



**HAL**  
open science

## Discovering motifs restricted in space-time

Heraldo Borges

► **To cite this version:**

Heraldo Borges. Discovering motifs restricted in space-time. Computer Science [cs]. CEFET-RJ, Rio de Janeiro, Brazil, 2021. English. NNT: . tel-04301030v2

**HAL Id: tel-04301030**

**<https://hal-lirmm.ccsd.cnrs.fr/tel-04301030v2>**

Submitted on 22 Nov 2023

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.



Distributed under a Creative Commons Attribution 4.0 International License



## DISCOVERING MOTIFS RESTRICTED IN SPACE-TIME

Heraldo Pimenta Borges Filho

Thesis submitted to the Postgraduate Program in Production Engineering and Systems of the Federal Center for Technological Education of Rio de Janeiro, CEFET/RJ, as partial fulfillment of the requirements for the degree of doctor.

Advisor: Eduardo Soares Ogasawara  
Co-advisor: Esther Pacitti

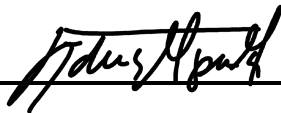
Rio de Janeiro,  
June, 2021

## DISCOVERING MOTIFS RESTRICTED IN SPACE-TIME

Thesis submitted to the Postgraduate Program in Production Engineering and Systems of the Federal Center for Technological Education of Rio de Janeiro, CEFET/RJ, as partial fulfillment of the requirements for the degree of doctor.

Heraldo Pimenta Borges Filho

Examining jury:



---

President, Eduardo Soares Ogasawara, D.Sc. (CEFET/RJ) (Advisor)



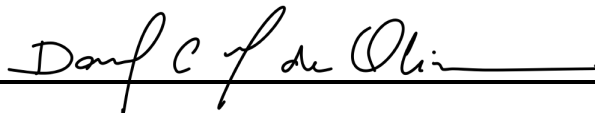
---

Esther Pacitti, D.Sc. (INRIA & LIRMM) (Co-advisor)



---

Fábio André Machado Porto, D.Sc. (LNCC)



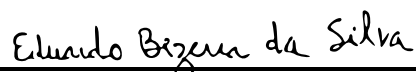
---

Daniel Cardoso Moraes de Oliveira, D.Sc. (UFF)



---

Leonardo Silva de Lima, D.Sc. (UFPR)



---

Eduardo Bezerra da Silva, D.Sc. (CEFET/RJ)

Rio de Janeiro,

June, 2021

Ficha catalográfica elaborada pela Biblioteca Central do CEFET/RJ

B732 Borges Filho, Heraldo Pimenta  
Discovering motifs restricted in space-time / Heraldo Pimenta Borges  
Filho. — 2022.  
66f. : il. color. , enc.

Tese (Doutorado) Centro Federal de Educação Tecnológica Celso  
Suckow da Fonseca, 2022.

Bibliografia : f. 61-66

Orientador: Eduardo Soares Ogasawara

Coorientadora: Esther Pacitti

1. Mineração de dados. 2. Espaço e tempo. 3. Sistemas de  
reconhecimentos de padrões. 4. Análise de séries temporais. I.  
Ogasawara, Eduardo Soares. (Orient.). II. Pacitti, Esther (Coorient.). III.  
Título.

CDD 006.312

## DEDICATION

Ao meu pai (*in memoriam*), pelo amor e exemplo.

Foi Ela quem tudo fez.

***São João Bosco***

## ACKNOWLEDGMENTS

À Deus, por tudo.

Aos meus pais e minha querida irmã, pelo exemplo, amor e incondicional apoio, mesmo com minha ausência na vida familiar. À minha esposa, Jéssica, por seu amor, paciência, compreensão e amizade. Ao meu filho Tomás, minha luz e inspiração.

Ao meu orientador, Professor Eduardo Ogasawara pela paciência e grande apoio para a realização deste trabalho. Pelos ensinamentos, incentivo e entusiasmo contagiante. À minha co-orientadora, Professora Esther Pacitti, por todo apoio, pelos ensinamentos e conhecimentos compartilhados. Pela grande ajuda em me receber na Universidade de Montpellier.

Aux Professeurs Patrick Valduriez, Florent Maseglia et Reza Akbarnia pour leur accueil à l'Université de Montpellier, pour leurs encouragements et échanges précieux. A mes coéquipiers de l'équipe *Zénith* du laboratoire *LIRMM*<sup>a</sup> et de l'institut *INRIA*<sup>b</sup> pour leurs connaissances partagées et leur amitié.

Aos amigos e professores da DIPPG<sup>c</sup> do CEFET/RJ, em especial aos professores Pedro Manoel e Luis Felipe Souza, pelo companheirismo, aprendizado e auxílio.

Aos amigos que colaboraram durante essa jornada, em especial Jéssica de Assis pela amizade, apoio e conversas.

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001

---

<sup>a</sup><https://team.inria.fr/zenith/>

<sup>b</sup><https://www.inria.fr>

<sup>c</sup><http://dippg.cefet-rj.br/>

# RESUMO

## Descoberta de Motifs Restritos no Espaço-Tempo

Muitos fenômenos podem ser observados e organizados como uma sequência de observações em uma linha do tempo que pode ser modelada como uma série temporal. Uma área relevante que está sendo explorada na análise de séries temporais é a de descoberta de padrões. Padrões são subsequências de séries temporais relacionadas a algumas propriedades ou comportamentos especiais. Um padrão particular que ocorre um número significativo de vezes em séries temporais é denominado *motif*. Vários fenômenos importantes de uma série temporal apresentam comportamentos diferentes quando observados em pontos do espaço (por exemplo, séries coletadas por sensores e IoT) e são melhor modelados como séries espaço-temporais, em que cada série temporal é associada a uma posição no espaço. Um padrão espaço-temporal é uma sequência de eventos que são limitados no espaço e no tempo. Encontrar padrões que são frequentes e restritos em um espaço e tempo pode nos permitir compreender como um fenômeno ocorre.

Diversos trabalhos foram desenvolvidos com o objetivo de identificar *motifs* em séries temporais, entretanto não foram identificados trabalhos que abordassem técnicas para dados espaço-temporais. Nesta tese comparamos diferentes abordagens de identificação de *motifs* em séries temporais com suas principais diferenças. Propomos dois métodos para a identificação automática de *motifs* restritos no espaço-tempo em séries espaço-temporais, o *CSA* e o *CSTMP*. Comparamos experimentalmente os métodos propostos com dois outros métodos alternativos: a técnica *Matrix Profile* e a combinação da técnica *Matrix Profile* e *DBScan*. Nossos resultados mostram que *CSA* e o *CSTMP* são inovadores e obtêm resultados superiores as técnicas estado da arte para séries temporais e suas adaptações para séries espaço-temporais, sendo avaliados em dois datasets.

Palavras-chave: Séries Espaço-Temporais; Motifs; Sequências Restritas

# ABSTRACT

## Discovering Motifs Restricted in Space-Time

Many phenomena can be observed and organized as a sequence of observations on a timeline that can be modeled as a time series. A relevant area that is being explored in time series analysis is pattern discovery. Patterns are subsequences of time series related to some special properties or behaviors. A particular pattern that occurs a significant number of times in time series is called *motif*. Several important phenomena of a time series present different behaviors when observed at points in space (for example, series collected by sensors and IoT) and are best modeled as space-time series. Each time series is associated with a position in space. A space-time pattern is a sequence of events that are limited in space and time. Finding patterns that are frequent and restricted in space and time can allow us to understand how a phenomenon occurs.

Several works have been developed to identify *motifs* in time series. However, studies that address spatiotemporal data techniques have not been identified. In this thesis, we compare different approaches to identifying *motifs* in time series with their main differences. We propose two methods for automatically identifying space-time restricted *motifs* in space-time series, the *CSA* and *CSTMP*. We experimentally compare the proposed methods with two other alternative methods: the *Matrix Profile* technique and the ensemble of the *Matrix Profile* and *DBScan* techniques. Our results show that *CSA* and *CSTMP* are innovative and obtain results that outperform the state-of-the-art techniques for time series and their adaptations for space-time series, being evaluated in two datasets.

Keywords: Spatial-Time Series; Motifs; Restricted Sequences



## LIST OF FIGURES

Figure 1	– Seismic Interpretation	15
Figure 2	– Time series, sub-sequences, and sliding windows	18
Figure 3	– A synthetic spatial-time dataset	31
Figure 4	– Synthetic dataset with identified motifs	42
Figure 5	– Seismic dataset with the mapped horizons	43
Figure 6	– $MP_1$ on synthetic dataset	44
Figure 7	– $MP_2$ on synthetic dataset	45
Figure 8	– $Baseline_1$	46
Figure 9	– $Baseline_2$	46
Figure 10	– $CSA_1$ - $CSA$ on synthetic dataset - ( $sb \approx tb$ )	47
Figure 11	– $CSA_1$ - $CSA$ on synthetic dataset - ( $sb < tb$ )	47
Figure 12	– $CSA_1$ - $CSA$ on synthetic dataset - ( $sb > tb$ )	48
Figure 13	– $CSTMP_1$ Synthetic	48
Figure 14	– $Baseline_1(eps = 4; minPts = 10)$ - Seismic Dataset	49
Figure 15	– $Baseline_1$ on seismic dataset	50
Figure 16	– $Baseline_1$ on seismic dataset	50
Figure 17	– $Baseline_2(eps = 8; minPts = 5)$ - Seismic Dataset	51
Figure 18	– $Baseline_2(eps = 24; minPts = 15)$ - Seismic Dataset	51
Figure 19	– $Baseline_2(eps = 48; minPts = 50)$ - Seismic Dataset	51
Figure 20	– $MSE$ for each alphabet size	52
Figure 21	– Top-ten discovered motifs	56
Figure 22	– Top discovered motifs according to the number of occurrences.	58
Figure 23	– Top discovered motifs according to spatial extension, filtering those that cover a greater number of spatial-time series.	58

## LIST OF TABLES

Table 1 –	Evaluated Approaches	44
Table 2 –	Overall performance of <i>CSA</i> under different block orientation	47
Table 3 –	<i>Baseline</i> Input Parameters	49
Table 4 –	<i>CSA</i> Input Parameters	52
Table 5 –	Overall performance of <i>CSA</i> under different block orientation	53
Table 6 –	Summary of Discovered Spatial-Time Motifs for different block orientation and word size	54
Table 7 –	Influence of $\sigma$ and $\kappa$ in the number of occurrences in Square (20x20) setup with word size $w = 4$	55
Table 8 –	Top five distinct motifs	55
Table 9 –	<i>CSTMP</i> Input Parameters	56
Table 10 –	Overall performance of <i>CSTMP</i> under different radius size	57
Table 11 –	Summary of Discovered Spatial-Time Motifs for different radius and word size	57
Table 12 –	Influence of $\gamma$ in the number of motifs and occurrences with word size $w = 4$	58

## LIST OF ALGORITHMS

Algorithm 1 –	STAMP ( $T_A, T_B, m$ )	22
Algorithm 2 –	STOMP ( $T, m$ )	24
Algorithm 3 –	AAMP Algorithm	25
Algorithm 4 –	Combined Series Approach	32
Algorithm 5 –	CSTMP	38
Algorithm 6 –	BlockSearch	39
Algorithm 7 –	STAAMP	40

# CONTENTS

<b>1</b>	<b>Introduction</b>	<b>14</b>
<b>2</b>	<b>State of the art</b>	<b>17</b>
2.1	Background	17
2.2	Motif discovery	19
2.3	Matrix Profile	21
2.3.1	STAMP	22
2.3.2	STOMP	23
2.3.3	AAMP	24
2.4	Related work	26
<b>3</b>	<b>Problem Statement</b>	<b>29</b>
<b>4</b>	<b>Combined Series Approach (<i>CSA</i>)</b>	<b>30</b>
4.1	General Principle	31
4.2	Normalization & SAX indexing	33
4.3	Spatial-time Motif Discovery	33
4.4	Rank motifs	34
<b>5</b>	<b>Constrained Spatio-Temporal Matrix Profile (<i>CSTMP</i>)</b>	<b>36</b>
5.1	General Principle	37
5.1.1	Block Search	37
<b>6</b>	<b>Experimental Evaluation</b>	<b>41</b>
6.1	Datasets	41
6.2	Analysis on the synthetic dataset	43
6.3	Analysis on the seismic dataset	49
<b>7</b>	<b>Conclusion</b>	<b>59</b>



## 1- Introduction

Under the data deluge scenario, data scientists are pushed to provide new ways for efficiently collecting, storing, processing, and organizing a large amount of data [Tsai et al., 2015]. We are immersed in a scenario with massively growing datasets from many sources, types, and formats. The analysis of this data can reveal useful information and significant insights about the domain. Such scenario opens a set of research opportunities involving knowledge discovery [Han et al., 2011; Shumway and Stoffer, 2017]. In this context, many phenomena can be observed and organized as a sequence of observations in a timeline that can be modeled as a time series, enabling discoveries.

A relevant area that is being explored in time series analysis is finding patterns [Patel et al., 2002]. Patterns are sub-sequences of time series that are related to some special properties or behaviors [Han et al., 2007]. Earlier efforts on frequent patterns discovery focused on finding association rules on transactional databases. In this context, a transaction is a set of items and we want to extract frequent subsets of items, also called itemsets, according to a given threshold. The problem of pattern mining successively evolved to augmented versions, according to data characteristics and analytics objectives, including sequential pattern mining where the temporal dimension of records is considered. It concerns, for instance, successive transactions belonging to the same customer [Han et al., 2007].

The discovery of pattern enables the understanding of some specific behaviors observed in time series, in many areas of knowledge, such as weather prediction [McGovern et al., 2011], wind generation [Fan and Kamath, 2015], image recognition [Chi et al., 2012], seismic amplitude [Cassisi et al., 2013]. A vast number of patterns discovery techniques, methods, and algorithms have been developed [Mueen, 2014; Serrà and Arcos, 2016; Torkamani and Lohweg, 2017; Yeh et al., 2018]. They include discovering patterns of a particular/variable length [Li and Lin, 2010; Tang and Liao, 2008], or without constraints (parameter-free algorithms) [Nunthanid et al., 2012], or in multivariate time series [Minnen et al., 2007; Yeh et al., 2017a].

However, various important time-series phenomena present different behaviors when observed at points of space (for example, series collected by sensors and IoT

devices) and are better modeled as spatial-time series, in which each time series is associated to a position in space [Fu, 2011]. This scenario motivates the definition of new techniques for pattern mining, dedicated to spatio-temporal datasets [Fu, 2011; Mooney and Roddick, 2013]. Under such scenarios, some patterns might not be discovered when we restrict the analysis to the time-only dimension. The challenge becomes to identify regions of space and time where the patterns are frequently observed. Finding patterns that are frequent in a constrained space and time, *i.e.*, finding spatial-time patterns, may enable us to comprehend how a phenomenon occurs.

Consider as an example the spatial-time series *dataset* at the seismic surveys. In some applications, there is a need for analyzing the subsoil structure of a large terrain area to discover specific interesting information like specific boundaries between geologic layers [Zhou, 2014]. For that purpose, a set of receivers are placed in specific fixed positions of a large terrain area and each of them captures the values reflected by the different materials of the subsoil. In this way, each receiver indicates the position of a time series where the time is related to the subsoil depth in a form that the earlier values are more superficial. This operation collects a huge volume of three-dimensional spatial-time data that needs to be analyzed to find specific interesting patterns. In this domain, finding a frequent pattern of textures in the entire *dataset* is of little interest. However, some patterns (Figure 1) occurring in a constrained depth interval and a specific space location may lead to highly valuable lithology change indicator such as *horizons*, indicated by the numbers 1, 3 and 4 (which stand for kind of layers) or *bright spots*, indicated by the number 2 (potential areas of hydrocarbon accumulation) [Zhou, 2014].

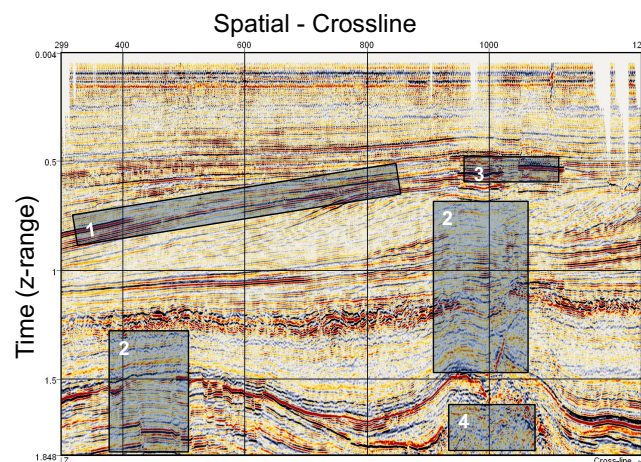


Figure 1 – Seismic Interpretation (adapted from Borges et al. [2020] )

This work addresses such problem by presenting approaches to discover patterns that are frequent on a constrained time and space and may not be frequent in the entire *dataset*. Defining the input data as a collection of spatio-temporal series, where each time series is a collection of observations and each observation is a continuous (non-discrete) value, we aim to find all patterns that are frequent in a constrained space and time. To tackle this problem, we propose in this work two evolving approaches.

At first, we propose an approach to discover and rank motifs in spatial-time series, denominated Combined Series Approach (*CSA*). *CSA* is based on partitioning the spatial-time series into blocks, combining subsequences of spatial-time series. Motifs are validated according to both temporal and spatial constraints and ranked according to their entropy, the number of occurrences, and the proximity of their occurrences.

At the second approach, we introduce the *Constrained Spatio-Temporal Matrix Profile (CSTMP)* algorithm. *CSTMP* it is an adaptation of the *Matrix Profile* approach Yeh et al. [2017b] in order to discover motifs that are restricted in space-time in spatio-temporal data that span two spatial dimensions. In *CSTMP*, the distance between the neighboring space-time series' sub-sequences is calculated iteratively by identifying and grouping the motifs that are directly reachable by density with each other.

We evaluated our approaches on a synthetic and on the seismic *dataset*. They were able to identify spatial-time patterns that could not be identified using the traditional approach. Besides, they were able to prioritize patterns that were meaningful according to their number of occurrences and quality. Our approaches outperform traditional methods designed only for time series, managing to find sets of motifs with different configurations as demonstrated in the synthetic dataset experiments. Moreover, the identified patterns on the seismic *dataset* correspond to candidate areas for seismic *horizons* and *bright spots* that are of high value for domain experts.

In addition to this introduction, this work is organized as follows. Chapter 2 presents the background for time series data mining and the Related Works from Motif Discovery, including the main concepts, evidencing the differences with the solution proposed in this work. Chapter 3 formalizes the problem definition. Chapter 4 presents the *CSA*, our first approach to discovery spatial-time motifs. In Chapter 5 we deeper our analysis and present *CSTMP*, which is an evolved approach based on *Matrix Profile*. Chapter 6 describes and explains the experiments using *CSA* and the *CSTMP*. Finally, chapter 7 presents the conclusion and future work.



## 2- State of the art

In this chapter, we introduce the general concepts about time series, spatial-time series, sub-sequences, sliding windows, normalization, and data indexing. Concepts related to motif discovery. In addition, state of the art motif discovery techniques are explored.

### 2.1- Background

A **time series**  $t$  is an ordered sequence of values in time:  $t = \langle t_1, t_2, \dots, t_m \rangle$ ,  $t_i \in \mathbb{R}$ , where each  $t_i$  is a value,  $|t| = m$  is the number of elements in  $t$ , and  $t_m$  is the most recent value in  $t$  [Shumway and Stoffer, 2017].

A **subsequence** is a continuous sample of a time series with a defined length. The  $p^{\text{th}}$  **subsequence** of size  $n$  in a time series  $t$ , represented as  $seq_{n,p}(t)$ , is an ordered sequence of values  $\langle t_p, t_{p+1}, \dots, t_{p+n-1} \rangle$ , where  $|seq_{n,p}(t)| = n$  and  $1 \leq p \leq |t| - n$ . Subsequences enable the analysis of data samples to evaluate some local properties [Chiu et al., 2003].

**Sliding windows** consist in exploring all possible subsequences of a time series [Keogh and Lin, 2005; Lampert et al., 2008]. Sliding windows produce a set of subsequences of the same length. A **sliding windows** of size  $n$  for a time series  $t$  is a function  $sw_n(t)$  that produces a matrix  $W$  of size  $(|t| - n + 1)$  by  $n$ . Each line  $w_i$  in  $W$  is the  $i$ -th subsequence of size  $n$  from  $t$ . Given  $W = sw_n(t)$ ,  $\forall w_i \in W$ ,  $w_i = seq_{n,i}(t)$ . This concept is widely used in time series analysis to make comparisons between subsequences to find their similarities [Van Hoan and Exbrayat, 2013].

Figure 2 depicts the application of the above definitions to a time series. The blue line represents a time series  $t$ , the red line represents a subsequence from the time series, and the green dashed lines are examples of some of the subsequences extracted from time series based on sliding windows.

A **spatial-time series**  $st$  can be described as a pair  $(t, p)$ , such as a time series  $t$

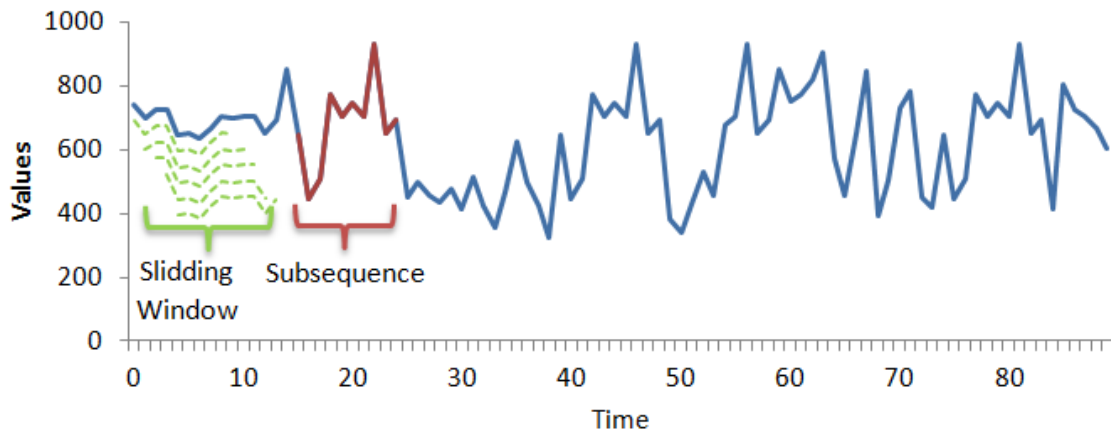


Figure 2 – An example of a time series, sub-sequences, and sliding windows [Borges et al., 2020]

with an associated position  $p$  [Shekhar et al., 2016]. The position can be its geographical coordinates or any other reference that can represent the place where data has been observed. If the position is a function of time, it is a trajectory spatial-time series. Otherwise, it is a permanent spatial-time series. In this work, we are interested in permanent spatial-time series and for the sake of simplicity, from now on, we are calling them spatial-time series.

Data preprocessing techniques are key activities for enabling or improving the quality of data mining. In time series context, data is usually a continuous numerical value. For pattern discovery, some methods do not process them efficiently [Daw et al., 2003]. Due to that, during pattern discovery, two data preprocessing techniques are commonly applied in sequence: (i) data normalization and (ii) Symbolic Aggregate Approximation (SAX).

Normalization is commonly used to enable the effectiveness of time series comparison methods. One of the most common normalization methods is z-score [Keogh and Ratanamahatana, 2005]. As a result of this normalization method, the normalized time series has *zero* as average and *one* as standard deviation. Equation 1 (a) describes z-score normalization, where  $t_i$  is an observation of the time series  $t$ ,  $\mu_t$  is the average,  $\sigma_t$  is the standard deviation of the time series, and  $t'$  is the transformed time series. Additionally, the min-max is another normalization method that applies a linear transformation to the original data, where the minimum value ( $\min(t)$ ) and the maximum value ( $\max(t)$ ) are used to transform each value  $t_i$  to another value  $t'_i$  in a range varying from  $[0, 1]$ , as shown in Equation 1(b) [Ogasawara et al., 2009].

$$\begin{aligned}
 t'_i &= \frac{(t_i - \mu_t)}{\sigma_t} & t'_i &= \frac{t_i - \min(t)}{\max(t) - \min(t)} \\
 (a) & & (b) &
 \end{aligned}
 \tag{1}$$

SAX is an indexing technique. It consists primarily in partitioning the domain of a variable into ranges such that each range is associated with a particular symbol [Lin et al., 2007]. The SAX alphabet size defines the number of partitions for the domain. Thus, all values are replaced by their respective associated symbol. Given an alphabet  $(a_1, \dots, a_n)$  of size  $n$ , the values of time series  $t$  are divided into  $n$  ranges  $([-\infty, \beta_1], \dots, [\beta_{n-1}, \infty])$  according to Gaussian function (with different sizes, but same probability), such that each value  $t_i$  is mapped to an alphabet value  $a_k$ , where  $1 \leq k \leq n$  [Lin et al., 2003].

## 2.2- Motif discovery

Given a sequence  $q$  and time series  $t$ ,  $q$  is a **motif** in  $t$  with support  $\sigma$ , if and only if  $q$  is included in  $t$  at least  $\sigma$  times. The length of a motif  $q$  ( $|q|$ ) is also known as word size. Formally, given a sequence  $q$  and a time series  $t$  where  $W = sw_{|q|}(t)$ ,  $motif(q, t, \sigma) \leftrightarrow \exists R \subseteq W, (|R| \geq \sigma)$ , such that  $\forall w_i \in R, w_i = q$  [Mueen, 2014]. An important property regarding motifs is that the repeated subsequence is not previously known and is discovered when scanning the entire data. It can be discovered by making a comparison between subsequences that are obtained using sliding windows [Chiu et al., 2003].

Many methods proposed in the literature to discover motifs in time series are computationally intensive [Patel et al., 2002]. Due to that, many works aim to improve the effectiveness of motif discovery and reduce the computational resources needed. Such a process requires some data preprocessing such as normalization and indexing before running the motif discovery algorithms to increase the performance and precision of results [Mueen, 2014].

There are some main approaches to discover motifs, such as (i) brute force; (ii) heuristic-based; and (iii) matrix profile. The brute force approach is the simplest method. It has a high computational cost, especially when used to discover sequences of greater size in large datasets [Mueen et al., 2009]. It is indicated for discovering sequences of smaller

size [Li and Nallela, 2009]. In this method, the coverage and accuracy are complete since it makes all possible comparisons between all subsequences of a time series. For the sake of our work, we can group time series motif discovery approaches according to the exactness (exact or approximate), length (fixed or variable), and dimension (single or multiple).

The brute force approach is the simplest method, but it has a high computational cost, specially when used for large dataset [Mueen et al., 2009]. It is indicated for low dimensional data [Li and Nallela, 2009]. In this method the coverage and accuracy is complete since it makes all possible comparisons between all subsequences of a time series.

The random projections approach was proposed to handle large dataset reducing dimensionality taking samples of data aleatory [Li and Nallela, 2009]. It optimizes the execution time and reduces the computational consumption in identifying motifs [Buhler and Tompa, 2002; Chiu et al., 2003]. In random projection, the search for similar subsequences is speed-up using hash-based methods for matching certain random elements of the sequence. It includes SAX indexing and sliding windows. Each symbolic subsequence is inserted in a subsequences matrix. Each line index corresponds to the initial position of subsequence in time axis. The next step is to build the collision matrix which is used to indicated potential motifs in the time series. The collision matrix is initially null and has the same quantity of lines and columns that correspond to the total number of subsequences identified. The collision matrix is built from the random projection process where two columns of the subsequence matrix are randomly selected and mask and for each position in mapped as a hash structure that has as input the symbolic values that correspond to position of selected columns. If two subsequences has the same symbolic value in the mask position then it is placed in hash structure.

The matrix profile is based on computing the distance of a sequence of size  $n$  with the most similar subsequence present in the time series. It is called matrix profile, since the naive implementation of this technique is to compute all pairwise distance for all sequences present in the time series. However, it can use efficient algorithms, such as Fast Fourier Transform (FFT) to enable very fast computation [Yeh et al., 2018]. Matrix profile has two manifold advantages. It can handle numeric data directly and uses a matrix for support comparisons of subsequences which can be used to other tasks rather than motif discovery [Yeh et al., 2017b,a, 2018; De Paepe et al., 2020]. Due to its important, a

entire section is dedicated to it.

### 2.3- Matrix Profile

Yeh et al. [2017a] published a novel technique to perform all-pairs-similarity-search on two time-series, producing two new series: the *Matrix Profile* and the *Matrix Profile Index*. The *Matrix Profile* is defined as the vector containing the z-normalized Euclidean distances between each subsequence from the first series and its closest matching subsequence from the second time series. The Matrix Profile Index contains the location of the closest matching subsequence in the second series for each subsequence. By itself, the Matrix Profile can be used for anomaly detection in contexts where anomalies are defined by unique behavior [De Paepe et al., 2020].

In general, given two series of  $n$  real values,  $\mathbf{S1} \in \mathbb{R}^n$  and  $\mathbf{S2} \in \mathbb{R}^n$  and a subsequence length  $m$ , the Matrix Profile  $\mathbf{M} \in \mathbb{R}^{n-m+1}$  and a Matrix Profile Index  $\mathbf{I} \in \mathbb{R}^{n-m+1}$  are new series such that for each  $i \in [0, n - m]$ ,  $I_i$  contains the index of the start of the subsequence of  $S2$  length  $m$  that best matches  $S1_{i,m}$  and  $M_i$  contains the corresponding distance. In the case a *self-join* is performed where  $S1 = S2$ , an additional constraint is added to prevent *trivial matches*, where subsequences match themselves or nearby subsequences, called *exclusion zone*.

The default distance measure used is the z-normalized Euclidean distance, which removes the effect of a changing data offset over time and thus focuses more on shape instead of amplitude. Typical causes of a changing offset are wandering baselines in sensors or natural phenomena (e.g., the gradual change in temperature throughout seasons) [De Paepe et al., 2020].

We present three important algorithms for calculating the Matrix Profile, which are described in Sections 2.3.1, 5.1.1, and 2.3.3.

### 2.3.1 STAMP

The Matrix Profile was originally published together with the STAMP (Scalable Time Series Anytime Matrix Profile) [Yeh et al., 2017b], an anytime algorithm to calculate the Matrix Profile over a time series and the corresponding Index. Internally, STAMP uses a similarity search algorithm called MASS [Mueen et al., 2017] that under z-normalized Euclidean iteratively calculates the distance of each subsequence to every subsequence by using the Fast Fourier Transform (FFT).

The STAMP is outlined in Algorithm 1. In line 2, the length of  $T_B$  is extracted. In line 3, the matrix profile  $P_{AB}$  and matrix profile index  $I_{AB}$  are initialized. From lines 4 to line 6, the distance profiles  $D$  are calculated, using each subsequence  $B[idx]$  in the time series  $T_B$  and  $T_A$ . The pairwise minimum for each element in  $D$  is performed with the paired element in  $P_{AB}$  (i.e.,  $\min(D[i], P_{AB}[i])$  for  $i = 0$  to  $\text{length}(D) - 1$ ). Then, as the minimum pair operations are performed,  $I_{AB}[i]$  is updated with  $idx$  (when  $D[i] \leq P_{AB}[i]$ ). Finally, the result  $P_{AB}$  and  $I_{AB}$  are returned in line 7. In this format, STAMP computes the MP for the general similarity join. It is possible to change the algorithm to compute the self-similarity join MP of a time series  $T_A$ , just by replacing  $T_B$  in line 2 with  $T_A$ , replace B with A in line 5, and ignore trivial matches in  $D$  when performing *ElementWiseMin* in line 6.

The overall complexity of the algorithm is  $O(n^2 \log n)$  where  $n$  is the length of the time series. Since all subsequences are compared using the MASS algorithm, the  $n \log n$  factor comes from the FFT subroutine.

---

#### Algorithm 1 – STAMP ( $T_A, T_B, m$ )

---

**Input:** Two time series,  $T_A$  and  $T_B$   
Interested subsequence length  $m$   
**Output:** A matrix profile  $P_{AB}$  and associated matrix profile index  $I_{AB}$  of  $T_A$  join  $T_B$

```

1 begin
2    $n_B \leftarrow \text{Length}(T_B)$ 
3    $P_{AB} \leftarrow \text{infs}, I_{AB} \leftarrow \text{zeros}, \text{idxes} \leftarrow 1 : n_B - m + 1$ 
4   for each  $idx$  in  $\text{idxes}$  do
5      $D \leftarrow \text{MASS}(B[idx], T_A)$ 
6      $P_{AB}, I_{AB} \leftarrow \text{ElementWiseMin}(P_{AB}, I_{AB}, D, idx)$ 
7   return  $P_{AB}, I_{AB}$ 

```

---

### 2.3.2 STOMP

STOMP is similar to STAMP [Keogh and Kasetty, 2003] in that it can be seen as highly optimized nested loop searches, with the repeated calculation of distance profiles as the inner loop. However, while STAMP must evaluate the distance profiles in random order (to allow its anytime behavior), STOMP performs an *ordered* search. It is by exploiting the locality of these searches that STOMP can reduce the time complexity by a factor of  $O(n \log n)$ . STOMP uses the z-normalized Euclidean distance  $d_{i,j}$ , as shown below, of two time series subsequences  $T_{i,m}$  and  $T_{j,m}$  using their dot product,  $QT_{i,j}$ :

$$d_{i,j} = \sqrt{2m \left( 1 - \frac{QT_{i,j} - m\mu_i\mu_j}{m\sigma_i\sigma_j} \right)} \quad (2)$$

Here  $m$  is the subsequence length,  $\mu_i$  is the mean of  $T_{i,m}$ ,  $\mu_j$  is the mean of  $T_{j,m}$ ,  $\sigma_i$  is the standard deviation of  $T_{i,m}$ , and  $\sigma_j$  is the standard deviation of  $T_{j,m}$ . Note that  $QT_{i,j}$  can be decomposed as:

$$QT_{i,j} = \sum_{k=0}^{m-1} T_{i+k}T_{j+k} \quad (3)$$

The time required to compute  $d_{i,j}$  depends only on the time required to compute  $QT_{i,j}$ . To solve this problem, STOMP pre-computes and stores the means and standard deviation in  $O(n)$  space and time, thus, it takes  $O(1)$  to compute  $d_{i,j}$  [Yeh et al., 2017b].

The pseudo-code of STOMP algorithm is shown in Algorithm 2. It begins in line 2 by computing the matrix profile length  $l$ . In line 3, it calculates the mean and standard deviation of every subsequence in  $T$ . Line 4 calculates the first dot product vector  $QT$ . Line 5 initializes the matrix profile  $P$  and matrix profile index  $I$ . The loop in lines 6-13 calculates the distance profile of every subsequence of  $T$  in sequential order, with lines 7-9 updating  $QT$  according to (3). Then update  $QT[1]$  in line 10 is done with the pre-computed  $QT_{first}$  in line 3. Line 11 calculates distance profile  $D$  according to (2). Finally, line 12 compares every element of  $P$  with  $D$ : if  $D[j] < P[j]$ , then  $P[j] = D[j]$ ,  $I[j] = i$ .

The time complexity of STOMP is  $O(n^2)$ . Thus, it can achieve a  $O(n \log n)$  factor speedup over STAMP [Keogh and Kasetty, 2003]. The  $O(n \log n)$  speedup clearly makes little difference for small datasets, however, when considering the datasets with millions of data points, this  $O(n \log n)$  factor begins to produce a significant performance gain.

---

Algorithm 2 – STOMP ( $T, m$ )

---

**Input:** A time series  $T$  and a subsequence length  $m$   
**Output:** Matrix profile  $P$  and the associated matrix profile index  $I$  of  $T$

```

1 begin
2    $n \leftarrow \text{Length}(T), l \leftarrow n - m + 1$ 
3    $\mu, \sigma \leftarrow \text{ComputeMeanStd}(T, m)$ 
4    $QT \leftarrow \text{SlidingDotProduct}(T[1 : m], T), QT_{first} \leftarrow QT$ 
5    $D \leftarrow \text{CalculateDistanceProfile}(QT, \mu, \sigma)$ 
6    $P \leftarrow D, I \leftarrow \text{ones}$  // initialization
7   for  $i=2$  to  $l$  do
8     for  $j=l$  downto  $2$  do
9        $QT[j] \leftarrow QT[j-1] - T[j-1] \times T[i-1] + T[j+m-1] \times T[i+m-1]$ 
10       $QT[1] \leftarrow QT_{first}[i]$ 
11       $D \leftarrow \text{CalculateDistanceProfile}(QT, \mu, \sigma, i)$ 
12       $P, I \leftarrow \text{ElementWiseMin}(P, I, D, i)$ 
13  return  $P, I$ 

```

---

### 2.3.3 AAMP

In many applications it is preferred to not normalize the time-series data because the anomalies can be detected based on the point values, and not the shapes. AAMP [Akbarinia and Cloez, 2019] has been designed for such applications. It is an efficient algorithm for computing matrix profile with the pure (non-normalized) Euclidean distance. AAMP is executed in a set of iterations, such that in each iteration the distance of subsequences is computed incrementally. The time complexity of AAMP is  $O(n \times (n - m))$  with small constants, where  $n$  is the time series length and  $m$  the subsequence length. The experiments reported in [Akbarinia and Cloez, 2019] show that the performance of AAMP is significantly better than that of STAMP and SCRIMP++ (an improved version of STOMP).

$$D_{i,j} = \sqrt{D_{i-1,j-1}^2 - (t_{i-1} - t_{j-1})^2 + (t_{i+m-1} - t_{j+m-1})^2} \quad (4)$$

The main idea behind AAMP is that for computing the distance between subsequences it uses *diagonal sliding windows*, such that in each sliding window, the Euclidean distance is incrementally computed only between the subsequences that have a precise difference in their *start position*. Let  $T_i = \langle t_i, t_{i+1}, \dots, t_{i+m-1} \rangle$  and  $T_j = \langle t_j, t_{j+1}, \dots, t_{j+m-1} \rangle$  be two subsequences. The sliding windows in AAMP allow to use Equation (4) for incre-



mental computation of the distance between subsequences  $T_i$  and  $T_j$  (denoted by  $D_{i,j}$ ) by using the yet computed distance between subsequences  $T_{i-1}$  and  $T_{j-1}$  (denoted as  $D_{i-1,j-1}$ ).

---

Algorithm 3 – AAMP Algorithm

---

**Input:**  $T$ : time series;  $n$ : length of time series;  $m$ : subsequence length  
**Output:**  $P$ : Matrix profile;

```

1 begin
2   for  $i=1$  to  $n$  do
3      $P[i] = \infty$ ;
4   for  $k=1$  to  $n-m-1$  do
5      $dist = EucDistance(T_{1,m}, T_{k,m})$  if  $dist \leq P[1]$  then
6        $P[1] = dist$ ;
7     if  $dist \leq P[k]$  then
8        $P[k] = dist$ ;
9     for  $i=2$  to  $n - m + 1 - k$  do
10       $dist = \sqrt{(dist^2 - (t_{i-1} - t_{i-1+k})^2 + (t_{i+m-1} - t_{i+m+k-1})^2)}$  if  $dist \leq P[i]$  then
11         $P[i] = dist$ ;
12      if  $dist \leq P[i+k]$  then
13         $P[i+k] = dist$ ;

```

---

Algorithm 3 shows the pseudo-code of AAMP. Initially, the algorithm sets all values of the matrix profile to infinity (*i.e.*, maximum distance). Then, it performs  $n - m - 1$  iterations using a variable  $k$  ( $1 \leq k \leq n - m - 1$ ). In each iteration  $k$ , the algorithm compares each subsequence  $T_{i,m}$  with the subsequence that is  $k$  positions far from it, *i.e.*,  $T_{i,m+k}$ . To do this, AAMP firstly computes the Euclidean distance of the first subsequence of the time series, *i.e.*,  $T_{1,m}$ , with the one that starts at position  $k$ , *i.e.*,  $T_{k,m}$ . This first distance computation is done using the normal formula of Euclidean distance. Then, in a sliding window, the algorithm incrementally computes the distance of other subsequences with the subsequences that are  $k$  position far from them, and this is done by using Equation (4) in  $O(1)$ . If the computed distance is smaller than the previous minimum distance that is kept in the matrix profile  $P$ , then the smaller distance is saved in the matrix profile.

## 2.4- Related work

After motif discovering process, an important task in motif analysis is how to sort the motifs according to their relevance [Castro and Azevedo, 2012]. A standard classification method is  $k$ -motif which considers the total number of occurrences of the motifs in time series. Also, motifs can be sorted according to their relevance degree. For instance, some motifs can be similar to a straight line (*i.e.*, constant observations) and depending on the data domain may not be relevant. Such motifs can be low qualified or discarded to avoid distorting the analysis [Chiu et al., 2003].

Some approaches to evaluate the significance and relevance of motifs were proposed in the literature. A statistical approach to assess the relevance of motifs is based on information gain which measures how expected is the motif to occurs [Castro and Azevedo, 2012]. The Log-odds considers the degree of how rare the motif is by comparing the amount of occurrence with the expected chance of having occurrence based on probabilistic distribution [Yang et al., 2004]. Castro and Azevedo [2012] proposed the estimation of expectation for the frequency of a motif based on Markov Chain models. The value is assessed making the comparison between actual frequency and estimated based on hypothesis tests.

Considering the exact motif discovery approach, some specific method to address the dimensionality and motif length problem were proposed. Jiang et al. [2008] proposed an efficient motif discovery algorithm PMDGS (P-Motif Discovery based on Grid Structure) that processes data streams. Mueen et al. [2009] proposed a motif discovery algorithm for exact time series called MK (Mueen-Keogh) and observed that MK was faster than brute-force [Chiu et al., 2003]. Narang and Bhattacharjee [2010] introduced the Par-MK, Par-MK-SLB, and Par-MK-DLB, which are all parallel multi-threaded algorithms for exact motif discovery that focus on load balancing. Mueen et al. [2011] proposed a disk-aware algorithm to discover exact motifs in large time series databases. Cassisi et al. [2013] applied a motif discovery technique for an exact time series to study recurrent patterns in seismic amplitude time series of the Etna 2011 periodic eruptive activity. Chi and Wang [2013] introduced a method based on cloud model theory to extract the top  $k$ -motifs. Truong and Anh [2015] proposed a fast method for motif discovery in time series based on Dynamic Time Warping distance.

When it comes to approximate motif discovery, the approaches aim to reduce the complexity and consequently the computational cost. Some work proposed approaches to improve the accuracy and efficiency of Random Projection Algorithm as proposed in Chiu et al. [2003]. Lin et al. [2007] created a new symbolic representation for time series (SAX) for indexing. Mohammad and Nishida [2009] proposed two algorithms called MCFull and MCInc that address constrained motif discovery problem. Castro and Azevedo [2010] addressed motif discovery problem as an approximate top  $k$ -frequent subsequence discovery problem. Lin et al. [2010] presented an approach that uses subseries joins to get similarity among subseries of the time series. Armstrong and Drewniak [2011] developed the algorithm MD-RP for unsupervised motif discovering in time series. Narang and Bhattacharjee [2011] designed the new sequential and parallel Motif discovery and data deduplication algorithms based on bloom filters.

Finally, Regarding variable-length motif discovery, Wilson et al. [2008] proposed the Motif Tracking Algorithm (MTA) that uses a small number of parameters based on the implementation of the Bell immune memory theory. Yankov et al. [2007] presented a novel algorithm that discovers motifs in time series with invariance to uniform scaling. It enables to reduce parameters such as motif length. Nunthanid et al. [2011] described the VLMD motif discovery algorithm that does not require the motif length parameter. Such an algorithm automatically returns motif lengths from all possible sliding window lengths reducing a set of possibilities of the sliding window lengths. Nunthanid et al. [2012] presented the  $k$ -Best Motif Discovery (kBMD) algorithm that is parameter-free. It produces a set of the best motif that is ranked by a scoring function based on similarity of motif locations and shapes. Mueen [2013] proposed the MOEN, an exact free-parameter algorithm to enumerate motifs that is faster than brute-force approach due to a novel bound on the similarity function that uses only linear space. Mohammad and Nishida [2014] proposed an extension of the MK algorithm called MK++ to handle multiple motifs of variable lengths considering maximum overlap of subsequences.

Finally, when it comes to space modeling, Du et al. [2009] modeled space by discrete attributes that resemble states of an object. In the context of their paper, they refer to the state of companies in the stock market. It is, in fact, a state-space model [Shumway and Stoffer, 2017] where a trajectory is the registration of state transitions. In this way, it differs significantly from our work, since such a phenomenon may not be constrained in space and time.

The approaches presented in the literature review propose to solve the problem of discovering patterns only in time series. We did not find studies that propose to solve the identification of patterns in space-time series. To the best of our knowledge, this is the first solution to address the problem of identifying frequent sequences constrained in space and time.

### 3- Problem Statement

The patterns discovery approaches presented in the literature review propose to solve the problem of discovering patterns on time series. In the context of spatial-time series, we observe a more complex scenario due to spatial constraints. Considering a *dataset* containing spatial-time series, each spatial-time series  $ST_s$  has a position in space. If we apply to this scenario a traditional known pattern discovery method on each spatial-time series, the patterns found will be in the same spatial-time series only. Also, even when some patterns are discovered, considering the entire *dataset*, those are not fully explored including subsequences appearing in neighboring spatial-time series are not discovered as patterns.

Depending on the *dataset*, such similar subsequences in neighboring time series can correspond to some relevant information. Discovering and grouping patterns in spatial-time series datasets can address some real-world problems. Such scenario was not studied in previous works as discussed in Section 2.4. The problem formalization for this new scenario is presented as follows.

A **spatial-time series dataset** (for short, dataset)  $S$  is a set of spatial-time series  $st$ . We are interested in finding patterns that occur in a constrained space and time. In our work, sequences may only be frequent inside spatial-time parallelogram (for short, block). A **block**  $b$  is a couple  $(\{st\}, i)$  where  $\{st\}$  is a subset of neighboring spatial-time series and  $i$  is a time interval. The size of a block  $b$  is the product of the number of spatial-time series by the interval length:  $|b| = |\{st\}| \cdot |i|$ .

Let  $B$  be a partition of  $S$  into blocks  $b$ . Let  $\sigma$  and  $\kappa$  be two support values such that  $\sigma \geq \kappa$ . A subsequence  $q$  is a **spatial-time pattern** if and only if there exists a block  $b$  such that  $q$  is included at least  $\sigma$  times in it and  $q$  occurs in at least  $\kappa$  different spatial-time series inside  $b$ .

From the definition above, the problem can be summarized as *the discovery of patterns in restricted spatial-time series dataset*. To address the problem, we propose two approaches that are discussed in the following chapters.

## 4- Combined Series Approach (*CSA*)

The motif discovery approaches presented in the literature review propose to solve the problem of discovering motifs on time series. In the context of spatial-time series, we observe a more complex scenario due to spatial constraints. In this chapter, we present the *CSA* approach to discovery space-time motifs.

In order to highlight this challenging problem, Figure 3a shows a synthetic dataset, containing twelve spatial-time series ( $ST1 \dots ST12$ ) where each spatial-time series  $ST_i$  has a position in space. Figure 3b presents a flat representation of the this dataset. The ordering of time-series obeys their spatial placement. For example,  $ST2$  is both close to  $ST1$  and  $ST3$ .

If we apply to this scenario a known motif discovery method on each spatial-time series for a support  $\sigma \geq 2$ , we can only observe the motifs which are marked as green worm-like shape found only in  $ST3$  as shown in Figure 3. Also, even when some motifs are discovered, considering the entire dataset, those motifs are not fully explored: there are many other equivalent worm-like shapes that are not discovered, although appearing in close spatial-time series ( $ST2, ST4$ ). It is also possible to observe that similar subsequences appearing in neighboring spatial-time series are not discovered as motifs. They are depicted in Figure 3 as orange trapezium-like and red stripe-like motifs.

Depending on the dataset, such similar subsequences in neighboring time series can correspond to some relevant information. Discovering and grouping motifs in spatial-time series datasets can address some real-world problems. Such scenario was not studied in previous works as discussed in Section 2.4.

We are interested in finding motifs that occur in a constrained space and time. In our work, sequences may only be frequent inside spatial-time blocks, where we find neighboring time series. A **spatial-time series dataset** (for short, dataset)  $S$  is a set of spatial-time series  $st$ . A **block**  $b$  is a couple  $(\{st\}, i)$  where  $\{st\}$  is a subset of neighboring spatial-time series and  $i$  is a time interval. The size of a block  $b$  is the product of the number of spatial-time series by the interval length:  $|b| = |\{st\}| \cdot |i|$ .

Let  $B$  be a partition of  $S$  into blocks  $b$ . Let  $\sigma$  and  $\kappa$  be two support values such that  $\sigma \geq \kappa$ . A subsequence  $q$  is a **spatial-time motif** if and only if there exists a block  $b$  such

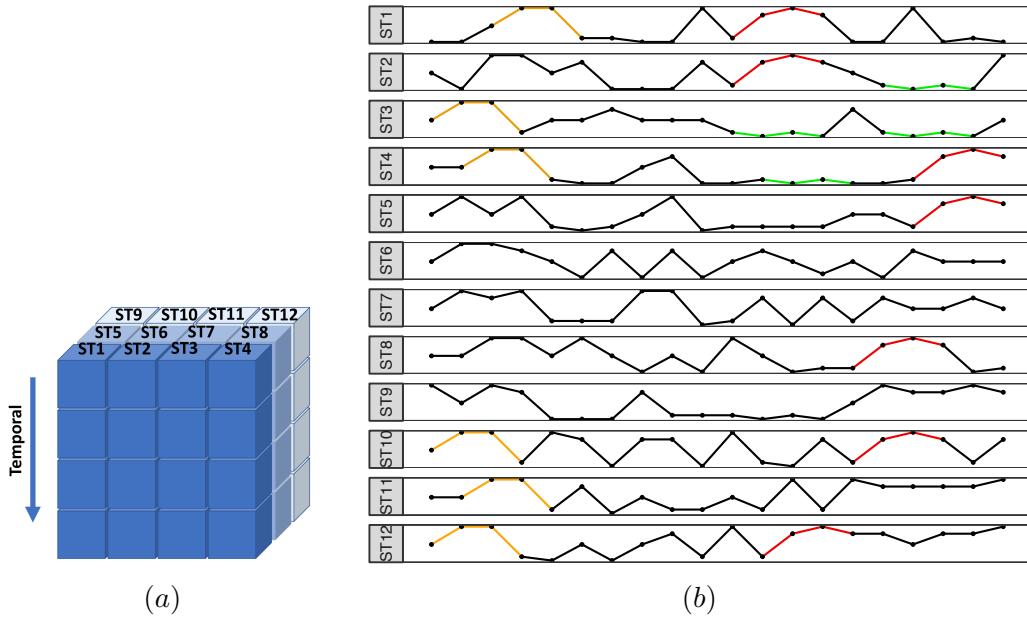


Figure 3 – (b) A synthetic dataset with twelve spatial-time series:  $ST1 \dots ST12$ . Traditional motif discovery algorithm applied in this spatial-time series dataset finds only the two green worm-like in  $ST3$  as motifs.

that  $q$  is included at least  $\sigma$  times in it and  $q$  occurs in at least  $\kappa$  different spatial-time series inside  $b$ .

From the definition above, the problem can be summarized as *the discovery of spatial-time motifs in a spatial-time series dataset*. To address the defined problem, we developed the Combined Series Approach (*CSA*).

#### 4.1- General Principle

The Combined Series Approach (*CSA*) is organized in three main steps: (i) normalization & SAX indexing; (ii) discovery of spatial-time motifs; (iii) ranking of spatial-time motifs. *CSA* is summarized in Algorithm 4. It takes as input a spatial-time series dataset  $S$ , a word size  $w$ , an alphabet size  $a$ ,  $sb_x$ ,  $sb_y$  and  $tb$  corresponding to spatial block sizes, temporal block sizes, and  $\sigma$  and  $\kappa$  constraints.

---

Algorithm 4 – Combined Series Approach

---

```

1 function CSA(S, w, a, sbx, sby, tb,  $\sigma$ ,  $\kappa$ )
2   DS  $\leftarrow$  normSAX(S, a)
3   stmotifs  $\leftarrow$  discoverSTMotifs(DS, w, sb, tb,  $\sigma$ ,  $\kappa$ )
4   rstmotifs  $\leftarrow$  rankSTMotifs(stmotifs)
5   return rstmotifs

6 function normSAX(S, a)
7   Sz  $\leftarrow$  zscore(S)
8   DS  $\leftarrow$  SAX(Sz, a)
9   return DS

10 function discoverSTMotifs(S, w, sbx, sby, tb,  $\sigma$ ,  $\kappa$ )
11   B  $\leftarrow$  partition(S, sbx, sby, tb)
12   stmotifs  $\leftarrow$   $\emptyset$ 
13   for each bi,j  $\in$  B do
14     cs  $\leftarrow$  combine(bi,j,k)
15     motifs  $\leftarrow$  discover(cs, w,  $\sigma$ )
16     stmotifs  $\leftarrow$  validate(motifs,  $\sigma$ ,  $\kappa$ )  $\cup$  stmotifs
17   return stmotifs

18 function rankSTMotifs(stmotifs)
19   stmotifs  $\leftarrow$  group(stmotifs)
20   for each mi  $\in$  stmotifs do
21      $ent_i = \sum_{k=1}^{|ft(m_i)|} \left( \frac{ft(m_i)_k}{n} \cdot \log_2 \left( \frac{ft(m_i)_k}{n} \right) \right)$ 
22     Oi  $\leftarrow$  occurrences(mi)
23     occi  $\leftarrow$   $\log_2(O_i)$ 
24      $prox_i \leftarrow \frac{1}{aw(mst(wam(O_i)))}$ 
25   rank = proj(norm(ent, occ, dist))
26   return order(stmotifs, rank)

```

---



## 4.2- Normalization & SAX indexing

The first step of the *CSA*, described by the *normSAX* function of Algorithm 4, applies z-score data normalization in the entire dataset. Right after normalization, SAX indexing method is applied for a given alphabet  $a$ . It transforms the numeric data from  $S$  into letters according to the data distribution, as described in Section 2.1. The output of such transformation is  $DS$ .

## 4.3- Spatial-time Motif Discovery

The second step of *CSA* corresponds to *discoverSTMotifs* function. In line 11, the indexed spatial-time series dataset  $S$  is partitioned into blocks ( $B$ ). These blocks are created based on spatial block size ( $sb_x, sb_y$ ) and temporal block size ( $tb$ ). The spatial block size ( $sb_x, sb_y$ ) delimit the number of neighboring spatial-time series inside each block. The  $tb$  specifies the time interval for subsequences of spatial-time series.  $B$  is the partition of  $S$  into a set of blocks  $\{b_{i,j,k}\}$ . Formally, each block  $b_{i,j,k}$  contains  $sb_x \cdot sb_y \cdot tb$  observations,  $\forall i \in [1, \frac{|st.t|}{tb}]$ ,  $\forall j \in [1, \frac{|S|}{sb_x}]$ ,  $\forall k \in [1, \frac{|S|}{sb_y}]$ . Each block  $b_{i,j,k}$  contains  $sb_x \cdot sb_y$  subsequences  $q_m$ , such that  $q_m = seq_{tb, (i-1) \cdot (tb)+1} (S_{(j-1) \cdot sb_x + m \cdot t})$ ,  $\forall m \in [1, sb]$ .

In line 14, all sequences inside a block are combined into a single time series  $cs$ , such that  $cs$  is the concatenation of sequences inside the block  $b_{i,j,k}$ . Formally,  $cs = q_1 || \dots || q_m$  and  $|cs| = \sum_{i=1}^m |q_i| = sb_x \cdot sb_y \cdot tb$ .

The *discover* function (line 15) checks all subsequences of size  $w$  in  $cs$ . It applies an adapted algorithm [Mueen, 2013] for time series identification of motifs of length  $w$ . It starts applying a hash function for registering the positions of each individual subsequence  $s_i$ . If the number of occurrences of  $s_i$  is greater than  $\sigma$ ,  $s_i$  is a temporal motif and included at *motifs*. Then, in line 16, *motifs* are validated according to both  $\sigma$  and  $\kappa$ . It checks the number of distinct spatial-time series for them is greater or equal to  $\kappa$ . It is worth mentioning that any motif whose sequence appears distributed between neighboring subsequences of a block (for example,  $q_k$  and  $q_{k+1}$ ) are fake occurrences and not considered during  $\sigma$  and  $\kappa$  validation. Motifs that validates both  $\sigma$  and  $\kappa$  are added at *stmotifs*.

#### 4.4- Rank motifs

Since the number of motifs can be high, especially when working with larger alphabets, it is important to establish ways to rank them in a way that more “interesting” ones are presented first. The third step of *CSA*, described by the *rankSTMotifs* function of Algorithm 4, makes a balance among three criteria: (i) the number of occurrences (significant higher occurrences are better); (ii) proximity (occurrences that are close to one another are better than the ones that are sparsely distributed in the dataset); (iii) entropy (higher entropy contains more information, which makes it more interesting).

Each motif corresponds to a sequence of SAX observations. All motifs that are discovered inside *discoverSTMotifs* are local block motifs. At *group* function (line 2), occurrences of motifs sharing the same sequence are grouped as long as they occur in neighboring blocks. Then, for each group of motifs  $m_i$ , metrics for ranking them are computed. In line 21, the entropy of a motif  $m_i$  of size  $n$  is computed ( $ent_i$ ). The  $ent_i$  is based on information theory and uses the frequency table ( $fm$ ) for the characters presented in a motif [Shannon, 1948] and is described in line 21. The higher is the entropy, the higher is the information that motif  $m_i$  encodes.

At line 22 the set of occurrences  $O_i$  for the motif  $m_i$  is obtained. Then, at line 23, the impact of the number of occurrences ( $O_i$ ) of the  $i$ -th motif in a logarithm scale is computed ( $occ_i$ ). This enables that only a significantly different number of occurrences becomes apparent.

In line 24, the weight of the occurrences according to their proximity is computed. Consider the pairs of position and time for the set of occurrences  $O_i$  of a motif  $m_i$  discovered in neighboring blocks. The distance between all these pairs is represented as a weighted adjacent matrix ( $wam$ ). Then, the minimum spanning tree ( $mst$ ) is built from the  $wam$ . Finally, the average weight ( $aw$ ) for the edges of the  $mst$  is computed. Thus,  $prox_i$  establishes the reciprocal measure for  $aw$  for the motif occurrences. The closer this measure is to 1, the closer are the occurrences in establishing a spatial-temporal pattern.

Once the entropy ( $ent_i$ ), the amount of occurrences ( $occ_i$ ), and the proximity ( $prox_i$ ) has been computed for each motif, the ranking procedure can be applied. During ranking, each of these dimensions are normalized using min-max and projected into the unit vector that combines these three dimensions. Such projection provides a balance among these

dimensions. The closer the projection of a motif  $m_i$  is to  $(1, 1, 1)$ , the better ranked it is. Function *rankSTMotifs* returns the *stmotifs* ordered according to the computed rank.

## 5- Constrained Spatio-Temporal Matrix Profile (*CSTMP*)

The traditional methods of identifying motifs (chapter 2) are restricted to the analysis of time series, not identifying spatially neighboring motifs when applied to space-time series. In the previous chapter, we introduced the Combined Series Approach (*CSA*), a method capable of discovering and grouping motifs in spatial-time series datasets. However, as observed, the *CSA* has some restrictions due to its modeling. With the use of non-overlapping blocks, as defined in chapter 3, the motifs present in intercession of blocks are not correctly identified. Besides, the *CSA* restricts the analysis of neighboring series that are only in the same spatial dimension.

To deal with these restrictions, we introduce a simple and efficient identification algorithm to discover motifs restricted in space-time, called *Constrained Spatio-temporal Matrix Profile (CSTMP)*, and conduct computational analyzes to measure the assertiveness and performance. Our method can identify and group similar spatial-time motifs that are density-reachable. *CSTMP* combines an evolution of the Matrix Profile approach for spatio-temporal data with an incremental distance computation strategy using Euclidean distance.

The overall method can be seen as a motif spatial-temporal identification and grouping algorithm, where initially, the space-time series  $sts$  are grouped given the distance *radius* to their neighbors. For each serie  $st$  and the set of its neighbors, the *CSTMP* incrementally selects a block of subsequences from these series in a time interval  $i$ , where it applies motif discovery. Finally, since all the series were visited, the motifs found are ranked according to their frequency. As demonstrated by our experiments on various datasets, *CSTMP* identifies space-time motifs with significantly higher quality than those provided by previous approaches. Its computational time is also lower than the other approaches, including the *CSA*.

Before presenting *CSTMP*, some definitions are necessary. A **block**  $b$  is a couple  $(t, s)$  where  $t$  is the time interval and  $s$  is the spatial location. The size of a block  $b$  is the product of range size with interval length:  $|b| = |b.r| \cdot |b.i|$ . We define the set of all possible blocks over  $D$  as  $PB(r, g)$ .

## 5.1- General Principle

The general scheme is given in Algorithm 25. The spatial-time series dataset is a set of series where each **spatial-time series**  $st$  can be described as a pair  $(t, p)$ , such as a time series  $t$  with an associated position  $p$  (2.1). Based on  $p$ , the algorithm first calculates the distance between all series. Then, for each  $st$  series, a set is created with the series that is up to  $radius$  of  $sts_n$ , given by *neighborhood* function (line 3-5). All these defined sets are organized in *STG*. For each of the neighboring series groups *group* of the set *STG* a for each loop (lines 7-19) performs a search for the frequent motifs. Each series in the set of neighbors, if it has not yet been marked as visited (line 9), is traversed in an iterative way (lines 8-22). At each iteration, the *coreIndex* element is defined (line 11), generating the spatio-temporal block (line 12). All sequences internal to the block are then generated in function *generateSequences*.

The *AAMP* function is then applied to *blockSeq* (line 14), calculating the matrix profile and generating all pairs with minimum distance from each other. From this set, the even sequences of the element *coreIndex* are then selected. If they have a quantity greater than the threshold defined in  $\gamma$  (line 15), they are added to the group *motifGroup* of the element *coreIndex*. Finally, the groups of motifs are ranked according to their frequency (line 24).

### 5.1.1 Block Search

In each generated block, we want to identify the subsequences that can be reached by the *coreIndex* element. We understand that these subsequences are those that, among all the subsequences in the block, have the shortest distance to the *CI* element, thus being most similar. For this purpose an algorithm must 1) incrementally calculate the distance between all sequences and 2) select the subsequences that have the shortest distance for the *coreIndex* element.

The goal of *blockSearch* Algorithm 6 is to calculate the distance between the subsequences and identify the most similar of *cI*. For this, first, *STAAMP* Algorithm 28

---

Algorithm 5 – CSTMP

---

```

Input:  $sts, ws, \gamma, radius$ 
Output:  $SMP : Spatial Matrix Profile$ 
1 begin
2    $SMP \leftarrow \infty$ 
3    $motifGroup \leftarrow 1$ 
4   foreach ( $st \in sts$ ) do
5      $STG[st] \leftarrow neighborhood(sts, st, radius)$ 
6   end
7   foreach ( $group \in STG$ ) do
8     foreach ( $st \in group$ ) do
9       if  $notVisited(st)$  then
10        for ( $i = 1$  to  $(length(st) - ws)$ ) do
11           $coreIndex \leftarrow c(st, i)$ 
12           $block \leftarrow getBlock(coreIndex, group)$ 
13           $blockSeq \leftarrow generateSequences(block, ws)$ 
14           $stMotifs \leftarrow blockSearch(coreIndex, blockSeq, ws, SMP)$ 
15          if  $length(stMotifs) > \gamma$  then
16             $SMP \leftarrow merge(SMP, coreIndex,$ 
17               $stMotifs, motifGroup)$ 
18             $motifGroup ++$ 
19          end
20        end
21         $st \leftarrow asVisited(st)$ 
22      end
23    end
24   $SMP \leftarrow rankSTMotifs(SMP)$ 
25 end

```

---

calculate the distance profile between all the sequences of the block (line 2). Then, a search is performed on the output looking for the subsequences with the shortest distance to the  $CI$  element (line 3-7). As a result, a list of subsequences is returned.

Akbarinia and Cloez [2019] introduces the *AAMP*, a simple but efficient algorithm for computing matrix profile with the pure Euclidean distance. *AAMP* is executed over a time series in a set of iterations, such that in each iteration the distance of subsequences is computed incrementally, as presented in Chapter 2. Since *AAMP* is a matrix profile-specific approach for time series, we develop the *STAAMP*, presented in Section 5.1.1, an adaptation of *AAMP* for our space-time series context. In this way, we can perform the all-pairs-similarity-search in sequences of neighboring space-time series. *STAAMP* has a worst-case complexity of  $O(n \times (n - m))$  with small constants, where  $n$  is the number of sequences in the block and  $m$  the sequence length.

---

Algorithm 6 – BlockSearch

---

**Input:**  $ci$ : coreIndex;  $blockSeq$ : set of subsequences;  $ws$ : subsequence length;  
 $SMP$ : spatial matrix profile  
**Output:**  $sSeq$ : Set of Subsequences;

```

1 begin
2    $SMP \leftarrow STAAMP(coreIndex, blockSeq, ws, SMP)$ 
3   for  $i = 1$  to  $size(blockSeq)$  do
4     if  $SMP[i].n == CI$  then
5        $sSeq.add(i)$ 
6     end
7   end
8 end

```

---

The *STAAMP* the algorithm receive as input a set the subsequences and the sequence length  $ws$ . It performs  $n$  iterations using a variable  $k$  ( $1 \leq k \leq n$ ), where  $n$  is the number of subsequences to be analyzed (Line 5). In each iteration  $k$ , the algorithm compares each subsequence  $T_{i,m}$  with the subsequence that is  $k$  positions far from it, *i.e.*,  $T_{i,m+k}$ . To do this, *STAAMP* firstly computes the Euclidean distance of the first subsequence, *i.e.*,  $T_{1,m}$ , with the one that starts at position  $k$ , *i.e.*,  $T_{k,m}$ . This first distance computation is done using the normal formula of Euclidean distance (Line 7). Then, in a sliding window, the algorithm incrementally computes the distance of other subsequences with the subsequences that are  $k$  position far from them, and this is done by using the incremental euclidean distance in  $O(1)$  (Line 17). If the computed distance is smaller than the previous minimum distance that is kept in the matrix profile  $SMP$ , then the smaller distance is saved in the matrix profile (Line 18-26).

---

Algorithm 7 – STAAMP

---

**Input:** *blockSeq*: set of subsequences; *ws*: sequence length; *SMP*: spatial matrix profile

**Output:** *SMP*: Spatial Matrix Profile;

```

1 begin
2    $n \leftarrow \text{size}(\text{blockSeq})$ 
3   for  $i = 1$  to  $n$  do
4      $SMP[i] \leftarrow \infty$ 
5   end
6   for  $k = 1$  to  $n$  do
7      $dist \leftarrow \text{Euc\_Distance}(T_{1,ws}, T_{k,ws})$ 
8     if  $dist < SMP[1].d$  then
9        $SMP[1].n \leftarrow k$ 
10       $SMP[1].d \leftarrow dist$ 
11    end
12    if  $dist < SMP[k].d$  then
13       $SMP[k].n \leftarrow 1$ 
14       $SMP[k].d \leftarrow dist$ 
15    end
16    for  $i = 2$  to  $n - ws + 1 - k$  do
17       $dist = \sqrt{(dist^2 - (t_{i-1} - t_{i-1+k})^2 + (t_{i+ws-1} - t_{i+ws+k-1})^2}$ 
18      if  $dist < SMP[i].d$  then
19         $SMP[i].n \leftarrow i + k$ 
20         $SMP[i].d \leftarrow dist$ 
21      end
22      if  $dist < SMP[i + k].d$  then
23         $SMP[i + k].n \leftarrow i$ 
24         $SMP[i + k].d \leftarrow dist$ 
25      end
26    end
27  end
28 end

```

---



## 6- Experimental Evaluation

In this chapter, an evaluation of the methods presented in this work is provided. We report the experimental results that show the effectiveness of *CSA* and *CSTMP* in the task of identifying motifs restricted in space-time in different datasets. We compared the results with the *Matrix Profile*, the state-of-the-art approach for detecting motifs in time series. In addition, we suggest a baseline approach, where we combine the *Matrix Profile* technique with the *DBSCAN* algorithm, adapting the approach to the spatio-temporal scenario.

In Section 6.1 below, the datasets used are presented, describing their characteristics and mapped results. Section 6.2 presents the comparative analysis of the methods applied to a synthetic dataset, detailing the employment and characteristics of these approaches. Finally, in Section 6.3, a comparative and sensitivity analysis of approaches is conducted in the seismic dataset, providing more information on the approach's behavior in real data.

### 6.1- Datasets

A synthetic and a real seismic spatial-time dataset have been used for evaluation. Despite their size differences, these datasets allow us to analyze different types of motifs restricted in space-time and provide a good discussion about each approach and its effectiveness. We can better explain about datasets as follows:

**Syntetic data:** This is a dataset designed to better elucidate and understand the problem of identifying restricted motifs in space-time. Each column is a spatial-time series (varying from positions 1 to 36) with 40 observations in time. This dataset contains a set of 108 mapped occurrences, each with size 4 ( $ws = 4$ ). Figure 4 shows the identified spatial-time motifs series and highlights three motif groups. Each group allows an analysis of the spatio-temporal distribution of the identified motifs, being a set more frequent in space (red), a second set with more frequency in time (green), and a third group homogeneously distributed both in space and time (green, blue).

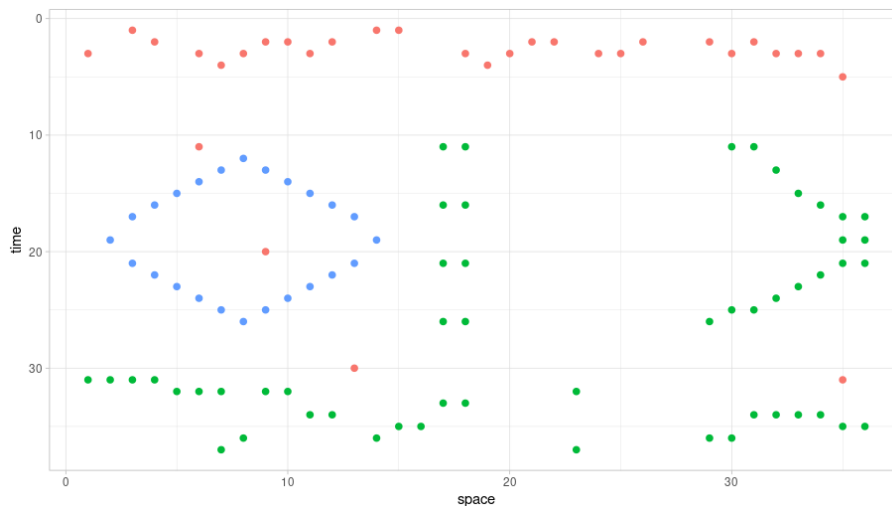


Figure 4 – Synthetic dataset with identified motifs

**Seismic data:** The seismic spatial-time series dataset, named F3 Block dgbes [2018] are produced by the seismic reflection method was collected in a region located in the Dutch sector of the North Sea. This method consists of generating artificial seismic waves with energy sources that disturb the medium, such as explosives or air guns (called seismic shots), and record the waveforms of the various interfaces in the subsoil using sensors (geophones or hydrophones) in that acquisition air guns and hydrophones were used. The generated wave propagates through the interior of the Earth and the Sea. The partially reflected waves are used to find interfaces between layers that have significant contrasting elastic properties. The arrival time of each reflection is related to the propagation velocities of the seismic wave in each layer. In a first approximation, the recorded amplitude is related to the contrast of the acoustic impedance, a product of velocity and density of the layers that define the interface.

In the F3 Block dataset, each spatial-time series has a position in which the hydrophone is placed. The dataset is organized into inlines (direction of the ship navigation). We selected inline 401 since it has been mapped by seismic specialists who have annotated some relevant information. Figure 5 presents inline 401, which consists of 920 spatial-time series with 440 observations in each. The horizontal axis represents the position of the receivers, and the vertical axis represents the time, which is also related to the depth at the subsoil. The location of seismic horizons noted by specialists is marked in the data.

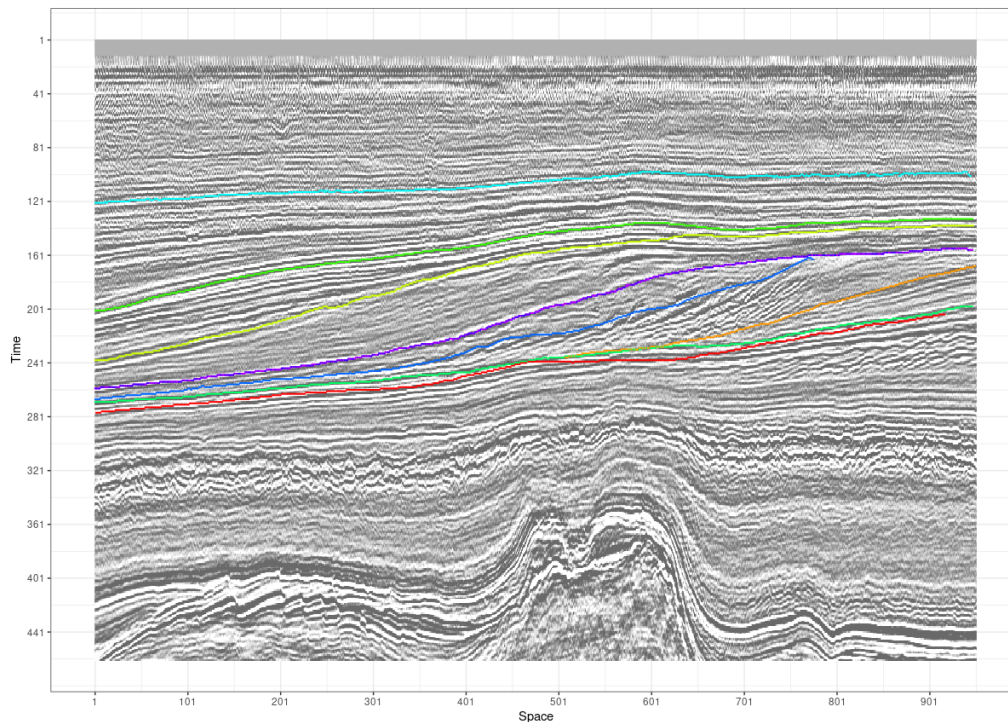


Figure 5 – Seismic dataset with the mapped horizons

## 6.2- Analysis on the synthetic dataset

First, we conducted the experiments on the synthetic dataset, which, although less complex, allows us to understand the proposed problem and the challenge of the proposed solution. The main objective is to study the number of motifs discovered and their occurrences and understand the computational cost.

We conducted our experiments by comparing the performance of our spatio-temporal motif identification methods (*CSA* [Borges et al., 2020] and *CSTMP*) with the variations of the *Matrix Profile* Yeh et al. [2018] ( $MP_1$  and  $MP_2$ ), in addition to two baseline approaches ( $B_1$  and  $B_2$ ) where we combine the *Matrix Profile* results using the *DBSCAN* algorithm. In Table 1, we list the approaches and their corresponding variations that are used in the discussion of this section. The *CSA* and the *Matrix Profile* implementation are available as an R Package in *STMotif*<sup>1</sup> and *tsmp*<sup>2</sup>, respectively. The *CSTMP* is available on the github platform<sup>3</sup>.

<sup>1</sup><https://cran.r-project.org/web/packages/STMotif/index.html>

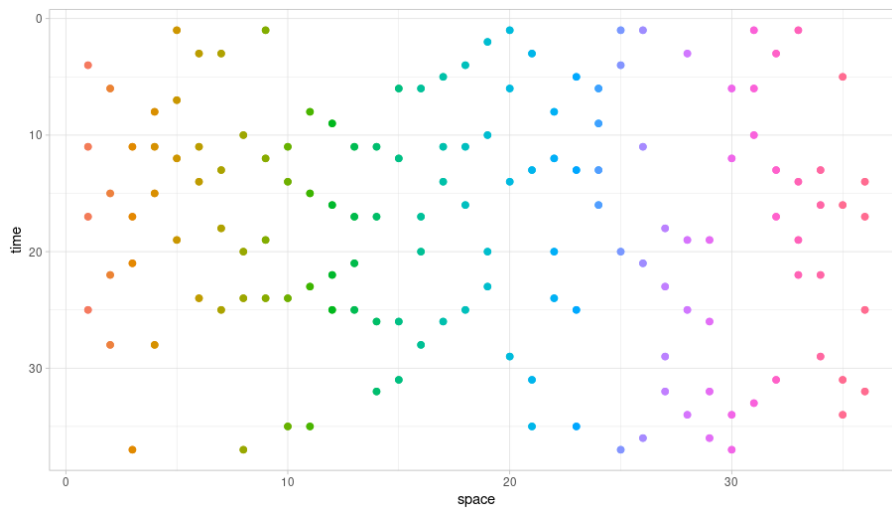
<sup>2</sup><https://cran.r-project.org/web/packages/tsmp/index.html>

<sup>3</sup><https://github.com/heraldoborges/cstmp>

Table 1 – Evaluated Approaches

Approach	Variation	Description
MP	$MP_1$	Applied to each ST-Series
	$MP_2$	Applied to the concatenated series
	$Baseline_1$	$MP_1$ with DBScan
	$Baseline_2$	$MP_2$ with DBScan
CSA	$CSA_1$	CSA with $sb = tb$
	$CSA_2$	CSA with $sb > tb$
	$CSA_3$	CSA with $sb < tb$
CSTMP	$CSTMP_1$	CSTMP

*Matrix Profile* is a technique that works for all-pair-similarity-search across a time series, as described in the section 2.1. *Matrix Profile* is a technique that works for all-pair-similarity-search across a time series, as described in the section 2.1. As it works only with time series, some adaptations are necessary to apply the technique to spatial-time series. At  $MP_1$ , we intuitively apply the technique to each series separately requesting one motif per series. At the end of the process, we have grouped all identified motif pairs into one large set. In Figure 6, we present the result of applying the  $MP_1$  approach to the synthetic data set. The approach found 36 motifs (72 occurrences), distributed by the data, with a high error rate, correctly identifying only 14 occurrences.

Figure 6 –  $MP_1$  - *Matrix Profile* on synthetic dataset

Another way to apply the *Matrix Profile*, proposed in  $MP_2$ , is to combine all spatial-time series (STS) into a single series so that the end of the  $sts_k$  series is concatenated with the beginning of  $sts_{k+1}$ . In Figure 7, we present the result of applying the  $MP_2$  approach to the synthetic data set, requesting 54 motifs (108 occurrences). The result is worse than

that presented in  $MP_1$ , featuring a large number of motif sets, correctly identifying only 12 occurrences.

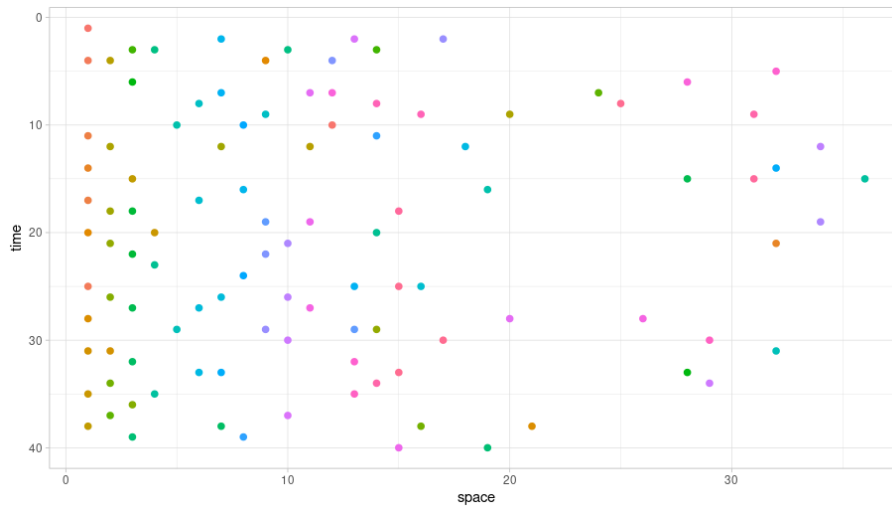


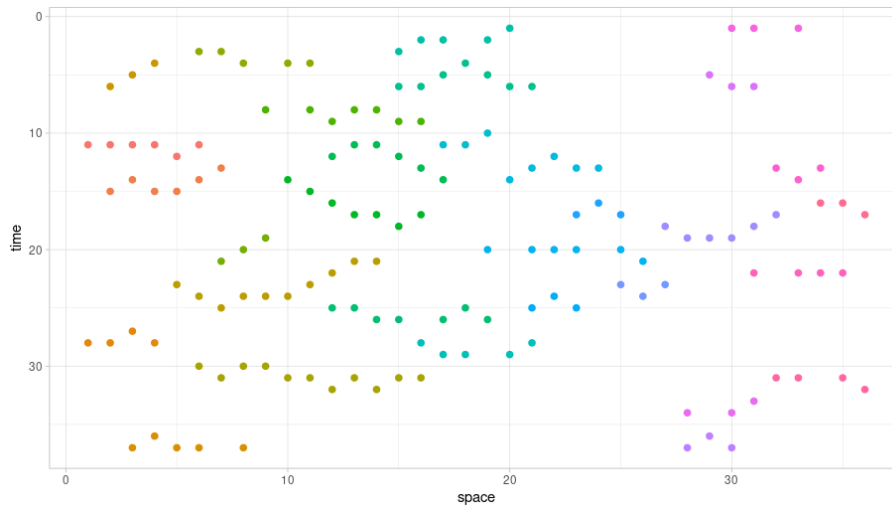
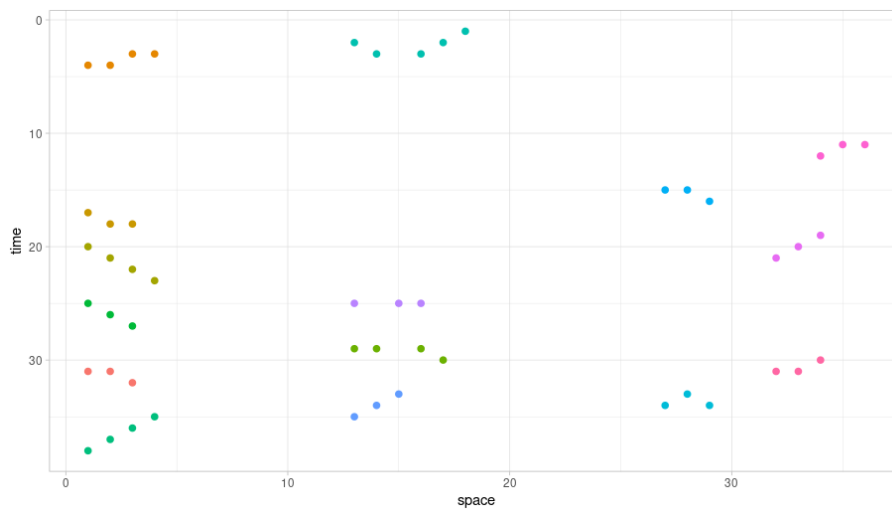
Figure 7 –  $MP_2$  - *Matrix Profile* on synthetic dataset

By conducting the experiments, it is possible to analyze that even increasing the number of requested motifs, both in approaches  $MP_1$  and in  $MP_2$ , they do not accurately detect motifs, gradually increasing the recall and decreasing the accuracy. The results become irrelevant since they present many errors in addition to identifying motifs that do not respect the space-time constraint.

In the face of these problems, as an alternative to remove outliers and restrict the observations identified in space and time, we present a baseline approach that combines the *Matrix Profile* approach with the *DBSCAN* clustering algorithm. Given the motifs identified by the *Matrix Profile*, the *DBSCAN* groups those close together into clusters and mark as outliers those who are distant, respecting the radius of a neighborhood  $\epsilon$  and a minimum number of points *minPts*. For the analysis in synthetic dataset, we considered a lower constraint,  $\epsilon = 2$ , *minPts* = 3.

In Figure 8, we present the result of *Baseline<sub>1</sub>* approach, applying *DBSCAN* to  $MP_1$  results, requesting 1000 motifs. Likewise, in *Baseline<sub>2</sub>*, we use the results of  $MP_2$ , requesting also 1000 motifs, shown in Figure 9. *DBSCAN* can identify and remove outliers, but the results show a high error rate, with low accuracy. Even performing a gradual increase in the number of motifs requested, in both approaches, the results remain with low precision, further decreasing the accuracy.

The Combined Series Approach (*CSA*) is an approach to discover and classify

Figure 8 – *Baseline<sub>1</sub>*Figure 9 – *Baseline<sub>2</sub>*

motifs in spatial time series. As the *CSA* is based on the partition of the space-time series into blocks, an important aspect of being evaluated is the influence of the block's orientation on the number of motifs discovered and their occurrences.

Thus, we evaluated three orientations: in *CSA<sub>1</sub>* using a square orientation ( $sb \approx tb$ ), in *CSA<sub>2</sub>* a vertical rectangle ( $sb > tb$ ) and in *CSA<sub>3</sub>* a horizontal rectangle ( $sb < tb$ ).

Figure 10 shows the result of *CSA<sub>1</sub>* approach, using  $tb = 9$  and  $sb = 10$ . For *CSA<sub>2</sub>* we use  $tb = 9$  and  $sb = 10$  and the result is shown in the figure 11. For *CSA<sub>3</sub>* we use a vertical rectangle with  $tb = 9$  and  $sb = 10$  with the result shown in figure 12.

Table 2 presents the overall performance of the *CSA* under different block orientation. The results demonstrate a strong influence of the orientation of the blocks on the

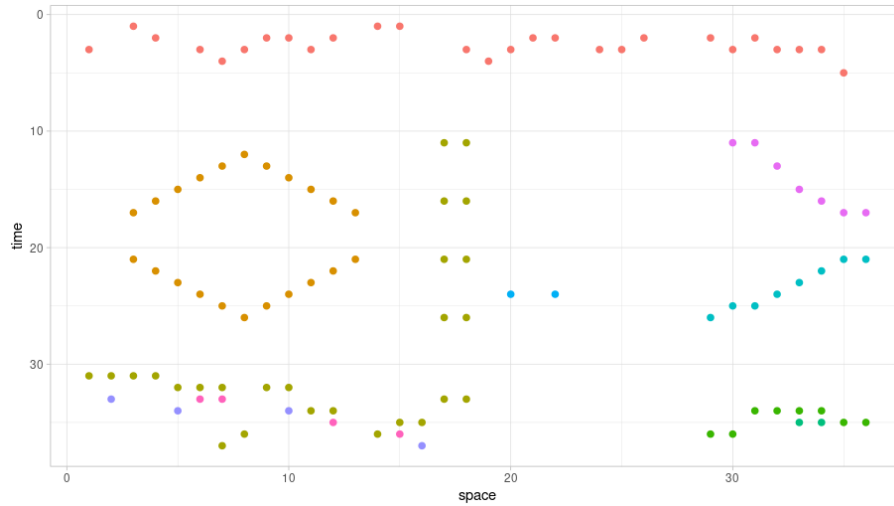


Figure 10 –  $CSA_1$  -  $CSA$  on synthetic dataset - ( $sb \approx tb$ )

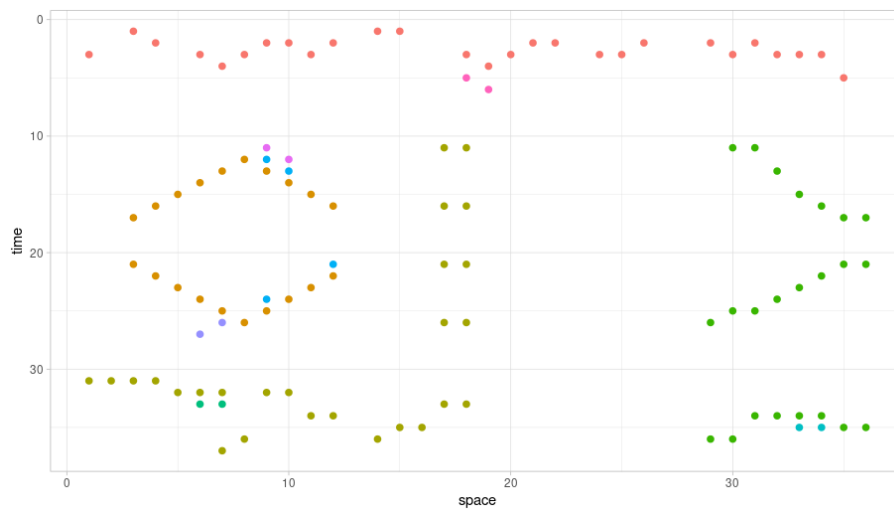


Figure 11 –  $CSA_1$  -  $CSA$  on synthetic dataset - ( $sb < tb$ )

results. In  $CSA_1$ , the method is able to correctly identify all motifs mapped in the original dataset.

Table 2 – Overall performance of  $CSA$  under different block orientation

Block orientation	motifs	sets of occur.
$CSA_1$ Square (9 x 10)	108	110
$CSA_2$ Horizontal (9 x 4)	108	110
$CSA_3$ Vertical (4 x 10)	21	108

The  $CSTMP$  approach, also proposed by this work, is developed specifically for space-time data. For the application of the technique in synthetic data, we used as a

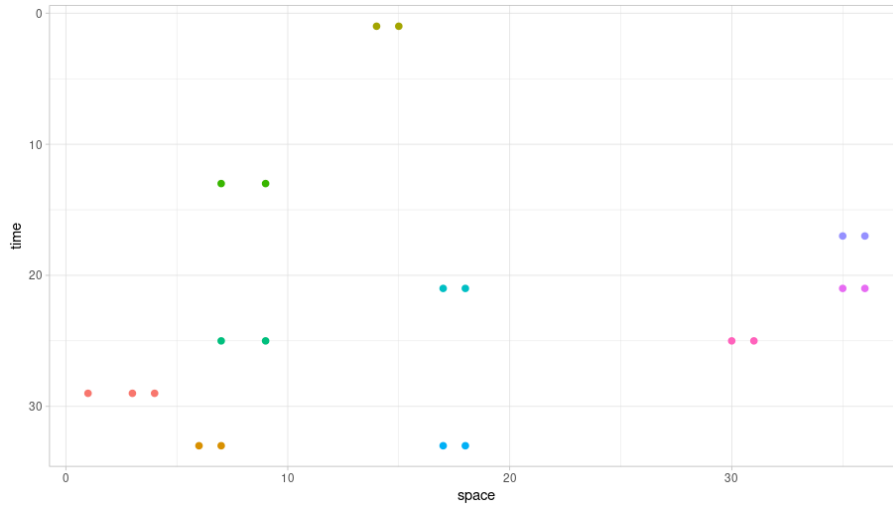


Figure 12 –  $CSA_1$  -  $CSA$  on synthetic dataset - ( $sb > tb$ )

parameter, the word size with  $ws = 4$  and  $radius = 7$ . The result of  $CSTMP$  is shown in the Figure 13. The approach is able to identify a greater number of motifs than the previous approaches, finding a set of 108 motifs, without making any mistakes. An important point is that the approach is able to identify motifs that were not identified by the  $CSA$  approach.

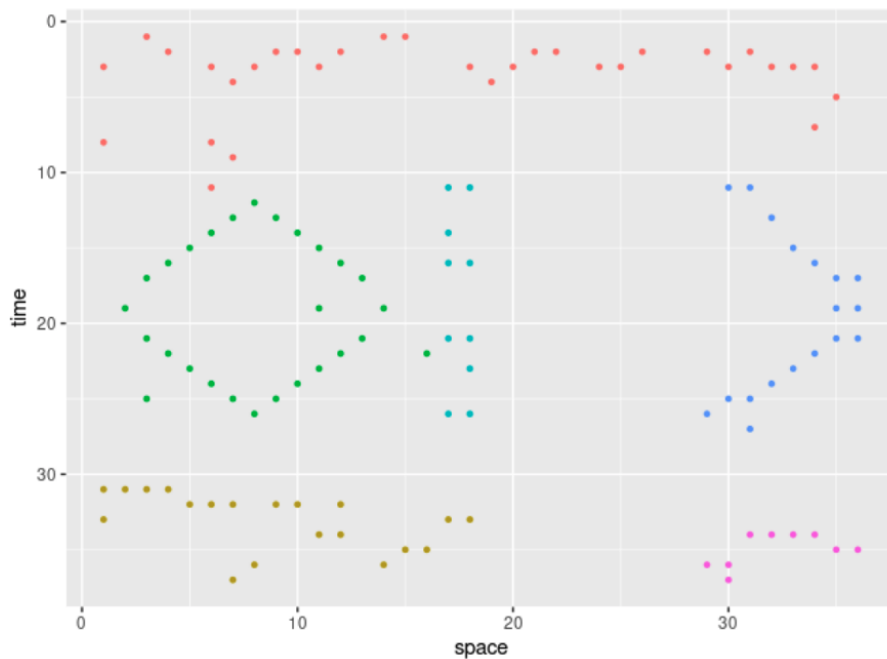


Figure 13 –  $CSTMP_1$  Synthetic



### 6.3- Analysis on the seismic dataset

In this section, we discuss the preliminary results of the approaches presented in this work in discovering spatial-time restricted motifs using the inline 401 of the Netherlands seismic spatial-time series dataset.

First, we evaluate the two suggested baseline approaches ( $Baseline_1$  and  $Baseline_2$ ). As input,  $Baseline$  enables users to set the following parameters: *word size*, which is the word size to be used by the *Matrix Profile*, varying between values 4 to 7. The *DBSCAN* receives from the *Matrix Profile* a set of positions of the found motifs and the parameters *eps* and *minPts*, generating a set of motif clusters. The *eps* parameter corresponds to the radius size between two motifs so that one is considered a neighbor to the other, and the *minPts* parameter is the minimum number of motifs in a neighborhood to be considered a central point. A list of the parameters used is shown in table 3 below.

Table 3 –  $Baseline$  Input Parameters

Parameter	Description (explored values)
<i>ws</i>	Length of motif word (from 4 to 7)
<i>eps</i>	size (radius) of the epsilon neighborhood.
<i>minPts</i>	number of minimum points required in the eps neighborhood for core points (including the point itself).

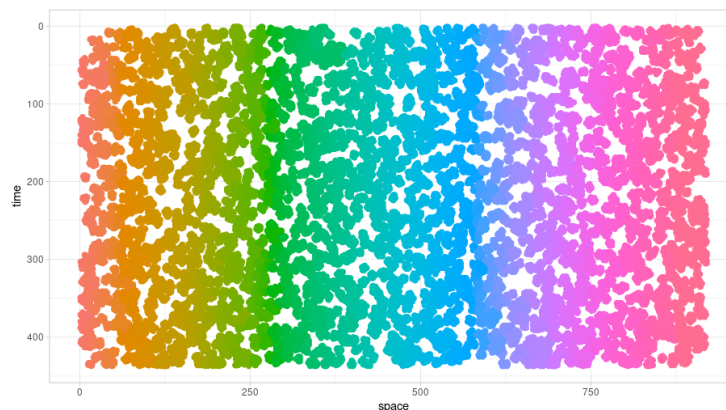


Figure 14 –  $Baseline_1(eps = 4; minPts = 10)$  - Seismic Dataset

To exemplify the application of  $Baseline_1$  in slice t401, three results are presented, as seen in the images below. The Figure 14 shows the result with the parameters  $ws = 4$ ,  $eps = 4$  and  $minPts = 10$ . The application of the *Matrix Profile* to each of the space-time

series, independently, has a low execution time, in this case 1'45'', but identifies a greater number of motifs for each series. Due to the large number of motifs found, *DBScan* builds larger clusters, covering the entire slice. We can increase the space-time constraint of the clusters by changing the parameter *minPts*, as we see in Figure 15 and Figure 16. However, for this example, few horizons are correctly mapped.

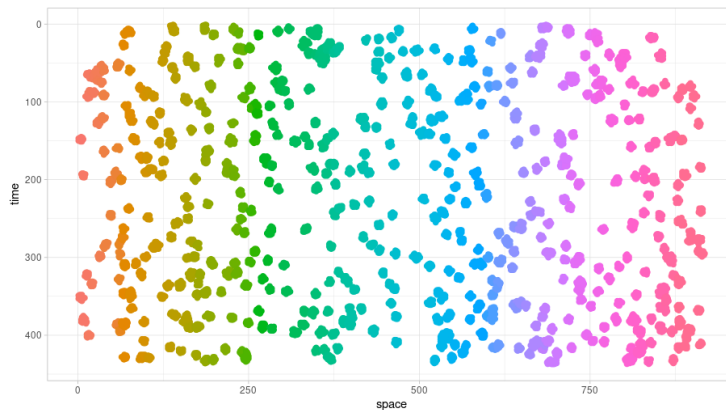


Figure 15 – *Baseline*<sub>1</sub>(*eps* = 4; *minPts* = 12) - Seismic Dataset

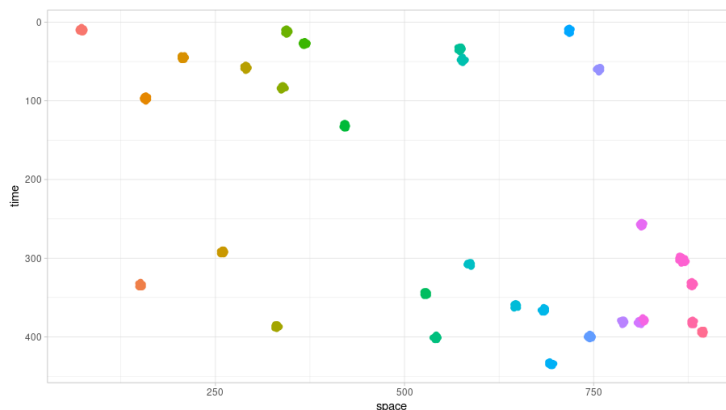


Figure 16 – *Baseline*<sub>1</sub>(*eps* = 4; *minPts* = 14) - Seismic Dataset

For the application of *Baseline*<sub>2</sub> the spatiotemporal series of slice t401 are concatenated, creating a single series  $t$  of size  $|t| = 404800$  (920 series times 440 time units). Conducting the matrix profile in this series requires approximately four hours of processing, regardless of the number of motifs you want to find. The Figure 17 shows the result with the parameters  $ws = 4$ ,  $eps = 8$  and  $minPts = 10$ . The mapped clusters present groups of motifs sparse by the slice, not being able to map any horizon. Even increasing the size of clusters, the result obtained remains very poor. Figures 18 and 19 exemplify these cases. In the first case, using the parameter  $eps = 24$ , some larger sets are formed, but with a

low recall, as many false-positive motifs are detected. In the second case, duplicating the  $eps$  parameter, we have a similar situation, where a large number of false-positive motifs are detected, without mapping and identifying the horizons.

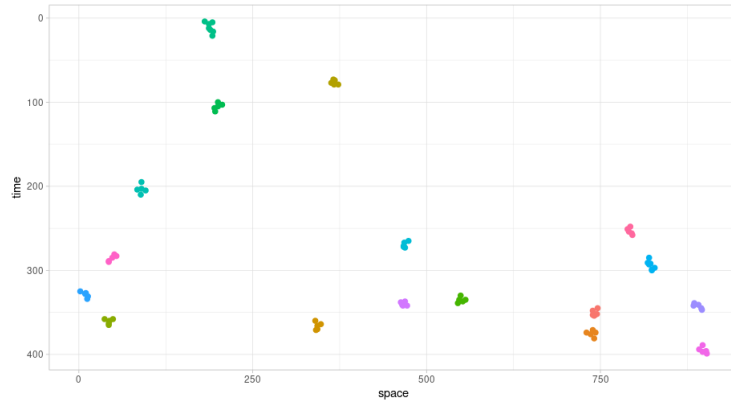


Figure 17 –  $Baseline_2(eps = 8; minPts = 5)$ - Seismic Dataset

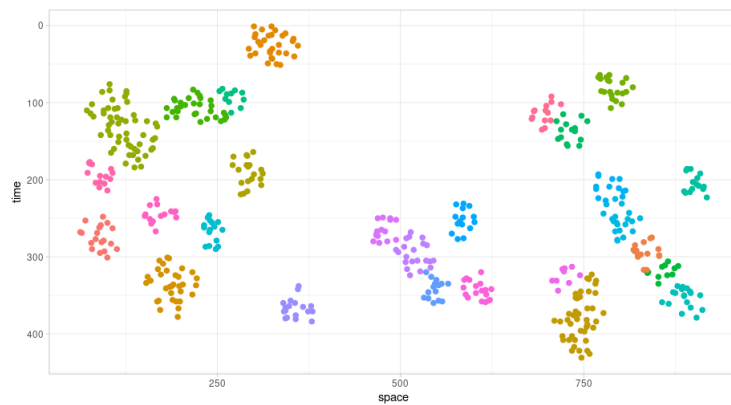


Figure 18 –  $Baseline_2(eps = 24; minPts = 15)$  - Seismic Dataset

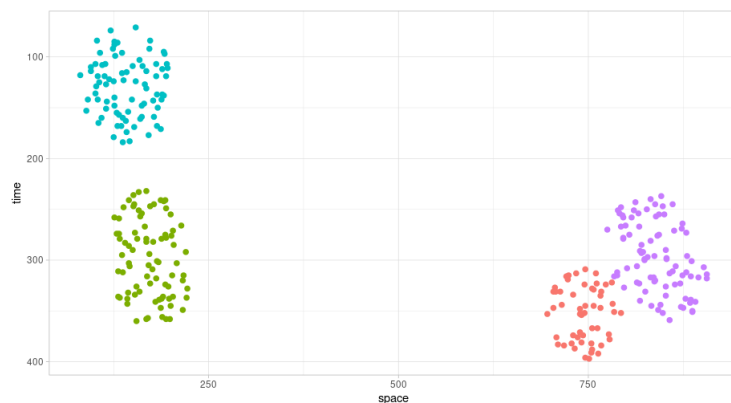


Figure 19 –  $Baseline_2(eps = 48; minPts = 50)$  - Seismic Dataset

The *CSA* Algorithm requires parameters  $alpha$ ,  $word$ ,  $tb$ ,  $sb$ ,  $\sigma$ , and  $\kappa$  to be

specified. The description of these parameters and the range explored are summarized in Table 4. These parameters influence both the quality of results and the computation elapsed time. The *alpha* was chosen based on the data adjustment. We varied the alphabet size for SAX encoding from 1 to 25, and measured the Mean Squared Error (*MSE*) for each observation concerning the mean of each SAX character. The higher the alphabet size, the lower is the *MSE*. The choice for the alphabet was identified by the maximum curvature analysis as depicted in Figure 20. The point where the maximum curvature is achieved (in red) indicates that increasing more the alphabet does not bring much more benefit concerning the *MSE*.

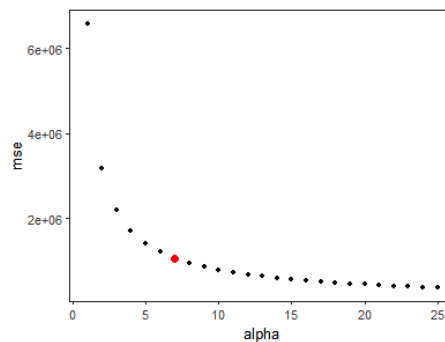


Figure 20 – *MSE* for each alphabet size

Table 4 – *CSA* Input Parameters

Parameter	Description (explored values)
<i>alpha</i>	Size of the alphabet for SAX indexing (fixed at 7)
<i>word</i>	Length of motif word (from 3 to 7)
<i>tb</i> x <i>sb</i>	Temporal and spatial block size (40x10, 20x20, 10x40)
$\sigma$	Minimum number of occurrences inside each block (from 2 to 7)
$\kappa$	Minimum number of spatial-time series with occurrences inside each block (from 1 to 5)

In this analysis, the goal is to study the number of discovered motifs and their occurrences and computational time as we vary block size (*tb* and *sb*), *word*,  $\sigma$ , and  $\kappa$ .

To evaluate the influence of block orientation, we set three orientations: vertical rectangle (*tb* = 40; *sb* = 10), square (*tb* = 20; *sb* = 20), and horizontal rectangle (*tb* = 10; *sb* = 40). For a fair comparison, all of them contain the same amount of observations.

Table 5 presents the overall performance of both traditional approach and *CSA* under different block orientation for all possible parameter combinations described in Table 4. The motifs column corresponds to the mean number of motifs whose occurrences were grouped with at least one neighboring block and contained more than seven occurrences

(the maximum  $\sigma$  value adopted). In the case of the traditional approach, we considered it as a block of 440 temporal observations with one spatial-time series, so that the same grouping criteria could also be applied.

Table 5 – Overall performance of CSA under different block orientation

Block orientation	motifs	sets of occur.	discovery time (min)	ranking time (min)	elapsed time (min)
Traditional (440x1)	43	449	1.8	2.0	4.7
CSA Vertical (40x10)	85	673	1.6	1.8	4.2
CSA Square (20x20)	<u>114</u>	<u>772</u>	1.4	1.6	3.8
CSA Horizontal (10x40)	105	705	0.9	1.2	2.9

The traditional approach, on average, discovered 43 different motifs under 449 sets of occurrences. It means that the same motif contains, on average, ten different spatial-temporal sets of occurrences. Also, the average discovery time and average time to rank motifs were, respectively, 1.8 and 2.0 minutes. The average elapsed time was 4.7 minutes, which also includes the time to do the data normalization and SAX encoding.

The time for discovering motifs was approximately the same for all configuration, except for Horizontal orientation. In this setup, as we increase the size of the word, there is a lower number of possible motifs to discover. It becomes unnecessary to check for motifs in between two consecutive spatial-time series. It makes less possible comparisons for this setup, also meaning that a lower number of motifs are discovered. However, all CSA block orientations discovered more motifs than traditional approach (the square had the better performance. It discovered more than 2.5 times more motifs than traditional approach).

Comparing the performance of different CSA orientations (Vertical, Square, and Horizontal), we may expect that typically Horizontal orientation might break temporal sequences. However, in our dataset, patterns often occurred in small time intervals spread in space. Such behavior justifies the better performance of Horizontal orientation over Vertical one. Additionally, Square orientation had a better balance between time and space and was able to identify more patterns. The choice of block orientation is fairly domain-dependent. Users may consider their knowledge about the data to set up this parameter.

Table 6 disclosures the results of Table 5 according to the word size. It presents

the number of discovered motifs and the sets of occurrences, applying the same criteria used to produce Table 5. It can be observed that as we increase the word size, the number of discovered motifs decreases. The same behavior occurs to the set of occurrences. The highest number of identified motifs occurred in CSA Square orientation for word size equals to four. Finally, the computation time (in minutes) for all discovered motifs also decreases as we increase the word size. It is due to the ranking function overhead. It has less impact on time when handling a lower number of occurrences.

Table 6 – Summary of Discovered Spatial-Time Motifs for different block orientation and word size

Block Orientation	word	motifs	set of occurrences	total time (min)
Traditional (440x1)	3	139	95862	9.5
	4	65	6809	5.4
	5	7	369	3.0
	6	2	73	2.7
	7	1	17	2.6
CSA Vertical (40x10)	3	168	62278	8.0
	4	163	13980	4.7
	5	60	2988	3.3
	6	23	761	2.7
	7	11	229	2.5
CSA Square (20x20)	3	184	62324	6.7
	4	221	16887	4.5
	5	103	4182	3.1
	6	42	1157	2.4
	7	19	352	2.1
CSA Horizontal (10x40)	3	187	52199	5.5
	4	219	12901	3.7
	5	89	2918	2.3
	6	25	628	1.6
	7	7	149	1.2

Table 12 presents the influence of  $\sigma$  and  $\kappa$  in the number of discovered motifs for the CSA according to the CSA Square block orientation for word size equals to four. It is possible to observe that as we increase  $\sigma$ , lower number of occurrences are identified. Also, as we increase  $\kappa$  constraint, the number of occurrences also decreases.

Finally, we analyzed the top-k spatial-time motifs discovered using CSA Square block orientation for word size of four, fixing  $\sigma$  equals to three and  $\kappa$  equals to three. In this configuration, as presented in Table 8, we computed the top-5 distinct motifs that accomplished the same criteria adopted to build Table 5.

The highest ranked motif (*aagg*) presented a good proximity value, an average

Table 7 – Influence of  $\sigma$  and  $\kappa$  in the number of occurrences in Square (20x20) setup with word size  $w = 4$

$\kappa$	$\sigma$					
	2	3	4	5	6	7
1	42725	30052	21349	13559	9621	6959
2	42253	29938	21297	13527	9589	6927
3	-	<u>29640</u>	21191	13461	9530	6895
4	-	-	20073	13184	9368	6758
5	-	-	-	11900	8800	6490

Table 8 – Top five distinct motifs

motif	proximity	entropy	occurrences	rank
<i>aagg</i>	0.74	1.0	8.28	1.57
<i>dfge</i>	0.83	2.0	3.16	1.46
<i>aaag</i>	0.85	0.8	7.06	1.45
<i>ggfa</i>	1.00	1.5	3.17	1.40
<i>egfa</i>	0.75	2.0	3.17	1.39

entropy value, and a high occurrences value. Such combination produced a rank value of 1.57. The second place (*dfge*), although exhibiting low occurrences value, has a good proximity and entropy values. The third place (*aaag*) was similar to the first motif, but with lower occurrences value. The fourth place (*ggfa*) compensated the low occurrences value with an excellent proximity value. Finally, the fifth place (*egfa*) is similar to the second, with a slightly lower proximity value.

In order to have an intuition on the quality of the ranked motifs, we have plotted the top-ten discovered motifs as we see in Figure 21(a), according to the ranking function, on top of the seismic dataset. The places where the motifs were plotted are in agreement with annotations from specialists where seismic horizons are located. Also the yellow ones are very close to a gas reservoir.

In a complementary analysis, we sorted the motifs according to the number of occurrences. Figure 21(b) plots the top-ten distinct motifs sorted by their occurrences. The set of occurrences for each motif was plotted, as long as their ranking value were greater than 1.0. It can be observed that the occurrences of motifs matched more regions where seismic horizons are located.

It is worth mentioning that the ranking function was conceived for general purpose usage and did not focus on any aspect to target seismic horizons. Nevertheless, they were able to discover the majority areas in which seismic horizons were annotated.

*CSTMP* is a hybrid approach that combines the features of a density-based clustering approach, such as *DBSCAN*, and motif identification in a local search. The

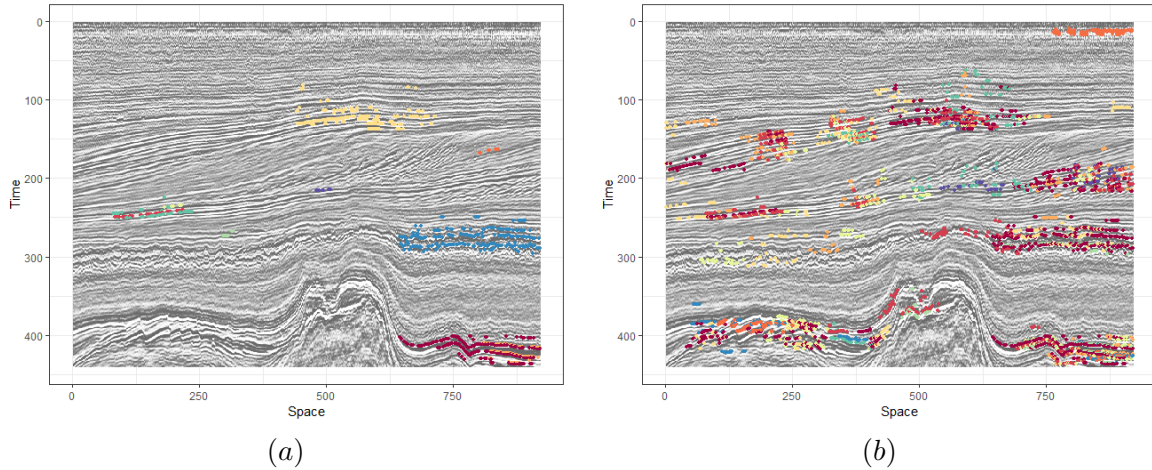


Figure 21 – (a) Top-ten discovered motifs according to the ranking function (b) Top-ten discovered motifs according to the number of occurrences, filtering the ones with ranking function lower than 1.0

algorithm requires parameters as  $ws$ ,  $\gamma$  and  $radius$  to be initialized. The description of these parameters and the range explored are summarized in Table 9. First, we study the effect of the parameters of *CSTMP* applied to the seismic dataset, how they influence the quality of results and processing time. Through this analysis, we want to study the identified motifs, the number of occurrences and the computational time as we vary the desired amount  $\gamma$ , the radius size and the word size.

Table 9 – *CSTMP* Input Parameters

Parameter	Description (explored values)
$ws$	Length of motif word (from 4 to 7)
$radius$	Size of the radius for local search (from 5 to 7)
$\gamma$	Minimum number of occurrences inside each block (from 3 to 7)

The choice of  $radius$  size is based on the neighborhood to be investigated. We varied the radius size from 5 to 7 and measure the quality and associated time. The larger the radius, the larger the search area and thus the longer is the processing time. For an equal comparison, all of them used the same word size ( $ws = 4$ ) and the same threshold ( $\gamma = 3$ ).

Table 10 presents the overall performance of *CSTMP* with variations in radius size. The motif column corresponds to the number of identified motif sets, where it maintains high similarity to each other (observing the minimum value  $\gamma = 3$  occurrences by set). The occurrences column corresponds to the total number of occurrences of the identified motifs. Lastly, the discovery time to identify motifs with this setting is presented.



Table 10 – Overall performance of CSTMP under different radius size

Radius size	word size	motifs	occur.	discovery time (min)
5 (11 x 6)	4	1596	8651	2.8
6 (13 x 7)	4	1745	9564	3.6
7 (15 x 8)	4	2092	11745	4.5

As the radius size increased, the time for discovering motifs also increased. This confirmation was already expected, since the increase in radius, with the consequent increase in the block, generates a proportionally greater number of comparisons between neighboring sequences.

Table 11 extends the results presented in Table 10 according to the radius size. For each radius and word size, it shows the number of identified motif sets and the number of occurrences, observing the minimum of  $\gamma = 3$  occurrences per set. It is possible to observe that the increase in word size generates a small increase in the number of identified motifs. This behavior occurs by generating small sets of motifs, with little similarity. Computation time is also presented, which grows as the radius is increased.

Table 11 – Summary of Discovered Spatial-Time Motifs for different radius and word size

Radius size	word	motifs	set of occurrences	total time (min)
5 (11x6)	4	1596	8651	2.8
	5	1651	9169	3.0
	6	1659	9309	3.3
	7	1679	9628	3.6
6 (13 x 7)	4	1745	9564	3.6
	5	1826	10367	3.8
	6	1946	11121	3.9
	7	2001	11756	4.1
7 (15 x8)	4	2092	11745	4.4
	5	2146	12359	4.5
	6	2237	13079	4.7
	7	2254	13506	4.9

Finally, Table 12 presents the influence of  $\gamma$  in the number of discovered motifs for the *CSTMP* as the size of the radius increases. It is possible to observe that as we increase  $\gamma$ , lower number of occurrences are identified.

To assess the quality of the identified motifs, we traced two results, according to the ranking of the group of motif sets found. In Figure 22, we present the top motifs with

Table 12 – Influence of  $\gamma$  in the number of motifs and occurrences with word size  $w = 4$ 

<i>Radius</i>	$\gamma$				
	3	4	5	6	7
5	1596 (8651)	210 (1377)	63 (667)	5 (52)	2 (27)
6	1745 (9564)	331 (2232)	54 (340)	5 (44)	1 (15)
7	2092 (11745)	474 (3013)	51 (354)	4 (36)	-

the highest number of occurrences. Then, in Figure 23, we present the top motifs with greater spatial coverage. It can be observed that the identified motifs cover the regions of the mapped horizons, identifying, beyond the horizons, areas with potential accumulation of hydrocarbons.

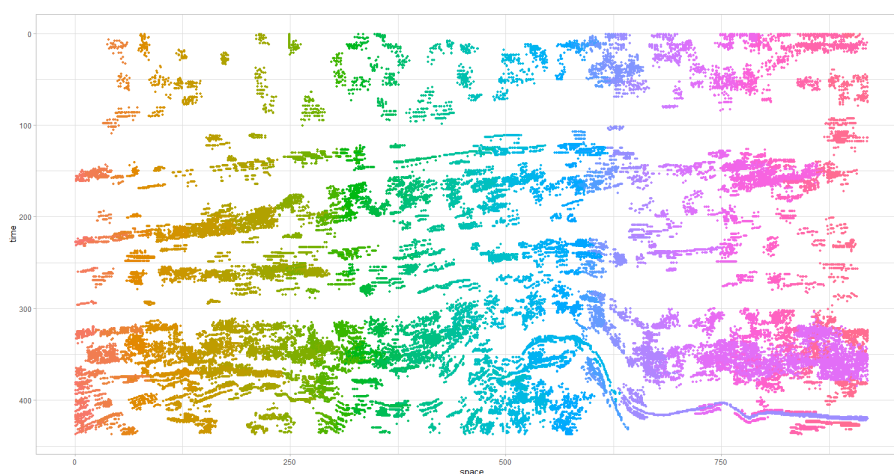


Figure 22 – Top discovered motifs according to the number of occurrences.

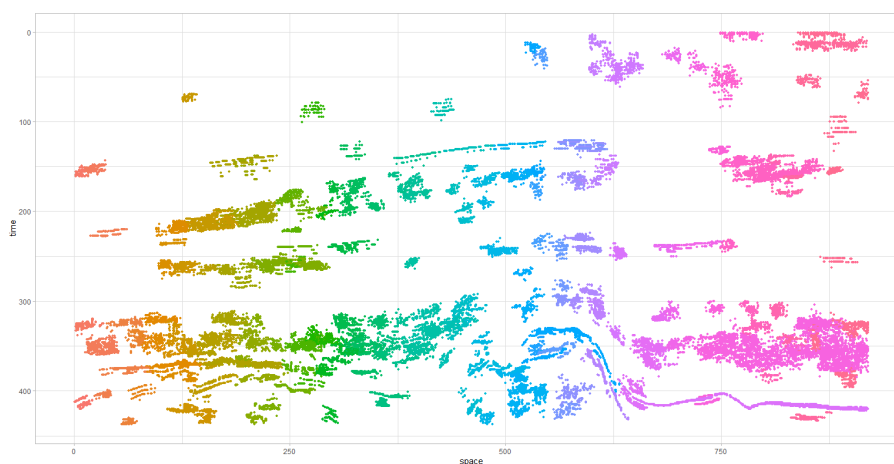


Figure 23 – Top discovered motifs according to spatial extension, filtering those that cover a greater number of spatial-time series.

## 7- Conclusion

Many applications observe phenomena whose values vary according to space and time dimensions. Discovering phenomena which are dependent on the occurrence in space and time requires extensions to traditional techniques adopted in time series analysis. In this work, we present a challenging problem with high impact potential: the discovery of motifs restricted in space and time. We tackle this problem by introducing two novel approach for spatial-time series motif discovery, the *CSA* (Combined Series Approach) and the *CSTMP* (Constrained Spatio-Temporal Matrix Profile). To the best of our knowledge, this is the first work to propose a complete approach for spatial-time motif discovery.

*CSA* supersedes traditional techniques when discovering spatial-time motifs, as it has been shown in our experimental evaluation. Additionally, *CSA* exhibits two major strengths points. Firstly, it is a divide-and-conquer algorithm that starts by discovering motifs inside a given spatial-time block. These blocks are then merged if neighboring blocks increase the number of occurrences of the discovered motifs. Such a technique makes the algorithm resilient to the initial block selection. Secondly, once the blocks have been defined, the algorithm is independent of the motif discovery algorithms applied. Such property enables exploring different motif discovery algorithms, such as *Random Projection* and *Matrix Profile*, to space-time series discovery.

We also propose *CSTMP*, an approach inspired by the *Matrix Profile* and *DBScan* approaches. Our method can identify and group similar spatial time motifs that are reachable by density. *CSTMP* handles some restrictions found in *CSA*, avoiding the problem of non-overlapping blocks. Through *CSTMP* it is also possible to perform data analysis with two spatial dimensions and one temporal dimension. The approach achieved better results than traditional techniques such as *Matrix Profile*, even in a tailored approach, with less processing time in many scenarios.

We have evaluated *CSA* and *CSTMP* against traditional approach using both synthetic and seismic dataset. The two approaches was able to identify more motifs and occurrences than the traditional approach. Also, the identified motifs were also well ranked considering both spatial-time constraints and number of occurrences.

Although the assessments were conducted using seismic data, the proposed approaches were conceived generically and applied in different scenarios with spatio-temporal data. Due to the potential of the techniques, opportunities open up to explore other real-world applications modeled as spatio-temporal series (such as ocean surface temperature data and pandemic propagation data). There are also opportunities to explore different classification functions to address domain-specific problems.

## Bibliography

- Akbarinia, R. and Cloez, B. (2019). Efficient Matrix Profile Computation Using Different Distance Functions. *arXiv:1901.05708 [cs, stat]*.
- Armstrong, T. and Drewniak, E. (2011). Unsupervised discovery of motifs under amplitude scaling and shifting in time series databases. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 6871 LNAI:539–552.
- Borges, H., Dutra, M., Bazaz, A., Coutinho, R., Perosi, F., Porto, F., Masegla, F., Pacitti, E., and Ogasawara, E. (2020). Spatial-time motifs discovery. *Intelligent Data Analysis*, 24(5):1121–1140.
- Buhler, J. and Tompa, M. (2002). Finding motifs using random projections. *Journal of Computational Biology*, 9(2):225–242.
- Cassisi, C., Aliotta, M., Cannata, A., Montalto, P., Patanè, D., Pulvirenti, A., and Spampinato, L. (2013). Motif Discovery on Seismic Amplitude Time Series: The Case Study of Mt Etna 2011 Eruptive Activity. *Pure and Applied Geophysics*, 170(4):529–545.
- Castro, N. and Azevedo, P. (2010). Multiresolution motif discovery in time series. In *Proceedings of the 10th SIAM International Conference on Data Mining, SDM 2010*, pages 665–676.
- Castro, N. and Azevedo, P. (2012). Significant motifs in time series. *Statistical Analysis and Data Mining*, 5(1):35–53.
- Chi, H. and Wang, S. (2013). Finding time series motifs based on cloud model. In *Proceedings - 2013 IEEE International Conference on Granular Computing, GrC 2013*, pages 70–75.
- Chi, L., Feng, Y., Chi, H., and Huang, Y. (2012). Face image recognition based on time series motif discovery. In *Proceedings - 2012 IEEE International Conference on Granular Computing, GrC 2012*, pages 72–77.

- Chiu, B., Keogh, E., and Lonardi, S. (2003). Probabilistic discovery of time series motifs. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 493–498.
- Daw, C., Finney, C., and Tracy, E. (2003). A review of symbolic analysis of experimental data. *Review of Scientific Instruments*, 74(2):915–930.
- De Paepe, D., Vanden Hautte, S., Steenwinckel, B., De Turck, F., Ongenae, F., Janssens, O., and Van Hoecke, S. (2020). A generalized matrix profile framework with support for contextual series analysis. *Engineering Applications of Artificial Intelligence*, 90.
- dgbes (2018). Netherlands Offshore F3 Block - Complete. Technical report, <https://opendtect.org/osr/Main/NetherlandsOffshoreF3BlockComplete4GB>.
- Du, X., Jin, R., Ding, L., Lee, V., and Thornton Jr., J. (2009). Migration motif: A spatial-temporal pattern mining approach for financial markets. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1135–1143.
- Fan, Y. and Kamath, C. (2015). Identifying and exploiting diurnal motifs in wind generation time series data. *International Journal of Pattern Recognition and Artificial Intelligence*, 29(2).
- Fu, T.-C. (2011). A review on time series data mining. *Engineering Applications of Artificial Intelligence*, 24(1):164–181.
- Han, J., Cheng, H., Xin, D., and Yan, X. (2007). Frequent pattern mining: Current status and future directions. *Data Mining and Knowledge Discovery*, 15(1):55–86.
- Han, J., Pei, J., and Kamber, M. (2011). *Data Mining: Concepts and Techniques*. Elsevier.
- Jiang, T., Feng, Y., Zhang, B., Shi, J., and Wang, Y. (2008). Finding motifs of financial data streams in real time. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 5370 LNCS:546–555.
- Keogh, E. and Kasetty, S. (2003). On the Need for Time Series Data Mining Benchmarks: A Survey and Empirical Demonstration. *Data Mining and Knowledge Discovery*, 7(4):349–371.

- Keogh, E. and Lin, J. (2005). Clustering of time-series subsequences is meaningless: Implications for previous and future research. *Knowledge and Information Systems*, 8(2):154–177.
- Keogh, E. and Ratanamahatana, C. (2005). Exact indexing of dynamic time warping. *Knowledge and Information Systems*, 7(3):358–386.
- Lampert, C., Blaschko, M., and Hofmann, T. (2008). Beyond sliding windows: Object localization by efficient subwindow search. In *26th IEEE Conference on Computer Vision and Pattern Recognition, CVPR*.
- Li, L. and Nallela, S. (2009). Probabilistic discovery of motifs in water level. In *2009 IEEE International Conference on Information Reuse and Integration, IRI 2009*, pages 388–393.
- Li, Y. and Lin, J. (2010). Approximate variable-length time series motif discovery using grammar inference. In *Proceedings of the 10th International Workshop on Multimedia Data Mining, MDMKDD '10*.
- Lin, J., Keogh, E., Lonardi, S., and Chiu, B. (2003). A symbolic representation of time series, with implications for streaming algorithms. In *Proceedings of the 8th ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery, DMKD '03*, pages 2–11.
- Lin, J., Keogh, E., Wei, L., and Lonardi, S. (2007). Experiencing SAX: A novel symbolic representation of time series. *Data Mining and Knowledge Discovery*, 15(2):107–144.
- Lin, Y., McCool, M., and Ghorbani, A. (2010). Motif and anomaly discovery of time series based on subseries join. In *Proceedings of the International MultiConference of Engineers and Computer Scientists 2010, IMECS 2010*, pages 481–486.
- McGovern, A., Rosendahl, D., Brown, R., and Droegemeier, K. (2011). Identifying predictive multi-dimensional time series motifs: An application to severe weather prediction. *Data Mining and Knowledge Discovery*, 22(1-2):232–258.
- Minnen, D., Isbell, C., Essa, I., and Starner, T. (2007). Discovering multivariate motifs using subsequence density estimation and greedy mixture learning. In *Proceedings of the National Conference on Artificial Intelligence*, volume 1, pages 615–620.

- Mohammad, Y. and Nishida, T. (2009). Constrained motif discovery in time series. *New Generation Computing*, 27(4):319–346.
- Mohammad, Y. and Nishida, T. (2014). Exact discovery of length-range motifs. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 8398 LNAI(PART 2):23–32.
- Mooney, C. and Roddick, J. (2013). Sequential pattern mining - Approaches and algorithms. *ACM Computing Surveys*, 45(2).
- Mueen, A. (2013). Enumeration of time series motifs of all lengths. In *Proceedings - IEEE International Conference on Data Mining, ICDM*, pages 547–556.
- Mueen, A. (2014). Time series motif discovery: Dimensions and applications. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 4(2):152–159.
- Mueen, A., Keogh, E., Zhu, Q., Cash, S., and Westover, B. (2009). Exact discovery of time series motifs. In *Society for Industrial and Applied Mathematics - 9th SIAM International Conference on Data Mining 2009, Proceedings in Applied Mathematics*, volume 1, pages 469–480.
- Mueen, A., Keogh, E., Zhu, Q., Cash, S., Westover, M., and Bigdely-Shamlo, N. (2011). A disk-aware algorithm for time series motif discovery. *Data Mining and Knowledge Discovery*, 22(1-2):73–105.
- Mueen, A., Zhu, Y., Yeh, M., Kamgar, K., Viswanathan, K., Gupta, C., and Keogh, E. (2017). The Fastest Similarity Search Algorithm for Time Series Subsequences under Euclidean Distance. Technical report, <http://www.cs.unm.edu/mueen/FastestSimilaritySearch.html>.
- Narang, A. and Bhattacharjee, S. (2010). Parallel exact time series motif discovery. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 6272 LNCS(PART 2):304–315.
- Narang, A. and Bhattacharjee, S. (2011). Real-time approximate range motif discovery & data redundancy removal algorithm. In *ACM International Conference Proceeding Series*, pages 485–496.
- Nunthanid, P., Niennattrakul, V., and Ratanamahatana, C. (2011). Discovery of variable length time series motif. In *ECTI-CON 2011 - 8th Electrical Engineering/ Electronics*,



- Computer, Telecommunications and Information Technology (ECTI) Association of Thailand - Conference 2011*, pages 472–475.
- Nunthanid, P., Niennattrakul, V., and Ratanamahatana, C. (2012). Parameter-free motif discovery for time series data. In *2012 9th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology, ECTI-CON 2012*.
- Ogasawara, E., Murta, L., Zimbrão, G., and Mattoso, M. (2009). Neural networks cartridges for data mining on time series. In *Proceedings of the International Joint Conference on Neural Networks*, pages 2302–2309.
- Patel, P., Keogh, E., Lin, J., and Lonardi, S. (2002). Mining motifs in massive time series databases. In *Proceedings - IEEE International Conference on Data Mining, ICDM*, pages 370–377.
- Serrà, J. and Arcos, J. (2016). Particle swarm optimization for time series motif discovery. *Knowledge-Based Systems*, 92:127–137.
- Shannon, C. (1948). A Mathematical Theory of Communication. *Bell System Technical Journal*, 27(3):379–423.
- Shekhar, S., Feiner, S., and Aref, W. (2016). Spatial computing. *Communications of the ACM*, 59(1):72–81.
- Shumway, R. H. and Stoffer, D. S. (2017). *Time Series Analysis and Its Applications: With R Examples*. Springer.
- Tang, H. and Liao, S. (2008). Discovering original motifs with different lengths from time series. *Knowledge-Based Systems*, 21(7):666–671.
- Torkamani, S. and Lohweg, V. (2017). Survey on time series motif discovery. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 7(2).
- Truong, C. and Anh, D. (2015). A fast method for motif discovery in large time series database under dynamic time warping. *Advances in Intelligent Systems and Computing*, 326:155–167.
- Tsai, C.-W., Lai, C.-F., Chao, H.-C., and Vasilakos, A. (2015). Big data analytics: a survey. *Journal of Big Data*, 2(1).

- Van Hoan, M. and Exbrayat, M. (2013). Time series symbolization and search for frequent patterns. In *ACM International Conference Proceeding Series*, pages 108–117.
- Wilson, W., Birkin, P., and Aickelin, U. (2008). The motif tracking algorithm. *International Journal of Automation and Computing*, 5(1):32–44.
- Yang, J., Wang, W., and Yu, P. (2004). Mining surprising periodic patterns. *Data Mining and Knowledge Discovery*, 9(2):189–216.
- Yankov, D., Keogh, E., Medina, J., Chiu, B., and Zordan, V. (2007). Detecting time series motifs under uniform scaling. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 844–853.
- Yeh, C.-C., Kavantzias, N., and Keogh, E. (2017a). Matrix profile VI: Meaningful multidimensional motif discovery. In *Proceedings - IEEE International Conference on Data Mining, ICDM*, volume 2017-November, pages 565–574.
- Yeh, C.-C., Zhu, Y., Ulanova, L., Begum, N., Ding, Y., Dau, H., Silva, D., Mueen, A., and Keogh, E. (2017b). Matrix profile I: All pairs similarity joins for time series: A unifying view that includes motifs, discords and shapelets. In *Proceedings - IEEE International Conference on Data Mining, ICDM*, pages 1317–1322.
- Yeh, C.-C., Zhu, Y., Ulanova, L., Begum, N., Ding, Y., Dau, H., Zimmerman, Z., Silva, D., Mueen, A., and Keogh, E. (2018). Time series joins, motifs, discords and shapelets: a unifying view that exploits the matrix profile. *Data Mining and Knowledge Discovery*, 32(1):83–123.
- Zhou, H.-W. (2014). *Practical Seismic Data Analysis*. Cambridge University Press.