remaining data types. Furthermore, using the modelling methods already identified, we can affirm that our objective is to transform data modelled through *temporal snapshots* to an *object change* representation, while quantifying and characterising the occurring change(s). Hence, the primary focus of this work will be on these two modelling methods.

2.1.2 Spatiotemporal decomposition of changes

Blok identifies four distinct categories when describing changes over spatiotemporal data [8]. These can be listed as follows: changes in the spatial domain, changes in the temporal domain, spatiotemporal patterns over long periods, and relative similarity of patterns.

In the first category, the author describes three basic concepts. The first is the **existential changes** meaning the appearance/disappearance of a phenomenon. It is appropriate to describe phenomena that can disappear over time. Natural phenomena, such as tornadoes or wildfires, are presented as examples of this spatial change. Appearances and disappearances can also describe split and merges of phenomena. Next, the **mutation** of phenomena, meaning the transformation of either a phenomenon's nominal attribute (e.g.: from rain to snow) or an ordinal change of an attribute (e.g.: wind's intensity increase). Lastly, the **movement**, which describes a change in the spatial position or geometry of a phenomenon. This last concept includes both the movement motion of the phenomenon along a path (encapsulating translations and rotations) or the phenomenon's boundary shift (encapsulates expansion/ shrinkage and deformations). Andrienko et al. identify similar types of changes that can occur, over time, on spatiotemporal data [3].

Regarding changes in the temporal domain, Blok identifies five concepts [8]. The first, ***Moment in time***, refers to the timestamp where a change starts occurring or occurs. Next, the ***Pace*** concept quantifies change over time. Hence, any spatial characteristic has a pace. ***Duration*** is the concept that characterises the period during which a change occurred. The ***Sequence*** concept refers to the order by which changes occur in the temporal domain. Lastly, ***Frequency*** refers to the number of times that a change repeatedly occurs.

According to Blok, spatiotemporal patterns over long periods divide into cycles or trends [8]. **Cycles** represent spatial events repeated regularly in the same order. **Trends** are structured acyclic patterns.

In the last category, Blok presents two different similarity categories of patterns: **equality** and **simultaneousness** [8].

Shekhar et al. identify three distinct data attributes on spatiotemporal data: non-spatiotemporal attributes, spatial attributes, and temporal attributes [78]. This decomposition corroborates the one presented by Blok [8] since the decomposition is very similar. The first attribute type is employed when describing an object's features unrelated to the spatiotemporal context, such as name, population, among others. Spatial attributes describe the spatial location (e.g. absolute coordinates) and the shape of objects. Lastly, temporal attributes include both the timestamp and duration of a process of a spatial object.

2.1.3 Modelling Movement

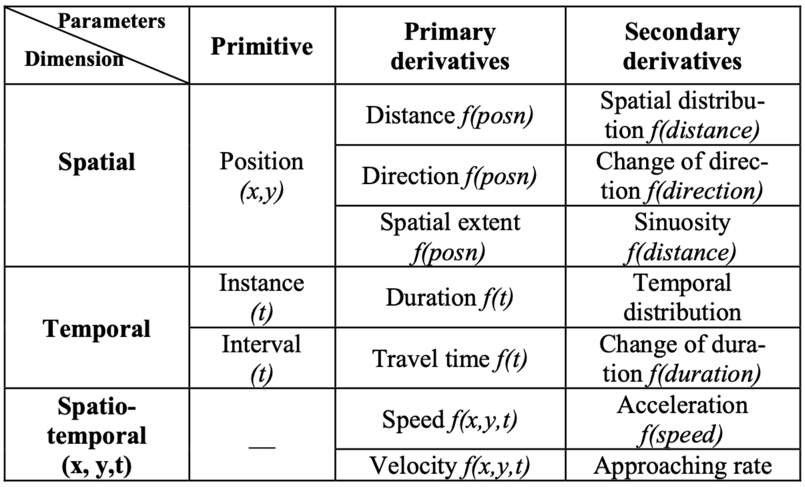
From the literature review already seen, we realise that movement emerges as a central area of focus in spatiotemporal modelling. Therefore, we examine it with particular attention.

Eschenbach categorises movement into two types: trajectory-oriented motion and internal motion [30]. On the one hand, **trajectory-oriented motion** focuses on the progress of individual bodies through space, denoted by successive spatial points. Consequently, the body's shape and spatial structure are irrelevant to the movement of the object. On the other hand, **internal motion** comprises the motion of some part of the object, instead of the whole entity. Therefore, it becomes necessary to consider the internal spatial structure. Growth, shrinkage, internal rotation and partial motion are some of the internal motion subcategories. The author defines growth and shrinkage by the gain or loss of area, respectively. Internal rotation consists of the motion of every object's part concerning an inner pivot. In partial motion, some parts of the object move while others do not. Thus, deformations belong to this category. Furthermore, the two distinct kinds of movement are not unrelated. Movements can result from a combination of other movements. Internal motions in short intervals can be combined in trajectory motions in larger intervals.

Dodge et al. define moving objects as 'entities whose positions or geometric attributes change over time' [23]. Furthermore, movement is defined as a change, occurring over time, in an object's location, while maintaining the objects' identity.

Dodge et al. introduce a conceptual framework with the intuit of better understanding movement, the parameters that define it, the external factors that influence it, and the possible types of movement patterns [23]. Moreover, the authors also classify and review the identified movement patterns. The authors also highlight the importance of defining movement in a *relative sense*. The term *relative sense* refers to the relation between two or more moving object's movements, and therefore, does not apply to this work, which is focused on a single phenomenon.

The conceptual framework presented considers three distinct movement parameters: *primitive parameters*, *primitive derivatives* (parameters derived from the first parameters), and *secondary derivatives* (derived from the primitive derivatives). This division of movement parameters has its foundations on Differential Geometry, where vector arithmetic and derivatives are used for computing movement patterns [36]. Additionally, the framework considers the existence of three dimensions (similar to the ones presented by previous works, such as [8, 78]): spatial, temporal and spatiotemporal. Figure 2.2 presents the different movement parameters that arise from the combination of dimensions with the different groups of movement parameters.



Parameters / **Dimension**	**Primitive**	**Primary derivatives**	**Secondary derivatives**
Spatial	Position *(x,y)*	Distance *f(posn)*	Spatial distribution *f(distance)*
		Direction *f(posn)*	Change of direction *f(direction)*
		Spatial extent *f(posn)*	Sinuosity *f(distance)*
Temporal	Instance *(t)*	Duration *f(t)*	Temporal distribution
	Interval *(t)*	Travel time *f(t)*	Change of duration *f(duration)*
Spatio-temporal (x, y,t)	—	Speed *f(x,y,t)*	Acceleration *f(speed)*
		Velocity *f(x,y,t)*	Approaching rate

Figure 2.2: Decomposition of movement parameters, as presented by [23].

Concerning the classification of movement patterns, the same three dimensions considered in the conceptual framework are used. Movement patterns can either be generic or behavioural. Generic patterns represent the simpler patterns that, in conjunction, form higher-level patterns. Behavioural patterns correspond to the higher-level movement patterns and are associated with the nature of the moving object. Generic patterns can also be decomposed in primitive or compound. Primitive patterns represent the most basic movement patterns, meaning only a movement parameter changes. Conversely, in compound patterns, several parameters change simultaneously. Figure 2.3 presents the classification system proposed by the authors, according to the mentioned dimensions and domain divisions.

Güting et al. use a *sliced representation* for representing deformable moving objects [32]. In this representation, moving objects consist of an ordered collection of units. Units represent the continuous evolution of the object between two consecutive observations. For each unit, the evolution of the moving object can be described using a time-dependent function $f(t)$. Figure 2.4 consists of a visual representation of this modelling technique.

Andrienko et al. introduce a taxonomy for representing and analysing movement data [2]. In a spatiotemporal analysis, they state the existence of three fundamental sets pertinent to movement: **space** S, representing a set of locations; **time** T, representing a set of timestamps, and **objects** O. Moving objects (also named movers) are objects having a position in space that changes over

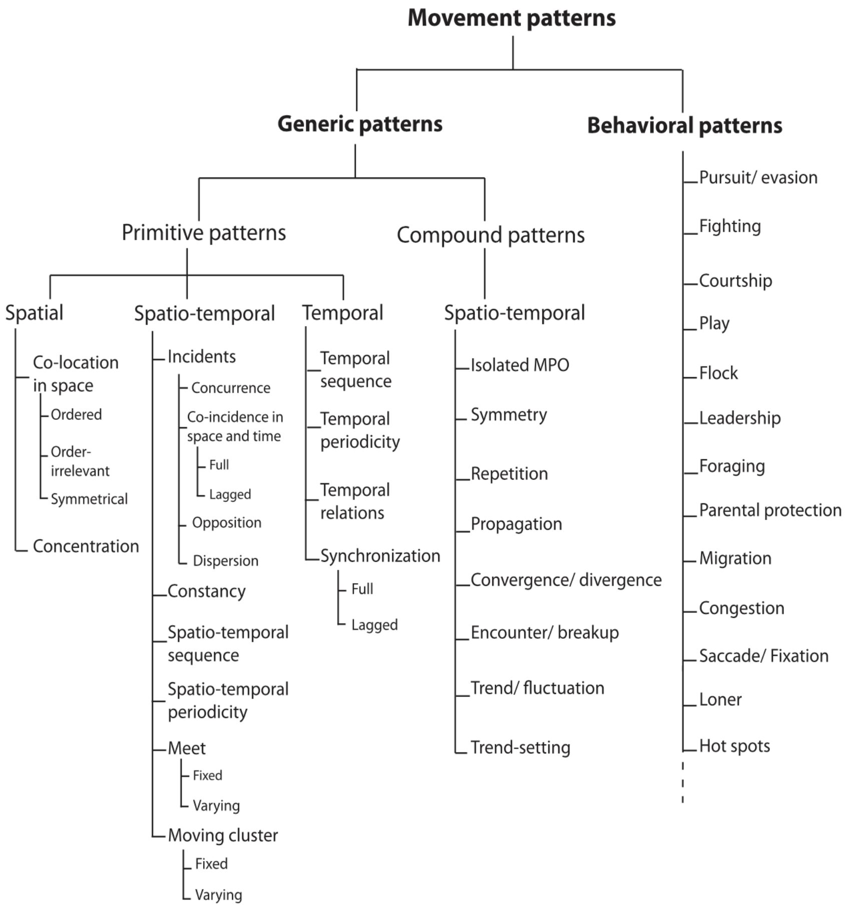


Figure 2.3: Classification system of movement patterns, presented by [23]

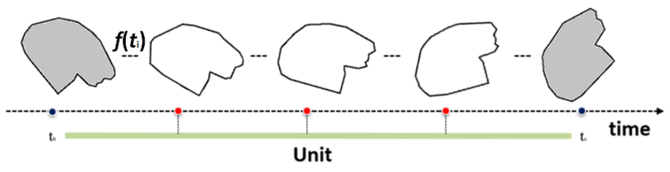


Figure 2.4: Sliced representation introduced by [32], as presented by [24].

time. **Thematic attributes** A, can be classified into static (when values do not change over time) or dynamic (the opposite case). Movement is the change of spatial position of a moving object over time. The authors represent Movements using the encoding: $O \times T \to S$ (interpreted as: a given mover, at a given unit, has a given spatial position) or the similar $O \to (T \to S)$. When considering thematic attributes, the adequate encoding is $O \times T \to S \times A$ (interpreted as: a given mover, at a given unit, has a given spatial position and thematic attributes' values). Another possible encoding is $O \to T \to S \times A$. Time units unrelated to locations are encoded as $T \to A$. Furthermore, dividing movement data into set of events is possible using the encoding $O \to ((T_1 \to S) \cup (T_2 \to S) \cup \cdots \cup (T_k \to S))$, where $T_1, T_2, \cdots, T_k$ represent non-overlapping subsets of the original T.

2.1.4 Geometrical approach

As previously observed, we are trying to output an *object change* representation of the input data. Consequently, we must identify the change that occurs, categorise it and give meaning to it. Change is strongly intertwined with the notion of transformation, and, as already noticed, in spatiotemporal data more specifically with geometrical transformations.

Furthermore, from the previous subsections, we can already see that spatiotemporal modelling is associated with geometry, since many of the features lead to known geometric transformations, such as translations and rotations.

Hence, it becomes relevant to understand the possible categories of geometric transformations and the characteristics inherent to each of those categories. One of these characteristics is the transformation **invariance properties**, meaning the object's features that are left unchanged under the transformation occurrence [35]. Below, we list the identified geometrical transformations categories while providing a brief explanation of each. With this purpose in mind, we support ourselves on the works of Wilkinson [95] and Galarza et al. [33].

1. **Isometries** are the group of transformations, belonging to euclidean geometry, that preserves the distance between points, across transformations. Thus, metrics such as perimeters, areas, angles and the geometry's shape created by the points are conserved (*invariance properties)*. Isometries can also be called **rigid transformations**.

2. **Similarity** transformations conserve the shape of the object, while not ensuring the perseverance of the distance between points. Thus, the angles and collinearity are maintained. Furthermore, parallel lines remain parallel after the transformation. Hence, these transformations are indicated for resizing.

3. **Affine** transformations maintain the ratios of distances between points belonging to a rectilinear segment and collinearity. Additionally, as in similarity, parallel lines remain parallel after the transformation. Hence, angles, distances between points, and consequently shapes, are not necessarily preserved.

4. **Projective** transformations, also known as **homographics** transformations [14], have its name deriving from the fact that they are able to represent projections. In this type of transformation, straight lines are preserved, but angles can be modified. This group of transformations is best visualised when thinking about the projection of light onto an object, from different angles.

5. **Conformal** transformations' main focus is conformal mappings, where there is the preservation of local angles in graphics, but there might be distortions of the object's global shape. Thus, when under a conformal transformation, smaller objects conserve their shape, while bigger objects are bent (making straight lines curve).

There is an inherent hierarchy to the enumerated transformations. For example, all rigid transformations are also affine transformations. Figure 2.5 highlights the hierarchy identified by [95].

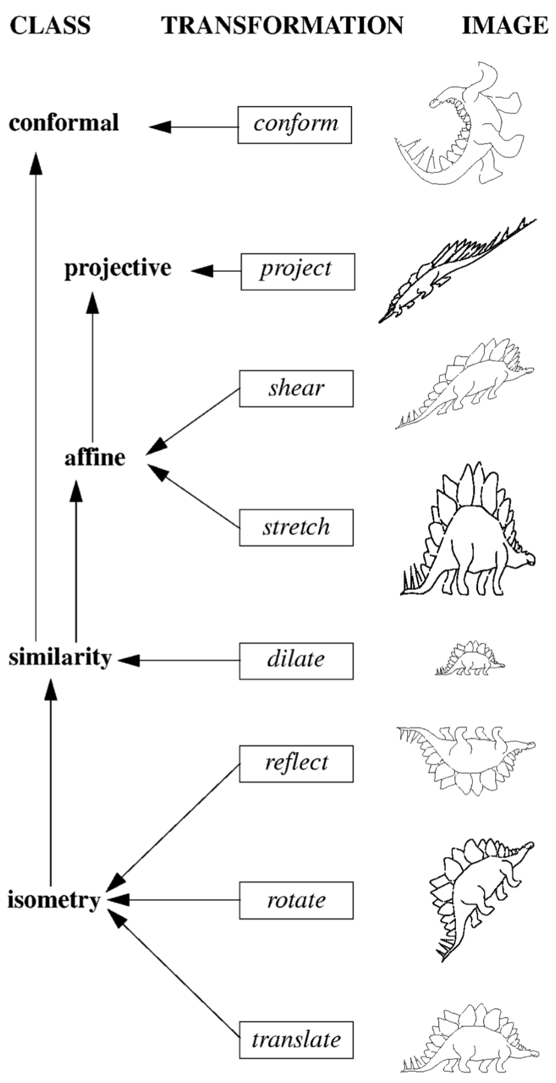


Figure 2.5: Examples of some geometrical transformations. The *CLASS* column identifies the category encompassing the transformation, according to the categories presented in sub-section 2.1.4. Image taken from [95].

There are still some transformations not mentioned by neither Wilkinson [95] nor Galarza et al. [33]. We join these others under the **General** transformations group, in a similar fashion to Burago et al. [11]. Therefore, general transformations encapsulate both diffeomorphisms [70] and homeomorphisms [71]. Homeomorphisms consist of the mapping of two sets of points while assuring continuity[1] and bijectivity[2]. Hence, classical shape deformations relate to this collection.

Further ahead, in section 2.2, we provide an enumeration of the identified transformations for each of the above categories.

[1] Absence of abrupt changes (discontinuities) [69].

[2] One-to-one mapping between the points of both point sets [68].

2.1.5 Continuous and Discrete Models

Changes can be interpreted as **discrete** or **continuous** [30]. On the one hand, the concept of *discrete change* represents the difference between an initial situation and the final situation, meaning the state after the occurrence of the change without any intermediate positions. On the other hand, the concept of *continuous change* focuses on the development of the situation while the change is occurring, and on the change's internal structure. Informally, continuous changes are smooth, meaning neither interruptions nor inconsistencies may be noticeable.

2.1.6 Summary

In this section, we analysed different manners of modelling spatiotemporal data. In finer detail, we discovered that several authors agreed on three different ways of modelling spatiotemporal data: using temporal snapshots, modelling object change, and through events and processes. Seeing that we receive a spatiotemporal phenomenon's data in the form of temporal snapshots as our input and that we intend to extract the changes that the phenomenon goes through, we mainly work the temporal snapshots model and object change model.

We also verified how to decompose spatiotemporal data, highlighting three different types of data attributes: non-spatiotemporal, spatial, and temporal attributes.

Considering that movement represents a significant component of change on spatial (and consequently spatiotemporal) data, we scrutinised its representation. Movement is essentially divided into two components: trajectory-oriented and internal motion.

After a more thorough analysis of movement, we verified the strong correlations with geometric modelling and geometric transformations. Therefore, we categorised and described different types of geometrical transformations.

Lastly, we identified the differences between discrete and continuous models.

2.2 Spatiotemporal change features

For a better understanding of the work at hand, it becomes a priority to define the meaning of **spatiotemporal change features**. In this work's context, a spatiotemporal change feature is the characterisation of an identifiable change occurring between two or more spatial representations of a phenomenon (two temporal snapshots). Henceforth, the term *change feature* or simply *feature* are other manners of referring to the afore presented concept.

Section 2.1 presented some of the available methods for modelling spatiotemporal data. Additionally, in order to make the user better understand the possible models, some examples of change features were given. In this section, we thoroughly analyse each of the possible spatiotemporal change features, defining each one and, when possible, visually representing them.

At the beginning of this chapter, we referred that our input consists of 2D representations of a phenomenon's shape. Hence, in this section, we focus on identifying related work towards spatiotemporal change features in the 2D domain.

We divide the identified features into five categories, based on the division/decomposition that has been employed by other authors, as seen in section 2.1. First, we identify spatial-related features. At its core, these features deal with shape changes. Since, in our case, we represent shapes using shape descriptors, we focus on the possible changes that can affect the boundaries of spatiotemporal phenomena. Therefore, essentially the first category aims at describing geometric transformations. Next, we describe the identified existence features, meaning features whose concerns are the phenomena's existence. Then, we analyse features that are mainly related to the time domain. The succeeding set of features are derivative features: features computed from other features previously identified. Lastly, we identify spatiotemporal patterns related features.

2.2.1 Spatial-related features

From the literature review, we identify geometrical transformations as the primary representation of spatial change features. Therefore, this subsection focuses mainly on the analysis of these transformations while employing algebraic notations.

In this subsection, the geometrical transformations presented, as well as the matrix notations, are firmly based on the works of Wilkinson [95], Galarza et al. [33], and Gentle [35].

For the following transformations, whenever a transformation's explanation refers to its effects on a generic point (x, y), assume that for a polygon the same effects apply to its constituting set of points.

Before initialising the listing of the identified features, it becomes relevant to introduce the concept of **homogeneous coordinates** [81]. Homogeneous coordinates are the natural coordinate system for projective geometry. For better explaining these, a comparison with cartesian coordinates is established. Cartesian coordinates are a set of numbers indicating the position of a point in space, relative to a fixed reference point named the origin, through the shortest distance to the axes that intersect the origin at right angles (90 degrees) [55]. Consider a point defined by the cartesian coordinates $(x_1, x_2, ..., x_n)$. In the homogeneous coordinates system, the same point representation is $(x_0^h, x_1^h, x_2^h, ..., x_n^h)$ or, positioning the new coordinate at the end, $(x_1^h, x_2^h, ..., x_n^h, x_0^h)$. The new coordinate, x_0^h, represents a hyperplane in the cartesian coordinate system. It is frequent for x_0^h to assume value 1, thus making the remaining coordinates map to their cartesian correspondent $(x_n^h = x_n)$ [35]. For exemplification, consider the point $(2, 4)$. Its correspondent using homogeneous coordinates, with $x_0^h = 1$, is $(2, 4, 1)$. The MATLAB programming platform proves helpful when determining homogeneous representations of geometrical transformations [58].

2.2.1.1 Translation

The translation is an isometric transformation. In translations, all of the polygon's points are moved by a distance $||\vec{v}||$, with $\vec{v}$ being the translation vector. Hence, translating a polygon means moving it in a given direction without altering its shape, orientation or size.

In another notation, consider the polygon represented by $((x_1,y_1),(x_2,y_2),(x_3,y_3))$. After applying a translation of (a,b) to the polygon, we obtain $((x_1+a,y_1+b),(x_2+a,y_2+b),(x_3+a,y_3+b))$.

Figure 2.5 presents a translation example.

Using a matrix notation, and homogeneous coordinates, translations can also be represented as:

$$\begin{bmatrix} x_T & y_T & 1 \end{bmatrix} = \begin{bmatrix} x & y & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ T_x & T_y & 1 \end{bmatrix}$$

2.2.1.2 Rotation

Rotation is an isometric transformation. Rotations around the origin, considering an angle θ, transform the coordinates of a generic point (x,y) into $(x+cos\theta - ysin\theta, xsin\theta + ycos\theta)$. Since rotations are isometric, they do not change the polygon's shape.

Figure 2.5 presents a rotation example, where the image is rotated 45^{o}.

Using a matrix notation, rotations can be portrayed as:

$$\begin{bmatrix} x_R & y_R \end{bmatrix} = \begin{bmatrix} x & y \end{bmatrix} \cdot \begin{bmatrix} cos\theta & sin\theta \\ -sin\theta & cos\theta \end{bmatrix}$$

Using homogeneous coordinates, the corresponding representation is:

$$\begin{bmatrix} x_R & y_R & 1 \end{bmatrix} = \begin{bmatrix} x & y & 1 \end{bmatrix} \cdot \begin{bmatrix} cos\theta & sin\theta & 0 \\ -sin\theta & cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Rotation, around a point — pivot — that is not the origin, is achieved by first performing a translation that positions the pivot at the origin. Then, the regular rotation is performed, followed by the inverse translation, thus repositioning the pivot at its initial position [27].

2.2.1.3 Reflection

Reflection is an isometric transformation. Reflections are often only considered regarding the axes. In such cases, an horizontal reflection transforms the generic point (x,y) into $(-x,y)$. A vertical reflection transforms the same point into $(x,-y)$.

Figure 2.5 presents a vertical reflection example.

However, in a more generic approach, reflections can be achieved over any straight line going through the origin. For instance, consider a straight line $L\phi$, where ϕ represents the angle between the line and the horizontal axis. Furthermore, assume that $L\phi$ intersects the origin. In such cases, and using a matrix notation, reflections can be represented as:

$$\begin{bmatrix} x_{Re_{L\phi}} & y_{Re_{L\phi}} \end{bmatrix} = \begin{bmatrix} x & y \end{bmatrix} \cdot \begin{bmatrix} cos2\phi & sin2\phi \\ sin2\phi & -cos2\phi \end{bmatrix}$$

Additionally, instead of cartesian coordinates, we can use homogeneous coordinates:

$$\begin{bmatrix} x_{Re_{L\phi}} & y_{Re_{L\phi}} & 1 \end{bmatrix} = \begin{bmatrix} x & y & 1 \end{bmatrix} \cdot \begin{bmatrix} cos2\phi & sin2\phi & 0 \\ sin2\phi & -cos2\phi & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

2.2.1.4 Uniform scaling

Uniform scaling, also known as dilation, represents a transformation belonging to the similarity group since it preserves the object's shape. Uniform scaling transforms the generic point (x,y) into (sx,sy), where s represents the scaling factor. When the factor is smaller than one, the object shrinks. Conversely, if this factor is higher than one, the object enlarges.

Figure 2.5 presents a uniform scaling example, where the object is shrunk by a scaling factor of 0.5.

Vector notation can describe affine transformations as $\vec{x_A} = \vec{x}L + \vec{a}$, where L is a non-singular linear transformation and $\vec{x_A}$, $\vec{x}$, and $\vec{a}$ are row vectors. Considering this notation, in uniform scaling, $\vec{a}$ is null while matrix L is of the form (s represents the scaling factor):

$$\begin{bmatrix} x_S & x_S \end{bmatrix} = \begin{bmatrix} x & y \end{bmatrix} \cdot \begin{bmatrix} s & 0 \\ 0 & s \end{bmatrix}$$

And the correspondent homogeneous coordinates representation:

$$\begin{bmatrix} x_S & x_S & 1 \end{bmatrix} = \begin{bmatrix} x & y & 1 \end{bmatrix} \cdot \begin{bmatrix} s & 0 & 0 \\ 0 & s & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

2.2.1.5 Anisotropic scaling

Anisotropic scaling is an affine transformation. Therefore, it is no longer assured the conservation of the object's shape under transformation.

Anisotropic scaling changes the object's aspect ratio. It transforms the generic point (x,y) into (sx,ty), where s and t represent the horizontal and vertical scaling factors, respectively. The same scaling factor properties existent in uniform scaling apply to this transformation.

Figure 2.5 presents an anisotropic scaling example, where the object is enlarged by a factor of 2 in the vertical axis.

Once more, using the notation for affine transformations, presented in the previous spatial feature, in anisotropic scaling, $\vec{a}$ is null while matrix L is of the form:

$$\begin{bmatrix} x_{As_s} & x_{As_t} \end{bmatrix} = \begin{bmatrix} x & y \end{bmatrix} \cdot \begin{bmatrix} s & 0 \\ 0 & t \end{bmatrix}$$

And using homogeneous coordinates:

$$\begin{bmatrix} x_{As_s} & x_{As_t} & 1 \end{bmatrix} = \begin{bmatrix} x & y & 1 \end{bmatrix} \cdot \begin{bmatrix} s & 0 & 0 \\ 0 & t & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

2.2.1.6 Shear

The shear transformation is also an affine transformation. Shears transform a generic point (x,y) into $(ax+cy, bx+dy)$, considering a,b,c, and d from the transformation matrix:

$$\begin{bmatrix} x_S & x_S \end{bmatrix} = \begin{bmatrix} x & y \end{bmatrix} \cdot \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

Using homogeneous coordinates:

$$\begin{bmatrix} x_S & x_S & 1 \end{bmatrix} = \begin{bmatrix} x & y & 1 \end{bmatrix} \cdot \begin{bmatrix} a & b & 0 \\ c & d & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Using the affine transformations' notation, in the shear transformation, $\vec{a}$ is null while matrix L is of the form presented above.

Figure 2.5 presents a shear, where a assumes the value 0.96, b assumes 0.3, c 0.95, and lastly d the value 0.96.

2.2.1.7 Projection

Projections belong to the group of projective transformations. For projections' matrix notation, we use homogeneous coordinates (as the explanation becomes simpler). Both matrix L and $\vec{a}$, from the affine transformations' notation, are combined into a single matrix A:

$$A = \begin{bmatrix} a & b & p \\ c & d & q \\ u & v & s \end{bmatrix}$$

Explaining matrix A values, elements a,b,c and d come from matrix L. Elements u and v represent $\vec{a}$. Lastly, p,q and s represent the projection. We can see the result of the application of a projection to the generic point, in homogeneous coordinates $(x,y,1)$, below:

$$\begin{bmatrix} ax+cy+u & bx+dy+v & px+qy+s \end{bmatrix} = \begin{bmatrix} x & y & 1 \end{bmatrix} \cdot A$$

Figure 2.5 presents a projection, using the coordinate transformation $(x,y) \to (\frac{1}{x}, \frac{y}{x})$.

2.2.1.8 Conformal Mappings

Conformal mappings are conformal transformations. For representing this class of transformations, we need to make use of the complex plane. Due to the complexity of this transformation, and the extensive correspondent explanation required, we only visualise the effects of this transformation. Nonetheless, Wilkinson [95] provides a detailed analysis of this transformation. Figure 2.5 presents a conformal mapping, using the transformation $w = \frac{1-z}{1+z}$, where $z = x + iy$.

2.2.1.9 Deformations

Under deformations, we consider all the changes of shape that were not classified by the previously presented geometrical transformations. Therefore, deformations encapsulate all of the transformations on the general transformations group identified in sub-section 2.1.4. Additionally, other identified (in section 2.1) but uncategorised shape changes, such as partial motion, fit under the deformations category.

Several classes further categorise, model and detail deformation itself, such as deformations that arise from merging with a simple shape [21], free-form deformations [77], or wire modelled deformations [80]. However, we will not further investigate the developments in this area, since we manage deformations as a whole entity, not making distinctions of its subtypes.

Figure 2.6 presents an example of a deformation, more specifically a homeomorphism, of a shape, between two consecutive timestamps. We expect to be working with the sort of deformations that this figure denotes.

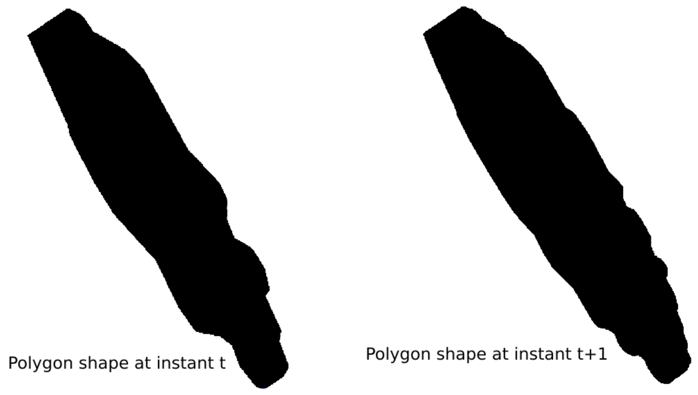


Figure 2.6: Example of a geometric deformation.

2.2.2 Existential features

In this subsection, we analyse features that alter the existential state of phenomena. Thus, with these features, we model the change that concerns the creation of new phenomena, or, conversely, the destruction of existing phenomena. Moreover, we also model the merge of distinct phenomena into a single phenomenon and, conversely, the split of phenomena. Unlike spatial-related features, that may spawn over several timestamps, existential features have an exact timestamp at which they occur. This timestamp can either be the moment the phenomenon is created or destroyed. Thus, there is a moment, in time and space, at which the phenomenon starts (and, later, may stop) existing.

This subset of spatiotemporal features was identified in the works of both Blok [8] and Shekhar et al. [78].

2.2.2.1 Appearance

The appearance of a phenomenon consists of the emergence of a new phenomenon (its "birth" [8]). We verify this feature whenever in a given timestamp *t* the phenomenon is inexistent, but in the next timestamp *t+1* the phenomenon is present. Figure 2.7 presents a visual example of an appearance.

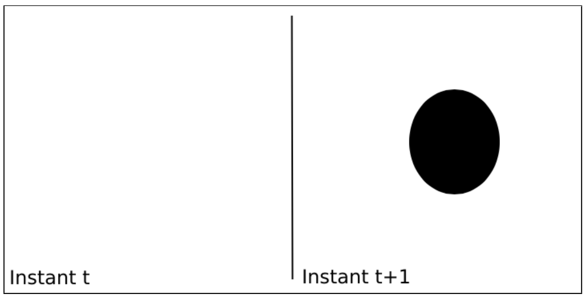


Figure 2.7: Example of a phenomenon appearance.

2.2.2.2 Disappearance

The disappearance of a phenomenon consists of the vanish of an existing phenomenon (its "death" [8]). The identification of this feature consists of a scenario wherein a given timestamp *t* the phenomenon is present, but in the next timestamp *t+1* the phenomenon is no longer observable. It stands as the inverse of the appearance feature. Figure 2.8 presents a visual example of a disappearance.

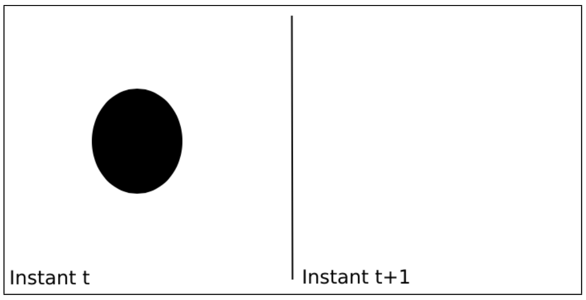


Figure 2.8: Example of a phenomenon disappearance.

2.2.2.3 Split

A split consists of the disaggregation of a phenomenon into two independent phenomena. The initial phenomenon that suffered the split can either disappear since the complete entity represented the phenomenon (e.g., the collapse of a bridge); or continue existing as one of the resulting split parts (as occurs in mitosis). Figure 2.9 presents a visual example of a split.

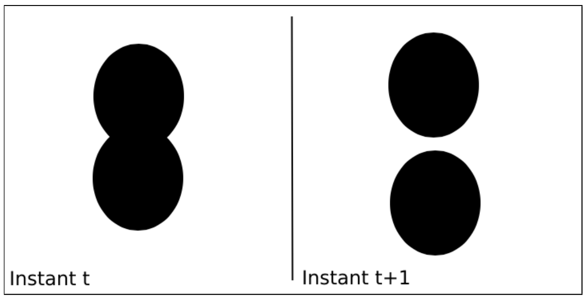


Figure 2.9: Example of a split of one phenomenon into two phenomena.

2.2.2.4 Merge

A merge consists of the aggregation of two phenomena into a single phenomenon. It stands as the inverse of the split feature. Therefore, the initial phenomena that suffered the merge can either disappear to be replaced by a new entity; or one of the original phenomena continues existing as the merge result. Figure 2.10 presents a visual example of a merge.

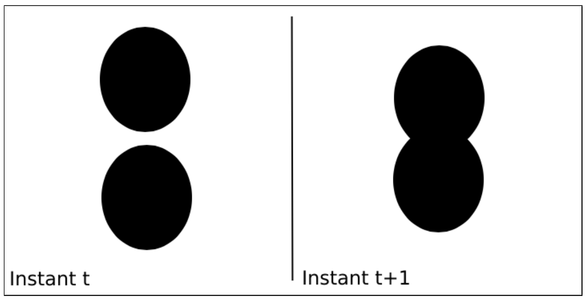


Figure 2.10: Example of a merge of two phenomena into one phenomenon.

2.2.3 Temporal-related features

In this subsection, we focus on features that depend on the occurring changes' temporal properties. Thus, these features, identified in Blok's work [8], rely upon the identification of other dynamic changes, such as spatial-related features.

Figure 2.11 presents a visual representation of all the time-related features presented below.

The temporal features are:

1. **Moment in time** refers to the timestamp at which a given change starts occurring;

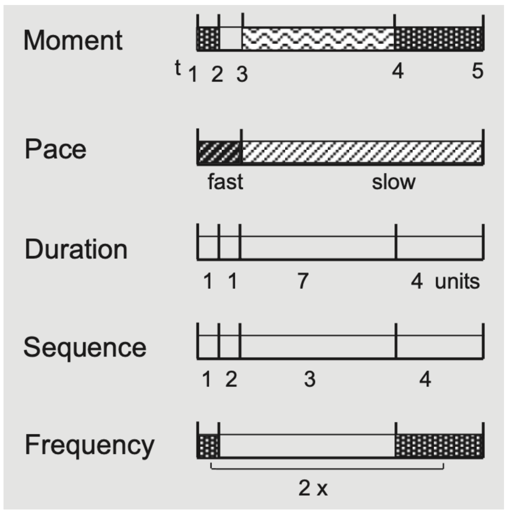


Figure 2.11: Characteristics of change in the temporal domain, as presented by [8].

2. **Duration** consists of the period during which a change occurs;
3. **Pace** quantifies the rate at which changes occur over time. For instance, we can interpret velocity as the pace of execution of a particular motion. Consequently, acceleration can be seen as the change of pace in velocity;
4. **Sequence** refers to the order by which changes occur in the temporal domain;
5. **Frequency** consists of the number of times that a given change occurs, over the temporal domain.

2.2.4 Movement-derived features

In this subsection, we focus on features that derive from other computed features, whether they are primary features, like the spatial and temporal-related features we already identified, or other derived features. We base this subset of features on the work of Dodge et al. [23]. This subset of features has its foundation on physics fundamentals, specifically in the kinematics area. Thus, for better explaining the identified features, we recur to Villate's work [90].

2.2.4.1 Velocity

Consider the presence of an object in instant t_i at position s_i and at instant t_{i+1} at position s_{i+1}, where $t_{i+1} = t_i + \Delta t_i$. The motion through the trajectory, at the interval $[t_i, t_{i+1}[$, can be computed as $\Delta s_i = s_{i+1} - s_i$. Then, we can compute the **average speed**, for the interval $[t_i, t_{i+1}[$, as the motion by unit of time, hence: $\bar{v}_i = \frac{\Delta s_i}{\Delta t_i}$. The absolute value of velocity, $|\bar{v}_i|$, is the speed at which the object moves. Units of speed are distance over time (e.g.: meters per second, m/s, or kilometre per hour, km/h). If the time interval, Δt, is too small (limit $\Delta t \to 0$), the average velocity value approximates to the **instantaneous velocity** value. This limit is the **derivative**. Hence, the velocity (implicit instantaneous velocity) can be described as: $v = \frac{\delta s}{\delta t}$.

This feature can be applied for computing the velocity of trajectory oriented movements such as translations and rotations [30].

2.2.4.2 Acceleration

As stated by Villate [90], acceleration can be decomposed into two components: tangential acceleration, the component dependent on the variation of trajectory; and the normal component, unrelated to velocity variations but related to the trajectory's curvature. For the remaining of this work, whenever we mention acceleration, assume we are talking about tangential acceleration.

Acceleration measures velocity's rate of change per unit of time. For an interval $[t, t+\Delta t[$, the **average acceleration** is equal to the increase of velocity per unit of time: $\bar{a}_t = \frac{\Delta v}{\Delta t}$. Conversely, we obtain the instantaneous acceleration, at instant t, through the limit $\Delta t \to 0$. Hence, **acceleration** a_t equals the derivative of velocity in order to time, thus being described as: $a_t = \frac{\delta v}{\delta t}$.

2.2.4.3 Change of direction

Change of direction is a secondary derivative feature that depends on a movement's direction. Thus, it becomes necessary to compute the phenomenon's translations first. We believe this feature's relevance is better explained through an example: consider a car on the road. The car is moving forward. However, the driver decides to stop and inserts the reverse gear, consequently making the car move backwards. If we decompose the movement into geometrical transformations, we identify two distinct translation vectors: one that represents the movement of the car moving forward, and another when the car moves backwards. However, the car did not change its orientation. Even though the car moves backwards, it still faces its initial direction.

The previous example highlights the importance of identifying the change of a movement's direction.

2.2.5 Spatiotemporal patterns

This subsection focuses on the recurrence of the previously mentioned features. In other words, spatiotemporal patterns focus on identifying regular, repeatable and understandable sequences of sets of changes.

The spatiotemporal patterns subset, identified in Blok's work [8], are:

1. **Cycles** represent changes that repeatedly occur in the same order, meaning the periodical return to a previous state of the phenomenon (e.g. changes in vegetation);

2. **Trends** represent structured acyclic patterns, meaning the tendency of a given change over time (e.g. steady increase of a movement's velocity).

2.2.6 Summary

In this section, we identified and categorised the spatiotemporal change features. We contemplate five different categories: spatial-related features, existential features, temporal-related features, derivative features and spatiotemporal patterns. Observe that both the temporal features and the derivative features are dependent on the spatial-related ones. Additionally, spatiotemporal patterns are dependent on any of the other features. Figure 2.12 shows all of the identified change features as well as the respective categories.

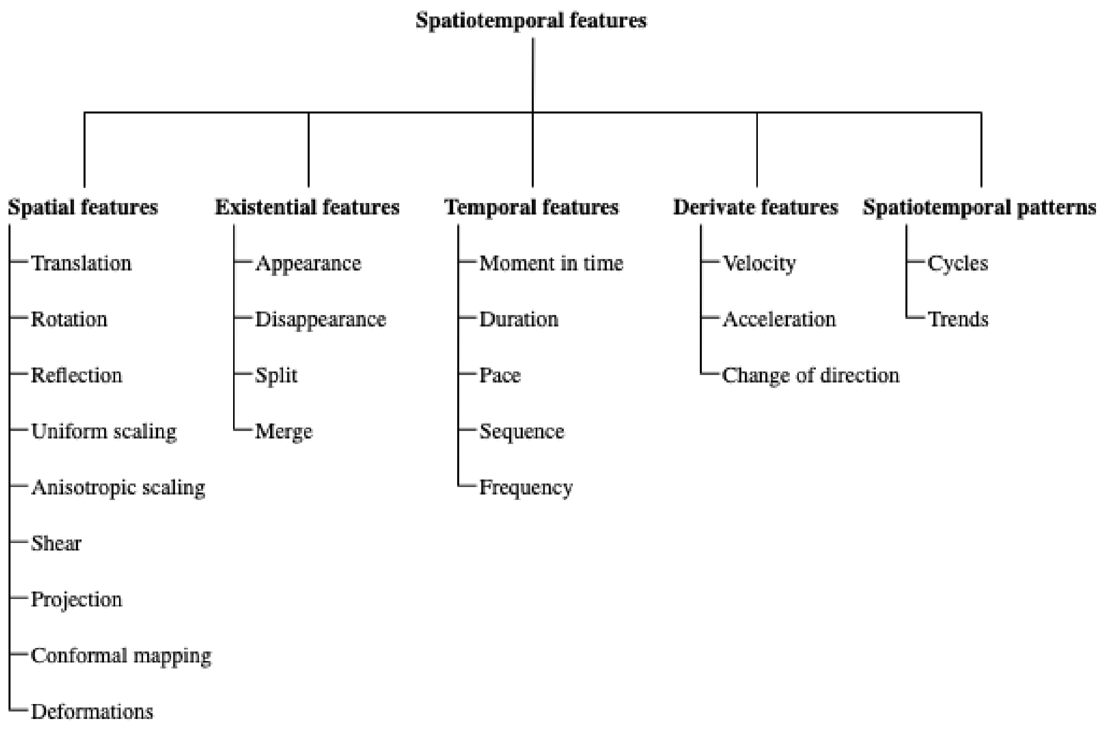


Figure 2.12: Identified spatiotemporal change features and respective categories.

2.3 Feature computation and interest

Next, we focus on the possible methodologies for computing the spatiotemporal change features distinguished in the previous section. We start by analysing the point set registration algorithm: an algorithm to estimate the geometric transformations occurring between two given temporal snapshots of a phenomenon. Following, we examine different similarity measures and how to compute each one. Both point set registration algorithms and similarity measures appear as suited approaches for computing the spatial-related change features. Afterwards, we analyse the calculation of derived features. Subsequently, we identify possible methodologies for the computation of spatiotemporal patterns. Next, we describe other authors' possible approaches to the evaluation of change features scenarios. Finally, we analyse methodologies to determine which change features are of interest.

In this section, we do not analyse temporal-related features. Temporal-related features are dependent on the identification of spatial-related features. Hence, whenever a spatial change is identified, some temporal features are instantaneously computed (moment in time and the duration). Besides pace — which we already identified as being inherently correlated with derived features —, the remaining temporal features, sequence and frequency, seem somewhat trivial to identify by looking at the identified features order.

2.3.1 Point set registration algorithms

Point set registration algorithms main focus is 'to find correspondences and to estimate the transformation between two or more point sets' [100]. The algorithm consists of **registering** (aligning) the source point set into the target point set. However, the point sets' characteristics can create challenges to the algorithm. Deformations such as changes in the object's viewpoint or pose, or noise such as the occlusion of some points or outliers, are some examples of the difficulties point set registration algorithms encounter. Figure 2.13 presents these same challenges while making use of visual examples to facilitate comprehension.

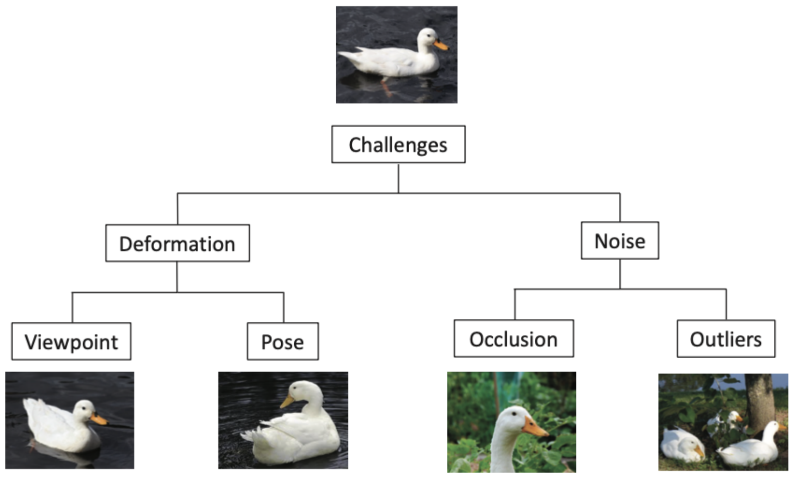


Figure 2.13: Main challenges of the point set registration algorithm, presented in [100].

To better understand the point set registration algorithm, figure 2.14 illustrates its application in a 3D scenario. In this figure, the blue circles are being aligned onto the red dots. Each column represents an iteration of the algorithm. The first row represents a scenario with occlusion of data (missing points). The second row represents a scenario with less missing points but also outliers.

As already observed in sub-section 2.1.4, geometric transformations can be classified into different categories. Section 2.2, provided us with the geometric transformations belonging to each category. Rigid transformations preserve the distance between points, thus encapsulating translations and rotations. Affine transformations conserve the ratios of the distances between points, hence encasing scalings (both uniform and anisotropic) and shears. Regarding the nature of the transformations that point set registration algorithms can detect, these are classified into rigid and non-rigid transformations. However, this classification does not match the one we presented, since, in the point set registration algorithm domain, rigid transformations include translations, rotations, and uniform scaling [63, 100]. Non-rigid transformations refer to transformations that

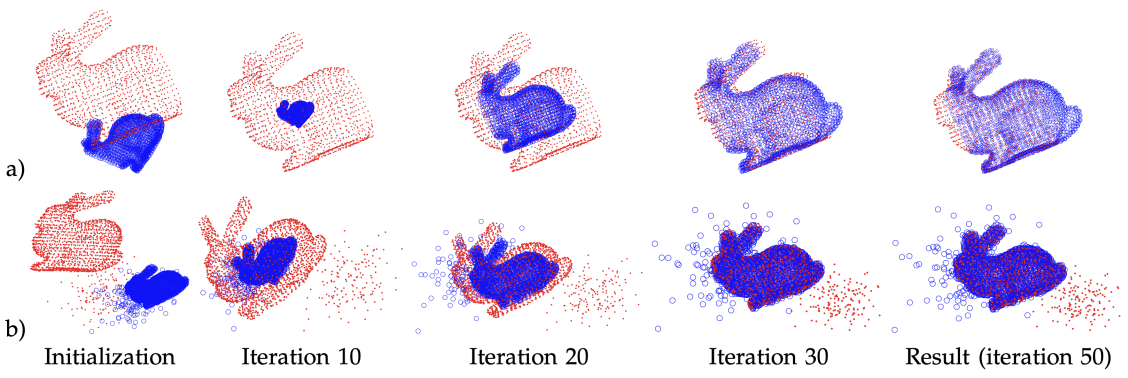


Figure 2.14: Example application of a point set registration algorithm, presented in [63].

are challenging to model and whose underlying model is often unknown. Affine transformations constitute the simplest case of non-rigid transformations [63].

Henceforth, when considering point set registration algorithms, we consider rigid transformations to include uniform scaling, to match the domain language used in the point set registration literature review.

Regarding the modelling methods in point set registration, they fit into one of two categories: parametric or non-parametric models [100]. If the model is **parametric**, then it depends only on a small set of parameters. Hence, on parametric models, the parameters provide insight regarding the transformations affecting the point set as a whole. Otherwise, in **non-parametric** models, the transformation cannot be represented by a set of parameters, implying the existence of deformations vectors for each point [73]. Thus, despite the registration being more precise, it is difficult to understand the transformations affecting the point set.

Point set registration algorithms can be of two types: pairwise or groupwise [100]. In **pairwise registration**, the algorithm receives two sets of points — the source and the target — as input. Contrarily, **groupwise registration** handles more than two point sets simultaneously. Seeing that our input consists of successive shape descriptors of a phenomenon, and our goal is understanding the change occurring between two of those shape descriptors, we focus on a more in-depth analysis of pairwise registration.

2.3.1.1 Pairwise registration

Jian and Vemuri described pairwise registration using the following mathematical representation [45]. Consider $\{\mathscr{M}, \mathscr{S}\}$ as the two finite sets of points, in a finite-dimensional real vector space $\mathbb{R}^d$, to be registered. $\mathscr{M}$ represents the moving "model" set — the source — while $\mathscr{S}$ represents the fixed "scene" set — the target. The point sets can have different sizes. The aim is to find the mapping, from $\mathbb{R}^d$ to $\mathbb{R}^d$, that produces the best alignment between the registered "model" point set and the "scene" point set. As observed, this mapping can either be a rigid or non-rigid transformation. Consider T to be the transformation mapping $\mathscr{M}$ into its registered representation

$T(\mathscr{M})$. The goal can be rewritten as finding the transformation T that minimises $dist(T(\mathscr{M}),\mathscr{S})$, where *dist* represents an algorithm for computing the distance between two point sets.

Hao et al. categorise pairwise registration methods into distance-based, filtering-based, or probability-based [100]. Next, we describe each one of these categories, while supporting ourselves in Hao et al.'s work.

Distance-based methods

Distance-based pairwise registration methods consist of two steps. First, compute the distance for each point in $\mathscr{M}$ to each point in $\mathscr{S}$ to find the correspondence between the point sets. The following step consists of attempting to minimise the distances between the corresponded points.

The *iterative closes point* (ICP) algorithm is a well understood and popular parametric distance-based method, introduced by Besl and McKay [7]. The algorithm assumes that the correspondent of a point in the $\mathscr{M}$ set is the closest point in the target $\mathscr{S}$ set. The algorithm consists of iteratively computing the transformation T, using least-squares[3] as the distance metric, and applying $T(\mathscr{M})$. The algorithm stops when a particular criterion is met. Many ICP variants have emerged since the publication of the original algorithm.

Another parametric distance-based algorithm is *robust point matching* (RPM), introduced by Gold et al. [37]. In RPM, the registration proceeds by using deterministic [4] and soft-assign optimisation to converge into the best-alignment transformation. The *thin-plate-spline robust point matching* (TPS-RPM) consists of an RPM variant, that enables non-rigid registration by parameterising the spatial transformation as a thin-plate spline[5].

Kernel correlation (KC), introduced by Tsin and Kanade [85], represents another parametric distance-based algorithm. In registration, the goal is to obtain the maximum kernel correlation of the point sets. According to the original authors, KC 'can be considered as a robust, multiply-linked ICP', since it is proven to be more robust in noisy scenarios.

Jian and Vemuri represented the point sets as *Gaussian mixture models* (GMM) [45]. The registration consists of aligning the two models. Furthermore, they use parameterisation by thin-plate splines to perform non-rigid registration.

Graph matching (GM) is a non-parametric distance-based algorithm. In this approach, the point sets are represented using graphs that store information regarding both edges and vertices. The registration consists of finding the correspondence between the two graphs, through the edges and vertices modelling.

Filtering-based methods

[3]Approximation technique for regression analysis [76]. The objective is to minimise the sum of the squares of the residuals of every equation.

[4]Deterministic variant of the Simulated Annealing technique [74]. Simulated Annealing is a probability-based technique to approximate the global optimum of a function.

[5]Spline-based technique for both data interpolation and smoothing [26].

For performing point set registration, filtering-based methods use state-space models (SSM). State-space models consist of ‘models that use state variables to describe a system by a set of first-order differential or difference equations’ [59]. There are no inherent subsets of algorithms, as there are in distance-based methods. Nonetheless, there exist several variations of filtering-based algorithms [100].

Probability-based methods

In probability-based methods, the alignment of the two point sets is addressed as a probability estimation problem. The *coherent point drift* (CPD) consists of a robust pairwise registration algorithm for both rigid and non-rigid transformations [63]. The authors propose an alignment as a probability density estimation problem, more specifically a maximum likelihood estimation[6] problem, using a Gaussian mixture model method. One of the point sets represents the GMM centroids, while the other represents the data points to be fitted. Next, using an expect-maximisation algorithm, they perform the maximum-likelihood optimisation. The CPD algorithm serves as the scientific basis for many other registration algorithms, such as the CPD-GL [56] or the CPD-Net [93].

Hao et al. chose some representative methods for pairwise point set registration and conducted some performance experiments on them [100]. They compare six different algorithms with different configurations, being those: the ICP [34], a *iterative closest point algorithm* algorithm; the TPS-RPM [17], a *robust point matching* algorithm; the KC [85], a *kernel correlation* algorithm; the CPD [63], a *coherent point drift* algorithm; the CPD-GL [56], also a *coherent point drift* algorithm; and the SCGF [91], a probability-based method. The authors run the experiments on two scenarios, the fish and Chinese character datasets, while varying noise, occlusion, rotations and object deformation. The algorithms ran until they converged or reached 50 iterations.

Figure 2.15 presents the results obtained. By the analysis of the results, Hao et al. verified that the SCGF algorithm had better accuracy than the remaining algorithms. However, its average runtime was, by a large margin, superior to the remaining algorithms — up to 74 times slower than the second slower algorithm (TPS-RPM) in the fish dataset (SCGF took 37.04s while TPS-RPM took 2.37s), and 37 times slower than the second slower (CPD-GL) in the Chinese dataset (SCGF took 27.64s while CPD-GL took 0.73s). Additionally, they also confirmed that the **CPD** algorithm proved to be the most time-efficient one (took 0.22s in both the fish and Chinese dataset). Furthermore, the accuracy of the CPD-GL algorithm is very close to the SCGF. Therefore, even though the SCGF had the best accuracy, the **CPD-GL** — *Coherent Point Drift - Global Local* — proved to be the algorithm with the best accuracy-average runtime trade-off, since it was able to achieve very similar results up to 37 times faster. The CPD-GL is a probability-based method, with a parametric model, for non-rigid transformations registration.

[6]Estimation method that works by maximising a likelihood function (measures the fit of a statistical model into a data sample) [76].

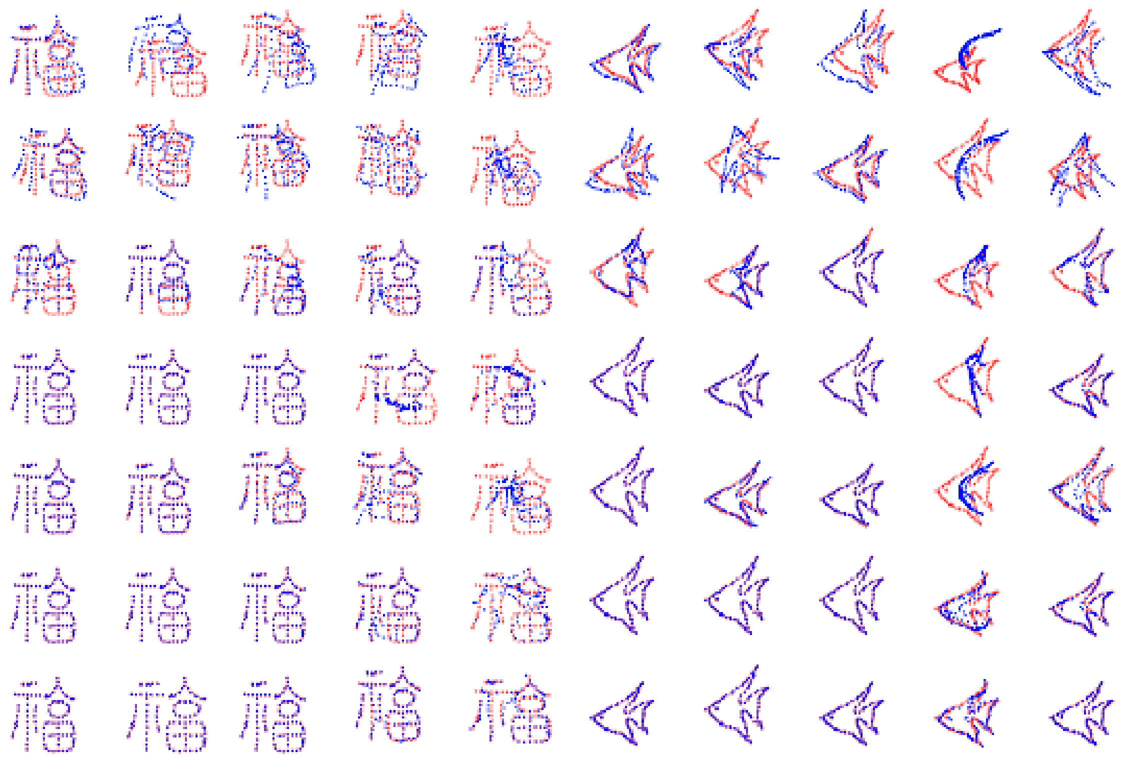


Figure 2.15: Experiments run by Hao et al. [100]. Each column represents a different registration, where the blue point set represents the "model", and the red point set represents the "scene". The first row represents the points sets when no algorithm is applied. The following rows represent the IPC, KC, ICP, TPS-RPM, CPD, CPD-GL, and SCGF results (meaning the "model" after applying the estimated transformation, superimposed on the "scene"), respectively. Original figure from [100].

2.3.2 Similarity measures

Point set registration algorithms are not capable of directly determining the more complex spatial-related features identified in section 2.2, such as conformal mappings or projections. Hence, similarity measures appear as a complement (and also as a possible alternative) to point set registration algorithms.

Veltkamp states that it is possible to use similarity measures for measuring the resemblance between two shapes [89]. With two shapes and a similarity measure, it becomes possible to decide whether the dissimilarity is smaller than a given threshold. Therefore, we can use similarity measures for computing the degree of deformation between two temporal snapshots. According to the author, ideally, similarity measures respect a set of three properties. The first property affirms that a distance function[7] (i.e., the dissimilarity between shapes) satisfies identity — distance between the shape and itself is zero —, uniqueness — if a distance between two shapes is zero then the shapes are equal —, and triangle inequality — for any triangle built out of three points, the sum of

[7]Function that defines the distance between two points. A point set with a metric distance is dubbed a metric space [13].

any two sides must greater than or equal to the remaining side [31]. The second property relates to continuity and robustness. A distance function should hold: perturbation robustness, crack robustness, blur robustness and noise and occlusion robustness. Lastly, a distance function should be invariant (the property of remaining unchanged independently of variations in the conditions of measurement[8]) under certain geometric transformations, such as isometric or affine transformations (see 2.1.4). However, not all the similarity measures described by the author respect simultaneously the three properties.

Veltkamp also presents several similarity measures [89]. The following list presents the ones that respect the above properties and suit our problem.

- L_p **Distance / Minkowsky Distance** [88]: this measure represents a generalisation of both the widespread Euclidean distance and the Manhattan distance. Therefore, one can compute the distance between two shape points and use it to infer the similarity between the shapes. The formula for calculating the Minkowsky distance is $(\sum_{n=1}^{k}|x_i - y_i|^p)^{1/p}$, where x and y represent two points in $\mathbb{R}^k$ and p is a parameter of the formula. For the triangle inequality property to hold, p must be greater than one. The Euclidean distance is the Minkowksy distance with $p = 2$, while the Manhattan distance has $p = 1$.

- **Bottleneck Distance**: given two point sets, with the same size, the bottleneck distance finds the one-to-one correspondence linking both point sets that minimises the maximum distance between the corresponded points [29]. For measuring the correspondences' distance, another metric must be used (e.g. the Minkowsky distance).

- **Hausdorff Distance**: represents the greatest distance of a shape's point to the closest point in the second shape. This metric is still valid in point sets without a one-to-one correspondence, hence being robust to noise and occlusion. Figure 2.16 presents a visual example of this metric application.

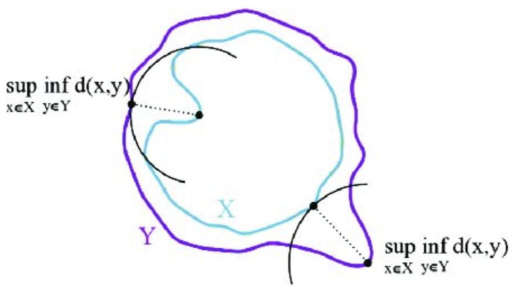

$\sup_{x\in X}\inf_{y\in Y} d(x,y)$
X
Y
$\sup_{x\in X}\inf_{y\in Y} d(x,y)$

Figure 2.16: Example of the *Hausdorff Distance*, presented in [87].

- **Turning Function Distance**: given two shapes, map each one into the turning function of its inner angles. Next, proceed with determining the distance between each of the obtained

[8]Dictionary.com. Definition of invariance. Available at https: https://www.dictionary.com/browse/invariance. Accessed in 2020-02-08.

turning functions. Narayanan presents a real-life application of this metric — analysis of the seasonal deformation of a lake — and respective effectiveness validation [64]. Figure 2.17 presents an example of the application of this metric.

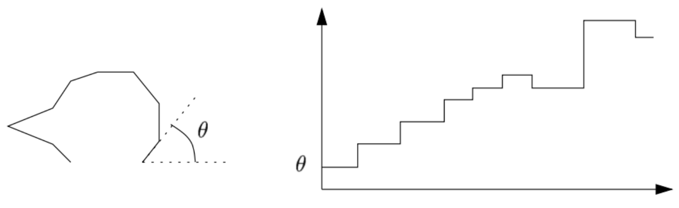


Figure 2.17: Example of a polygon's mapping into its turning function, presented in [89].

- **Fréchet distance**: minimum distance existent between correspondences of two shapes. The correspondences are obtained through increasing monotone parametrisations of both shapes. Figure 2.18 presents an example of the *Fréchet distance* applied to two lines.

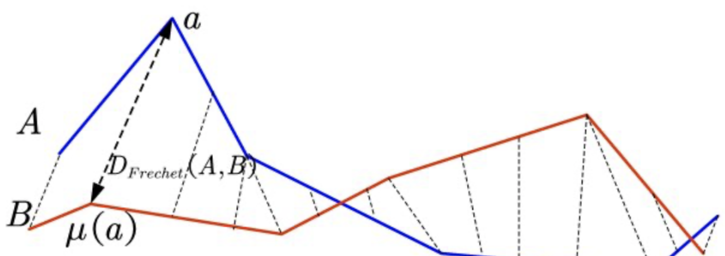


Figure 2.18: Example of the *Fréchet distance*, presented in [39].

- **Template metric**: considering two shapes (S_1 and S_2), the *template metric* assumes value equal to $area((S_1 - S_2) \cup (S_2 - S_1))$. The centroids of both shapes must coincide when calculating this metric.

- **Transport Distance**: metric equals the minimum amount of work (amount of energy to move a mass) needed to transform a shape into another. This metric has its foundations on physics. Su et al. present a more time-efficient variation and prove the efficacy and efficiency of this measure [83].

Since Veltkamp's work, other similarity measures have emerged, such as:

- Nasreddine et al. propose an approach based on **shape geodesics** [65]. A Geodesic line is a line 'denoting the shortest possible line between two points on a sphere or other curved surface'[9]. This methodology is invariant to geometric transformations, such as translations, rotations, and scaling. Furthermore, the similarity measure is robust to outliers.

[9]Lexico. Meaning of Geodesic by Lexico. Available at https: `https://www.lexico.com/definition/geodesic`. Accessed in 2020-02-02.

- Mémoli [60] presents a similarity measure based on the **Gromov-Hausdorff distance** [86], a feature dependent on the already observed Haussdorf distance. Mémoli introduces a more natural and general approach that can also be applied to unsmoothed datasets.

Besides the similarity measures already observed, Veltkamp also describes two more generic algorithms for measuring shape similarity [89]. The first, **voting schemes**, consists of counting votes to identify the target point set that maps to the given query point set. Geometric hashing [96] is an example of a voting scheme algorithm for matching geometric features from a query point set with a database (the hash) containing similar features.

The second algorithm is **subdivision schemes**. This algorithm consists of a geometric branch and bound technique. A progressive subdivision of the transformation space decides against the set of transformations occurring.

Similarity measures can also be used for finding the geometrical transformation that maximises the similarity between the two given shapes. In such cases, shape matching becomes an optimisation problem — the goal is to find the transformation that presents the global minimum for the dissimilarity measure computation. Since solving optimisation problems might lead to time-inefficient solutions, approximate-optimisation approaches have also been devised. It is possible to use the *Bottleneck distance* metric to understand if there exists a translation between two shapes whose dissimilarity measure is inferior to a given threshold [28, 29]. In a similar approach, it is feasible to identify the rigid transformation between two shapes, using the *Hausdorff distance* [16]. Point set registration algorithms using distance-based methods make use of similarity measures to evaluate the distance between the two scenes. For example, the original ICP algorithm uses Euclidean distance (Minkowsky distance with $p = 2$) as its distance function [7].

2.3.3 Movement-derived features

Assuming the use of a *point set registration* algorithm that provides the affine transformations affecting a phenomenon, in this subsection, we identify how to proceed for the computation of the identified movement-derived change features: velocity, acceleration, and change of direction.

Regarding the **velocity** feature, in section 2.2 we verified that the average speed formula is $\bar{v}_i = \frac{\Delta s_i}{\Delta t_i}$ [90]. Using the *point set registration* algorithm, the translation vector or the rotation angle and pivot are obtained, while also knowing the timestamps corresponding to the shape descriptors used in the algorithm. Therefore, it is possible to compute the average speed using the previous methodologies. In translations, the travelled distance is computed by calculating the translation vector's length (also known as magnitude or Euclidean norm), using the formula: $||\vec{v}|| = \sqrt{v_1^2 + v_2^2 + \ldots + v_n^2}$, where $\vec{v} = (v_1^2, v_2^2, \ldots, v_n^2)$ [4]. Considering the two timestamps as t_i and t_j, where $t_j > t_i$, velocity is $\frac{||\vec{v}||}{t_j - t_i}$. For rotations, consider a rotation of an angle θ, in radians, and with a radius R (computable as the distance separating the rotation pivot and the rotation starting point). Velocity is computed as $v = R\omega$, where $\omega = \frac{\Delta\theta}{\Delta t}$ [90]. Since the starting and finishing timestamps (t_i and t_j, respectively) and the rotation angle are known, the velocity calculus is $R\frac{\theta}{t_j - t_i}$.

As previously observed, **acceleration** is computed through $\frac{\Delta v}{\Delta t}$. Hence, using the calculated velocities and the corresponding timestamps, the acceleration for that interval is calculated.

For computing the **change of direction**, the *dot product*, also named *scalar product* [82], appears as a valid approach. The dot product permits the discovery of the angle between two vectors. In this work context, both vectors represent different translations, and the angle between them represents the change of direction. The algebraic definition of the dot product is $\vec{A} \cdot \vec{B} = \sum_{i=1}^{n} a_i b_i$, with $\vec{A} = [a_1, a_2, ..., a_n]$ and $\vec{B} = [b_1, b_2, ..., b_n]$ [52]. The geometric definition of the dot product is $\vec{A} \cdot \vec{B} = ||\vec{A}|| ||\vec{B}|| \cos(\theta)$, where θ is the angle between the vectors [82]. Knowing both vectors and using the above formulas, θ can be calculated as:

$$\theta = \cos^{-1}\left(\frac{\sum_{i=1}^{n} a_i b_i}{||\vec{A}|| ||\vec{B}||}\right)$$

2.3.4 Spatiotemporal patterns

As seen in section 2.2.5, spatiotemporal patterns can be decomposed in cycles or trends.

In the most simple scenarios, where a single change feature is identified, and it repeatedly occurs, either in cyclic or acyclic manner, the identification of spatiotemporal patterns is rather trivial. By looking at the order of the identified change features, the spatiotemporal patterns can be easily detected.

However, in more chaotic scenarios where several change features co-occur, for an extended period, the problem of identifying spatiotemporal patterns becomes considerably more complicated. Seeing that the problem consists of finding the set of change features that repeatedly occur, a possible approach is to address it as a combinatorial problem where all possible combinations of change features are tested as a spatiotemporal pattern [92, 18]. The main downside of this approach is that combinatorial space directly increases with the number of change features. Another possible approach is the usage of neural networks for the detection of spatiotemporal patterns [75, 22, 1].

2.3.5 Other approaches

In this subsection, we analyse other approaches, used by other researchers, for the identification of spatiotemporal change features.

Change detection based on Artificial Intelligence is also a common approach for identifying changes in spatiotemporal data [79]. However, machine learning approaches imply the existence of previous datasets for training the prediction model. Moreover, these approaches are also focused on specific spatiotemporal contexts (e.g. [61, 62]), seeing that the data sources must be homogeneous for the model to be able to predict the change features [79]. Furthermore, this is also the reason why the analysis of point set registration algorithms focused on machine learning, such as the CPD-Net [93] or the Deep Closest Point [94], has not been given further examination.

Bogaert et al. make use of a decision tree to determine which one of ten spatial processes, based on pattern geometry, occurred in a landscape transformation [9]. Deformations, enlargements and creations are some of the few patterns the authors determine that are interesting to us.

The Decision Tree algorithm receives as input the area, perimeter and number of patches regarding the land-cover given as input. Furthermore, the authors do explain how the input parameters relate to each one of the selected spatial processes. Figure 2.19 presents the decision tree used.

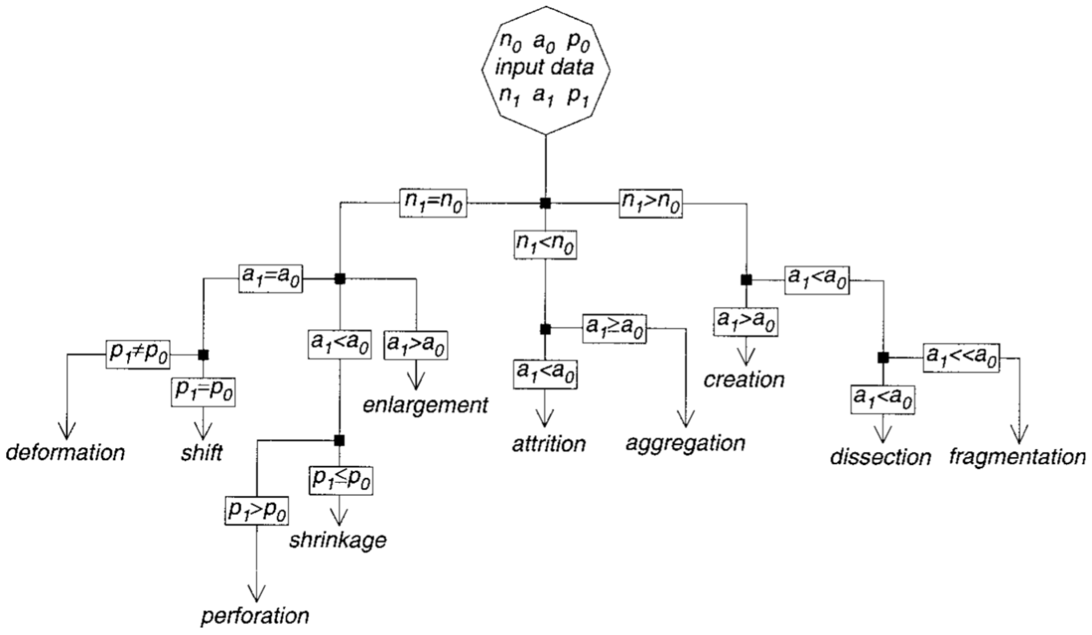


Figure 2.19: Decision Tree used in spatial processes classification, presented by [9]. Areas are represented using the a_k variable, perimeters with the p_k, and number of patches with the n_k variable.

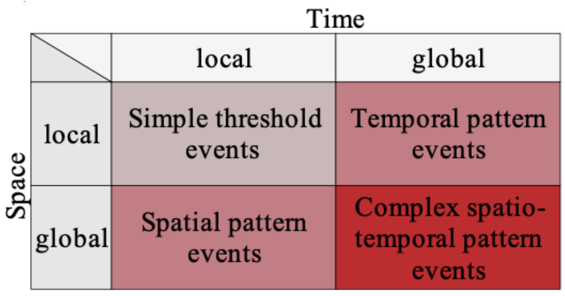
2.3.6 Determining features of interest

In section 2.2, we have determined a set of possible features identifiable in spatiotemporal phenomena. However, we have not yet disclosed how a spatiotemporal change feature is considered to be of interest to the user. In this section, we will focus on possible methodologies for classifying features (also referred to as events) as interesting or not.

In large datasets, information hiding emerges as a relevant topic [84], since there will be irrelevant information to the user that must be filtered. The work presented in [84] focuses mainly on spatiotemporal information visualisation and information hiding. Regarding information hiding, the authors define an event-based approach where the user defines events of interest. To this extent, users describe their events of interest using natural language operators. These operators are then processed into a set of thresholds, thus allowing the quantification of events. The ones surpassing the defined thresholds are considered interesting. The authors consider the existence of two types of thresholds:

- the exceeding of a certain threshold, **accumulated over different time steps**;
- the exceeding of a certain increase of an attribute value, **from one time step to another**.

The authors also consider the existence of different types of events of interest. Figure 2.20 presents these. Local events are events referring to one time step (temporal view) or a single area (spatial view). Conversely, global events are events detected in the entirety of the temporal continuum or the total spatial area.



Space \ Time	local	global
local	Simple threshold events	Temporal pattern events
global	Spatial pattern events	Complex spatio-temporal pattern events

Figure 2.20: Different type of events of interest, as presented by [84].

Similarly, to enable the user access to desired climatic data, Pederson et al. also employ thresholds of interest [72]. In this work, the user can parametrise the average and maximum temperatures.

2.3.7 Summary

In this section, we analysed the methods for computing the spatiotemporal change features identified in section 2.2.

Point set registration algorithms focus on finding correspondences between point sets and estimating the transformation between them. We have already observed that our input consists of shape descriptors as a list of points modelling the phenomenon — hence, point sets. Thus, the pertinence of the point set registration algorithms, for mainly identifying spatial-related features.

Next, section 2.3.2 described different similarity measures for evaluating the resemblance between two shapes, while explaining each one of them. Furthermore, it also examined their possible impact on determining spatial-related features.

Section 2.3.3 determined how to compute the derived features — velocity, acceleration, and change of direction — using the data obtained from the previous feature computation techniques.

Afterwards, section 2.3.4 described several approaches to the computation of spatiotemporal patterns.

Section 2.3.5 surveyed other possible approaches to the computation of spatiotemporal change features.

Finally, section 2.3.6 examined a possible methodology for classifying change features as interesting or not. This methodology is thresholds of interest. Thresholds of interest consist of letting the end-user define what is interesting, through inputted numerical thresholds that quantify change.

2.4 Similar approaches

After having identified how to compute and classify as interesting the various spatiotemporal change features, it becomes relevant to study which existing framework are capable of detecting and quantifying change on a spatiotemporal phenomenon. Therefore, in this section, we explore other approaches to similar problems and compare them to our problem.

Yi et al. present a framework for the study of spatiotemporal changes of dynamic geographic phenomena [98]. The framework uses a three-layer hierarchical organisation for modelling dynamic phenomena. At the basis of this hierarchy, there are static structures, meaning an entity's geometric shape as well as other attributes. Therefore, the hierarchy input is similar to the temporal snapshots we receive as our input. The framework then processes the static structures to extract spatiotemporal changes. In this work, the spatiotemporal changes consist only of what we refer to as existential features (see section 2.2). Thus, they do not consider any spatial transformation. For determining the existential features, the authors recur to image analysis of the phenomena, using pixel colours and contours to determine the existence of change. The obtained knowledge regarding the spatiotemporal changes is then stored in a database, where columns identify different phenomena and rows timestamps. Figure 2.21 presents the architecture of the framework when processing the example scenario of ocean eddies.

The framework presented by Yi et al. is unadaptable to our scenario since they only identify existential features and the algorithm to do so does not suit our input.

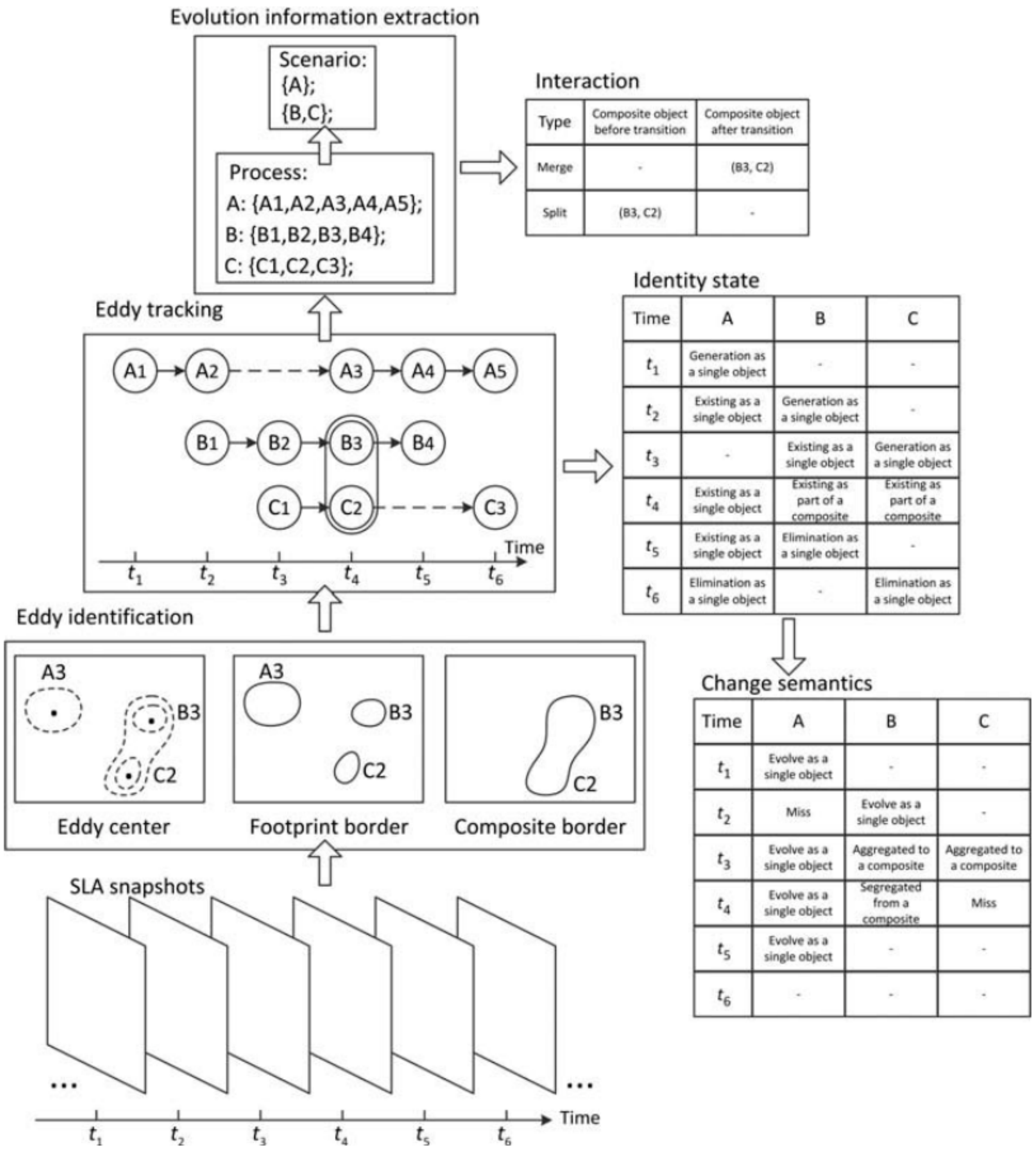
Duarte et al. present a framework for the analysis of interpolation on deformable moving regions [25]. The framework, SPT Data Lab, permits the end-user to visualise and refine 2D moving regions obtained from a set of temporal snapshots, through the use of interpolation methods. Moreover, the framework also allows the user to visualise and compare the results of different interpolation methods. For reporting and giving more insights to the user regarding the similarity of the interpolated moving regions, the framework employs a set of similarity metrics, namely Hausdorff distance and Jaccard Index [43].

Even though the input and domain of the framework proposed by Duarte et al. are similar to ours, the framework does not fit our scenario since they do not identify nor categorise changes occurring between moving regions.

2.5 Summary

In this chapter, we focused on the analysis of the literature related to the representation and quantification of change on spatiotemporal phenomena.

In section 2.1, we analysed different manners of modelling spatiotemporal data. In finer detail, we discovered that several authors agreed on three different ways of modelling and representing spatiotemporal data: using temporal snapshots, modelling object change, and through events and processes. Seeing that we receive a spatiotemporal phenomenon's data in the form of temporal snapshots as our input and that we intend on extracting the change that the phenomenon



Type	Composite object before transition	Composite object after transition
Merge	-	(B3, C2)
Split	(B3, C2)	-

Time	A	B	C
t_1	Generation as a single object	-	-
t_2	Existing as a single object	Generation as a single object	-
t_3	-	Existing as a single object	Generation as a single object
t_4	Existing as a single object	Existing as part of a composite	Existing as part of a composite
t_5	Existing as a single object	Elimination as a single object	-
t_6	Elimination as a single object	-	Elimination as a single object

Time	A	B	C
t_1	Evolve as a single object	-	-
t_2	Miss	Evolve as a single object	-
t_3	Evolve as a single object	Aggregated to a composite	Aggregated to a composite
t_4	Evolve as a single object	Segregated from a composite	Miss
t_5	Evolve as a single object	-	-
t_6	-	-	-

Figure 2.21: Framework introduced by [98], in the example scenario of the evolution of ocean eddies.

goes through, we work mainly on temporal snapshots and object change models. We also verified how to decompose spatiotemporal data, highlighting three different types of data attributes: non-spatiotemporal, spatial, and temporal attributes. Considering that movement represents a significant component of change on spatial (and consequently spatiotemporal) data, we scrutinised its modelling. Movement is essentially divided into two components: trajectory-oriented and internal motion. After a more thorough analysis of movement, we verified the strong correlations with geometric modelling and geometric transformations. Therefore, we categorised and described different types of geometrical transformations. Lastly, we identified the differences between discrete and continuous models.

In section 2.2, focused on spatiotemporal change features, we identified and categorised the spatiotemporal change features. We contemplate five different categories: spatial-related features, existential features, temporal-related features, derivative features and spatiotemporal patterns. We surveyed that both the temporal features and the derivative features are dependent on the spatial-related ones. Additionally, spatiotemporal patterns are dependent on any of the other features. Figure 2.12 shows all of the identified change features as well as the respective categories.

In section 2.3, on feature computation, we analysed the methods for computing the spatiotemporal change features identified in the previous section. Point set registration algorithms focus on finding correspondences between point sets and estimating the transformation between them. We have already observed that our input consists of shape descriptors as a list of points modelling the phenomenon — hence, point sets. Thus, the pertinence of the point set registration algorithms, for mainly identifying spatial-related features. Next, we described different similarity measures for evaluating the resemblance between two shapes, as an alternative/complement to point set registration algorithms. We also explained each one of them. Furthermore, we examined their possible impact on determining spatial-related features. We then determined how to compute the derived features — velocity, acceleration, and change of direction — using the data obtained from the previous feature computation techniques. In the following section, we determined several approaches for the computation of spatiotemporal patterns. Afterwards, we investigated other possible approaches to the computation of spatiotemporal change features. Finally, we examined a possible methodology for classifying change features as interesting or not. This methodology is thresholds of interest. Thresholds of interest consist of letting the end-user define what is interesting, through inputted numerical thresholds that quantify change.

In section 2.4, we explored other approaches to similar problems and identified the nuances that made it not viable to adapt their solution to our problem. Hence, from the literature review, we verified that the identified approaches are incapable of determining the spatiotemporal change features of interest. Therefore, in the following chapter, we detail the main dissertation problem with the intent of addressing it.

Chapter 3

Problem Statement

This chapter presents a detailed analysis of the problem initially described in section 1.2. Section 3.1 details the problem and the features of the most similar approach of the literature review is used as a benchmark. Next, section 3.2 details the research questions that this work intends to answer. Following, section 3.3 describes the requirements a solution needs to fulfil to tackle the problems presented in the first section. Finally, section 3.4 examines the scope of this work.

Before presenting this work's problem, it becomes necessary to establish the definition of three principal concepts in this work's context, extensively used henceforth, being those:

- **Spatiotemporal change feature** (also change feature), having already been presented in section 2.2, meaning the characterisation of an identifiable change occurring between two or more spatial representations of a phenomenon (two temporal snapshots);

- **Spatiotemporal transformations** (also transformations), which are spatiotemporal change features that describe the occurrence of change related to the spatial domain, such as translations, rotations, and others (see section 2.2.1 for the full disclosure of the possible spatial features);

- **Features of interest** (also changes of interest), which are spatiotemporal change features that are of interest to the user.

3.1 Problem

According to the performed literature review, more specifically section 2.4, few approaches exist that focus on extracting the spatiotemporal change features from the data original representation. From the surveyed alternatives, none is capable of being applied to a multitude of scenarios and extracting a variety of spatiotemporal features, related to the different domains inherent to spatiotemporal data. From the available approaches, the one presented by Yi et al. [98] is the one that more closely addresses our problem. Therefore, using Yi et al.'s approach as a benchmark, the research problem can be succinctly defined as a combination of several more precise sub-problems:

1. **Context-agnostic:** the inexistence of a context-agnostic spatiotemporal framework leads to the appearance of overfitted context-designed tools. The framework presented by Yi et al. uses an algorithm focused on the change of colours and contours; a characteristic that may not be present in all spatiotemporal scenarios;

2. **Analysis of spatiotemporal features:** the lack of a framework capable of analysing all the spatiotemporal features identified in section 2.2, leads to context and problem-oriented tools. A context-independent framework has added value when providing the analysis of the majority of the features that can be of interest to the end-user. The framework presented by Yi et al. only examines a subset of the available spatiotemporal change features: the existential features;

3. **Retrieval of information of interest to the user:** a generic framework should be customisable as to fit the context and goals of the end-user (and of its more specific scenario). When working with spatiotemporal data, different users might have different analysis objectives regarding the same data (e.g. a user is interested in the movement of the phenomenon while other is interested in the object's inner-motion). Therefore, the framework must actively provide a manner to specify the data or results that best fit the user needs. The framework of Yi et al. creates a database schema describing the change semantics of ocean eddies. However, this approach does not actively provide a manner for users to retrieve data of interest: they must first understand the data to posteriorly create a database query that finds the data that matches their interests.

3.2 Research Questions

Given the state-of-the-art solutions available, the main purpose of this work is to answer the following research question:

RQ: Considering a spatiotemporal phenomenon, how to automatically detect, extract and represent the changes of interest that encompass equivalent knowledge to the original representation? The change-based representation must not affect the end-user capability of understanding the spatiotemporal change features occurring in the original representation.

Since the research question presented is considerably abstract and complex and implies addressing a plethora of thematics (e.g. automatic generation, equivalent representations, among others), we chose to subdivide the research question into a set of research questions that complement each other. These research questions, as a result of the subdivision of the initial research question, end up having a more specific domain, narrower contexts, and thus more direct answers. The research questions resultant of the subdivision are:

RQ1: How to extract the spatiotemporal features occurring in the original representation? The basis for building a change-based representation is the identification of the spatiotemporal transformations occurring between the successive continuous spatial configurations of the phenomenon (i.e. the temporal snapshots describing the phenomenon). While for a human, the identification of a transformation can be intuitive and simple, for a machine, it is not trivial to identify which transformations occur between two spatial configurations. On the contrary, some other features can be imperceptible to humans (changes that were not obvious) but detectable by a machine.

RQ2: How to find the spatiotemporal features that interest the user? Different users can have different interests, and consequently, focus on different changes when consuming the original representation. Different focuses lead to the attainment of different knowledge. Hence, a change-based representation must be adjustable to the end-user interest, thus providing the end-user with valuable information.

RQ3: How to assert if the two representations are equivalent? Having generated an inequivalent change-based representation from the original representation, if an end-user consumes the change-based representation, the knowledge obtained is not similar to the one obtained when consuming the original representation. Therefore, it is crucial to assure that the generated change-based representation correctly identifies the changes in which the users are interested, so that they can obtain equivalent knowledge to the one obtained when consuming the original representation.

RQ4: How to automate the process? The original purpose of having end-users consuming a change-based representation rather than the original representation is to more efficiently and quickly provide them with knowledge (as seen in section 1.1). Thus, a process that requires a minimal amount of human assistance will satisfy the identified requirements.

3.3 Solution requirements

The requirements that a solution needs to fulfil to address the entirety of the problems presented in section 3.1, and thus, provide an answer to the research questions, are:

D1: Provide the end-user autonomy in dataset usage so that there are no constraints regarding the used datasets, therefore contributing to context-independency;

D2: Choose the spatiotemporal features that are analysed so that the solution only retrieves the features considered of interest (e.g. analyse transformations while not analysing uniform scalings);

D3: For a spatiotemporal feature, define in which conditions the feature is of interest so that only changes of significance are retrieved;

D4: Cover the whole temporal range queried so that all the changes of interest are described. Covering the whole temporal range queried also ensures that the full original representation is mapped into the change-based representation;

D5: **Retrieve changes that are considered of interest** by the standards provided as an input by the user;

D6: **Retrieve the resulting values associated with the identified features of interest** to be able to recreate a continuum representation that covers the features of interest and their evolution.

3.4 Scope

Considering that the temporal horizon available for the development of a solution is limited, it becomes relevant to establish which features of the solution have priority in terms of prototype implementation, experiments, collection and analysis of results. The selected features with most priority are (1) a solution capable of recognising the most elementary spatiotemporal features (e.g. rigid transformations) and (2) a solution easily extendable and reusable. For the time being, certain characteristics, such as better performance and more complex and unusual spatiotemporal features, are considered having less priority.

It also becomes important to characterise the solution's end-user, with the goal of understanding who can benefit from such a solution. Therefore, the foreseen end-users are:

1. People using large amounts of spatiotemporal data in their work;
2. People needing to describe, using a change-based representation, the spatiotemporal features of interest occurring over a temporal range;
3. In a more general way, anyone with basic knowledge regarding spatiotemporal data, which may need to consume/analyse the behaviours of spatiotemporal phenomena.

3.5 Summary

Section 3.1 starts by presenting the problem as an aggregation of smaller issues and comparing each of the issues to the approach adopted in the framework proposed by Yi et al. [98]. Next, section 3.2 identifies the main research question, that is further subdivided into four complementary narrower research questions, that this work intends to answer. Section 3.3 described a total of six requirements that a solution needs to fulfil to address the identified problems. Lastly, section 3.4 examined the scope of the problem, established the features that had priority in the implementation of a solution and identified the end-user of the developed solution.

Chapter 4

Features of interest quantification

This chapter presents the proposed solution for identifying and quantifying features of interest. Section 4.1 presents the main guidelines and concepts of the devised solution. Next, section 4.2 provides a more in-depth analysis of each of the main components of the solution. Likewise, section 4.3 details the secondary components that integrate the solution. Section 4.4 describes the output of the solution's processing pipeline. Finally, section 4.5 presents a summary of this chapter's contents.

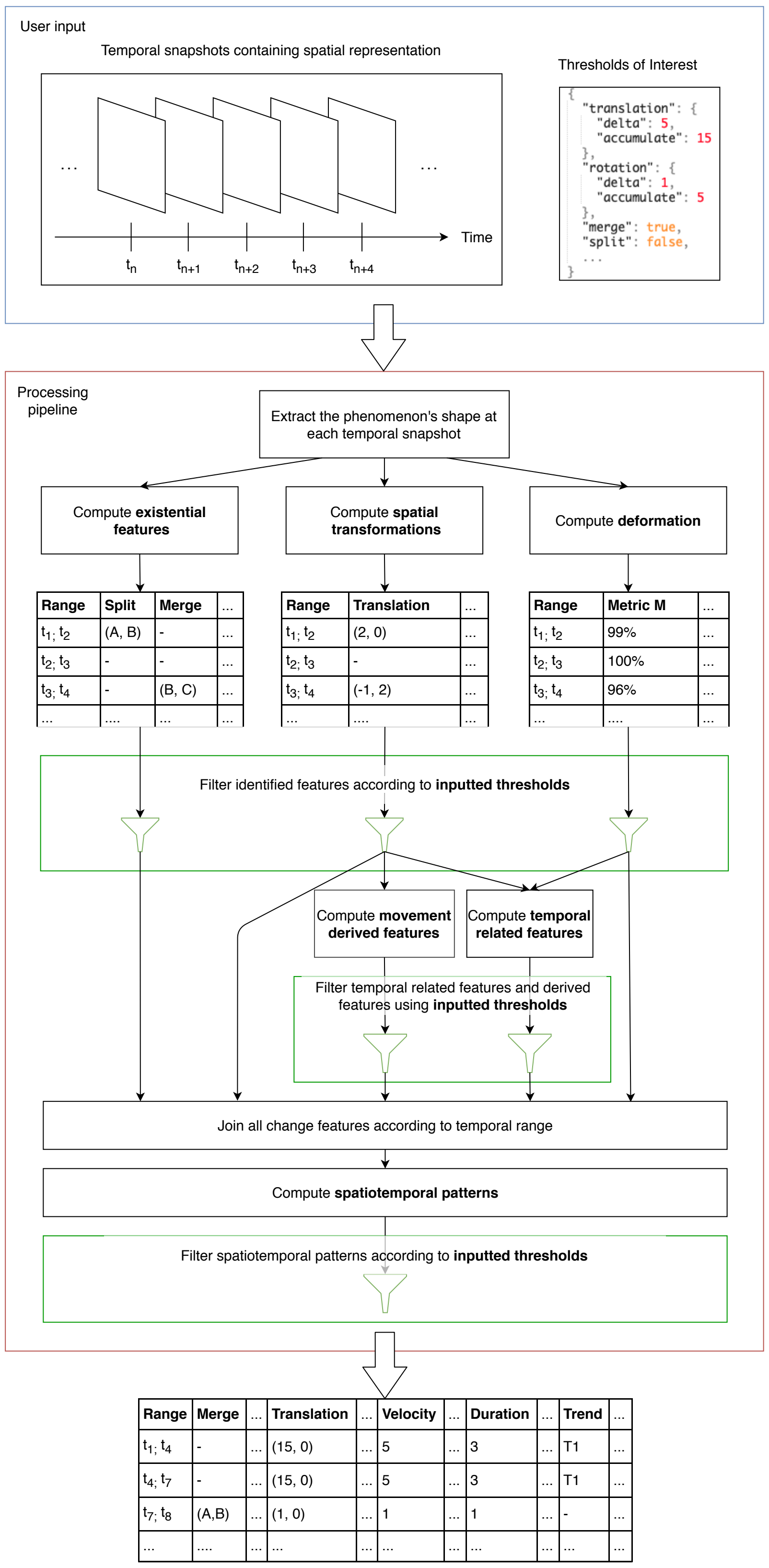
4.1 Proposed solution

From the literature review (chapter 2) and to the best of our knowledge, currently, there is not a single solution that addresses what we attempt to achieve: a framework for identifying and quantifying change on spatiotemporal data that is of interest to the end-user.

Figure 4.1 visually describes our proposal of a conceptual framework and its functioning process. The devised framework presupposes that the provided information is capable of representing spatiotemporal phenomena (rather than a single phenomenon), as this information is necessary for the computation of existential change features. Therefore, as input, the framework assumes a sequence of temporal snapshots. Each snapshot, associated with a timestamp, contains a mapping between a phenomenon ID and the phenomenon's shape. The phenomenon's shape is described by a polygon, represented as an array of vertices. However, as already stated at the beginning of chapter 2, in-truth the input data of the developed prototype (chapter 5) consists of spatiotemporal data describing a single phenomenon.

Based on Figure 4.1, the general functioning guidelines of the proposed framework can be briefly explained as:

1. The proposed framework starts by receiving as input a spatiotemporal dataset containing an ordered sequence of temporal snapshots. Each snapshot describes the shape of the phenomenon at the associated timestamp. Additionally, the user also provides various thresholds characterising its interest over different spatiotemporal change features (similar to [84]);



User input
Temporal snapshots containing spatial representation
...
...
Time
t_n
t_{n+1}
t_{n+2}
t_{n+3}
t_{n+4}
Thresholds of Interest
Processing pipeline
Extract the phenomenon's shape at each temporal snapshot
Compute **existential features**
Compute **spatial transformations**
Compute **deformation**
Filter identified features according to **inputted thresholds**
Compute **movement derived features**
Compute **temporal related features**
Filter temporal related features and derived features using **inputted thresholds**
Join all change features according to temporal range
Compute **spatiotemporal patterns**
Filter spatiotemporal patterns according to **inputted thresholds**

```
{
  "translation": {
    "delta": 5,
    "accumulate": 15
  },
  "rotation": {
    "delta": 1,
    "accumulate": 5
  },
  "merge": true,
  "split": false,
  ...
}
```

Range	**Split**	**Merge**	...
t_1; t_2	(A, B)	-	...
t_2; t_3	-	-	...
t_3; t_4	-	(B, C)	...
...	....	...	...

Range	**Translation**	...
t_1; t_2	(2, 0)	...
t_2; t_3	-	...
t_3; t_4	(-1, 2)	...
...	....	...

Range	**Metric M**	...
t_1; t_2	99%	...
t_2; t_3	100%	...
t_3; t_4	96%	...
...	....	...

Range	**Merge**	...	**Translation**	...	**Velocity**	...	**Duration**	...	**Trend**	...
t_1; t_4	-	...	(15, 0)	...	5	...	3	...	T1	...
t_4; t_7	-	...	(15, 0)	...	5	...	3	...	T1	...
t_7; t_8	(A,B)	...	(1, 0)	...	1	...	1	...	-	...
...	....	...	...	...	...	...	...	...	...	...

Figure 4.1: Visual representation of the proposed conceptual framework.

2. Having received the user input, the processing pipeline begins with the extraction of the shapes of the phenomenon from the spatial representations present in the temporal snapshots;

3. For each snapshot pair $[t_n, t_{n+1}]$, compute (identify and quantify) the existential features, the spatial transformations and the similarity between shapes. Therefore, the *object change* modelling approach, identified in section 2.1, is used for representing the computed spatiotemporal change features. The calculation can be concurrent since the features being computed are independent;

4. Filter the features parsed in the previous step according to the inputted thresholds. The aggregation of sequential spatiotemporal change features can occur in order to trigger the inputted thresholds (e.g., the aggregation of two sequential rotations of 5 degrees to surpass a threshold of 10 degrees). Section 4.2.7 further explains the thresholds of interest workings;

5. Given the filtered features of interest, the processing proceeds with the computation of both the movement-derived features and the temporal-related features. The newly identified features are then also filtered according to the respective inputted thresholds;

6. Join the spatiotemporal change features of the five categories previously calculated. To facilitate spatiotemporal analysis and interpretation of the data, index it by temporal range, meaning having the mapping between the temporal range (key) and the occurring features of interest (value);

7. Compute spatiotemporal patterns from the previous joint features of interest and filter them according to the inputted thresholds;

8. Join the spatiotemporal patterns with the remaining features of interest. Concluding the processing pipeline, the framework outputs the detected features of interest, once more indexed by temporal range.

The retrieved data consists of a representation based on the spatiotemporal changes, of interest to the user, affecting a phenomenon.

The proposed conceptual framework supports both 2D and 3D shape representations. When implementing this conceptual framework, it is the responsibility of the developer to choose appropriate algorithms (that fit the intended spatial representation) for the computation of the spatiotemporal change features.

With this conceptual framework, we present a solution that we consider appropriate to tackle the problem detailed in chapter 3.

More thoroughly, the problems defined in section 3.1 are handled in the following manner:

1. **Context-agnostic:** the devised solution is agnostic to the data context since as input, the framework manages the temporal evolution of the shape of the object - a feature inherent to all spatiotemporal datasets.

2. **Analysis of spatiotemporal features:** as seen from figure 4.1, the devised framework can support the different categories of change features identified. However, depending on the implementation, some features may not be computable.

3. **Retrieval of information of interest to the user:** the end-user can express its interests by defining the thresholds of interest (similar to [84]). When a threshold is defined, only spatiotemporal change features that surpass the user-defined threshold are retrieved. Moreover, by defining or omitting the thresholds of interest, the end-user can control which spatiotemporal features are processed. This process allows the conceptual framework to identify features of interest.

4.2 Main Components

This section details the main components of the conceptual framework solution proposed in section 4.1. We start by analysing the most appropriate methods for the computation of all of the identified spatiotemporal change features categories. Next, we examine how to identify the spatiotemporal change features of interest to the end-user. Subsequently, we present two novel spatiotemporal features. Lastly, a summary of this section contents is presented.

It is worth mentioning that the devised solution and respective approaches for the computation of the different modules are not the only methods for addressing the problem. However, from the performed literature review, it was concluded that these were the most appropriate approaches.

4.2.1 Quantification of spatial change features

For the quantification of spatial transformations, meaning the estimation of the spatial transformations occurring between successive temporal snapshots, point set registration algorithms appear as an appropriate solution. As verified in section 2.3.1, point set registration algorithms are capable of modelling both rigid (in this context rigid transformations stand for translations, rotations and uniform scaling) and affine (anisotropic scaling and shear) transformations.

However, to obtain such information, there must be a concern regarding the algorithm choice. In this scenario, the point set registration algorithm must be pairwise and parametric, subsequently employing affine registration. The algorithm is required to be pairwise since, as shown in section 4.1, the extraction of the spatial change features occurs between two consecutive shapes of the phenomenon. Groupwise registration is appropriate for situations where more than two point sets are being registered, but in our case, we never compare more than two shapes. A non-parametric algorithm makes it much harder to extract the underlying spatial transformations, as the output is modelled as a set of deformation vectors (one for each point of the source point set). Consequently, non-parametric algorithms, such as non-rigid algorithms, are not viable in this scenario. Hence, the necessity for a parametric algorithm. Finally, there is the requirement for the algorithm to support affine registration: from affine registration, we can obtain both the rigid and affine transformations, while from rigid transformations we can only obtain the rigid transformations. Therefore, algorithms that support both registrations (e.g. CPD algorithm [63] supports rigid, affine and non-rigid registrations) facilitate the identification and quantification of the spatial change features.

Regarding the most complex spatial features, such as projections and conformal mappings, no appropriate methodology was found for their extraction from consecutive phenomenon shapes. Henceforth, these spatial features are aggregated under deformations (the most generic category of the identified spatial change features).

4.2.2 Similarity analysis

As reported in the previous subsection, point set registration algorithms are unable to identify the most complex spatial change features. As such, these are aggregated under the category of deformations. Moreover, many point set registration algorithms are resilient to deformations. Thus, to identify and quantify deformations, the analysis of the similarity between the consecutive phenomenon shapes emerges as an adequate methodology.

The detection of deformations also encompasses a primary module of the framework, seeing that natural phenomena do not often perform spatial transformations that follow the exact function of a transformation. In the example of the area modified in a landslide, depending on the soil characteristics, some ground sections may advance faster than others [66]. Therefore, this phenomenon can not be modelled as a single scaling, but possibly as a conjunction of scalings paired with deformations.

In the performed literature review (specifically section 2.3.2), we verified that there are two viable approaches for estimating the similarity between two shapes: usage of a single similarity metric or of voting schemes (that can further utilise standalone similarity metrics). An approach utilising voting schemes is naturally more resilient as the results obtained by several assessments are crossed, with the most frequent assessment being chosen as the final one (voting system). Moreover, the usage of a voting scheme, employing distinct metrics as the voters, provides different perspectives over the occurred deformation.

4.2.3 Existential Features

For identifying existential features, as these can not be quantified since a phenomenon either exists or it does not, we rely on the metadata associated with the different phenomena shapes described in a temporal snapshot. Considering the presumption presented at the beginning of section 4.1, that each shape object has an associated ID, the succession of temporal snapshots informs us of the disappearance, appearance, split, or merge of phenomena. For exemplification, consider the existence of a temporal snapshot at instant t containing the phenomenon with ID A. However, if in the next instant the phenomenon with ID A is no longer present, that corresponds to a disappearance change feature.

4.2.4 Temporal features

For the following spatiotemporal change features, namely temporal-related features (this section), movement-derived features (section 4.2.5) and spatiotemporal patterns (section 4.2.6), their identification depends on the detection of the previously presented features. To justify this statement, consider an example scenario in which a spatiotemporal pattern is identified without any other underlying change feature. Users consuming the framework output will be able to recognise that a pattern was identified. However, they can not understand which change features led to the identification of the spatiotemporal pattern. Moreover, it is also presumable that end-users will not be interested in patterns constituted by change features that were not of interest to them.

Temporal change features are somewhat trivial to identify. The identification of one of the previously mentioned change features instantaneously provides the moment in time as well as the duration. Existential features stand as an exception since, as already seen, they occur in an exact moment in time, thus not making sense to have an associated duration.

Furthermore, by looking at the sequence of identified change features, it is also trivial to identify sequences — study the order by which the features were identified —, the frequency of a given change feature — find how many times change features with similar characteristics were identified —, and its pace — divide the value of the quantified temporal change over its change duration.

4.2.5 Movement-derived features

Movement-derived patterns are computed by applying kinematics physics fundamentals (analysed in section 2.3.3), using the travelled distance of trajectory-oriented movements, such as translations and rotations, and the duration of the respective movement. As implied by its naming, this category of change features depends on the previous identification and quantification of spatial change features.

4.2.6 Spatiotemporal patterns

Identifying the most simple spatiotemporal patterns can be achieved by analysing the order of the identified change features. If a given change feature cyclically repeats itself, then it is a cycle. Conversely, if a change feature repeats itself in an acyclic manner, then it is a trend.

For the identification of more complex scenarios, both combinatorial analysis and neural networks stand as viable solutions. However, both approaches have downsides that have to be taken into account when implementing a solution. In combinatorial pattern finding algorithms, the processing time increases with the number of identified change features, seeing that the number of tested combinations also increases. Neural networks suffer from the same problem as every other machine learning algorithm, meaning the dependability of the quality of the data used to train the model. However, seeing that the framework needs to be agnostic to the data context and that the machine learning algorithms require training the model in previous similar data, we propose the combinatorial pattern finding algorithms as the most appropriate approach.

4.2.7 Thresholds of interest

For the identification of spatiotemporal change features that are of interest to the user, the work presented by Tominsky et al. [84] emerges as the most suitable method. In this work, the user defines numerical thresholds that quantify change. If the quantification of a given spatiotemporal change feature surpasses the respective thresholds, then the change feature is considered a feature of interest. Hence, having this work as fundament, we identify two types of thresholds:

1. **Delta thresholds**, meaning the exceeding of a threshold by a change feature identified from two sequential temporal snapshots;
2. **Accumulated thresholds**, meaning the exceeding of a threshold by the accumulation of quantified change features over more than two sequential temporal snapshots.

In order to provide more control to the end-users utilising the application and making it easier to retrieve the features that are of real interest to them, the framework receives as input the presented thresholds for all spatiotemporal change features (a delta threshold and an accumulate threshold for each feature). The omission of a threshold leads to the framework not processing that threshold. Moreover, the omission of all the thresholds of a change feature leads to the framework not processing the change feature entirely. With this approach, we maximise the capability of the end-user to retrieve information of its interest and discard irrelevant one.

However, numerical thresholds do not characterise all of the features. The user interest by existential features, such as appearances, disappearances, splits, and merges, is defined through boolean flags indicating its analysis. This approach is necessary since it is not possible to quantify existential features: a phenomenon either exists, or it does not.

4.2.8 Additional Features

When employing thresholds of interest to obtain features of interest, the identification of features of interest for all possible temporal ranges is not guaranteed. However, and in order to meet the requirements presented in section 3.3, the devised solution must describe in totality the full temporal range queried. It is erroneous to state that if no feature of interest was identified for a given temporal range, then no spatiotemporal change features occur during that temporal range. Not having a feature of interest identified for a given temporal range means that, for the duration of that temporal range, no change occurred that was important enough to be considered interesting to the end-user. Hence, and respecting the given explanation, temporal ranges without any associated feature of interest are categorised as **unimportant**.

Even though the conceptual framework's primary goal is to identify spatiotemporal change features, we consider the total absence of motion (be it deformations or movement) as a particular circumstance worth highlighting. Hence, the framework should also be capable of identifying **immutability**. A distinct threshold of interest should be devised for this spatiotemporal feature, thus allowing the user to retrieve it if interested. In this scenario, the threshold of interest is the amount of time above which immutability is interesting.

4.2.9 Summary

This section detailed the most appropriate methodologies for the computation of the identified spatiotemporal change features, being these: point set registration algorithms for quantifying spatiotemporal change features, voting schemes operating similarity metrics for quantifying deformations, database identifiers for existential features, kinematics physics fundamentals for both temporal features and movement-derived features, and lastly, order analysis for simple spatiotemporal patterns and combinatorial analysis for more complicated patterns. Moreover, this section also presented the thresholds of interest as the mechanism for detecting features of interest. Finally, two additional features were described.

4.3 Secondary components

This section details the secondary components of the proposed solution. Even though these components do not perform essential steps in the processing pipeline presented in figure 4.1, they significantly increase the quality of the returned features of interest. We start by describing noise filtering of spatiotemporal change features and subsequently detail coalescing of features of interest.

4.3.1 Noise Filtering

The approaches detailed in the previous section, section 4.2, are expected to identify and quantify spatiotemporal change features with high-precision. Nonetheless, it is foreseen the production of some small noise by these methodologies. For example, the application of a point set registration algorithm in a phenomenon that is kept unchanged from one time step to another can result in the algorithm retrieving a rotation of 0.003 degrees. The estimated rotation is incredibly precise but not perfect.

Moreover, considering that this value is different from 0, which represents the absence of rotation, it can lead to the framework starting accumulating rotations, and eventually surpass the accumulated threshold.

Furthermore, such imprecisions can also result in the framework not being able to detect immutability, since the point set registration algorithm detected a rotation.

Noise filters appear as an appropriate solution for the enumerated problems. By filtering the noise produced by the employed methodologies, accumulated thresholds consider only significant change features and the detection of immutability is possible. Noise filters should be defined similarly to thresholds of interest — define a threshold, for each change feature, below which quantified change is considered noise — thus granting the end-user the possibility to control what information should be considered as noise. If the end-user does not define noise filter thresholds, then no noise filtering occurs.

Considering the processing pipeline presented in figure 4.1, noise filters are applied before filtering any change feature according to the inputted thresholds. However, this additional process is not needed if the solution components responsible for quantifying change features can do it with perfect precision.

4.3.2 Coalescing

Coalescing is responsible for aggregating temporal ranges that encompass similar change features. To better understand the utility of this component, consider that a user queries the framework and as a result obtains 10 temporal ranges where each temporal range is solely constituted of an accumulated translation of 0 units in the x-axis and 5 units in the y-axis. Seeing that the transformations contained in the retrieved temporal ranges are identical, they can be coalesced into a single temporal range. The coalesced result starts at the first temporal range start time, ends at the last temporal range end time and is constituted by an accumulated translation of 0 units in the x-axis and 50 units in the y-axis.

This solution permits retrieving to the end-user the result of the processing pipeline more compactly and straightforwardly.

Considering the processing pipeline presented in figure 4.1, the coalescing process occurs after having joined all change features according to temporal range.

4.4 Output

As shown in section 4.1, the proposed framework retrieves the identified features of interest indexed by temporal range. More concretely, the devised solution should return, for each temporal range, the features of interest that co-occur. The temporal ranges are continuous and describe the full temporal range queried.

Figure 4.2 presents an illustration of the designed desired output and how several features of interest should be merged in order to satisfy the described output format. This figure illustrates

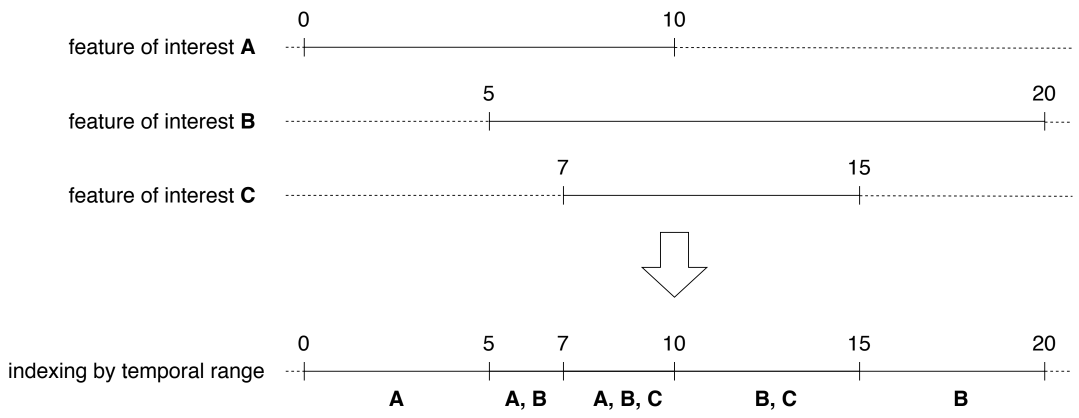


Figure 4.2: Example of the modelling of three features of interest, in temporal range 0 to 20: indexed by change feature and indexed by temporal range.

three distinct features of interest, identified through accumulated thresholds (since they spawn over more than one time step), resulting from the query that obtains the feature of interest occurring in the temporal range 0 to 20. The first feature of interest, A, occurs during temporal range 0 to 10. The second one, B, during temporal range 5 to 20. The third one, C, during temporal range 7 to 15. This same features of interest, indexed by temporal range, can be described as: from time step 0 to time step 5, A occurs; from 5 to 7, A and B co-occur; from 7 to 10, A, B and C co-occur; from 10 to 15, B and C co-occur; and lastly, from 15 to 20, B occurs. If the temporal range queried was from 0 to 25 rather than from 0 to 20, then there would be an extra *unimportant* feature of interest, spawning from time step 20 to time step 25 (as introduced in section 4.2.8).

The example presented in figure 4.2 highlights a characteristic of indexing by temporal range: features of interest detected by accumulation may have to be segregated in several temporal ranges that on their own do not surpass the given thresholds.

4.5 Summary

In this chapter, we proposed a conceptual framework design that we consider effectively addresses the problems that were identified in chapter 3. Section 4.1 presents the conceptual framework and its general functioning guidelines. Section 4.2 introduces the different main components that constitute the conceptual framework and what methodology to use to compute the associated change features. Some referred methodologies are point set registration algorithms, voting schemes utilising similarity metrics, physics kinematics fundamentals, features of interest, among others. Section 4.3 details noise filtering and coalescing: two secondary conceptual framework modules that, despite not being essential, may significantly improve the quality of the returned features of interest. Finally, section 4.4 describes the output format of the results retrieved by the devised conceptual framework.

Chapter 5

Spatiotemporal Features Extractor

This chapter describes the prototype implementation of the conceptual framework detailed in the previous chapter. Section 5.1 overviews the devised prototype while also presenting some key considerations. Afterwards, section 5.2 details the prototype architecture. Next, section 5.3 describes the API while section 5.4 examines the deployment methods. Section 5.5 provides some pointers on how to extend the devised prototype. Finally, section 5.6 presents a summary of this chapter's contents.

5.1 Overview

The **SpatioTemporal Features eXtractor (*stfX*)** [12] is the prototype implementation of the conceptual framework detailed in chapter 4. Thus, the *stfX* focuses on the quantification of the features of interest occurring over user-inputted spatiotemporal phenomena, while completing all the requirements established in section 3.3. The *stfX* is capable of quantifying a subset of the spatiotemporal change features identified in the literature review, namely: translations, rotations, uniform scaling, changes of direction, moment in time and duration.

The *stfX* receives as input the continuous representation of a spatiotemporal phenomenon modelled as a sequence of temporal snapshots. Each temporal snapshot contains a shape descriptor, meaning a simplified representation of a 2D shape in the form of a vector that geometrically describes the shape [47]. The choice of input (input format and analysis of a single phenomenon) is justified by the usage of the framework presented by Duarte et al. [24], SPTDataLab, as the provider of spatiotemporal datasets used for testing. Section 5.3 more thoroughly details the prototype input data.

Considering the absence of a framework with similar capabilities to the one detailed in chapter 4, the *stfX* represents the developed prototype of what is a pioneering research stream. Hence, what we intend with this first version of the *stfX* is to establish a robust underlying foundation of the sophisticated and complete framework that the *stfX* can one day be, provided further study is performed as presented in section 7.3. Therefore, and considering future iterations over the *stfX*, we determined as a priority developing extensible, loosely-coupled and easily-scalable code.

The *stfX* can be utilised as a data provider in a plethora of situations: used as a standalone framework whose output is further handled by end-user tools (e.g. applications in machine learning); used in conjunction with a visualisation tool, such as the one presented by Marques [57], to visually narrate the retrieved features of interest; raw consumption of the outputted data by the end-user; among others.

5.2 Architecture and Components

This section starts by analysing the architecture of the devised prototype, while also identifying the implemented capabilities (in comparison to the conceptual framework). Next, it describes in detail each of the developed microservices, while analysing its functionalities and implementation aspects.

The SpatioTemporal Features eXtractor (*stfX*) is a server-side application utilising the microservices architectural approach, thus emphasising "self-management and lightweightness as the means to improve software agility, scalability, and autonomy" [44]. In the *stfX*, each microservice represents a module, presented in section 4.2, responsible for the quantification of a given category of spatiotemporal change features.

Regarding the different functioning guidelines displayed in section 4.1, and establishing a mapping with the proposed conceptual framework, the *stfX* current version complies with the following:

1. Receives as input both the spatiotemporal dataset modelled as a sequence of temporal snapshots and the thresholds that express the end-user interests;

2. Extracts both the shapes and associated timestamps from each temporal snapshot;

3. For each temporal snapshot, quantifies the rigid transformations that occurred (translation, uniform scaling and rotation);

4. From the identified movement derived features, computes change of direction (the motive behind solely considering the change of direction is explained further ahead in section 5.2.2.1);

5. Filters the identified spatiotemporal change features according to the user-inputted thresholds;

6. Computes, from the identified temporal features, duration and moment in time, even though no thresholds are provided for these features of interest;

7. Indexes the identified features of interest by temporal range.

Considering the described *stfX* capabilities, figure 5.1 displays its microservices architecture.

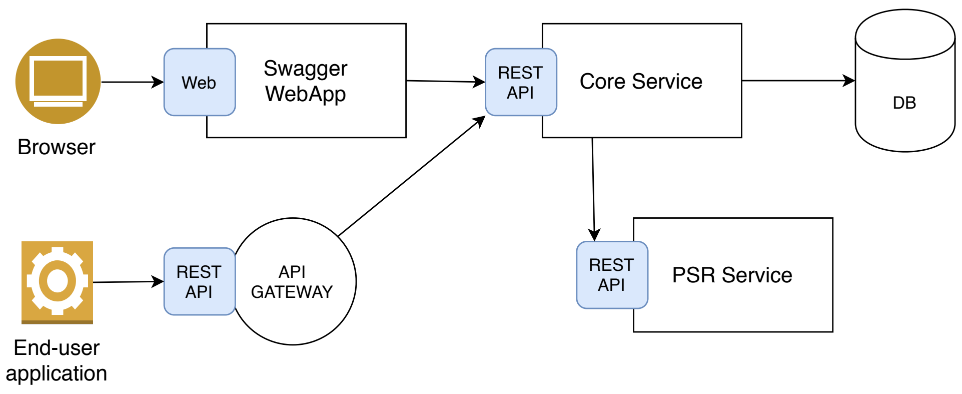


Figure 5.1: Visual representation of the *stfX* microservices architecture.

As presented in figure 5.1, the *stfX* presents two interfaces to the end-user: (1) a web application, built with Swagger[1], displaying the *stfX* API documentation and facilitating the usage of the *stfX* API; (2) the *stfX* API proxied by a NGINX[2] gateway. All requests are redirected to the core service.

The core service represents, as its name suggests, the central microservice of the *stfX*. The core service, built using spring[3], is the service responsible for everything that is not the computation of spatiotemporal change features (e.g. data storage, features of interest attainment, etc.). This service is further detailed in subsection 5.2.2.

The other service displayed in figure 5.1, named PSR service, consists of the implementation of a Point Set Registration (PSR) algorithm, the approach selected to quantify the spatial change features. This service was built using Flask[4]. More details regarding this service are provided in subsection 5.2.1.

Before proceeding with the architecture analysis, it becomes relevant to introduce the notion of **Storyboard**. A Storyboard is the *stfX* internal representation of a dataset, the associated metadata and the spatiotemporal change features that were computed using the dataset.

The *stfX* workflow is also slightly different from the one described in section 4.1. In section 4.1, the end-users provided simultaneously the dataset, modelled as a sequence of temporal snapshots, and the thresholds defining their interests. However, in the *stfX*, instead of devising a single endpoint that handles both inputs, we chose to separate the input of the dataset and the thresholds into two separate endpoints. The motive behind this decision is the fact that end-users might want to tune the thresholds after obtaining the results of an initial search or even analyse the same dataset later. When using a single endpoint approach, the end-user has to provide the dataset every time and repeat the computation of the spatiotemporal change features between temporal snapshots. This process is both time-consuming and unnecessary. With the usage of two separate

[1]SmartBear Software. Swagger. Available at https: https://swagger.io. Accessed in 2020-06-17.
[2]F5. NGINX. Available at https: https://www.nginx.com. Accessed in 2020-06-17.
[3]VMware. spring. Available at https: https://spring.io. Accessed in 2020-06-17.
[4]Pallets. Welcome to Flask - Flask Documentation (1.1.x). Available at https: https://flask.palletsprojects.com/en/1.1.x/. Accessed in 2020-06-17.

endpoints, the user provides the dataset once, receives an ID representing the created Storyboard and then uses the ID in conjunction with the thresholds to obtain the features of interest of the respective Storyboard. Figure 5.2 shows the sequence diagram, between microservices, of loading a dataset to the *stfX*. Naturally, an endpoint to delete a given loaded dataset has also been created.

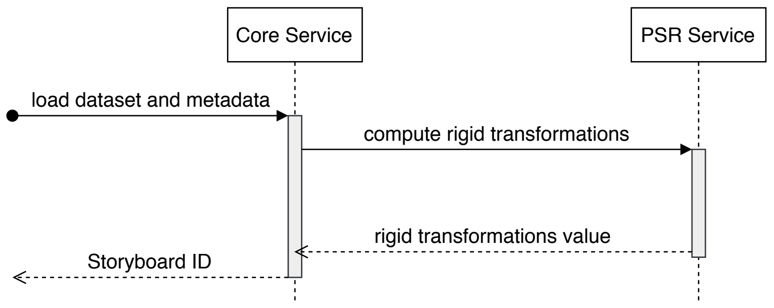

Core Service
PSR Service
load dataset and metadata
compute rigid transformations
rigid transformations value
Storyboard ID

Figure 5.2: Sequence diagram of loading a dataset in the *stfX*.

5.2.1 Point set registration service

The point set registration (PSR) service is responsible for quantifying spatial change features, using point set registration algorithms.

There are a vast set of point set registration algorithms. However, as shown in chapter 4.2.1, the algorithm employed must be pairwise, parametric and support affine transformations. Moreover, considering that the *stfX* can be managed as a data provider to other applications, algorithms with high accuracy to runtime trade-off present themselves as the most appropriate candidates. From the performed literature review (specifically section 2.3.1), we identified SCGF [91] as having the highest accuracy and CPD-GL [56] as having the best accuracy to runtime trade-off. However, both algorithms are non-rigid and thus, non-parametric. Hence, neither SCGF nor CPD-GL are suitable for the *stfX*. The next best option is **CPD** [63]. CPD is the most time-efficient algorithm and, from the ones performing parametric registration, also the one with the best accuracy.

Hence, and after a first attempt, which proved futile, at extracting the rigid parameters from the non-rigid algorithm CPD-GL, we chose CPD as our point set registration algorithm. Instead of implementing CPD from scratch, we used *pycpd*[5], a pure NumPy[6] implementation of the CPD algorithm. Seeing that *pycpd* is implemented in python, we chose to develop the server using Flask. Henceforth, we might also refer to the PSR service as the CPD service.

The *pycpd* package allows for both rigid, affine and non-rigid registration. All registrations require as input a source and a target point set, but other algorithm parameters can also be defined. In rigid registration, CPD returns the uniform scaling factor, the rotation matrix and the initial translation vector. In affine registration, CPD returns the affine transformations matrix and the initial translation vector. Both the uniform scaling factor and the rotation angle can be extracted

[5]Python Software Foundation. pycpd 2.0.0. Available at https: `https://pypi.org/project/pycpd/`. Accessed in 2020-06-17.

[6]NumPy. NumPy. Available at https: `https://numpy.org`. Accessed in 2020-06-17.

from the affine transformations matrix. As already seen, non-rigid registration does not fit our problem and hence will not be further investigated.

5.2.1.1 Communication Protocol

As shown in section 5.2, the PSR service serves as a microservice in the *stfX* architecture. Therefore, a communication protocol between microservices is necessary. In the current *stfX* architecture, the PSR service receives requests from the core service and does not initiate any communication with other microservices. The PSR-service API is constituted by two endpoints:

- `/cpd`: Endpoint expecting a request with the post method. This endpoint's purpose is to analyse the rigid transformations between two point sets. Listings 5.1 and 5.2 show an example of a request and respective response. As input, this endpoint receives the two point sets, with "X" being the source and "Y" the target. The output consists of the identified rigid transformations (translation vector, uniform scaling factor and rotation angle in degrees).

```
{
    "X": [
        [0, 1],
        [1, 0],
        [0, 0]
    ],
    "Y": [
        [4, 5],
        [5, 4],
        [4, 4]
    ]
}
```

Listing 5.1: Example body of a request to the `/cpd` endpoint.

```
{
    "translation": [4, 4],
    "rotation": 0,
    "scale": 1

}
```

Listing 5.2: Example body of a `/cpd` endpoint response.

- `/cpd-all`: Endpoint expecting a request with the post method. This endpoint's purpose is to analyse the rigid transformations between several pairs of point sets. Summarily, this endpoint consists of the application of the `/cpd` process to all elements of a list. Listings 5.3 and 5.4 shows an example of a request and respective response. As input, the endpoint receives a list of point set pairs. In each pair, "X" represents the source point set and "Y" the target point set. The output is the list of identified rigid transformations, where an element in the list corresponds to the rigid registration of the element in the same index in the input list.

```
[
    {
        "X": [
            [0, 1],
            [1, 0],
            [0, 0]
        ],
        "Y": [
            [4, 5],
            [5, 4],
            [4, 4]
        ]
    }, {
        ...
    },
    ...
]
```

Listing 5.3: Example body of a request to the `/cpd-all` endpoint.

```
[
    {
        "translation": [4, 4],
        "rotation": 0,
        "scale": 1

    },
    ...
]
```

Listing 5.4: Example body of a `/cpd-all` endpoint response.

5.2.2 Core service

The core service is the central service of the *stfX* architecture and the component bearing more responsibilities, namely:

1. aggregation of the spatiotemporal change features computed by the remaining services;
2. storing temporal snapshots as well as the corresponding extracted spatiotemporal change features;
3. filter the obtained spatiotemporal change features according to the user inputted thresholds, thus obtaining the features of interest;
4. identification of immutability (feature presented in section 4.2.8);
5. filtering of noise produced by the utilised computation methods (explained in section 4.3.1);
6. indexing of spatiotemporal change features according to the temporal range (explained in section 4.4);
7. identification of unimportant temporal ranges (feature presented in section 4.2.8);
8. coalescing of features of interest (explained in section 4.3.2).

The service follows the Model-View-Controller (MVC) architecture, reinforced by the spring framework. Regarding storage, an H2[7] in-memory database is used.

[7]H2. H2 Database Engine. Available at https: `http://www.h2database.com`. Accessed in 2020-06-18.

Following the MVC architecture, the following packages of the *stfX* represent the structural foundation of the application:

- The **Models** package purpose is to provide a CRUD interface for the data entities managed in the application. Models do not handle any business logic, solely entity self-contained logic;
- **Controllers** are responsible for handling the *stfX* API endpoints and further delegate actions to the other architecture components;
- **Repositories** specify persistence in the database using ORM;
- **Services** fetch data from the Models, handle it and then provide the processed data to the Controllers. The majority of the business logic is implemented here.

The View, meaning the display of the application's state [49], is represented by the JSON returned by the API endpoints (further described in section 5.3).

In the following subsections, we examine the *stfX* most relevant components.

5.2.2.1 Directed accumulated threshold

While devising the *stfX* prototype, we identified a new threshold category — the **directed accumulated threshold**. This new threshold represents the quantified change features accumulated while not changing direction, over more than two sequential temporal snapshots. The reasons that led to the identification of this new threshold category were:

1. From the analysis of the identified movement-derived change features, in section 2.3.3, we verify that changes of direction have a unique computation nature: while velocity and acceleration depend on the travelled distance and the duration of the movement, change of direction depends on the angle between two movement vectors (a movement and its predecessor). However, while in a translation this angle can have any value between -180 degrees and 180 degrees, on uniform scaling, for example, the angle can only be either 0 degrees (movement maintained direction) or -180^o degrees (movement considerably changed course by inverting direction). Hence, it becomes intricate to establish a standard for defining thresholds for the change of direction feature.

2. While analysing and developing the conceptual framework, we perceived that the accumulated threshold (introduced in section 4.2.7) encapsulated a significant quantity of spatiotemporal information. For example, considering an absolute threshold set to 10 and two sequential rotations of 5 degrees and -5 degrees, the threshold is surpassed even though, in the retrieved movement that surpassed the threshold, the phenomenon initial position and the last position are the same.

Seeing that the directed accumulated threshold represents a subcategory of the initially defined accumulated threshold, we decided to rename the accumulated threshold to **absolute accumulated threshold** as to best describe the nature of the change features this threshold detects.

5.2.2.2 Parsing features of interest

The detection of features of interest is an essential function of the *stfX*, that is highly dependent on the thresholds inputted by the end-user. With the intuit of easing this section upcoming analysis, Listing 5.5 shows a possible example of the thresholds inputted by the end-user.

```json
{
    "parameters": {
        "translation": {
            "delta": 3.1,
            "directedAcc": 8,
            "absoluteAcc": 10.2,
            "noiseEpsilon": 0.02
        },
        "rotation": {
            "delta": 3.1,
            "directedAcc": 8
        },
        "immutability": 100
    }
}
```

Listing 5.5: Example of thresholds inputted by the end-user.

In the input example shown in Listing 5.5, the end-user establishes thresholds for two transformations: translations and rotations. Hence, the remaining spatiotemporal change features are not analysed. Regarding translations, the user defines the delta threshold as 3.1, the directed accumulated threshold (*directedAcc*) as 8, and the absolute accumulated threshold as 10.2 (*absoluteAcc*). Additionally, the user also defines the ***noiseEpsilon*** threshold. This threshold, identified as necessary in section 4.3.1, represents the limit below which the spatiotemporal change feature identified (in this case a translation) is considered noise. When noise is detected the *stfX* assumes that the spatiotemporal change feature is null. The threshold naming refers to the feature purpose: examine noise using the expectedly low (epsilon) threshold.

The *stfX* prototype capability of filtering features of interest is implemented in the Services package, more concretely in the Parsers package. The `ParserFactory` is the entry point of this package. It is responsible for starting the concurrent parsing of the features of interest, using the user-inputted thresholds in conjunction with the respective spatiotemporal change feature parsers. Listing 5.6 displays the features that are being detected (the parsing only proceeds if the respective threshold is not null), in the current version of the *stfX*.

```java
ITransformationParser[] concurrentParsers = {
    new TranslationParser(thresholds.getTranslation()),
    new RotationParser(thresholds.getRotation()),
    new UniformScaleParser(thresholds.getScale()),
    new ImmutabilityParser(thresholds)
};
```

Listing 5.6: Change features being detected in the *stfX*.

As shown in Listing 5.6 there are a total of 4 non-abstract parsers, corresponding to the features being detected that require thresholds: translations, rotations, uniform scalings and immutability. Figure 5.3 illustrates the hierarchy between the parsers available in the Parser package.

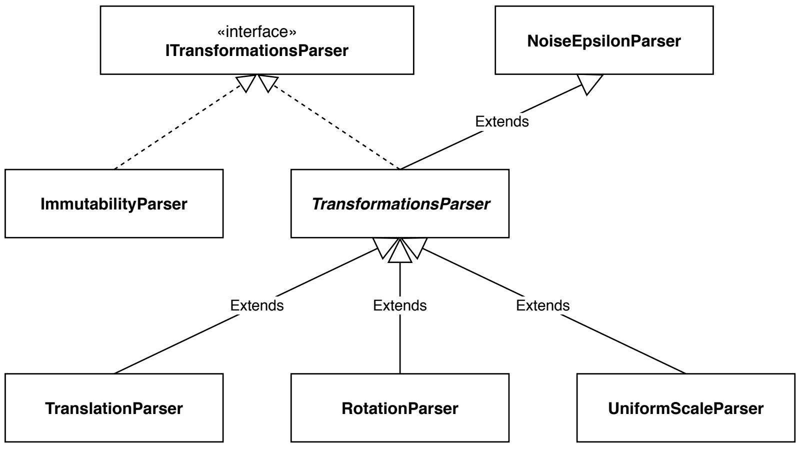


Figure 5.3: Parser classes hierarchy.

The `NoiseEpsilonParser`, displayed in figure 5.3, is responsible for parsing spatiotemporal change features and discarding those which are considered as noise. The parsing of noise only occurs if the *noiseEpsilon* threshold is defined. For computing the limit below which change features are considered noise, the *stfX* makes use of both the delta threshold and the *noiseEpsilon* threshold. If the delta threshold is not defined, then the noise limit equals the *noiseEpsilon* threshold. Otherwise, the noise limit is defined by the maximum value between the *noiseEpsilon* threshold and one-hundredth of the delta threshold value. We opted for this approach, as a result of experiments, where we verified that the usage of minimal *noiseEpsilon* thresholds would often fail to detect noise. Nonetheless, future work includes studying appropriateness and alternatives regarding the current approach for selecting the noise limit.

All change features parsers implement the `ITransformationParser` interface and, consequently, the `parse()` function, i.e. the function responsible for identifying features of interest. Figure 5.3 shows that two classes implement this interface: the `ImmutabilityParser` and the `TransformationsParser`.

The `ImmutabilityParser` is the class responsible for identifying immutability. With this intent, in its implementation of the `parse()` function, the list of spatiotemporal change features is iterated. If there is a total absence of motion, for a temporal range superior to the value representing the immutability threshold, then the parser detects an immutability of interest.

The `TransformationsParser` is an abstract class representing the parsers that interpret a set of thresholds similar to the one defined in the code excerpt above for the translation: constituted by *delta*, *directedAcc*, *absoluteAcc* and *noiseEpsilon* thresholds. The `parse()` function starts by applying the *noiseEpsilon* threshold first, thus discarding noise (this class extends the `NoiseEpsilonParser` class). Next, the spatiotemporal change features are iterated, and each of the remaining thresholds is verified, in the following order: *delta*, *directedAcc*, and *absoluteAcc*. When a threshold is surpassed, the other thresholds' counters are reset (e.g., reset the accumulator of absolute translations values when a directed translation is detected). We opt to reset all threshold accumulators since otherwise, several features of interest regarding the same change feature could co-occur. Such an approach would introduce redundancy and raise difficulties to visualisation tools (e.g. presenting several translations affecting simultaneously the same phenomenon), like the solution proposed by Marques [57].

Classes `TranslationParser`, `RotationParser` and `UniformScaleParser` extend the `TransformationsParser` class and represent the applications of the parser to the respective spatiotemporal change features.

We foresee that the introduction of the remaining spatiotemporal change features not being identified in the current *stfX* version (e.g. existential features) implies the design of new parsers. Furthermore, each of the current parser implementations can be replaced by upgraded versions resulting from outcomes of future work.

5.2.2.3 Events and Data Types

In the previous subsection, we examined the implementation details regarding the method used for obtaining the spatiotemporal change features of interest to the end-user. In this subsection, we verify how the *stfX* internally manages features of interest. The package responsible for this management is the Events package inside the Services package.

Internally, the *stfX* represents a feature of interest using the `Event` class. An `Event` stores: the type of thresholds that identified the features as interesting (threshold triggers); the associated spatiotemporal change feature; a list of mappings between a timestamp and the associated quantification of the spatiotemporal transformation (with the timestamps of the first and last list elements we obtain the moment in time and duration features). Listing 5.7 demonstrates the possible threshold triggers and spatiotemporal change features.

```java
public enum ThresholdTrigger {
    DELTA,
    DIRECTED_ACC,
    ABSOLUTE_ACC
}
public enum Transformation {
    UNIMPORTANT,
    IMMUTABILITY,
    TRANSLATION,
    ROTATION,
    UNIFORM_SCALE
}
```

Listing 5.7: Possible threshold triggers and change features that an `Event` may store.

Different spatiotemporal change features have distinct forms of quantifying change. For example, translations between temporal snapshots can be modelled using a translation vector, hence a list of values. On the other hand, rotations can be represented using the angle rotated in degrees, thus a single value where a rotation of 0 degrees denotes a null change. Lastly, uniform scales can be modelled using the scaling factor – a unique value where a scale factor of 1 indicates the absence of change.

Seeing the disparities between the different forms of quantifying change, we introduce the `TransformationDataType` abstract class (Listing 5.8), with the purpose of establishing a norm for the change quantifiers of the various features of interest.

```java
public abstract class TransformationDataType<T> {
    protected T value;
    TransformationDataType(T value) {this.value = value;}
    public abstract boolean verifyNull();
    public abstract TransformationDataType<T> nullValue();
    public T getTransformation() {return value;}
    public abstract float numericalValue();
    public abstract TransformationDataType<T> add(T value);
    public abstract TransformationDataType<T> subtract(T value);
    public abstract boolean changeDirection(TransformationDataType<T>
        transformation);
}
```

Listing 5.8: `TransformationDataType` class.

The class template parameter `T` represents the underlying data type (e.g. on translations `T` is a list of values, on rotations and uniform scales `T` is a single value).

Hence, each `Event` instance has an associated `TransformationDataType` (used in the mapping between timestamps and quantified change previously referred), corresponding to the feature of interest that the `Event` instance represents.

5.2.2.4 Index by temporal range

Previously, in section 4.4, we identified how the features of interest should be retrieved to the end-user: indexed by temporal range rather than by feature of interest. However, the `ParserFactory` class returns a list of `Event` instances, hence indexing by features of interest.

The package Frames, inside the Services package, is responsible for converting the list of `Event` instances into a list of temporal ranges containing the co-occurring features of interest. Before proceeding, it becomes necessary to introduce two main classes of the Frames package:

- `Frame` is the class responsible for representing the set of features of interest associated with a temporal range. Therefore, a `Frame` stores a temporal range, the associated list of features of interest co-occurring in that temporal range, as well as the phenomenon representation for each timestep of the temporal range;

- `FramedDataset` represents the list of `Frame` instances that describe the queried temporal range of the Storyboard. It is also responsible for converting the list of `Event` instances into a list of `Frame` instances.

The methodology used to parse the list of `Event` instances into a list of `Frame` instances consists of an event-oriented algorithm. For each `Event` instance, two events are created: the beginning of the `Event` and its ending. Then, using a priority queue sorted by the event associated timestamp, we iterate over the created events. Whenever an event (or several events on the same timestamp) is polled from the priority queue, a new `Frame` instance is created.

Figure 5.4 presents the mapping between the described classes and the output example presented in section 4.4. As shown in the figure, when indexing by temporal range, feature of interest *A* is divided into three different segments belonging to three different temporal ranges. This division leads to the change quantified in each of the produced segments not being enough to surpass the provided thresholds. However, when observing the full extent of feature of interest *A*, the end-users verify that the sum of the quantified change on all segments is superior to the thresholds provided by them.

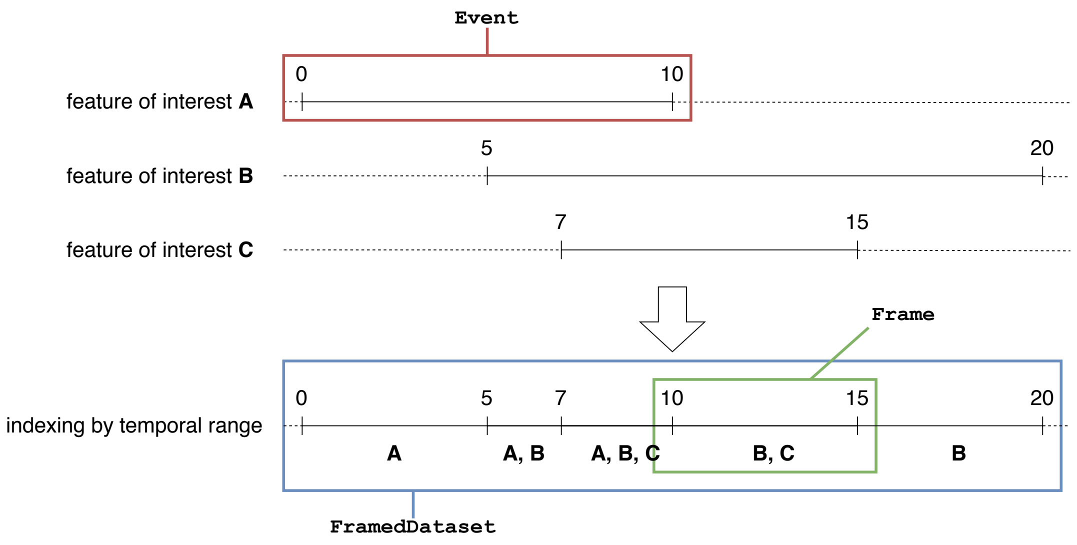


Figure 5.4: Mapping between the illustration elements presented in Figure 4.2 and some of the *stfX* classes.

Another aspect of this approach is that a `Frame` resulting from an `Event` fragmentation might contain segments with no change, which might be counter-intuitive to the end-user. For example, consider that a rotation that surpassed a directed accumulated threshold of 8 degrees gets segmented into three frames with equal temporal ranges: a rotation of 5 degrees, a rotation of 0 degrees and lastly a rotation of 3 degrees. The middle frame is contemplated as a feature of interest despite bearing no change.

5.2.2.5 Unimportant temporal ranges and Coalescing

To compute the unimportant temporal ranges (section 4.2.8), we iterate over the list of `Frame` instances obtained in the `FramedDataset` class. Seeing that there are no `Frame` instances with overlapping temporal ranges, the process of identifying unimportant temporal ranges consists of iterating over the list and verifying if there are any time gaps. A time gap is identified if the end timestamp of a `Frame` does not match the beginning timestamp of its successor. Moreover, it is also necessary to verify the beginning timestamp of the first `Frame` with the beginning of the range queried. The process is also repeated for the last `Frame` and the end timestamp of the range queried.

To proceed with coalescing (section 4.3.2), we once more iterate over the list of `Frame` instances and aggregate sequential features of interest that are identical. Two `Frame` instances are considered identical if both contain the same number of features of interest and if, for each feature of interest of a `Frame`, its successor has a feature of the same type and triggered with the same threshold. For features of interest triggered with the *directedAcc* threshold, the feature and its correspondent need to have the same direction. Coalescing two `Frame` instances means replacing them by a single `Frame` that has aggregated temporal range, aggregated phenomenon representations and summed identical features of interest. For example, having a frame *A* with temporal range 0 to 30 and a 10 degrees rotation, triggered by a *directedAcc* threshold, and a successive frame *B* with temporal range 30 to 45 and a 30 degrees rotation, also triggered in the same way, results in a coalesced frame with temporal range 0 to 45 and a 40 degrees rotation. Figure 5.5 illustrates this example.

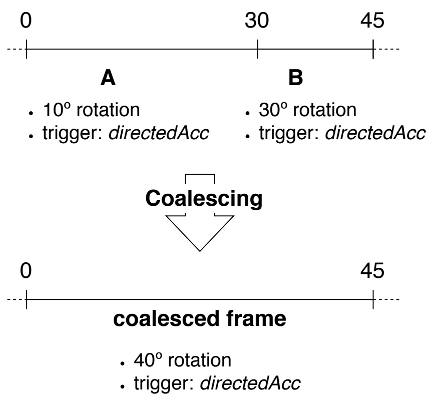


Figure 5.5: Coalescing example.

To facilitate the inclusion (or removal) of these capabilities to the *stfX*, we implement them using the Decorator design pattern [19], as shown in Listing 5.9.

```
return new CoalescedFramedDataset(
    new FramedDatasetWithUnimportantFrames(
        new FramedDataset(storyboard, thresholds)
            .restrictInterval(initialTimestamp, finalTimestamp)))
    .getFrames();
```

Listing 5.9: Coalescing and Unimportant temporal ranges implementation, using the Decorator design pattern.

5.2.2.6 Components Interaction

In order to understand how the previously presented components of the *stfX* interact, figure 5.6 illustrates the sequence diagram for getting the frames of interest. This sequence diagram refers only to core service components.

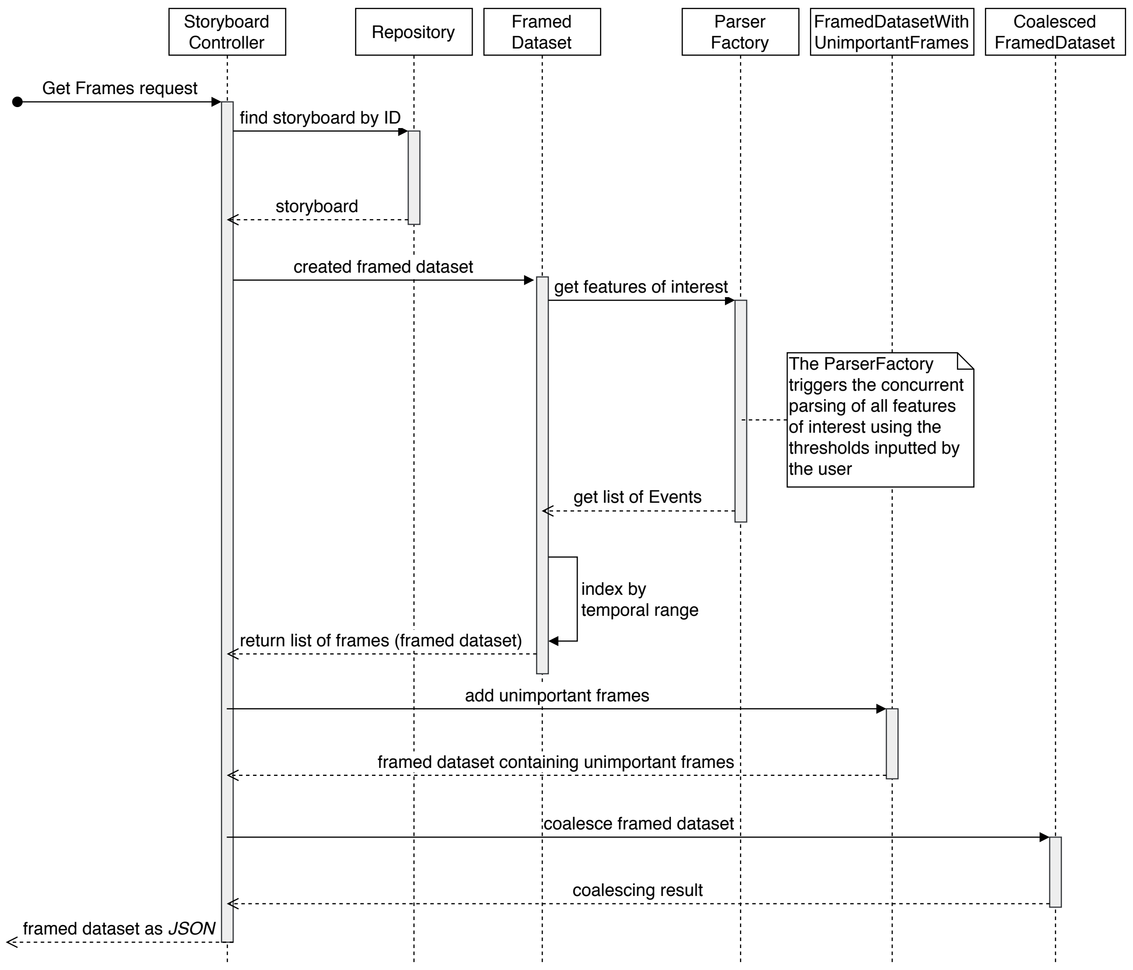


Figure 5.6: Sequence diagram of getting the frames of interest in the *stfX* core service.

First, the `StoryboardController` starts by receiving the end-user request. This request contains the ID of the referred `Storyboard`, the thresholds determining which spatiotemporal change features are interesting and the temporal range queried. Using the received ID, the *stfX* finds the corresponding `Storyboard`. Afterwards, the `FramedDataset` corresponding to the queried temporal range is created. This process starts with the extraction of the features of interest, using the `ParserFactory` in conjunction with the inputted thresholds. The `ParserFactory` triggers the concurrent parsing of the various change features using the different parsers introduced in section 5.2.2.2. When the parsing is done, a list with all the identified features of interest (modelled as `Event` instances) is retrieved. Next, a list of `Frame` instances (a `FramedDataset`) is created, by indexing the previous array of features of interest by temporal range. Having the `FramedDataset`, the unimportant temporal ranges are added to it, and subsequently, the *stfX* coalesces identical frames. Lastly, the resulting `FramedDataset`, coalesced and containing unimportant temporal ranges, is retrieved to the end-user as a JSON object. The retrieved JSON is further analysed in section 5.3.

5.2.3 Summary

In this section, we detail the architecture of the *stfX* and analyse how the different components of the various services interact. The *stfX* is a server-side application utilising the microservices architectural approach. In the current version, the *stfX* is composed of two services: the PSR-service and the Core service.

Subsection 5.2.1 describes the PSR-service: the service responsible for computing the spatial change features occurring in a phenomenon. Moreover, the communication protocol between the two services is also specified in this subsection.

Subsection 5.2.2 analyses the Core service: the central service of the *stfX* and the one responsible for several tasks, such as storing data in an in-memory database and filtering the features of interest. This subsection presents a new accumulated threshold named directed accumulated threshold (*directedAcc*). Moreover, it also examines how the identification of features of interest occurs in the *stfX* and describes the methodologies for (1) indexing by temporal range, (2) identifying unimportant temporal ranges and (3) coalescing identical `Frame` instances. Lastly, it determines how the distinct core service components interact to retrieve the features of interest.

5.3 API

In section 5.2, we identified *stfX* as a server-side application with an exposed REST API. Hence, in this section, we describe the REST API while also analysing template responses for each endpoint. For more details, refer to the API documentation[8].

The endpoints composing the *stfX* API are:

[8] SwaggerHub. stfX API. Available at https: https://app.swaggerhub.com/apis/EdgarACarneiro/thesis/2.1.1. Accessed in 2020-06-21.

- POST `/storyboard`: endpoint responsible for loading the dataset and the respective metadata to the *stfX*. Expects to receive in the request body an object containing the fields *dataset* and *metadata*. The dataset must be modelled as an array of the phenomenon shape descriptors. A phenomenon shape is modelled as an array of points, where each point is represented by an array containing the point coordinates. The metadata field, regarding the dataset, must contain two properties: *startTime*, indicating the timestamp, in milliseconds, corresponding to the dataset first temporal snapshot; and *timePeriod*, the number of milliseconds separating each dataset temporal snapshot. Moreover, a third metadata parameter can also be provided: the dataset name. After processing, the *stfX* returns the ID of the Storyboard created using the provided data. Listing 5.10 displays an input example.

```
{
    "dataset": [
        [
            [0, 1],
            [1, 0],
            [0, 0],
        ], [
            [4, 5],
            [5, 4],
            [4, 4]
        ],
        ...
    ],
    "metadata": {
        "startTime": 1593730800000,
        "timePeriod": 86400000,
        "name": "ExampleDataset"
    }
}
```

Listing 5.10: Example body of a request to the POST `/storyboard` endpoint.

- POST `/storyboard/file`: endpoint identical to `/storyboard` that receives the input through a JSON file, rather than the request body.

- POST `/storyboard/{id}`: endpoint responsible for retrieving the frames of a particular Storyboard. The path parameter `{id}` indicates the Storyboard from which the features of interest will be retrieved. Through the request body, the *stfX* expects to receive a *json* object describing the thresholds that shall be used to identify features of interest. Listing 5.5 is a valid example of a set of thresholds. If the thresholds regarding a feature are not provided (.e.g in Listing 5.5 no thresholds are provided for uniform scales) then that change feature is not analysed. If part of the thresholds regarding a feature are not provided (e.g. in Listing 5.5, thresholds *absoluteAcc* and *noiseEpsilon*) then those thresholds are not analysed.

Additionally, two query parameters can also be introduced by the user: *initialTimestamp*, meaning the timestamp above which temporal snapshots are parsed (if none is given the *stfX* assumes the timestamp associated to the Storyboard first temporal snapshot); *finalTimestamp*, meaning the timestamp below which temporal snapshots are parsed (if none is given the *stfX* assumes the timestamp associated to the Storyboard last temporal snapshot).

Listing 5.11 displays an example response given by the *stfX*.

```
[
  {
    "events": [
      {
        "threshold": "DIRECTED_ACC",
        "type": "TRANSLATION",
        "trigger": {"transformation": [1,0]}
      }, {
        "threshold": "ABSOLUTE_ACC",
        "type": "ROTATION",
        "trigger": {"transformation": 15}
      }
    ],
    "temporalRange": [84, 99],
    "phenomena": [
      {
        "representation": [...],
        "timestamp": 84
      },
      ...
    ]
  }, {
    "events": [
      {
        "type": "IMMUTABILITY",
        "trigger": {"transformation": 6}
      }
    ],
    "temporalRange": [99, 105],
    "phenomena": [...]
  }, {
    "events": [
      {"type": "UNIMPORTANT"}
    ],
    "temporalRange": [105, 120],
    "phenomena": [...]
  }
]
```

Listing 5.11: Example body of a POST `/storyboard/{id}` endpoint response.

The example shown in Listing 5.11 contains a total of three `Frame` instances (subsection 5.2.2.4). Each `Frame` is composed by (1) a list of the identified co-occurring features of interest (named as events), (2) a temporal range, indicating when the features of interest occur, and (3) the phenomenon representation, as a succession of objects containing the phenomenon shape descriptors (similar format to the one used for the dataset in the POST `/storyboard` endpoint) and the associated timestamp.

1. The first `Frame`, from timestamp 84 to timestamp 99, encompasses two different features of interest: a translation of 1 unit in the x-axis, triggered by a directed accumulated threshold; and a rotation of 15 degrees counterclockwise; triggered by an absolute accumulated threshold. The impact of having distinct `TransformationDataType` (subsection 5.2.2.3) is perceivable in this example, as a list of values quantifies the translation while a single value quantifies the rotation.
2. The second `Frame`, from timestamp 99 to 105, is composed by a feature of interest of the immutability type, that lasts six milliseconds. Features of interest of the immutability type do not have an associated threshold type.
3. The third `Frame`, from timestamp 105 to 120. This temporal range is considered unimportant (a concept introduced in section 4.2.8) since it contains a single feature of interest of the unimportant type. Features of interest of the unimportant type do not have an associated threshold type, nor an associated trigger transformation as no thresholds detected this feature.

- DELETE `/storyboard/{id}`: endpoint responsible for deleting a storyboard and all of its associated information (e.g. change features computed). The path parameter `{id}` indicates the Storyboard to be deleted.

- GET `/storyboard/metadata/{id}`: endpoint responsible for retrieving the dataset metadata initially submitted in the Storyboard creation. The path parameter `{id}` indicates the Storyboard from which the user is requesting to get the dataset metadata. This endpoint response follows the same JSON schema as the metadata object provided in endpoint POST `/storyboard`.

5.4 Deployment

To ease the *stfX* usage and also facilitate future deployments, the *stfX* has been containerised. With this intent, we used both Docker[9] and Docker Compose[10]. The *docker-compose* developed for the *stfX* is responsible for orchestrating a total of four containers:

1. the *cpd-service* container is responsible for running the PSR service (section 5.2.1);

[9] Docker Inc. Docker. Available at https: `https://www.docker.com`. Accessed in 2020-06-22.

[10] Docker Inc. Overview of Docker Compose. Available at https: `https://docs.docker.com/compose/`. Accessed in 2020-06-22.

2. the *stfx-core* container is responsible for running the Core service (section 5.2.2);
3. the *swagger* container purpose is to serve the web application displaying an API interface and documentation (built with Swagger), connected to the Core service;
4. the *nginx* container purpose is to proxy the requests from `/stfx` to the *stfx-core* service and the requests from `/stfx/api-docs/` to the *swagger* web application.

5.5 Extension

Seeing the pioneering nature of the current study, we foresee that future developments will most likely start with improving or extracting other change features from spatiotemporal data. Hence, in this section, we provide some guidelines for the extension of the *stfX* current capabilities, towards new change features detection:

1. Start by developing the spatiotemporal change features extractor as a separate microservice;
2. The `StoryboardController` should call the developed microservice and store the retrieved information, similarly to the spatial change features already implemented;
3. Adapt the threshold models so that the *stfX* can receive thresholds relative to the new change features;
4. For each new change feature, create a new `TransformationDataType` class that accurately represents its quantification, if one does not already exist;
5. Create a class implementing `ITransformationParser` that is capable of parsing the extracted spatiotemporal change features, using the previously defined thresholds;
6. Add the new parser to the `ParserFactory`.
7. Update both the `ThresholdTrigger` and `Transformation` enumerations, in the `Event` class to support the newly defined change features.

5.6 Summary

In this chapter, we provided the most significant details of the prototype developed in this dissertation for implementing the conceptual framework detailed in chapter 4: the SpatioTemporal Features eXtractor (*stfX*). Section 5.1 presented an overview of the *stfX*, followed by section 5.2 which detailed the architecture of the *stfX* and analysed how the different components of the various services interact. The *stfX* is a server-side application utilising the microservices architectural approach, composed of two services: the PSR service (subsection 5.2.1) and the Core service (subsection 5.2.2). The PSR service is responsible for computing the spatial change features occurring in a phenomenon. The Core service is the central service of the *stfX* and the one responsible for several tasks, such as storing data in an in-memory database, filtering the features of interest, indexing them by temporal range, coalescing similar temporal ranges and identifying unimportant ones. Moreover, this section also describes a new accumulated threshold named *directed accumulated threshold* (*directedAcc*). Following, section 5.3 examined the *stfX* API while also presenting template requests and responses to the identified endpoints. Section 5.4 described the containerisation approach adopted, using Docker, in order to ease future deployments. Finally, section 5.5 provided some guidelines for the extension of the *stfX*.

Chapter 6

Validation and Results

This chapter describes the methods utilised to validate the devised prototype, as well as analyses the validation results. Section 6.1 outlines the developed unit tests while also determining which prototype components they validate. Afterwards, section 6.2 details the methodology used to assess the results obtained. Subsequently, section 6.3 presents the conducted experiments while also analysing and discussing the results obtained. Finally, section 6.4 assesses whether the designed prototype satisfies the established requirements and determines the answers to the research questions raised in chapter 3.

Before proceeding, it is relevant to recall the meaning of frame in this work's context (firstly introduced in subsection 5.2.2.4): a frame represents an entity aggregating a temporal range, as well as the identified features of interest and phenomenon representations for that temporal range. Moreover, it also becomes pertinent to highlight that in this section, the concepts of change feature and of transformation are used interchangeably, as a consequence of the features of interest that the *stfX* is capable of identifying.

All *python* scripts, resources needed to run the experiments described and the obtained results are available in [12].

6.1 Validation through unit testing

In order to ensure the *stfX* proper behaviour, validate the implementation and make future extensions easier, we developed a set of unit tests that verify the correct operation of the *stfX* core functionalities. These unit tests represent 45% of the code lines developed and have a code coverage of 92% (line coverage). Since many tests share the same validation purpose, we decided to aggregate them accordingly. The validation purpose of each of the aggregated set of unit tests, along with the respective section that represents the associated *stfX* functionality in chapter 5, are as follows:

UT1: Validate all routes in the API, the ability to deal with erroneous input and the correct functioning when proper input is given.

UT2: Test the ability to receive and deal with the query parameters that control the temporal range of the Storyboard from which the *stfX* will extract the features of interest (feature presented in 5.3).

UT3: Test the ability to coalesce frames, in situations in which they can and cannot be coalesced (feature presented in 5.2.2.5).

UT4: Validate if noise filtering effectively filters noise produced by the services responsible for computing spatiotemporal change features (feature presented in 5.2.2.2).

UT5: Assert that when a given threshold regarding a spatiotemporal change feature is not provided, the corresponding feature is not analysed (feature presented in 5.3).

UT6: Verify that even though a given set of thresholds did not identify features of interest for the full temporal range, the temporal range is fully described, with frames that have no features of interest being classified as unimportant (feature presented in 5.2.2.5).

UT7: Validate that the *stfX* can partition features of interest to aggregate all co-occurring change features in a given temporal range together (feature presented in 5.2.2.4).

UT8: Validate that the *stfX* can successfully retrieve all the features of interest indexed by temporal range, rather than by the features of interest themselves (feature presented in 5.2.2.4).

The developed unit tests are useful to assert the *stfX* capability of handling simple test scenarios (e.g. generate an unimportant temporal range if no features of interest were detected). However, for more complex scenarios and to assert the *stfX* efficiency regarding the retrieved features of interest (i.e., verify if they accurately describe the Storyboard), a more elaborate approach is necessary.

The presented unit tests are used in continuous integration (CI), meaning that every new commit to the repository containing the *stfX* code is tested, thus facilitating the detection of errors in the *stfX* functioning.

6.2 Validation methodology on experiments

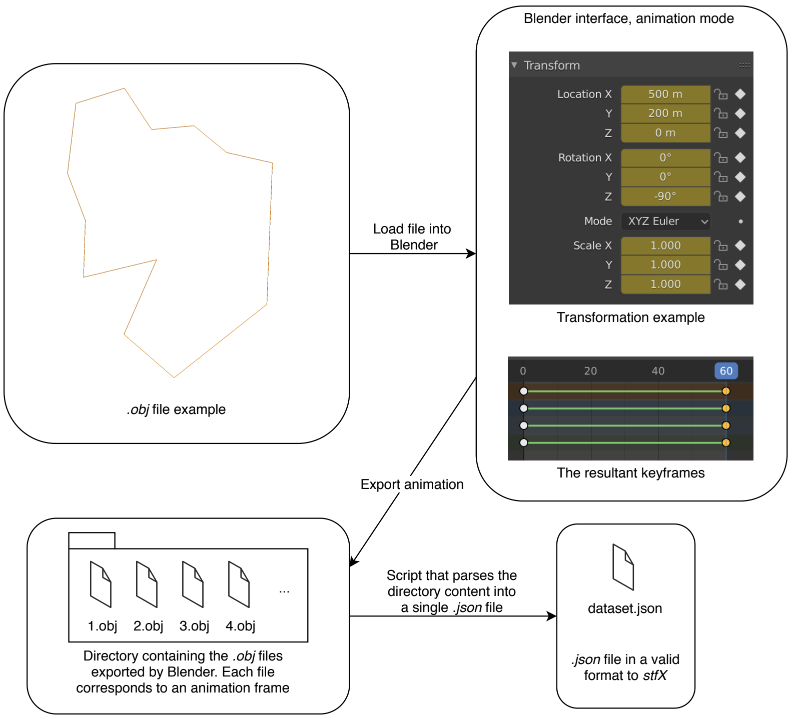
This section describes the experiments pipeline and validates the solution presented in chapter 4 by running the experiments. For these experiments, we used synthetic test datasets with the goal of using the known ground-truth to validate the experiments. Section 6.2.1 starts by analysing the process used to generate the synthetic test datasets, in a way that we can guarantee the integrity of the datasets. Next, section 6.2.2 describes how the validation, using the test datasets, is done and which metrics are used to assert the solution efficacy.

6.2.1 Test datasets generation

The first step of the validation process consists of building synthetic spatiotemporal datasets. We chose to use synthetic datasets seeing that it granted us complete control over the changes affecting a phenomenon. Therefore, the change features utilised to create the datasets are used as the ground truth necessary for measuring the efficiency of the developed solution.

The datasets generation process consists of creating a polygon, animating it by applying a set of transformations S, and computing the temporal snapshots to feed to the system. To help in this, it is useful to use an animation tool that can handle a shape description in a known format and output the animation results also in a treatable format. We chose to use the *obj* file format for the shape description, and the animation tool Blender[1]. Blender is an open-source 3D software, used for modelling, animation, simulation, rendering, among other functionalities. Using Blender's animation mode, whose purpose is to make an object move or change shape over time, we create the keyframes of the animation. The keyframes represent the application of spatial transformations to the loaded phenomenon. Additionally, it is also possible to change the interpolation mode between keyframes. At last, we export the developed animation as an ordered set of *obj* files (one per animation frame).

Considering Blender's output format and *stfX* input format, we then parsed the directory containing the animation modelled as *obj* files into a valid JSON file. With this intent, a *python* script was developed. Afterwards, the metadata object is added manually, thus composing a viable dataset to the *stfX* server. Figure 6.1 illustrates the test generation pipeline.


Blender interface, animation mode
Transform
Transformation example
0
20
40
60
The resultant keyframes
.obj file example
Load file into Blender
Export animation
Script that parses the directory content into a single *.json* file
1.obj
2.obj
3.obj
4.obj
...
Directory containing the *.obj* files exported by Blender. Each file corresponds to an animation frame
dataset.json
.json file in a valid format to *stfX*

Location X	500 m
Y	200 m
Z	0 m
Rotation X	0°
Y	0°
Z	-90°
Mode	XYZ Euler
Scale X	1.000
Y	1.000
Z	1.000

Figure 6.1: Test generation pipeline.

[1] Blender. Blender. Available at https: https://www.blender.org. Accessed in 2020-05-18

6.2.2 Validation Metrics

The validation is performed through automatic processes that are developed specifically for the purpose. The main goal of using automated validation is to dismiss the necessity of integrating a user in the validation process. By employing automatic validation, objectivity, rigour, scalability and process control are assured, while discarding possible errors that derive from perceptual subjectivity. Furthermore, automated validation makes recreation and revalidation of the developed tests more feasible.

For validation, we use a *python* script that outputs the efficiency of the solution in the test scenarios. An experiment consists of a dataset, a set of defined thresholds and the expected result. The latter consists of the ground truth modelled in a format similar to the one returned by the server. For each experiment, the dataset is loaded into the *stfX* server. Using the *ID* retrieved by the server and the thresholds previously defined, the features of interest are obtained. To validate the solution, we compare the features of interest returned by the server with the expected result. Furthermore, we also compare the spatial disposition of the phenomenon at the end of the ground truth animation with the one resulting from the features determined by the *stfX*.

Seeing that few frameworks with identical goals exist (Yi et al. [98] being the one with most resemblances), validation metrics were researched in domains with similar problems. In Yi et al.'s work [98], the validation is achieved by applying the framework in a well-documented study of oceanic eddies [15] and verifying if the output is similar to the one obtained in previous studies. Regarding Point Set Registration algorithms, that aim at discovering the transformation between two point sets, the validation metric is an average distance function between a point in the warped model shape and the ground truth corresponding point (e.g. mean squared error distance [100], average Euclidean distance [63, 56]). However, these metrics have increased value for validating a solution when there are benchmarks and other performances for comparison. Otherwise, in the absence of comparison, a distance metric gives fewer insights into the solution performance (e.g. the distance metric has different meanings on shapes of different size). Considering the absence of similar frameworks and, consequently, of datasets and results for comparison, the validation of our novel solution suffers from the lack of a reference metric that can ascertain its efficiency.

To validate our tool, we present two original distinct metrics we studied, based on well understood and widespread distance functions. The metrics are:

M1: metric that consists of comparing the ground truth final representation with the representation obtained by applying the transformations outputted by the *stfX*. The goal is to identify if the transformations recognised by the *stfX* result in equivalent representations to the ground truth. With this goal in mind, distance functions emerge as the first approach. However, as already referred, without a benchmark for comparison, this approach has decreased value. Therefore, we devised a solution inspired by the ratio between areas of polygons and convex hulls [38], that follows the distance function principle of penalising more distant polygons. However, this metric is agnostic to the transformations that led to the final representation (e.g. a translation followed by a change in rotation might give the same result as a rotation followed by a translation).

To compute this metric, we use the following procedures:

(a) Start by retrieving the final phenomenon representation (last representation of the last frame of the response body returned) and the transformations that the *stfX* identified. Apply the identified transformations to the first representation of the first frame of the response body (initial phenomenon representation), obtaining the estimated final phenomenon state.

(b) Having both the real final representation, P_R, and the estimated final representation, P_E, we can proceed with their comparison. We start by identifying the minimum area between the convex hull of P_R and the convex hull of P_E. Next, we obtain the area of the convex hull including the points from both representations (joint convex hull). At last, the ratio between the minimum area and the area of the joint convex hull is calculated. To compute the areas of the convex hulls we employ the surveyor's formula (also known as shoelace formula or Gauss's area formula) [10]. Therefore, *M1* can, in a simplified manner, be described as:

$$M1 = \frac{min(area(convex\_hull(P_R)), area(convex\_hull(P_E)))}{area(convex\_hull(P_R, P_E))}$$

where P_R and P_E stand for polygons, modelled as $((x_1,y_2),(x_1,y_2),\cdots,(x_n,y_n))$. P_R represents the real final representation, while P_E represents the estimated final representation. Figure 6.2 illustrates the convex hulls used in the computation of metric *M1*. For figure 6.2, and for all figures depicting similar information to it, the red area consists of the joint convex hull. In contrast, the blue area consists of the polygon with the minimum area (between the real representation, P_R and the estimated representation, P_E). Therefore, the area depicted in purple represents the overlapping between the two previously introduced areas. Regarding the drawn polygons, the blue polygon contour line depicts the real representation, P_R, while the red polygon contour line depicts the estimated representation, P_E.

M2: metric that consists of analysing the similarity between the set of transformations outputted by the tool and the ground truth transformations. The goal is to identify how similar are both sets of transformations and, therefore, how effective is the devised solution. To do so, we base ourselves on the longest common subsequence distance [67, 5] (allows insertions and removals, costing 1), a distance function deriving from the widespread Levenshtein distance (often named edit distance) [51]. However, *M2* is agnostic to the identified transformations' values and, consequently, to the representations resultant of applying the transformations. To compute this metric, we use the following procedures:

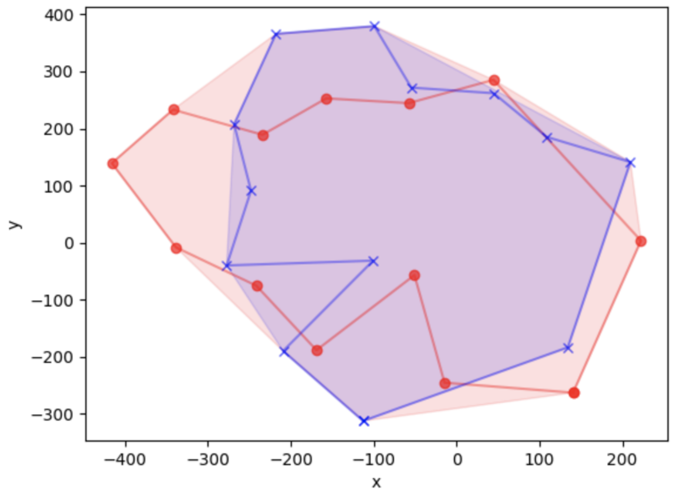


Figure 6.2: Example of visual application of metric *M1*.

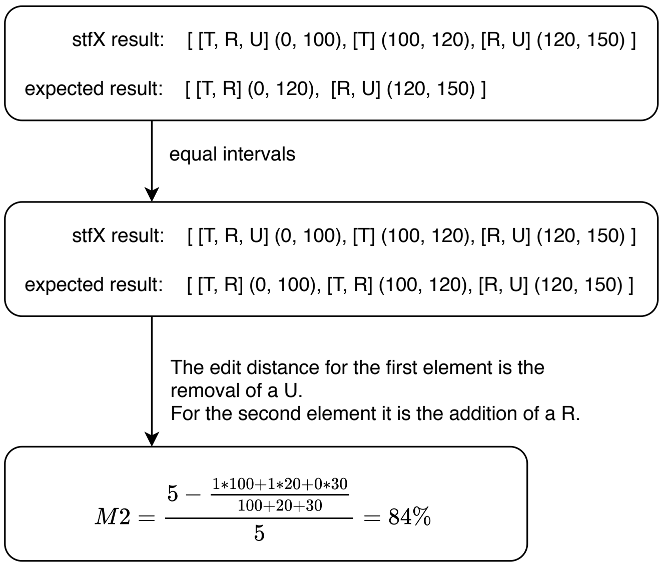
(a) Parse the obtained result and the expected result so that both representations have the same number of identified transformations, with matching temporal ranges. Achievable by splitting temporal ranges.

(b) For each temporal range of the estimated result, and the correspondent temporal range in the real result, find the minimum number of operations (only insertions and removals) that allow transforming the estimated set of transformations into the real set of transformations. Employing set theory [50], and considering the estimated set of transformations as S_E and the real set of transformations as S_R, the minimum number of operations transforming S_E into S_R can be modelled as $|(S_E - S_R) \cup (S_R - S_E)|$, where $S_E - S_R$ represents the necessary removals, $S_R - S_E$ the necessary insertions, and where $|S|$ represents the cardinality of set S. As the worst-case scenario, occur $|S_E| + |S_R|$ operations: the removal of all elements from S_E and insertion of all elements from S_R.

(c) The metric is then obtained by computing the ratio between the weighted average of the minimum distance between sets and the worst-case scenario. The weighted average takes into account the set temporal range and the total temporal range. Therefore, the metric can be modelled as:

$$M2 = 1 - \frac{\sum_{n=1}^{N} \frac{|(A_n - B_n) \cup (B_n - A_n)| * R_n}{|A_n| + |B_n|}}{\sum_{n=1}^{N} R_n}$$

where A and B are arrays of sets of equal size where $\forall A_n \in A : A_n \subseteq T$ and $\forall B_n \in B : B_n \subseteq T$, where T is the set of possible transformations, and R is an array containing the temporal ranges duration. Figure 6.3 shows an example computation of this metric. In this figure, the elements inside the square brackets are representative of the transformations, while the proceeding round brackets indicate the associated temporal range.


stfX result: [[T, R, U] (0, 100), [T] (100, 120), [R, U] (120, 150)]
expected result: [[T, R] (0, 120), [R, U] (120, 150)]
equal intervals
stfX result: [[T, R, U] (0, 100), [T] (100, 120), [R, U] (120, 150)]
expected result: [[T, R] (0, 100), [T, R] (100, 120), [R, U] (120, 150)]
The edit distance for the first element is the removal of a U.
For the second element it is the addition of a R.

$$M2 = \frac{5 - \frac{1*100+1*20+0*30}{100+20+30}}{5} = 84\%$$

Figure 6.3: Example computation of the *M2* metric.

The objective with the identification of the two referred metrics is to evaluate the several characteristics that comprise an effective solution. Hence:

- Metric *M1* evaluates the outputted features of interest transformations values, and the resemblance of the actual representation with the representation resultant of the application of the *stfX* identified features of interest. Therefore, this metric focuses on the final result, rather than on the process that led to the outcome. As a consequence, both the frames order and the temporal ranges are neglected, leading to the possibility of erroneous sequences of features of interest being highly rated.

- Metric *M2* evaluates the sequence of transformations (extracted from the frames), as well as the respective temporal ranges, while disregarding the transformation values. Hence, this metric focus on the sequence of features of interest leading to the result, rather than the result itself. As a downside, transformations with disproportionate values (in comparison to the actual ones) still score high.

Therefore, metrics *M1* and *M2* should be employed together, as they are complementary: successful test scenarios imply that both metrics have a high score (we consider values above 0.7 as satisfactory and above 0.9 as exceptional).

From our analysis, we identified a circumstance that can elude the conjunct application of these metrics. This circumstance is verified when the sequence of transformations is correctly identified but with unexpected trigger values that, nonetheless, still lead to the correct final representation. A possible example is a scenario in which the *stfX* identifies a rotation of 10^{o} degrees followed by a rotation of -10^{o}degrees. However, the ground-truth rotations are of 50^{o} and -50^{o}, respectively. In this example, both metrics would get a perfect score, even though the sequence of change features did not correctly describe reality. This circumstance results from metric *M2* not taking into account the values of the transformations (as stated at the beginning of metric *M2* description).

Nonetheless, we still believe that the exhibited metrics present a valid methodology for validating the developed solution, as the *stfX* is based on previously studied methods [63] and, therefore, is not expected to give results so discrepant from reality.

6.2.3 Summary

This section describes the validation methodology used in the experiments. Subsection 6.2.1 details the pipeline used to generate the test datasets. Summarily, it consists of defining the transformations over a phenomenon utilising Blender keyframes and then exporting to a format that the *stfX* can interpret. Subsection 6.2.2 introduces two novel metrics to evaluate the results obtained in the experiments. Metric *M1* evaluates the resemblance between the obtained phenomenon and the ground-truth final representations, while metric *M2* evaluates the correctness of the identified features of interest and respective temporal ranges. The presented metrics are complementary and should be utilised in conjunction to obtain more reliable results.

6.3 Experiments and results

In this section, we describe the experiments used for validating the proposed solution. Moreover, we analyse each of the obtained results and discuss them.

Throughout the experiments presented in this section, we use the polygon shown in figure 6.4. We chose to use this polygon as it is non-convex, asymmetric and synthetically generated.

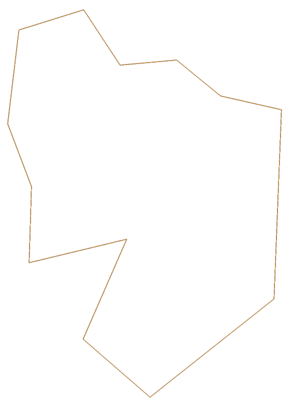


Figure 6.4: Polygon used in the experiments presented in section 6.3.

For all the performed tests, we present the score obtained using the metrics defined in section 6.2.2. Furthermore, in cases demanding more insights on the metrics, we identify for metric

M1 the convex hulls used in the ratio computation (as depicted in figure 6.2), and for metric *M2* the differing insertion and removals, as well as the associated temporal ranges.

For the remaining of this section, each subsection outlines an experiment with a different goal. An experiment can be composed of various tests, as long as these contribute to the same purpose.

6.3.1 Single change features

The goal of this experiment is to evaluate if the *stfX* is capable of identifying individual transformations when given appropriate thresholds. An appropriate threshold stands for a threshold matching the exact change in the ground-truth (e.g., if a rotation of 50º occurred, we define the directed accumulated threshold — *directedAcc* — as 50). We ran a test for each of the spatiotemporal change features capable of being identified, meaning that a total of four tests were performed. The different tests consist of:

- a translation of 500 units in the x-axis and 200 units in the y-axis, during 60 time units;
- a final rotation of 10º clockwise, resultant of a first rotation of 20º counterclockwise, during 30 time units, followed by a second rotation of 30º counterclockwise, also during 30 time units;
- a uniform scale with value 2.52, resultant of the accumulation of three distinct uniform scales: a scaling of 1.4 in the first 20 time units, a scaling of 1.2 in the next 20 time units, and a scaling of 1.5 in the last 20 time units;
- the phenomenon remains immutable during 60 time units.

All transformations progress linearly[2], meaning that the rate of change is equal throughout the entire transformation.

Results

The results obtained are presented in Table 6.1. Additionally, figure 6.5 presents the obtained visual representation for metric *M1* in the rotation test. This figure is presented with intuit of highlighting the complete overlapping when metric *M1* has a maximum score.

Table 6.1: Score obtained for each of the fundamental transformations.

Transformation	***M1* score**	***M2* score**
Translation	1.0	1.0
Rotation	1.0	1.0
Scale	1.0	1.0
Immutability	1.0	1.0

[2]Blender Manual. F-Curves Introduction. Available at https: `https://docs.blender.org/manual/en/latest/editors/graph_editor/fcurves/introduction.html#interpolation-mode`. Accessed in 2020-05-31.

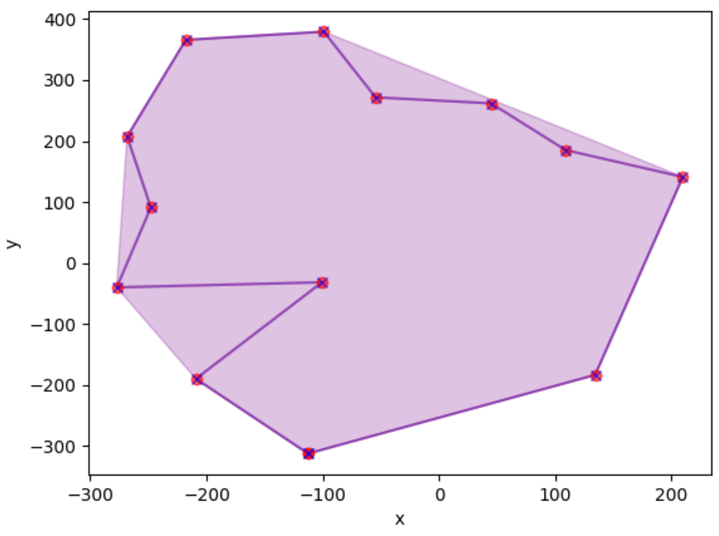


Figure 6.5: Metric *M1* visual representation in a maximum score.

Discussion

From the analysis of the performed tests, we can conclude that the *stfX* is capable of effectively identifying singular transformations when the thresholds are appropriate. Despite also having defined the remaining thresholds, no other features of interest were identified, besides the ones supposed to. Therefore, in these experiments, no relevant noise was produced by the service responsible for quantifying the spatial change features between temporal snapshots, the PSR service (introduced in section 5.2.1).

6.3.2 Sequence of change features

The goal of this experiment is to evaluate if the *stfX* is capable of identifying the occurrence of several different sequential transformations. In this experiment, the transformations vary, and there are no multiple transformations (meaning that there are no co-occurring spatial change features). Additionally, appropriate thresholds are once more employed. The ground-truth sequence of transformations consists of: immutability, from time unit 0 to time unit 30; a translation of 1000 units in the x-axis and minus 400 units in the y-axis, from time unit 30 to time unit 90; a rotation of 50^{o} counterclockwise, from time unit 90 to time unit 120; a rotation of 50^{o} clockwise, from time unit 120 to time unit 150; a uniform scaling of 1.3, from time unit 120 to time unit 180; immutability, from time unit 180 to time unit 210; and a uniform scaling of approximately 0.377, from time unit 240 to time unit 270.

Results

The obtained results consist of the maximum classification for both metrics, meaning the score for *M1* and *M2* is 1.0. Figure 6.6 presents representation estimated by the *stfX*, using the visualisation framework proposed by Marques [57]. In this visualisation framework, Marques uses the real representation to draw the polygons, while displaying the transformations identified by the *stfX*.

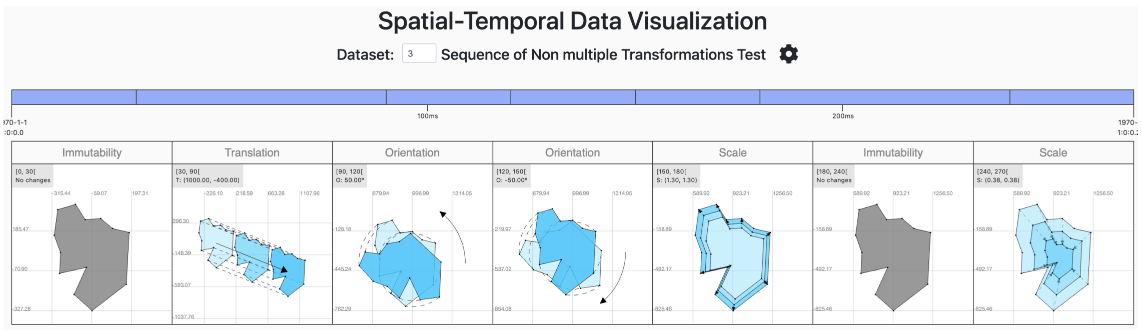

Spatial-Temporal Data Visualization
Dataset: 3 Sequence of Non multiple Transformations Test
100ms
200ms
Immutability
Translation
Orientation
Orientation
Scale
Immutability
Scale
[0, 30[No changes
[30, 90[T: (1000.00, -400.00)
[90, 120[O: 50.00°
[120, 150[O: -50.00°
[150, 180[S: (1.30, 1.30)
[180, 240[No changes
[240, 270[S: (0.38, 0.38)

Figure 6.6: Estimated representation obtained for experiment 6.3.2, using the visualisation framework proposed by Marques [57].

Discussion

From the performed experiment, we can conclude that the *stfX* is capable of correctly identifying the sequence of transformations, using appropriate thresholds. For all transformations, only *directedAcc* thresholds were defined, as well as *noiseEpsilons* (introduced in subsection 5.2.2.2). Therefore, only the accumulation of directed transformations can trigger the thresholds. It is also important to highlight that some thresholds are defined slightly below the ground-truth value (e.g., the *directedAcc* threshold of rotations is defined as 49.5, even though all ground-truth rotations are of 50º). The necessity for this minor tweak is further analysed in the next experiment (subsection 6.3.3).

6.3.3 Sequence of change features, with modified thresholds

This experiment's goal is similar to the previous one (subsection 6.3.2). However, some modifications regarding the thresholds were made. These modifications consist of: for translations, we use the *absoluteAcc* thresholds instead of the *directedAcc* threshold; for rotations, the *directedAcc* threshold is set to 50, rather than 49.5. Therefore, by our definition of appropriate thresholds, presented in subsection 6.3.1, the chosen thresholds are still appropriate. The dataset used in this experiment is the same utilised in the experiment described in section 6.3.2.

Results

When running this experiment, the score obtained for metrics *M1* and *M2* was, respectively, 0.476 and 0.740. Figure 6.7 and Listing 6.1 present, in a more thorough manner, the obtained results. Figure 6.7 represents the overlapping of convex hulls obtained for metric *M1*, while Listing 6.1 represents metric *M2* report, meaning the edit distance between the obtained sequence of features and the expected one. Therefore, the JSON displayed in Listing 6.1 is interpreted as: identified an unexpected translation in the temporal range of 90 to 120; identified an unexpected translation and did not identify an expected rotation in the temporal range of 120 to 150 and so on. If a given temporal range does not appear in the *M2* report, then all the features of interest pertaining to this temporal range were correctly identified.

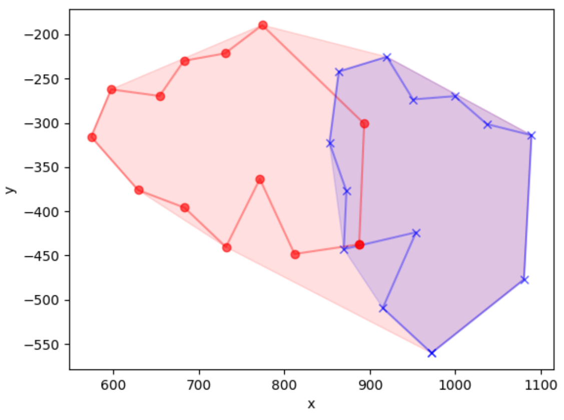


Figure 6.7: Metric *M1* details in experiment 6.3.3.

```
[
    {
        "temporalRange": [90, 120],
        "removals": ["TRANSLATION"]
    }, {
        "temporalRange": [120, 150],
        "removals": ["TRANSLATION"],
        "insertions": ["ROTATION"]
    }, {
        "temporalRange": [150, 180],
        "removals": ["TRANSLATION"]
    }, {
        "temporalRange": [180, 240],
        "removals": ["TRANSLATION"]
    }
]
```

Listing 6.1: Metric *M2* details in experiment 6.3.3.

Figure 6.8 presents the result obtained from the *stfX*, using the visualisation framework proposed by Marques [57].

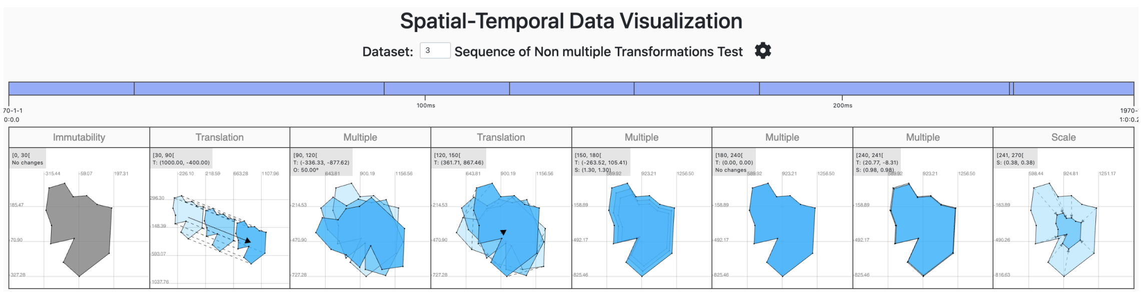

Spatial-Temporal Data Visualization
Dataset: 3 Sequence of Non multiple Transformations Test
100ms
200ms
Immutability
[0, 30[No changes
Translation
[30, 90[T: (1000.00, -400.00)
Multiple
[90, 120[T: (-336.33, -877.62) O: 50.00°
Translation
[120, 150[T: (361.71, 867.46)
Multiple
[150, 180[T: (-263.52, 105.41) S: (1.30, 1.30)
Multiple
[180, 240[T: (0.00, 0.00) No changes
Multiple
[240, 241[T: (20.77, -8.31) S: (0.98, 0.98)
Scale
[241, 270[S: (0.38, 0.38)

Figure 6.8: Estimated representation obtained for experiment 6.3.3, using the visualisation framework proposed by Marques [57].

Discussion

From the results obtained in this experiment, we can conclude that the small modifications made, in comparison to the previous experiment, deeply affected the *stfX* performance (considering as the performance the metrics score).

On the one hand, the change of the *directedAcc* threshold to *absoluteAcc* threshold, in translations, led to the identification of unexpected translations, accumulated from time unit 90 to time unit 240. However, from the ground-truth, it is known that there occurs a single translation (that is successfully identified). Therefore, all the unexpected translations are the result of the accumulation of significant random noise produced by the *stfX* PSR service. Hence, we can conclude that the absolute accumulator defined by the *AbsoluteAcc* is most propitious to accumulate noise since in no case is this accumulator reset. Contrarily, the directed accumulator, defined by the *directedAcc* threshold, is reset when the direction changes.

On the other hand, the change of the *directedAcc* threshold, on rotations, from 49.5 to 50, led to the clockwise rotation of 50^o, from time unit 120 to time unit 150, not being identified. This circumstance is the consequence of the *stfX* PSR service being very precise but not exact. Consequently, minor errors can be accumulated throughout the processing, thus causing the accumulated value to be inferior to the defined threshold. Therefore, we can conclude that as a measure of precaution, it is good practice to establish the threshold slightly below the transformation value of interest.

6.3.4 Co-occurring transformations

In this experiment, the objective is to determine whether *stfX* is capable of successfully identifying co-occurring transformations (multiple transformations) in a simple test scenario. Furthermore, by applying three different sets of thresholds to the same test dataset, we also determine the impact of the choice of thresholds in the quality of the results.

The ground-truth transformations consist of, simultaneously, a translation of minus 200 units in the x-axis and 40 units in the y-axis; a rotation of 45^o clockwise; and a uniform scaling of scaling factor 2. All the transformations co-occur in a smooth manner (the first and last transformations are smaller, thus causing the transformation to be smoother[3]) during the temporal range 0 to 60. Figure 6.9 illustrates the described transformations.

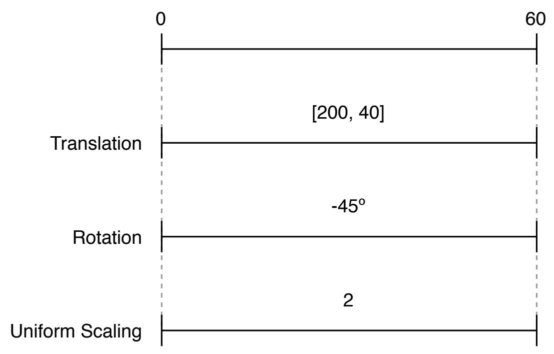


Figure 6.9: Visual representation of the transformations of the dataset used in experiment 6.3.4.

Regarding the three different sets of thresholds, they all contain the same *noiseEpsilon* threshold for all transformations, the same *delta* threshold and the same immutability threshold value. However, they differ in the remaining parameters, in the following manner:

[3]Blender Manual. F-Curves Introduction. Available at https: `https://docs.blender.org/manual/en/latest/editors/graph_editor/fcurves/introduction.html#interpolation-mode`. Accessed in 2020-05-31.

T1: The first set of thresholds consists of appropriate thresholds, meaning that thresholds are equal to the exact values of the occurring ground-truth transformations. Therefore, *absoluteAcc* on translations is set to 203.9, *directedAcc* on rotations to 45 and *directedAcc* on uniform scaling is set to 1.

T2: The second set of thresholds takes into account the conclusion obtained from experiment 6.3.3. Hence, the thresholds are set to slightly smaller values, namely: translation *absoluteAcc* is set to 202, rotation *directedAcc* to 44 and uniform scaling *directedAcc* to 0.9.

T3: The third set of thresholds is based on the *stfX* capability of coalescing similar frames, as shown in subsection 5.2.2.5. Thus, the thresholds are set to minimal values. As a consequence, in an initial stage, the *stfX* identifies many distinct frames. Later, these frames are merged into smaller coalesced ones. Regarding the threshold values, those are: on translations, *absoluteAcc* is set to 1; on rotation, *directedAcc* is set to 1; on uniform scaling, *directedAcc* is set to 0.02.

Results

The results obtained are shown in Table 6.2. Through its analysis, it is perceivable that the usage of thresholds with values equal to the ground-truth values (*T1*) leads to the worst-performance. Even more, metric *M2* was 0.0, meaning that no features of interest were correctly identified, for any temporal range. The approach using minimal threshold values (*T3*) obtained the best score, in both metrics. Moreover, the metric scores are both high and similar, thus showing that this approach is the most cohesive and adequate one. Comparing the results obtained by the set of thresholds *T2* and *T3*, we verify that the most significant discrepancy in the score is on metric *M2*. While metric *M1* increased 0.073 (increase of 10%), metric *M2* increased 0.24 (increase of 35%).

Table 6.2: Score obtained by the application of each set of thresholds.

Set of thresholds	***M1* score**	***M2* score**
T1	0.247	0.000
T2	0.768	0.685
T3	0.841	0.925

Discussion

The results obtained in this experiment, namely the comparison between the scores of the set of thresholds *T1* with the set of thresholds *T2*, further corroborate the conclusion obtained in experiment 6.3.3 that establishing the thresholds slightly below the desired transformation value is an appropriate approach.

The main takeaway from this experience is that the usage of minimal threshold values leads to better performance. The increasing granularity in the search for features of interest, and consequently, the more detailed retrieved features, justify the increased performance. The downside of this approach is the retrieval of many frames, thus causing the visualisation process for a user (e.g., using the framework of Marques [57]), to be very time-consuming. Seeing that the *stfX* applies coalescing to the retrieved frames (see 5.2.2.5), this downside is handled. By using minimal thresholds, the inclusion of noise (with values superior to the noise filter) in the solution is also facilitated, as it becomes easier to trigger the thresholds. Nonetheless, in this experiment, this approach outperforms the remaining possible methods.

Regarding the comparison of the results obtained by the set of thresholds *T2* and *T3*, we verify that the usage of minimal thresholds increases mostly metric *M2* score. As already referred, the usage of minimal thresholds leads to the identification of more detailed transformations. As a consequence, these transformations have a low impact in the spatial configuration of the phenomenon, thus explaining why the increase of the metric *M2* score does not lead into a significant increase of the metric *M1* score. Moreover, we can also verify that by using thresholds slightly below the ground-truth value, the *stfX* is capable of grasping the core transformations affecting the phenomenon (only missing the more subtle ones).

Concerning the *stfX* ability to identify co-occurring transformations, when employing minimal thresholds, these transformations are successfully detected.

6.3.5 Impact of the service used for quantifying spatial transformations

Throughout the previously performed experiments, we verified that the *stfX* module responsible for quantifying the spatial transformations — the PSR service (introduced in section 5.2.1) — can output noise (experiment 6.3.3). Besides, it was also verified that despite the results being very precise, they are not exact (experiment 6.3.3). As a consequence, to obtain better scores, the usage of thresholds slightly below the ground-truth value or the use of minimal thresholds is required (experiment 6.3.4). Hence, in this experiment, we analyse the *stfX* performance in a scenario in which the module responsible for quantifying the spatial transformations returns perfect results. For this purpose, we developed a simulation server that returns the exact set of transformations occurring between temporal snapshots (known from the ground-truth data). We then re-run the previous experiments using this server and compare the obtained scores with the ones previously collected. This experiment's objective is to assert how the quality of the quantified spatiotemporal transformations affects the *stfX* performance.

In this experiment, we mimicked the transformations occurring in the datasets used in experiments described in sections 6.3.3 and 6.3.4 using the open-source tool *json-server*[4], that allows mocking a REST API quickly. For every test in those experiments, the same dataset, thresholds and expected results files were employed.

[4]GitHub. JSON Server. Available at https: `https://github.com/typicode/json-server`. Accessed in 2020-05-31.

Results

The results obtained are shown in Table 6.3. By its analysis, we observe that the scores obtained using the mock API were improved for all metrics, except for the metric *M2* score in the test using the set of thresholds *T3*.

Table 6.3: Comparison of the scores obtained using *stfX* module responsible for quantifying spatial transformations and using a mock REST API.

		Original Scores		**Scores with mock API**	
Experiment	**Set of thresholds**	***M1* score**	***M2* score**	***M1* score**	***M2* score**
6.3.3	-	0.476	0.740	0.985	0.996
6.3.4	T1	0.247	0.000	0.250	0.500
	T2	0.768	0.685	0.915	0.902
	T3	0.841	0.925	0.938	0.925

Regarding experiment 6.3.3, using the mock API, the error — maximum score (1.0 for both metrics) minus the actual score — associated with both metrics greatly diminished, reaching minimal values (0.015 for metric *M1* and 0.004 for metric *M2*). Moreover, the metric *M1* score increases 106% while the metric *M2* score increases 35%.

Regarding experiment 6.3.4, we verify that in the set of thresholds *T1*, using the mock API, the score for metric *M2* increased from 0.0 to 0.5. This value is explained by the fact that *stfX* was able to identify the occurring translation correctly. However, the score of metric *M1* only suffered a small change of 0.003 (an increase of 1%). Figure 6.10 shows the motive behind such a small increase. Despite the translation being correctly identified, the real representation P_R still majorly encapsulates the resultant estimated representation P_E. Thus, since the translation that occurs remains contained in the real representation P_R, the score for metric *M1* remains almost unaltered.

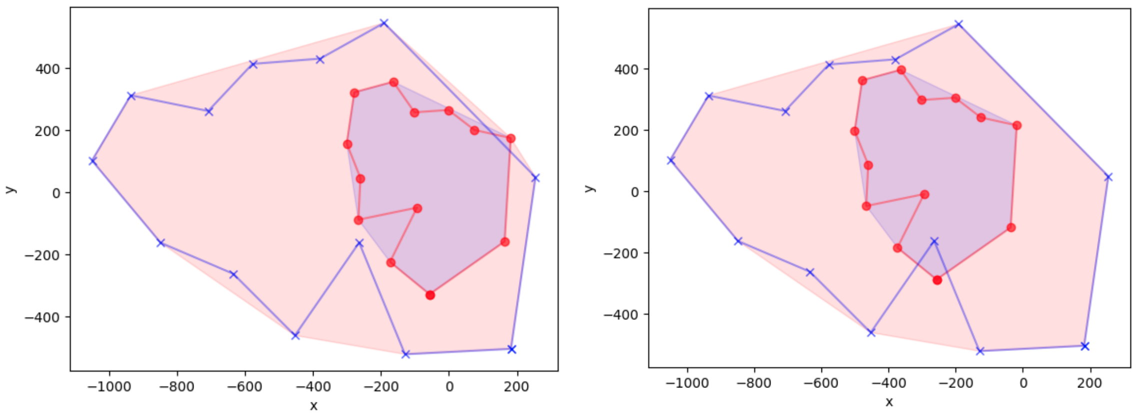


Figure 6.10: Visual representation for metric *M1*, for experiment 6.3.4 using the set of thresholds *T1*, on the original scenario (left) and on when using the mock API (right).

By the analysis of the test scores using the set of thresholds *T2*, we verify an increase of performance that makes the accuracy of this test very similar to the test using the set of thresholds

T3. In the original scores, the difference between the scores was 0.073 (increase of 10%) for metric *M1* and 0.24 (increase of 35%) for metric *M2*. Then employing the mock API, the difference between the scores was, for both metric *M1* and metric *M2*, of 0.023 (increase of 1%).

Discussion

From the results obtained in this experience, we can conclude that the usage of a service capable of quantifying the spatial transformations with high precision majorly increases the quality of the results retrieved by the *stfX*. Additionally, since inadequate quantification of spatiotemporal transformations leads to poor *stfX* performance and precise quantification leads to good *stfX* performance, we can conclude that the *stfX* performance highly depends on the performance of the underlying PSR service. Hence, when looking to increase the *stfX*'s overall performance, a central concern is the search for an algorithm quantifying spatial transformations with high precision.

Regarding all the obtained results, when using the mock API that retrieves the exact transformations occurring, it could be expected that the score for both metrics would be 1.0 (since the transformations are perfectly identified and retrieved by the simulation server). However, this is not the case. For experiment 6.3.3, the choice of thresholds explains the scores obtained. Nevertheless, it is worth mentioning that the difference to the maximum score is minimal. For experiment 6.3.4, noise filtering is the explanation for the results not being maxed out. Seeing that all transformations occur smoothly, the initial and last transformations are not significant enough to surpass the imposed noise thresholds. As a consequence, both the initial and final motions are not detected, thus affecting the scores obtained. Listing 6.2 that displays the output of metric *M2* for the test using the set of thresholds *T3* further corroborates this circumstance since all the missing transformations focus on the temporal ranges 0 to 3 and 52 to 60.

```
[
    {
        "temporalRange": [0, 1],
        "insertions": ["UNIFORM_SCALE", "ROTATION"]
    }, {
        "temporalRange": [1, 3],
        "insertions": ["UNIFORM_SCALE"]
    }, {
        "temporalRange": [52, 55],
        "insertions": ["UNIFORM_SCALE"]
    }, {
        "temporalRange": [55, 59],
        "insertions": ["UNIFORM_SCALE", "ROTATION"]
    }, {
        "temporalRange": [59, 60],
        "insertions": ["TRANSLATION", "UNIFORM_SCALE", "ROTATION"]
    }
]
```

Listing 6.2: Detailed representation of the metric *M2* obtained using the set of thresholds *T3*, using the mocked API.

6.3.6 Noise filtering impact

In the discussion section of experiment 6.3.5, we determined that noise filtering was the reason why the scores were not maxed out when using the mocked API. Hence, as a follow-up experiment, we strive to assert the impact of using noise filtering as well as demonstrate its usefulness. With this intent, we repeat the tests performed in experiment 6.3.2 and experiment 6.3.5 (namely, the one using the mock API and the set of thresholds *T3*) but without applying noise filters.

Results

Table 6.4 shows the results obtained. From its analysis, we verify that in experiment 6.3.2 the removal of the noise filters led to a decrease of 33% in the metric *M2* score, while in experiment 6.3.5 led to an increase of 5% in metric *M1* and 4% in metric *M2*. In a more in-depth analysis of the score obtained for metric *M2*, in the first test, we discovered that without noise filters, the immutability transformations are not identified.

Table 6.4: Comparison between the scores of experiments 6.3.2 and 6.3.5 (with mock API, thresholds *T3*), with and without noise filters.

	With noise filters		**Without noise filters**	
Experiment	***M1* score**	***M2* score**	***M1* score**	***M2* score**
6.3.2	1.000	1.000	1.000	0.665
6.3.5 (with mock API, thresholds *T3*)	0.938	0.925	0.987	0.960

Discussion

Regarding the first test conducted in this experiment, we conclude that the removal of noise filters led to the *stfX* not being able to identify immutability transformations. This circumstance, already foreseen in section 4.3.1, explains why metric *M1* remains unaltered while metric *M2* score decreases. In a scenario in which smaller thresholds are used (e.g., when using minimal thresholds), the absence of noise filtering can also lead to the accumulation of the noise surpassing thresholds, thus resulting in the identification of erroneous transformations. However, in this test, this is not verified as we are employing appropriate thresholds and none of the accumulations of noise is significant enough to surpass the thresholds.

Lastly, in the case of a service capable of perfectly quantifying spatial transformations (represented as the second test of this experiment), we conclude that noise filters are not needed since no noise is produced. Using noise filters in such cases only leads to losing detailed transformations.

6.3.7 Co-occurring transformations and non-linear animations

The objective of this experiment is to study how the *stfX* performs when given more complex and chaotic scenarios. To test the *stfX* in such a scenario, we produced a ground-truth dataset that consists of: a time-consuming translation, of 20 units in the x-axis and minus 10 units in the y-axis, from time unit 1 to time unit 190; simultaneous to the described translation, a rotation of 10^o clockwise, from time unit 110 to 120, and a rotation of 10^o counterclockwise, from time unit 150 to 160; immutability, from time unit 190 to time unit 210; a quick-paced change, from time unit 210 to time unit 225, where a translation of -30 units in the x-axis and 10 units in the y-axis and a uniform scaling of factor 0.4 co-occur; simultaneously, a translation of 5 units in the x-axis and a uniform scaling of factor 3, in the temporal range 225 to 255; and, at last, a rotation of 85^o counterclockwise, in the temporal range 255 to 275, with a uniform scaling of factor 0.95 also occurring, from time unit 260 to time unit 265. All the transformations occur smoothly (applying Blender's sinusoidal interpolation[5]). Seeing that the set of transformations is reasonably complicated, Figure 6.11 illustrates the described transformations.

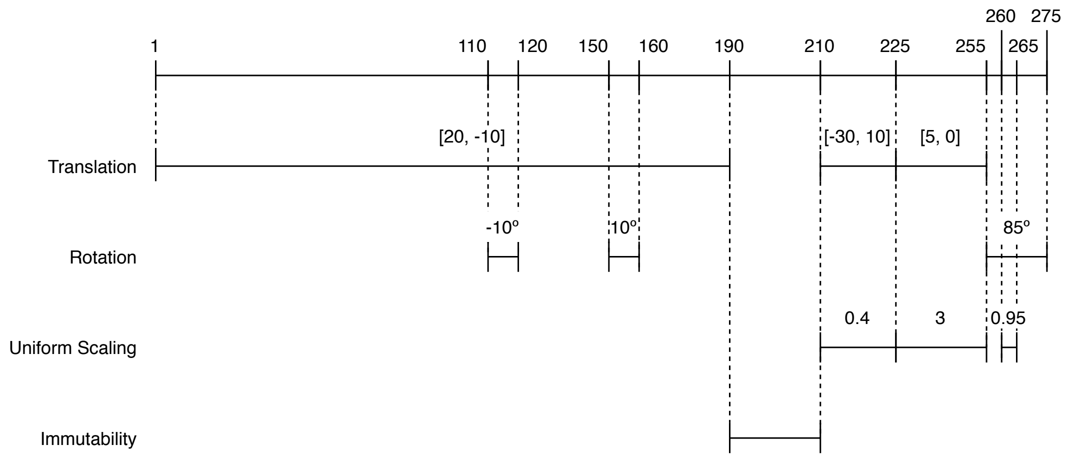


Figure 6.11: Visual representation of the transformations of the dataset used in experiment 6.3.7.

Throughout the previous experiments, we verified that the selection of thresholds has a high impact on the quality of the outputted results. Hence, we attempt to use the possible best thresholds with the intent of demonstrating the *stfX* full capabilities. Using the mock REST API, introduced in experiment 6.3.5, we determined the set of thresholds that maximised the obtained scores.

Results

Table 6.5 presents the obtained scores, for both metrics, with and without employing the mock API. The scores are not maxed out (in the test using the mock API) as noise filters are preserved seeing that the unmocked *stfX* version manages the same set of thresholds.

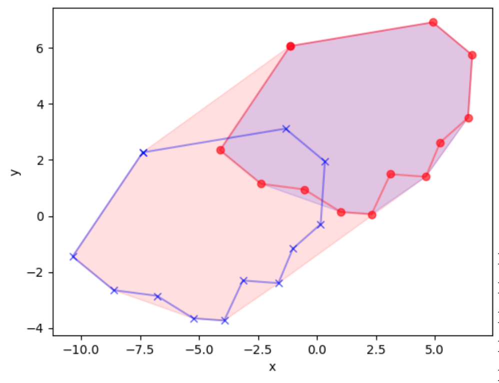
[5]Blender Manual. F-Curves Introduction. Available at https: `https://docs.blender.org/manual/en/latest/editors/graph_editor/fcurves/introduction.html#easing-by-strength`. Accessed in 2020-05-31.

Table 6.5: Scores obtained for experiment 6.3.7, using thresholds that max out the scores with the mock API.

Scores with mock API		**Original Scores**	
***M1* score**	***M2* score**	***M1* score**	***M2* score**
0.998	0.974	0.496	0.954

From Table 6.5, we verify that the most significant difference between the scores when using the mock API and the original scores subsides in metric *M1*: while in metric *M2* the scores are very similar (difference of 0.02 corresponding to a decrease of 2%), in metric *M1*, when using the mock API, the obtained score more than doubles the original score (difference of 0.458 corresponding to a decrease of 50%).

Figure 6.12 shows the visual representation for both metrics when using the original scores. From the analysis of this figure, we verify that the *stfX* identified several translations that in the ground-truth did not exist. The accumulation of these several translations is the motive behind the obtained metric *M1* score.



```
[
    {
        "temporalRange": [190, 210],
        "removals": ["TRANSLATION"]
    }, {
        "temporalRange": [255, 260],
        "removals": ["TRANSLATION"]
    }, {
        "temporalRange": [260, 265],
        "removals": ["TRANSLATION"]
    }, {
        "temporalRange": [265, 275],
        "removals": ["TRANSLATION"]
    }
]
```

Figure 6.12: Detailed representation of the metrics, for the original scores, obtained in the first set of experiment 6.3.7.

Seeing that in the obtained results the identification of translations stemming from noise led to a low score on metric *M1*, we repeated the same test, while using higher thresholds on translations (including the noise filter thresholds). To assert a measure of comparison, we also ran these same thresholds while operating the mock API. Table 6.6 presents the obtained results.

From Table 6.6, we verify that the disparity between the scores with mock API and the original scores is no longer existent (difference of 0.009, corresponding to an increase of 2%, for metric *M1* and difference of 0.005, corresponding to a decrease of 1%, for metric *M2*). Comparing these scores with the original scores of Table 6.5, we verify that the metric *M1* score increased 0.062 (increase of 12%), while metric *M2* decreased 0.154 (decrease of 16%). Nevertheless, the disparity

Table 6.6: Scores obtained for experiment 6.3.7, using thresholds that maximise the filtering of noise.

Scores with mock API		**Original Scores**	
***M1* score**	***M2* score**	***M1* score**	***M2* score**
0.567	0.805	0.558	0.800

between the scores of both metrics was decreased, while improving one of the metrics. Regarding the content of the identified transformations, the previously detected noise was no longer included. However, some translations — the less significant ones — also got filtered as noise. Hence, only the most relevant transformations are detected.

Discussion

From the results obtained in this experience, we can conclude that:

1. The usage of minimal thresholds and noise filters allows the inclusion of more detail in retrieved identified transformations (as already shown in 6.3.5). However, in scenarios in which the quantification of transformations produces noise, lower thresholds also facilitate retrieving the noise as a significant transformation (thus corroborating the obtained conclusions in experiment 6.3.6). The usage of higher thresholds — that lead to discarding detailed transformations and possible noise — makes the *stfX* only grasp the most significant transformations (core transformations), thus corroborating the conclusions reached in experiment 6.3.4;

2. The initial and final moments of a smooth transformation are often considered to be noise. The impact in the scores of its non-detection is minimal;

3. In conjunction with experiments 6.3.3, we verified that in the rare cases where the *stfX* PSR service introduces noise, it frequently affects translations;

4. The *stfX* is capable of analysing and correctly detecting the majority of the transformations in complex and chaotic scenarios. However, the inclusion (or not) of noise in the quantification of the occurring spatiotemporal transformations plays one of the most important roles in the framework's efficiency. When noise is produced, higher thresholds and noise filters should be employed.

6.3.8 Summary

Sections 6.3.1 to 6.3.7 detail the performed experiments, while also analysing the obtained results and discussing them. Experiment 6.3.1 determines the *stfX* capability of detecting singular transformations. Afterwards, experiment 6.3.2 ascertains that the *stfX* can correctly identify the transformations in a sequence of non-multiple change features (no two features of interest co-occur). Subsequently, experiment 6.3.3 we studied how small changes in the thresholds (like choosing a value slightly below the transformation of interest) impact the results obtained. Experiment 6.3.4 studied the *stfX* performance on a scenario with co-occurring transformations. The results showed that choosing minimal thresholds led to a significant improvement in the quality of the frames retrieved by the *stfX*. Next, experiment 6.3.5 identifies the quality of the results retrieved by the service responsible for quantifying spatial change features as a fundamental impact factor on the scores obtained. After, experiment 6.3.6 ascertains that the absence of noise filters decreases the *stfX* performance. Finally, experiment 6.3.7 helped corroborate many of the conclusions previously achieved while also determining that the *stfX* can correctly detect the features of interest in chaotic and complex scenarios, if the spatial features quantification service retrieves high-quality results.

6.4 Assessment

In this section, and using the performed evaluation, we assess the work developed and analyse to what extent the research objectives have been accomplished. Hence, we start by examining if the devised solution satisfies all of the defined requirements. Next, we determine if the proposed solution answers the identified research questions.

6.4.1 Solution requirements

Previously, in section 3.3, we defined the requirements that a solution needs to fulfil to address the entirety of the problems presented in section 3.1. Therefore, and with the intuit of validating our proposed solution, we analyse whether the presented requirements were fulfilled:

D1: Provide the end-user autonomy — By the usage of the API routes, validated through unit tests *UT1*, the user can load its desired dataset. The dataset needs only to be defined in JSON, and contain the attributes *dataset* and *metadata* (section 5.3). Moreover, when retrieving the features of interest, the users can also set the thresholds parameters to express their interests (validated through unit tests *UT2* and *UT5*).

D2: Choose the spatiotemporal features that are analysed — By not including the thresholds of a given transformation, the transformation is not analysed by the *stfX*. Unit tests *UT5* validate this capability.

D3: For a spatiotemporal feature, define in which conditions the feature is of interest — Achieved by the usage of thresholds. The values of the thresholds define the condition for considering a spatiotemporal change feature as interesting. Experiments 6.3.3, 6.3.4 and 6.3.7 study how the usage of different thresholds affects the retrieved results.

D4: Cover the whole temporal range queried — ensured by the *stfX* capability of identifying unimportant temporal ranges (subsection 5.2.2.5). Validated by unit tests *UT6*.

D5: Retrieve changes of interest — All the experiments performed end up validating this requirement. However, the ones with a primary focus on examining the retrieved features of interest are experiments 6.3.1, 6.3.2, 6.3.4 and 6.3.7.

D6: Retrieve the values associated with the identified features of interest — To compute metric *M1*, the values of the retrieved transformations are needed. Thus, since all experiments make usage of metric *M1* to evaluate the *stfX* performance, all experiments validate this requirement.

Concluding, the proposed solution satisfies all of the identified requirements. It also becomes relevant to highlight that not all of the unit tests and experiments are directly correlated to a requirement. For instance, experiments 6.3.5 and 6.3.6 study the impact of distinct circumstances on the *stfX*, without directly mapping to any requirement.

6.4.2 Research Questions

Through the development of the solution proposed in chapter 4, validated throughout the previous sections of this chapter, we obtained the results that support the answers for the research questions raised in section 3.2.

Contrary to the order by which they were presented in section 3.2, in this section, we start by analysing the most specific research questions first.

RQ1: How to extract the spatiotemporal features occurring in the original representation? By resorting to the solution presented in chapter 4, more specifically to section 4.2, spatiotemporal features can be extracted from the original representation by quantifying the spatiotemporal transformations occurring between successive temporal snapshots. With this intent, in the *stfX*, for quantifying the core spatial transformations, such as translations, rotations, and uniform scalings, a point set registration algorithm is used (subsection 5.2.1).

RQ2: How to find the spatiotemporal features that interest the user? Using the thresholds mechanism, described in subsection 4.2.7 with a detailed explanation of the *stfX* approach in subsection 5.2.2.2, the end-users can determine when a given transformation interests them. A transformation, or set of transformations when considering accumulated thresholds, is considered of interest if its value is superior to the threshold inputted by the end-user. With this approach, by changing the values of the thresholds, end-users are capable of reflecting their interests.

RQ3: How to assert if the two representations are equivalent? The change-based representation and the original representation are equivalent when the knowledge obtained by the analysis of both representations is also equivalent. Considering the problem domain addressed in this work, the knowledge obtained from the original representation consists of the spatiotemporal transformations that affect the phenomenon. Hence, to be able to assert if the two representations are equivalent, we propose two complementary scoring metrics that analyse the correctness of the spatiotemporal transformations identified in the change-based representation when having the original representation as ground-truth. The scoring metrics are further detailed in section 6.2.2.

RQ4: **How to automate the process?** An automated process is a process that requires minimal human assistance. Therefore, by making agnostic to the end-user the process of quantification of spatiotemporal transformations, the verification of thresholds and other processes like coalescing and noise filtering, the human interaction is reduced to the minimum. So, to automate the process, user interaction must be reduced to situations where it is indispensable, namely, to provide the necessary input.

Having answered the more specific research questions, we answer the broader research question.

RQ: **Considering a spatiotemporal phenomenon, how to automatically detect, extract and represent the changes of interest that encompass equivalent knowledge to the original representation?** The main actions necessary to generate a change-based representation that encompasses equivalent knowledge to the original representation are: (1) to extract the spatiotemporal transformations occurring between temporal snapshots and (2) to filter the extracted transformations accordingly to the inputted thresholds of interest. The performed experiments, namely experiments 6.3.3, 6.3.4, 6.3.5 and 6.3.7, proved that both the service responsible for quantifying the spatial transformations between snapshots, as well as the inputted thresholds, represent the factors that mostly impact the quality of the resulting change-based representation. Consequently, these are the factors that have the most impact on how equivalent is the change-based representation to the original representation.

6.5 Summary

In this chapter, we presented the validation methodology employed, described the performed experiments as well as the obtained results and assessed if the requirements and research questions raised in chapter 3 have been satisfied/answered.

Section 6.1 describes the unit tests performed and the component of the *stfX* that each intends to validate.

Section 6.2 presents the pipeline for generating the experiments as well as two complementary novel metrics utilised to evaluate the results obtained: metric *M1* evaluates the resemblance between the obtained phenomenon and the ground-truth final representations. In contrast, metric *M2* evaluates the correctness of the identified features of interest and respective temporal ranges.

Section 6.3 details the performed experiments, while also analysing the obtained results and discussing them. A total of seven distinct experiments were performed, with various intents: (1) validate the *stfX* capability of the detecting singular transformations; (2) ascertain that the *stfX* can correctly identify the transformations in a sequence of non-multiple change features; (3) study how small changes in the thresholds impact the results obtained; (4) examine the *stfX* performance on a scenario with co-occurring transformations; (5) determine how the quality of the results retrieved by the service responsible for quantifying spatial change impacted the results; (6) ascertain that the absence of noise filters decreases the *stfX* performance; and (7) determine if the *stfX* can correctly detect the features of interest in chaotic and complex scenarios.

Finally, section 6.4 verified that the devised solution and corresponding prototype successfully satisfy all the requirements established in section 3.3. Moreover, the research questions raised in section 3.2 have also been answered.

Chapter 7

Conclusions

This chapter presents this dissertation's conclusions. Section 7.1 describes the findings obtained at each step of the research pipeline. Next, section 7.2 describes the most significant contributions resulting from this work. Finally, section 7.3 details the identified research lines of potential value stemming from the developed work.

7.1 Conclusions

The fast development of technological devices capable of collecting spatiotemporal data leads to the necessity of spatiotemporal data mining frameworks, capable of automatically extracting information of interest to the end-user, from ever-growing data. Considering the vast domain of spatiotemporal data mining, we focus on the more specific field of change detection, where the knowledge comes from being able to detect, extract and represent changes affecting phenomena. From the literature review, we determined that the currently available frameworks were not suited to address this dissertation's main problem: the identification of the changes that are of particular interest to the end-user, occurring on spatiotemporal phenomena. Hence, with the intent of further examining the problem, we mapped it into the research question: how to automatically detect, extract and represent the changes of interest that encompass equivalent knowledge to the original representation?

Through the literature review performed, presented in chapter 2, we distinguished a plethora of spatiotemporal change features that can possibly be identified when detecting changes between successive temporal snapshots. Spatial transformations, like translations, rotations and so forth, are the more basilar change features on which many other features build upon. Moreover, for each identified spatiotemporal change feature, we also identified an adequate method for its computation. Once more, considering the case of spatial transformation detection, point set registration algorithms appeared as the most appropriate approach. To distinguish the change features of interest, thresholds of interest — limits imposed by the end-user above which features are considered as interesting — appeared as the most suitable approach. Regarding similar frameworks, we concluded that the approaches most closely addressing the problem were either too restrictive regarding the spatiotemporal data domain or only able to identify a subset of spatiotemporal changes.

Seeing that the state-of-the-art frameworks analysed are unable to solve this dissertation's problem and provide the answer to the respective research question, we proposed a conceptual framework that we consider effectively addresses these issues. This original conceptual framework focuses on the computation of different spatiotemporal change features, using methods like point set registration algorithms for spatial features, similarity measures for deformation, kinematics physics fundamentals for movement-derived features, among others. Thresholds of interest are employed to distinguish the spatiotemporal change features of interest.

With the intent of verifying if the proposed conceptual framework effectively answers the identified research question, thus presenting a viable solution for this dissertation's problem, we developed the SpatioTemporal Features eXtractor (*stfX*): a context-agnostic prototype capable of identifying spatiotemporal change features of interest. The *stfX* can detect rigid transformations and changes of direction and further filter them using thresholds of interest provided by the end-user. Moreover, the *stfX* provides some additional capabilities, namely noise filtering and coalescing of similar temporal ranges, with the intent of improving the quality of the results retrieved.

Towards assessing the *stfX* efficacy and ability to comply with the established requirements, we validated it using unit testing as well as various experiments. To the best of our knowledge, there is not a single framework with a similar nature to ours. Therefore, finding a method to evaluate the experiment's results proved a challenge. Hence, we designed two novel validation metrics, metrics *M1* and *M2*, that when employed in conjunction can provide a comprehensive analysis of the results retrieved. Each experiment was then scored using both metrics. The results obtained allowed us to conclude that the *stfX* is highly dependent on the accuracy of the point set registration algorithm chosen to compute the spatial transformations. When the algorithm provides high-accuracy estimates of the transformations occurring, the *stfX* is capable of detecting the change features correctly on all tested scenarios. Moreover, noise filtering and coalescing proved to have a significant impact on the quality of the retrieved results.

From the obtained results, we conclude that the *stfX* meets all of the established requirements as well as answers this dissertation's main research question (for the presented test scenarios, for the features that the *stfX* can compute). The main actions necessary to generate a change-based representation of interest encompassing equivalent knowledge, include first computing the spatiotemporal change features and then filtering those according to the provided thresholds of interest. The quality of the resulting change-based representation is strongly impacted by both the provided thresholds as well as the accuracy of the modules responsible for computing the spatiotemporal change features.

7.2 Contributions

With this work, we aimed at contributing to the spatiotemporal data mining domain and laying out a foundation for others to work upon. As such, the main contributions of this dissertation are the following:

- **Conceptual framework:** a solution that presents the general guidelines for implementing a system capable of retrieving all of the spatiotemporal change features identified in chapter 2, according to the provided user interests;
- **SpatioTemporal Features eXtractor (*stfX*):** an open-source implementation of a prototype, built to be easily extensible, that defines an underlying foundation for the conceptual framework proposed;
- **Metrics *M1* and *M2*:** Novel evaluation metrics that, when used in conjunction, can provide a comprehensive analysis of the accuracy of the methods employed to generate the change-based representation of interest.
- **Benchmarks:** Considering the absence of similar frameworks, the results obtained by the *stfX* can be utilised as benchmarks for future frameworks having similar objectives.

The *stfX* open-source implementation, the resources necessary to run the experiments described in chapter 6, as well as the *stfX* results are available at [12].

7.3 Future work

Seeing that this work represents a pioneering approach to the development of a framework capable of quantifying the spatiotemporal change features of interest to the end-user, there is a vast set of research streams that can be further explored or examined. Hence, possible future work encompasses:

1. **Test the *stfX* on real-world datasets:** In order to further corroborate the proposed solution and the respective prototype, it is appropriate to study the *stfX* performance in non-synthetic datasets;
2. **Extend the *stfX*:** As shown in chapter 5, the current *stfX* version does not implement all of the features of the framework proposed in chapter 4. Hence, the remaining spatiotemporal change features can be implemented using the methodologies identified in chapter 4, such as: voting schemes manipulating multiple similarity metrics for deformation analysis, combinatorial analysis for identifying spatiotemporal patterns and trends, and so forth;
3. **Support new input data types:** As shown in section 5.3, the current *stfX* version only accepts JSON as input. Therefore, other spatiotemporal data formats, like shapefile (*.shp*) or well-known text geometry (*.wkt*), could be directly included in the framework;

4. **Add support for handling multiple phenomena:** The current *stfX* version accepts a JSON describing the evolution of a single phenomenon. However, this approach limits the spatiotemporal change features that can be identified, as it does not support existential change features. By extending *stfX* with the capability to handle multiple phenomena, it becomes possible to detect this subset of spatiotemporal change features.

5. **Study noise filtering:** In the current *stfX* version, noise filtering is achieved using thresholds of interest. However, it is yet to be assessed if this is the optimal approach;

6. **Further explore point set registration algorithms and compare results:** To support the selected point set registration algorithm, we based ourselves in the study performed by Hao et al. [100]. However, considering our restriction of pairwise and parametric algorithms supporting affine registration (as shown in section 4.2.1), a more detailed study over that family of algorithms could prove beneficial;

7. **Test using 3D data:** Augment the prototype to support 3D data. Many of the *stfX* components already support the additional dimension (e.g. point set registration algorithm). However, going from 2D to 3D is not straightforward, and further study is necessary to ensure prerequisites, conditions and correct handling of 3D data;

8. **Investigate thresholds of interest representation:** All of the spatiotemporal change features being parsed by the *stfX* are mapped into a single threshold value (e.g. rotation into the rotation angle, uniform scaling into the scaling factor, etc.). However, for certain spatiotemporal change features (e.g. shear), the modulation into a single threshold value is non-trivial. We foresee describing thresholds using preponderance vectors as a viable possibility. However, this is an area needing further investigation;

9. **Examine parsing of thresholds:** Despite making sense from a conceptual point-of-view, the logic for parsing thresholds being currently employed in the *stfX* (e.g., resetting the other thresholds' accumulators upon surpassing a threshold), presented in section 5.2.2.2, needs further testing and experimentation, with the intent of determining if it is the optimal approach;

10. **Interdependent thresholds:** Allow the definition of more complex thresholds, by using interdependencies between change features (e.g. only retrieve uniform scalings that co-occur with rotations);

11. **Automatic parameter tuning:** With the inclusion of additional spatiotemporal change features, the number of parameters to be defined by the end-user will vastly increase. Hence, studying a mechanism for the automatic tuning of the thresholds of interest stands out as a promising research stream. For the automatic tuning of thresholds, metric *M1* can represent the loss function used for optimising the parameters, as it does not require any previous knowledge of the occurring change features.

References

[1] Abu Yousuf Md Abdullah, Arif Masrur, Mohammed Sarfaraz Gani Adnan, Md Baky, Abdullah Al, Quazi K Hassan, and Ashraf Dewan. Spatio-temporal patterns of land use/land cover change in the heterogeneous coastal region of bangladesh between 1990 and 2017. *Remote Sensing*, 11(7):790, 2019.

[2] Gennady Andrienko, Natalia Andrienko, Peter Bak, Daniel Keim, Slava Kisilevich, and Stefan Wrobel. A conceptual framework and taxonomy of techniques for analyzing movement. *J. Vis. Lang. Comput.*, 22:213–232, 06 2011.

[3] Natalia Andrienko, Gennady Andrienko, and Peter Gatalsky. Exploratory spatio-temporal visualization: An analytical review. *Journal of Visual Languages & Computing*, 14:503–541, 12 2003.

[4] H. Anton and C. Rorres. *Elementary Linear Algebra: Applications Version*. John Wiley & Sons, 2010.

[5] Alberto Apostolico and Concettina Guerra. The longest common subsequence problem revisited. *Algorithmica*, 2(1-4):315–336, 1987.

[6] Berkay Aydin, Vijay Akkineni, and Rafal A. Angryk. Modeling and indexing spatiotemporal trajectory data in non-relational databases. In *Managing Big Data in Cloud Computing Environments*, 2016.

[7] Paul Besl and H.D. McKay. A method for registration of 3-d shapes. ieee trans pattern anal mach intell. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 14:239–256, 03 1992.

[8] Connie Blok. Monitoring Change: Characteristics of Dynamic Geo-spatial Phenomena for Visual Exploration. In *Spatial Cognition II*, pages 16–30. Springer, 2000.

[9] Jan Bogaert, Reinhart Ceulemans, and David Salvador-Van Eysenrode. Decision Tree Algorithm for Detection of Spatial Processes in Landscape Transformation. *Environmental Management*, 33(1):62–73, jan 2004.

[10] Bart Braden. The surveyor's area formula. *The College Mathematics Journal*, 17(4):326–337, 1986.

[11] Yu. D. Burago and V. A. Zalgaller, editors. *Geometry III*, volume 48 of *Encyclopaedia of Mathematical Sciences*. Springer Berlin Heidelberg, Berlin, Heidelberg, 1992.

[12] Edgar Carneiro. stfX: SpatioTemporal Features eXtractor. `https://github.com/EdgarACarneiro/stfX`. Online; accessed 2020-07-04.

[13] E. Čech. *Point Sets*. Academic Press, 1969.

[14] Edmond Chalom, Eran Asa, and Elior Biton. Measuring image similarity: An overview of some useful applications, 2013.

[15] Gengxin Chen, Yijun Hou, and Xiaoqing Chu. Mesoscale eddies in the south china sea: Mean properties, spatiotemporal variability, and impact on thermohaline structure. *Journal of Geophysical Research: Oceans*, 116(C6), 2011.

[16] L.Paul Chew, Michael T. Goodrich, Daniel P. Huttenlocher, Klara Kedem, Jon M. Kleinberg, and Dina Kravets. Geometric pattern matching under euclidean motion. *Computational Geometry*, 7(1):113 – 124, 1997.

[17] Haili Chui and Anand Rangarajan. A new point matching algorithm for non-rigid registration. *Computer Vision and Image Understanding*, 89:114–141, 02 2003.

[18] C.J. Colbourn and E.S. Mahmoodian. *Combinatorics Advances*. Mathematics and Its Applications. Springer US, 1995.

[19] J.W. Cooper. *Java Design Patterns: A Tutorial*. Addison-Wesley, 2000.

[20] Sergi Costafreda-Aumedes, Carles Comas, and Cristina Vega-Garcia. Characterizing configurations of fire ignition points through spatiotemporal point processes. *Natural Hazards and Earth System Sciences Discussions*, 2, 03 2014.

[21] Philippe Decaudin. Geometric deformation by merging a 3d-object with a simple shape. In *Graphics Interface*, 1996.

[22] K. Dhoble, N. Nuntalid, G. Indiveri, and N. Kasabov. Online spatio-temporal pattern recognition with evolving spiking neural networks utilising address event representation, rank order, and temporal spike learning. In *The 2012 International Joint Conference on Neural Networks (IJCNN)*, pages 1–7, 2012.

[23] Somayeh Dodge, Robert Weibel, and Anna Katharina Lautenschütz. Towards a taxonomy of movement patterns. In *Information Visualization*, volume 7, pages 240–252, dec 2008.

[24] José Duarte, Paulo Dias, and José Moreira. *An Evaluation of Smoothing and Remeshing Techniques to Represent the Evolution of Real-World Phenomena: 13th International Symposium, ISVC 2018, Las Vegas, NV, USA, November 19 – 21, 2018, Proceedings*, pages 57–67. Springer, 11 2018.

[25] José Duarte, Bruno Silva, José Moreira, Paulo Dias, Enrico Miranda, and Rogério Luís Costa. Towards a qualitative analysis of interpolation methods for deformable moving regions. In *the 27th ACM SIGSPATIAL International Conference*, pages 592–595, 11 2019.

[26] Jean Duchon. Splines minimizing rotation-invariant semi-norms in sobolev spaces. In Walter Schempp and Karl Zeller, editors, *Constructive Theory of Functions of Several Variables*, pages 85–100, Berlin, Heidelberg, 1977. Springer Berlin Heidelberg.

[27] David J. Eck. *Introduction to Computer Graphics*. David J. Eck, 2016.

[28] Alon Efrat and Alon Itai. Improvements on Bottleneck Matching and Related Problems Using Geometry. In *Proceedings of the Annual Symposium on Computational Geometry*, pages 301–310, 1996.

[29] Alon Efrat, Alon Itai, and Matthew J. Katz. Geometry helps in bottleneck matching and related problems. *Algorithmica*, 31:1–28, 2001.

[30] Carola Eschenbach. Research abstract on dynamic phenomena in space and their representation. Presented at NCGIA's Varenius Project Meeting on Cognitive Models of Dynamic Phenomena and their Representations, University of Pittsburgh, PA, U.S.A, 1998. `http://www2.sis.pitt.edu/cogmap/ncgia/eschenbach.html`.

[31] Ronald Fagin and Larry Stockmeyer. Relaxing the triangle inequality in pattern matching. *International Journal of Computer Vision*, 30, 02 1999.

[32] Luca Forlizzi, Ralf Güting, Enrico Nardelli, and Markus Schneider. A data model and data structures for moving objects databases. In *SIGMOD '00: Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, volume 29, pages 319–330, 06 2000.

[33] A.I.R. Galarza and J. Seade. *Introduction to Classical Geometries*. Birkhäuser Basel, 2007.

[34] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Proceedings / CVPR, IEEE Computer Society Conference on Computer Vision and Pattern Recognition. IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 3354–3361, 05 2012.

[35] James E. Gentle. Matrix Transformations and Factorizations. In *Matrix Algebra*, chapter 5, pages 227–263. Springer International Publishing, 2 edition, 2017.

[36] F. Giannotti and D. Pedreschi. *Mobility, Data Mining and Privacy: Geographic Knowledge Discovery*. SpringerLink: Springer e-Books. Springer Berlin Heidelberg, 2008.

[37] Steven Gold, Chien-Ping Lu, Anand Rangarajan, Suguna Pappu, and Eric Mjolsness. New algorithms for 2d and 3d point matching: Pose estimation and correspondence. In *Advances in Neural Information Processing Systems 7, [NIPS Conference, Denver, Colorado, USA, 1994]*, volume 31, pages 957–964, 01 1994.

[38] Ronald L Graham and F Frances Yao. Finding the convex hull of a simple polygon. *Journal of Algorithms*, 4(4):324–331, 1983.

[39] Ning Guo, Mengyu Ma, Wei Xiong, Luo Chen, and Ning Jing. An efficient query algorithm for trajectory similarity based on fréchet distance threshold. *ISPRS International Journal of Geo-Information*, 6:326, 10 2017.

[40] Alison C. Hale, Fernando Sánchez-Vizcaíno, Barry Rowlingson, Alan D. Radford, Emanuele Giorgi, Sarah J. O'Brien, and Peter J. Diggle. A real-time spatio-temporal syndromic surveillance system with application to small companion animals. *Scientific Reports*, 9(1):17738, Nov 2019.

[41] Jiawei Han, Micheline Kamber, and Jian Pei. *Data Mining Trends and Research Frontiers*, pages 585–631. Elsevier, 12 2012.

[42] Kathleen Hornsby and Max J. Egenhofer. Identity-based change: a foundation for spatio-temporal knowledge representation. *International Journal of Geographical Information Science*, 14(3):207–224, 2000.

[43] Paul Jaccard. Etude de la distribution florale dans une portion des alpes et du jura. *Bulletin de la Societe Vaudoise des Sciences Naturelles*, 37:547–579, 01 1901.

[44] P. Jamshidi, C. Pahl, N. C. Mendonça, J. Lewis, and S. Tilkov. Microservices: The journey so far and challenges ahead. *IEEE Software*, 35(3):24–35, 2018.

[45] Bing Jian and Baba Vemuri. Robust point set registration using gaussian mixture models. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 33:1633 – 1645, 09 2011.

[46] Ioannis Karatzas and Steven E. Shreve. *Brownian Motion*, pages 47–127. Springer New York, New York, NY, 1998.

[47] Ismail Khalid Kazmi, Lihua You, and Jian Jun Zhang. A survey of 2D and 3D shape descriptors. In *Proceedings - 10th International Conference Computer Graphics, Imaging, and Visualization, CGIV 2013*, pages 1–10. IEEE Computer Society, 2013.

[48] Fatima Khalique, Shoab Ahmed Khan, Wasi Haider Butt, and Irum Matloob. An integrated approach for spatio-temporal cholera disease hotspot relation mining for public health management in punjab, pakistan. *International Journal of Environmental Research and Public Health*, 17(11), 2020.

[49] Glenn E Krasner, Stephen T Pope, et al. A description of the model-view-controller user interface paradigm in the smalltalk-80 system. *Journal of object oriented programming*, 1(3):26–49, 1988.

[50] Kenneth Kunen. *Set theory an introduction to independence proofs*. Elsevier, 2014.

[51] Vladimir Levenshtein. Binary codes capable of correcting spurious insertions and deletion of ones. *Problems of information Transmission*, 1(1):8–17, 1965.

[52] S. Lipschutz and M. Lipson. *Schaum's Outline of Linear Algebra Fourth Edition*. Schaum's Outline Series. McGraw-Hill Education, 2008.

[53] Alex Lohfink, Tom Carnduff, Nathan Thomas, and Mark Ware. An object-oriented approach to the representation of spatiotemporal geographic features. In *GIS: Proceedings of the ACM International Symposium on Advances in Geographic Information Systems*, pages 268–275, 2007.

[54] Steve Lohr. The age of big data. *New York Times*, 11(2012), 2012.

[55] S.L. Loney. *The Elements of Coordinate Geometry*. Number pt. 1 in The Elements of Coordinate Geometry. Macmillan, 1964.

[56] Jiayi Ma, Ji Zhao, and Alan Yuille. Non-rigid point set registration by preserving global and local structures. *IEEE transactions on image processing : a publication of the IEEE Signal Processing Society*, 25, 08 2015.

[57] Daniel Marques. Spatiotemporal phenomena summarization through static visual narratives. Unpublished Master Dissertation, 2020.

[58] Mathworks. Matrix representation of geometric transformations. https://www.mathworks.com/help/images/matrix-representation-of-geometric-transformations.html. Online; accessed 2020-03-07.

[59] Mathworks. What are state-space models? https://www.mathworks.com/help/ident/ug/what-are-state-space-models.html. Online; accessed 2020-03-07.

[60] Facundo Mémoli. Gromov–wasserstein distances and the metric approach to object matching. *Foundations of computational mathematics*, 11(4):417–487, 2011.

[61] Lichao Mou, Lorenzo Bruzzone, and Xiao Xiang Zhu. Learning spectral-spatial-temporal features via a recurrent convolutional neural network for change detection in multispectral imagery. *IEEE Transactions on Geoscience and Remote Sensing*, 57(2):924–935, 2018.

[62] G. Muszynski, K. Kashinath, V. Kurlin, M. Wehner, and Prabhat. Topological data analysis and machine learning for recognizing atmospheric river patterns in large climate datasets. *Geoscientific Model Development*, 12(2):613–628, 2019.

[63] Andriy Myronenko and Xubo Song. Point set registration: Coherent point drift. *IEEE transactions on pattern analysis and machine intelligence*, 32:2262–75, 12 2010.

[64] Ram Narayanan. *A Shape-Based Approach to Change Detection and Information Mining in Remote Sensing*, page 63–86. World Scientific, 01 2003.

[65] Kamal Nasreddine, Abdesslam Benzinou, and Ronan Fablet. Variational shape matching for shape classification and retrieval. *Pattern Recognition Letters*, 31(12):1650–1657, sep 2010.

[66] E.M. National Academies of Sciences, D.E.P. Sciences, S.S. Board, and C.D.S.E.S.A. Space. *Thriving on Our Changing Planet: A Decadal Strategy for Earth Observation from Space*. National Academies Press, 2019.

[67] Saul B Needleman and Christian D Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of molecular biology*, 48(3):443–453, 1970.

[68] Encyclopedia of Mathematics. Bijection. http://www.encyclopediaofmath.org/index.php?title=Bijection&oldid=30987. Online; accessed 2020-03-07.

[69] Encyclopedia of Mathematics. Continuous function. http://www.encyclopediaofmath.org/index.php?title=Continuous_function&oldid=15327. Online; accessed 2020-03-07.

[70] Encyclopedia of Mathematics. Diffeomorphism. http://www.encyclopediaofmath.org/index.php?title=Diffeomorphism&oldid=24410. Online; accessed 2020-03-07.

[71] Encyclopedia of Mathematics. Homeomorphism. http://www.encyclopediaofmath.org/index.php?title=Homeomorphism&oldid=33549. Online; accessed 2020-03-07.

[72] Gregory Pederson, Daniel Fagre, Todd Kipfer, and Clint Muhlfeld. A century of climate and ecosystem change in western montana: What do temperature trends portend? *Climatic Change*, 98:133–154, 08 2010.

[73] Leonardo Romero and Felix Calderon. *A Tutorial on Parametric Image Registration*, chapter 10, pages 167–184. IntechOpen, 06 2007.

[74] Kenneth Rose, Eitan Gurewitz, and Geoffrey Fox. A deterministic annealing approach to clustering. *Pattern Recognition Letters*, 11:589–594, 09 1990.

[75] Stefan Schliebs, Nuttapod Nuntalid, and Nikola Kasabov. Towards spatio-temporal pattern recognition using evolving spiking neural networks. In Kok Wai Wong, B. Sumudu U. Mendis, and Abdesselam Bouzerdoum, editors, *Neural Information Processing. Theory and Algorithms*, pages 163–170, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[76] M. Schomaker, C.R. Rao, H. Toutenburg, and C. Heumann. *Linear Models and Generalizations: Least Squares and Alternatives*. Springer Series in Statistics. Springer Berlin Heidelberg, 2007.

[77] Thomas Sederberg and Scott Parry. Free-form deformation of solid geometric models. In *SIGGRAPH '86: Proceedings of the 13th annual conference on Computer graphics and interactive techniques*, volume 20, pages 151–160, 08 1986.

[78] Shashi Shekhar, Zhe Jiang, Reem Ali, Emre Eftelioglu, Xun Tang, Venkata Gunturi, and Xun Zhou. Spatiotemporal Data Mining: A Computational Perspective. *ISPRS International Journal of Geo-Information*, 4(4):2306–2338, oct 2015.

[79] Wenzhong Shi, Min Zhang, Rui Zhang, Shanxiong Chen, and Zhao Zhan. Change detection based on artificial intelligence: State-of-the-art and challenges. *Remote Sensing*, 12(10), 2020.

[80] Karan Singh and Eugene Fiume. Wires: a geometric deformation technique. In *SIGGRAPH '98*, 1998.

[81] D.E. Smith. *History of Modern Mathematics*. Mathematical monographs. J. Wiley & Sons, 1906.

[82] M.R. Spiegel and S. Lipschutz. *Schaum's Outline of Vector Analysis, 2ed*. Schaum's Outline Series. McGraw-Hill Education, 2009.

[83] Zhengyu Su, Yalin Wang, Rui Shi, Wei Zeng, Jian Sun, Feng Luo, and Xianfeng Gu. Optimal mass transport for shape matching and comparison. *IEEE transactions on pattern analysis and machine intelligence*, 37(11):2246–2259, 2015.

[84] Christian Tominski, Petra Schulze-Wollgast, and Heidrun Schumann. 3D information visualization for time dependent data on maps. In *Proceedings of the International Conference on Information Visualisation*, volume 2005, pages 175–181, 2005.

[85] Yanghai Tsin and Takeo Kanade. A correlation-based approach to robust point set registration. In *Computer Vision - ECCV 2004, 8th European Conference on Computer Vision, Prague, Czech Republic, May 11-14, 2004. Proceedings, Part III*, volume 3, pages 558–569, 01 2004.

[86] Alexey A Tuzhilin. Who invented the gromov-hausdorff distance? *arXiv preprint arXiv:1612.00728*, 2016.

[87] Francesca Uccheddu, Michaela Servi, Rocco Furferi, and Lapo Governi. Comparison of mesh simplification tools in a 3d watermarking framework. In *Intelligent Interactive Multimedia Systems and Services 2017*, pages 60–69, 05 2018.

[88] John P Van de Geer. *Some aspects of Minkowski distance*. Leiden University, Department of Data Theory, 1995.

[89] Remco C. Veltkamp. Shape matching: Similarity measures and algorithms. In *Proceedings - International Conference on Shape Modeling and Applications, SMI 2001*, pages 188–197, 2001.

[90] Jaime E. Villate. *Dinâmica e Sistemas Dinâmicos*. Jaime E. Villate, first edition, 2015.

[91] Gang Wang, Qiangqiang Zhou, and Yufei Chen. Robust non-rigid point set registration using spatially constrained gaussian fields. *IEEE Transactions on Image Processing*, PP:1–1, 01 2017.

[92] Jason Tsong-Li Wang, Gung-Wei Chirn, Thomas G. Marr, Bruce Shapiro, Dennis Shasha, and Kaizhong Zhang. Combinatorial pattern discovery for scientific data: Some preliminary results. *SIGMOD Rec.*, 23(2):115–125, May 1994.

[93] Lingjing Wang, Xiang Li, Jianchun Chen, and Yi Fang. Coherent point drift networks: Unsupervised learning of non-rigid point set registration, 2019.

[94] Yue Wang and Justin M Solomon. Deep closest point: Learning representations for point cloud registration. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3523–3532, 2019.

[95] Leland Wilkinson. *The Grammar of Graphics*. Springer, second edition, 1999.

[96] Haim J Wolfson and Isidore Rigoutsos. Geometric hashing: An overview. *IEEE computational science and engineering*, 4(4):10–21, 1997.

[97] Michael Worboys. Event-oriented approaches to geographic phenomena. *International Journal of Geographical Information Science*, 19(1):1–28, jan 2005.

[98] Jiawei Yi, Yunyan Du, Fuyuan Liang, Chenghu Zhou, Di Wu, and Yang Mo. A representation framework for studying spatiotemporal changes and interactions of dynamic geographic phenomena. *International Journal of Geographical Information Science*, 28(5):1010–1027, 2014.

[99] May Yuan. Temporal GIS and Spatio-temporal Modeling. In *Proceedings of the Third International Conference Workshop on Integrating GIS and Environment Modeling*, January 1996.

[100] Hao Zhu, Bin Guo, Ke Zou, Yongfu Li, Ka-Veng Yuen, Lyudmila Mihaylova, and Henry Leung. A review of point set registration: From pairwise registration to groupwise registration. *Sensors*, 19:1191, 03 2019.