Master's thesis

Zhicheng Hu

Encoding and Visualizing Temporal Changes of Topology Density Maps

Master's thesis in Simulation and Visualization
Supervisor: Ricardo Da Silva Torres
Co-supervisor: Agus Hasan
June 2022

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of ICT and Natural Sciences

Zhicheng Hu

Encoding and Visualizing Temporal Changes of Topology Density Maps

Master's thesis in Simulation and Visualization
Supervisor: Ricardo Da Silva Torres
Co-supervisor: Agus Hasan
June 2022

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of ICT and Natural Sciences

NTNU
Innovation and Creativity

Encoding and Visualizing Temporal Changes of Topology Density Maps

Zhicheng Hu

June 2022

MASTER THESIS
Department of ICT and Engineering
Norwegian University of Science and Technology

Supervisor: Prof. Ricardo Da Silva Torres

Co-supervisor: Prof. Agus Hasan

Preface

The research work presented in this master thesis was conducted at the Department of ICT and Natural Sciences, Faculty of Information Technology and Electrical Engineering at the Norwegian University of Science and Technology (NTNU), Ålesund, Norway. It has been written to complete the essential part for fulfilling the requirements of the Master Program in Simulation and Visualization.

In April 2021, I was lucky to invite my best friends Amirashkan Haghshenas and Amirabbas Hojjati to work as a small team to explore the potential research work directions under the supervision of Prof. Ricardo da Silva Torres. With his encouragement and guidance, we researched the state-of-the-art data visualization methods and decided to optimize an existing visualization method aiming its application in some funded projects. That is also the original idea related to this master thesis research. In August 2021, we had to work individually because of the study schedule, and I decided to continue the research work alone with Prof. Ricardo. Until December 2021, the software prototype and one draft report of a journal paper had been produced as a result of our efforts. In Spring 2022, Prof. Ricardo proposed this master thesis topic and encouraged me to continue this work, with more emphasis on computation efficiency improvement, visualization prototype design, case studies research, etc.

This research work has been considered in the context of the NORDARK project, funded by NordForsk, and the Smart Plan [grant number #310056] and Twin Fjord [grant number #320627] projects funded by the Research Council of Norway. The research questions were formulated with my main supervisor Prof. Ricardo da Silva Torres and co-supervisor Prof. Agus Hasan. The research was quite challenging, but conducting comprehensive research has answered the questions we raised. The conducted research includes the design and implementation of new algorithms, the creation of software prototypes, and their validation in terms of their performance and also in the context of their utilization in two relevant case studies. The whole research addressed existing literature gaps related to the encoding and visualization of temporal changes in the context of topology density maps.

Acknowledgment

First, I would like to thank my supervisor, Prof. Ricardo da Silva Torres, and co-supervisor, Prof. Agus Hasan, for their time and patience in guiding me to overcome the difficulties even during their holidays. Many thanks to Amirashkan Haghshenas and Amirabbas Hojjati, who helped us formulate the original idea of this research work and made the initial achievements with the group efforts. I also wish to thank Claudia Viviana Lopez Alfaro, Di Wu, Sihan Gao, Léo Francois Jean Leplat, Syed Hammad Hussain Shah and many other friends who are often willing to give me feedback and kindly provide assistance.

Last but not least, my wife Hanlu Lou and our daughter Hanni Hu deserve my special note of thanks. Your arrival to Norway and presence with me from November 2021 always keep me motivated. Fortunately, I have the chance to accomplish the master's program soon with the ending of the tough corona situations. However, it will not only be the end of my master's study but also be a new start of my Ph.D. study at NTNU, Trondheim, from September 2022. After working for 14 years before, I hope that I can continuously devote myself to the research work of Data Science in more disciplines in my life. Wish you enjoy reading this master thesis and get some research inspiration after that.

Zhicheng Hu

Ålesund, Norway
June 2022

Summary

Spatiotemporal urban data analysis is one of the most relevant and challenging topics in the computer graphics and information visualization research areas. This master project introduces a novel solution, named Temporal Topology Density Map (TTDM), to represent 2D discrete spatial data with temporal variations into a 2D continuous spatial space constrained by a topology. Two algorithms are introduced in the conducted research. The first one integrates Image-Foresting Transform (IFT) into the computation of Topology Density Map (TDM), which leads to an efficient solution. The second one, in turn, combines topological density maps with Change Frequency Heatmap (CFH) to convey visual information on changes over time, which lead to a new visualization method. Two case studies related to the analysis of response time associated with emergency services and walkability changes over time for areas of interest demonstrate the effectiveness of the proposed approach in challenging scenarios. We could observe that the proposed solution provides an intuitive visualization for supporting the accurate analysis of spatiotemporal data changes over time using topology density maps.

Abbreviations

API	Application Programming Interface.
CFH	Change Frequency Heatmap.
CSV	comma-separated values.
DAG	directed acyclic graph.
DLL	Dynamic Link Library.
IFT	Image-Foresting Transform.
ITDM	IFT-based Topology Density Map.
JSON	JavaScript Object Notation.
KDE	Kernel Density Estimation.
NKDE	Network-constrained Kernel Density Estimation.
POI	points-of-interest.
SDK	Software Development Kit.
ST	spatiotemporal.
TDM	Topology Density Map.
TTDM	Temporal Topology Density Map.
UI	user interface.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This chapter presents motivational aspects related to the conducted work as well as the objective and associated research questions. It also provides an overview of the proposed approach, and presents a summary of the main contributions of this work. Finally, this chapter outlines the organization of the document.

1.1 Background

Advances in terms of processing and storage capacity of devices have led to the creation of large data collections in several domains. In particular, a large amount of spatiotemporal (ST) data is being produced and consumed, especially associated with the widely use of sensing technologies or as a result of the application of algorithms (data-driven or simulation-based). The proper analysis of these data, associated with the identification and understanding of their relevance based on trends and patterns, is of paramount importance for supporting proper decision-making. In particular, analyzing and understanding changes related to multiple variables (e.g., attributes) over space and time is essential.

Information visualization has become a relevant discipline in Computer Science dedicated to the development of tools to improve the understanding of multiple and complex data. Information visualization has often been associated with transforming data into a visual form. In the context of spatiotemporal data analysis, a practical and intuitive visualization design is more demanding because of space- and time-related challenges. For example, multiscale data intricates relations of entities and their attributes across time and space.

The visualization and analysis of ST data are especially worthwhile in the study of ecology [1, 2, 3], biology [4, 5], transportation [6, 7], meteorology [8, 9], medicine [10, 11], etc. This work has been centered in the definition of information visualization algorithms to support mobility analysis for urban planning applications. That is partially motivated by needs and demands of Norwegian cities concerning the implementation of sustainable operations for mobility demands. Ålesund, Norway, for example, has become part of the United Nations Smart Cities program, which defines a set of Key Performance Indicators (KPI) to support decision-making. Some indicators are connected with the traffic and mobility conditions. In the present time, there are some groups of engaged research projects connected to this program. The Nordforsk NORDARK,[1] for example, aims to identify and characterize the impact of light infrastructure in green urban areas. Digital twins to support the intervention planning and visualization of collected data concerning light impact on outdoor environments, and human and animal behaviour are expected to be developed. Smart Plan[2] and TwinFjord[3] are other two related projects, both funded by the Research Council of Norway. These last projects concern the design and implementation of decision-making systems for public planning processes with new digital tools, and visualization methods associated with multidimensional data. Most of the collected data used in those projects are spatiotemporal. In all projects, novel visualization schemes are needed to support the professional analysis of these data, which should be not only be flexible on the mobility-related problem under investigation, but also effective and efficient.

The visualization of spatiotemporal data is one of the most promising and challenging topics in the field of information visualization. This type of data should reflect three distinct types of attributes; non-spatiotemporal, spatial, and temporal attributes simultaneously in a limited 2D display space.

Density map visualization is one approach often explored to generate areas around groups of points of data in the view [12]. It encodes the continuous distribution of scalar fields in a 2D space. These approaches promise to visualize the projected spatial patterns associated with various types of data. By dividing the data space into an arbitrary number of density fields and then visualizing them, it is possible to distinctly discern the patterns that are present within particular datasets [13, 14]. Density map visualizations have been successfully explored in several applications, especially in the context of urban data management. Those approaches, for example, support analyses related to traffic conditions [15], pollution distribution [16], and demographic evolution [17].

Despite the success of existing methods in encoding spatial distributions (e.g., using kernel-based methods [18, 14]), few initiatives have dedicated to the presentation of density maps taking into account topological information found in networks (e.g., road networks [17]). Feng et al. [17] introduced recently a promising approach for the computation of Topology Density

[1] `https://nordark.org/` (As of May 2022).

[2] `https://www.unitedfuturelab.no/en/projects/smart-plan----planning-through-visualization-and-simulation/` (As of May 2022).

[3] `https://www.twinfjord.no/` (As of May 2022).

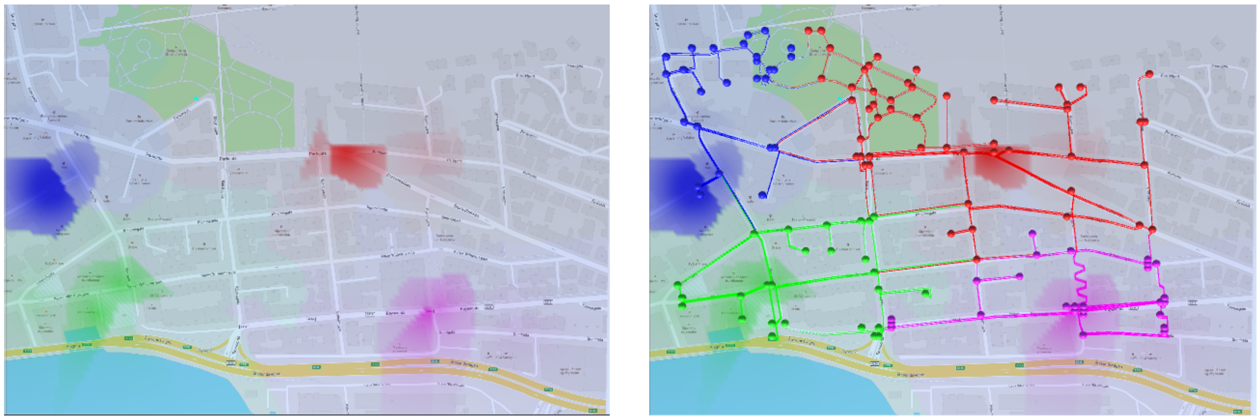


Figure 1.1: Examples of density maps. The left figure is a density map, while the right one is a Topology Density Map (TDM).

Map (TDM). In their method, the computation process extends the density estimation from a 1D network to a 2D space, providing *correct* and *intuitive* visualizations. The *correct* visualization means the accurate reflection of the directional and topological road network and dynamic path costs directly linked to the traffic conditions. The *intuitive* aspect refers to providing density fields on 2D planar fields intuitively. Figure 1.1 illustrates a density map (left) and a Topology Density Map (TDM – right) associated with the analysis of spatial data. The color intensity is proportional to the density for each position. The TDM result on the right side in the figure provides the 1D road network with colored taper lines, which greatly help users to visualize density variations along with road segments. The density intensity change is challenging to be represented on density map calculation only (left).

The algorithm of Feng et al. [17] is very promising and was validated in the context of compelling applications related to urban mobility analysis. Their solution, however, is not efficient to determine the density maps for regions not belonging to the input network. Another problem is the lack of support for representing changes of topology density maps over time. In fact, to support the proper analyses of urban data is not only relevant the spatial distribution of scalar field, but also the temporal variation. In several applications, understanding trends and patterns over time of spatial data are of a key element to support better-informed decision-making. To the best of our knowledge, the problem of encoding temporal changes visually and associated with topology density maps is still a problem overlooked in the literature.

One suitable alternative to address the first limitation relies on employing efficient algorithms for computing influence zones of network nodes. A promising alternative is the Image-Foresting Transform (IFT) [19]. IFT is a graph-based approach [19] to the design of image processing operators based on connectivity. It basically computes the Dijkstra's shortest-path algorithm based on the Euclidean distance, partitioning the input image according to a given seed pixel set. It can be seen as region growing algorithm based on seed pixels.

In the context of change pattern representation and characterization, several approaches have been proposed in the literature [20, 21, 22, 23]. In particular, Change Frequency Heatmap (CFH) [24] has established as a promising method to encode temporal changes according to application-oriented customized metrics and behavior patterns. It has been successfully employed in the analysis of multivariate temporal data associated with plant phenology data [24, 25], including both data obtained from direct observation of individuals [24] over time and variations defined by sequences of images [25].

This master thesis encompasses the investigation of information visualization approaches to support the understanding of temporal changes connected with topology density map. It connects with the raised issues concerning the use of TDM for spatiotemporal data analysis. The first issue is the low efficiency in the density intensity computation. We investigate the use of IFT to speedup the computation of TDMs. The second problem is the encoding and representation of temporal changes. In our research, we explore the use of CFH to encode temporal changes associated with TDMs and provide a visualization of the interesting change patterns.

1.2 Objective and Research Questions

The objective of this work is to design, implement, and validate *efficient* and *effective* methods for the computation of temporal changes in topology density maps. In this work, we explore and design a new algorithm based on three background concepts, which are Topology Density Map (TDM), Change Frequency Heatmap (CFH), and Image-Foresting Transform (IFT) (Chapter 2). Moreover, we introduce two prototypes, which are validated and evaluated in compelling usage scenarios with real data.

This work addresses four research questions, defined as follow:

- **RQ1:** How to compute Topology Density Map (TDM) using the Image-Foresting Transform (IFT)?

  The Topology Density Map (TDM) algorithm [17] calculates first the shortest path with the minimum cost from points-of-interest (POI) nodes to other nodes (non-POI) in a 1D road network. The network topology and edge costs can reflect the road information and traffic conditions. Second, the algorithm computes density field on each edge, which is the density estimation on the 1D road network. And last, the algorithm determines the shortest access time from each POI to an arbitrary point on the 2D planar surface and computes density fields. The computation of the access time from each POI to a point depends on the Euclidean distance calculation. This is a time-consuming step because it needs the iterated execution for each node and each position.

The image partition based on the calculation of the Euclidean distance with a given seed set is also one kind of region growing problem. Therefore, the IFT is selected to compute the propagation step of TDM. We choose pixels associated with non-POI nodes as the seeds. The IFT directly outputs the Euclidean distances between any arbitrary pixel to the belonging seed, which leads to a partition of the space. Each region can be seen as a propagation (influence zone) associated with one seed. The algorithm, therefore, computes the POI accessibility for any pixel. This strategy greatly saves time when the resolution of 2D space is higher, as the IFT reduces the calculation time for the TDM in the computation of density field in 2D planar surface. The details and analysis related to the use of IFT are described in Chapter 4.

- **RQ2:** How to encode and visualize temporal changes on Topology Density Map (TDM) using Change Frequency Heatmap (CFH)?

  To address this research question, we investigate different formulations of CFH for the encoding and representation of temporal changes associated with TDMs. We then introduce Temporal Topology Density Map (TTDM), a new algorithm that allows for encoding and visually representing changes on TDMs. TTDM integrates the efficiency of the IFT with the representation power of CFH. It is constructed with a stack of graphs including temporal changes of edge costs as the input. After the computation, it produces three outputs (the visualized 1D network, 2D planar texture color map, and a 3D mesh with height map) for the final visualization. Chapter 5 presents the whole solution pipeline with the theoretical basis, outlines the associated algorithm, and illustrates its execution in a running example.

- **RQ3:** Would the use of Image-Foresting Transform (IFT) lead to a more efficient computation of Topology Density Map (TDM)? To what extent is the new algorithm different from the TDM?

  To emphasize this research question, we explore the use of the IFT algorithm for computing density and label maps. Furthermore, we assess the computation time of its different components, considering various usage scenarios (topologies with diverse number of nodes and distinct density map resolutions). The differences of computed density map intensities with distinct parameters when compared with the outputs produced by the algorithm of Feng et al. [17] are also evaluated. The experimental data refer to random temporal changes applied to real road networks. The evaluated parameters include the resolution scale, the size of network, the number of POIs, and the length of time considered. The indicators refer to the differences among density maps produced by the proposed algorithm and the ones computed by the original algorithm [17].

- **RQ4:** Would the use of Temporal Topology Density Map (TTDM) be effective for analyzing changes over time in mobility-related applications?

  Walk network (small region, slower speed) and drive network (large region, faster speed) are two typical kinds of directed networks in mobility-related applications. The accessibility time for both are affected by the severity of weather conditions, such as the frequency of snow and rain. Those conditions are critical factors especially in Nordic countries. In this work, we validate TTDM in the context of compelling mobility-related urban analysis using real data of Ålesund, Norway. In the considered case studies, we assume that the snow intensity on the roads directly affects the moving speed of pedestrians and vehicles. Then, the access time related to the path from a POI to another geographical position is also correspondingly changed (defined in terms of the road distance divided by speed). Finally, it causes the temporal changes of density values in the target region. We analyze two case studies associated with different kinds of temporal changes in Section 6.3. Case Study 1 concerns the identification of the optimal location for public services considering a walk network. Case study 2, in turn, refers to the analysis of change patterns related to topology density maps computed around a drive network. The final results are discussed based on visual patterns associated with sample regions and alternative visualization designs.

Figure 1.2 indicates the whole research approach outline. It starts from three background concepts (TDM, CFH, and IFT) and two related research fields. Afterwards, our research explores the design, analysis, and evaluation of different algorithms for the *efficient* and *effective* computation of Temporal Topology Density Maps in the following three chapters. The figure outlines the relations among background concepts and research questions. In the end, we draw the final conclusions on the results.

1.3 Contributions

The contributions of this work can be summarized as follows:

1. It introduces a new algorithm, named IFT-based Topology Density Map (ITDM) – based on the Image-Foresting Transform (IFT) – for the efficient computation of Topology Density Map (TDM) .
2. It introduces a new algorithm, named Temporal Topology Density Map (TTDM), for the encoding of temporal changes associated with Topology Density Map (TDM).
3. It proposes a new method for the visualization of temporal changes associated with TDM.

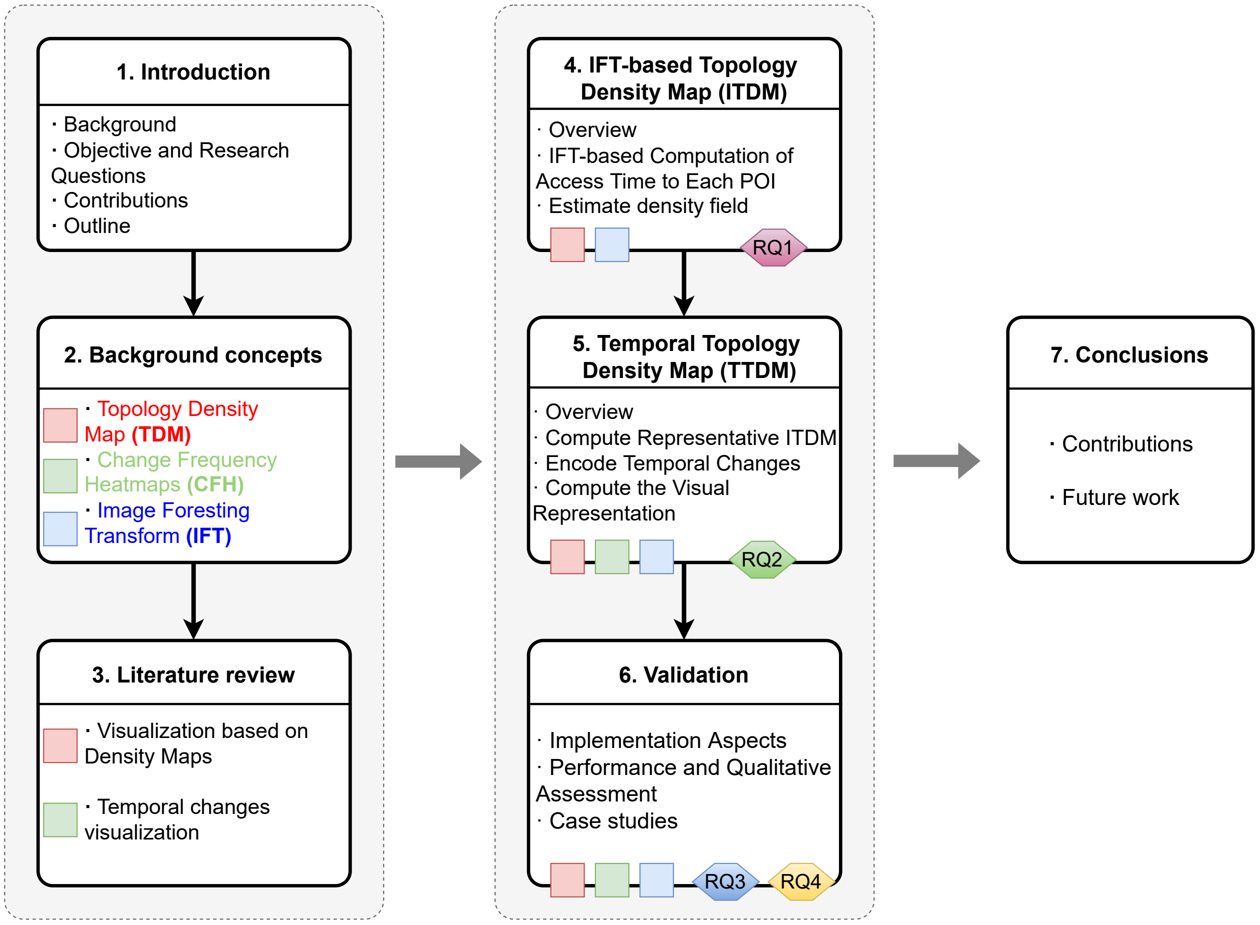

1. Introduction
· Background
· Objective and Research Questions
· Contributions
· Outline
2. Background concepts
· Topology Density Map (TDM)
· Change Frequency Heatmaps (CFH)
· Image Foresting Transform (IFT)
3. Literature review
· Visualization based on Density Maps
· Temporal changes visualization
4. IFT-based Topology Density Map (ITDM)
· Overview
· IFT-based Computation of Access Time to Each POI
· Estimate density field
RQ1
5. Temporal Topology Density Map (TTDM)
· Overview
· Compute Representative ITDM
· Encode Temporal Changes
· Compute the Visual Representation
RQ2
6. Validation
· Implementation Aspects
· Performance and Qualitative Assessment
· Case studies
RQ3
RQ4
7. Conclusions
· Contributions
· Future work

Figure 1.2: Research outline presenting the relationships among each chapter's content, background concepts (the colored squares) and research questions (the numbered hexagons). The figure is adapted from the ones presented in [26, 27].

4. It presents a software prototype, named TTDM computation analysis tool (Desktop), in Unity,[4] based on the devised algorithms, for supporting the analysis of temporal changes associated with Topology Density Maps. It supports the GeoJSON[5] and customized format file as input and output data. A web-end open-source prototype, referred to as TTDM visualization analysis tool (Web server)[6,7], is also designed for the user-friendly visualization and evaluation of different visual layouts.

5. It demonstrates the feasibility and usability of the proposed approach in the context of spatiotemporal (ST) urban data analysis related to the visualization of changes over time.

[4]https://unity.com/ (As of May 2022).
[5]https://geojson.org/ (As of May 2022).
[6]https://folk.ntnu.no/zhichenh/MasterThesis (As of May 2022).
[7]https://felando1984.github.io/WebTTDM (As of May 2022).

1.4 Outline

The remaining of this document is organized as follows:

- **Chapter 2** introduces relevant background concepts, such as Topology Density Map (TDM), Change Frequency Heatmap (CFH), and Image-Foresting Transform (IFT). The description of the background concepts also includes the discussion of running examples. These three concepts serve as the basis for the proposed algorithms.

- **Chapter 3** provides an overview of related work in two corresponding research areas considered in our work: visualization based on density maps and temporal change visualization.

- **Chapter 4** presents IFT-based Topology Density Map (ITDM), a new algorithm for Topology Density Map computation that combines TDM and IFT. This is the core part of the proposed algorithm for encoding and visually representing temporal changes.

- **Chapter 5** introduces a new algorithm, named Temporal Topology Density Map (TTDM), for encoding temporal changes associated with Topology Density Map (TDM).

- **Chapter 6** addresses validation aspects. It covers the evaluation protocol and the presentation and discussion of results related to the efficiency analysis of the proposed algorithms. The chapter also presents and discusses the use of the Temporal Topology Density Map (TTDM) and real usage scenarios with real data.

- **Chapter 7** summarizes the main conclusions and contributions of the research work and points out directions for future work.

Chapter 2

Background Concepts

This chapter presents background concepts, utilized in the proposed formulation for computing Temporal Topology Density Map. First, we introduce a recent formulation for computing Topology Density Map (Section 2.1). Next, we present the Change Frequency Heatmap and how this representation encodes temporal changes (Section 2.2). Finally, we introduce the Image-Foresting Transform algorithm (Section 2.3).

2.1 Topology Density Map (TDM)

Density map is an effective visualization method, which could present the continuous distribution of scalar fields in a 2D planar space by assigning a specific color to each scalar vertex. It is widely used in many applications of urban analysis, such as traffic conditions analyses [15], demographic evolution detection [17] and air pollution distribution assessment [16]. Recently, Feng et al. [17] introduced a new method for computing density maps, named Topology Density Map (TDM). Their goal was to create more *correct* and *intuitive* density maps in the context of urban data visualizations. Their work explored the use of a directed acyclic graph (DAG) to propagate nonlinear scalar fields along 1D road networks. Next, their formulation extends the density calculation from the scalar fields to 2D planar surface by identifying POI nodes and computing density scalar fields for all points in this 2D space.

Figure 2.1 illustrates the pipeline, composed of six modules, for computing a TDM. The figure also provides our simplified presentation and the relationship between them. Our formulation of the TDM computation was designed for easy understanding and linking with our new algorithm by using similar terms. It comprises three modules: encode network data, compute accessibility data, and surface mapping. The first module is responsible for encoding the network data, graph $G = (V, E)$, which includes the node set V and directional edges set E. The node set $V = \{v_1, v_2, \ldots, v_i, \ldots, v_{nodeNum}\}$, where v_i is a vertex and *nodeNum* is the number of nodes in V. To each node, attributes, such as name, location, or if it is a POI, are assigned. The edge set $E = \{e_1, e_2, \ldots, e_i, \ldots, e_{edgeNum}\}$, where $e_j = (v_x, v_y)$ is a ordered pair of vertices ($(v_x, v_y) \in V^2$, and $v_x \neq v_y$) and *edgeNum* is the number of edges in E. A weight may be associated with the edge (v_x, v_y), representing the *cost* from v_x to v_y, such as time, distance, or other customized scalar. In this research, the cost is typically the access time rather than the physical distance (e.g., Euclidean distance). Moreover, the terms network, graph and topology will be used interchangeably in our research.

The output of the first module is a wrapped structured DAGs data (G_{DAG}), computed from different POIs. Implemented using the Dijkstra shortest path algorithm, the second module is responsible for calculating a set of shortest path costs between every non-POI node and POI nodes based on G_{DAG}. Then, it constructs the accessibility data G_{cost}, which includes all accessibility data information for each non-POI node (the shortest path cost from the nearest POI as well as the POI name). The accessibility data G_{cost} is utilized as the input of the last module, surface mapping, which is responsible for computing (G_{TDM}), the estimated density field values for a 2D planar surface.

The following density estimation Equation 2.1 [17] is used to calculate the density field value for any arbitrary point P on the 2D planar surface:

$$\lambda(P|POI_{nr}, G) = \frac{1}{r} F_k\left(\frac{(F_c(POI_{nr}, N_{nr}) + F_c(N_{nr}, P))^{-\alpha}}{r}\right) \tag{2.1}$$

where λ is the density estimation, P is any point on the 2D map, POI_{nr} is the nearest POI node, G is the road graph which is a tuple composed of nodes and edges with specific costs, N_{nr} is the nearest node to point P, α is the accessibility coefficient. According to the literature, the value of α is usually between 0.9 and 2.29 [28]. F_K is the kernel function with a kernel radius r. $F_c(v_x, v_y)$ is the cost function of two vertices v_x and v_y. If both vertices are input nodes, F_c is the path cost, which is the sum of the weights (cost) in the edges in the path from v_x to v_y. When v_y is an arbitrary vertex like P, the cost function F_c is calculated by Equation 2.2.

$$F_c(N_{nr}, P) = \frac{d(N_{nr}, P)}{s_{avg}} \tag{2.2}$$

where d is the distance between point P and the closest non-POI node N_{nr}, s_{avg} is the average speed for the path between N_{nr} and P.

In [17], the Gaussian ($F_{k-Gaussian}$) and Sigmoid ($F_{k-Sigmoid}$) kernels are considered in the im-

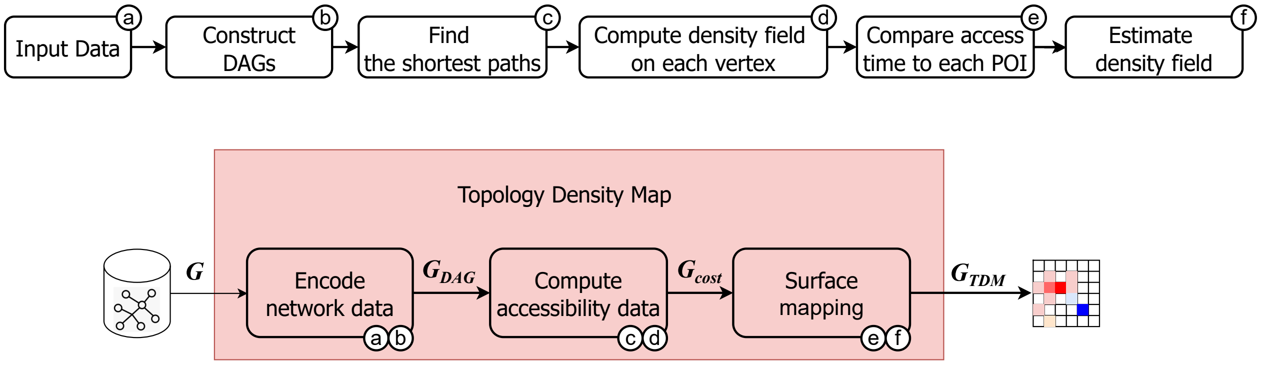


Figure 2.1: Topology Density Map (TDM) algorithm pipeline. The six white background modules emphasized as ⓐ - ⓕ are original definition for TDM [17]. The three red background steps are our simplified formulation of TDM, marked with the related defined modules in the top-right corner.

plementation of $F_k(x)$, where x is the input value. Those kernels are defined as follows:

$$F_{k-Gaussian}(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}x^2} \tag{2.3}$$

$$F_{k-Sigmoid}(x) = \frac{1}{1+e^x} \tag{2.4}$$

The main novelty of the formulation of Feng et al. [17] relies on the extension of 1D-limited network discrete information to a 2D-continuous space. It greatly improves the insight depth in spatial aspect. The approach was validated in two relevant applications related to the analyses of road topology and traffic conditions on accessibility and to the identification of optimal location of new public facilities.

Figure 2.2 provides an example related to the computation of a TDM. The input network data includes six nodes. We assume that $H1$ and $H2$ are POIs and the other four nodes (A, B, C, D) are non-POIs. After the encoding of network data, the graph G will be wrapped up to two DAGs for POIs $H1$ and $H2$. The cost F_c is calculated from the POI to non-POI nodes and finally to any point in the 2D planar space. The density field is the complete 2D planar surface for this network. In this figure, the color of the nodes and the tapered edges reflect the accessibility data G_{cost}. The visualization of G_{TDM} concerns one density estimation field with colors that represent the propagation of the density values along the edges and the 2D planar surface. For example, the blue region with the nodes $H1$, A, B has more variations in the intensities of density fields when compared to the red region with the nodes D, C, $H2$.

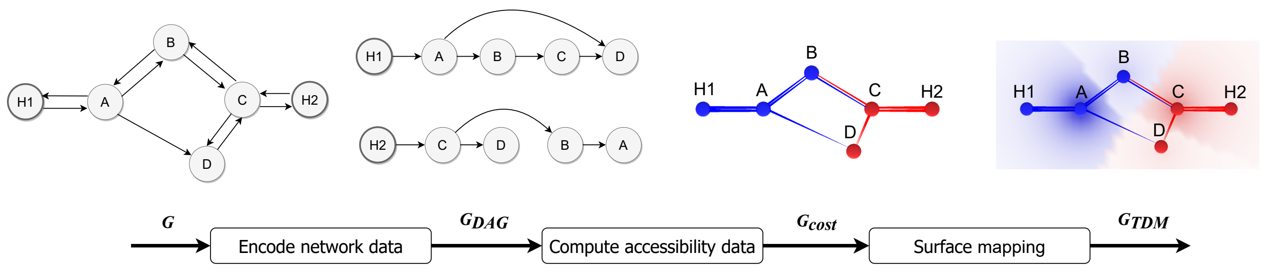


Figure 2.2: A running example for TDM. The network data G includes two POI nodes $(H1, H2)$ and four non-POI nodes (A, B, C, D). For each POI node, G_{DAG} is one DAG connecting non-POI nodes from it. G_{cost} is the accessibility data, which include the path cost and related POI label. It can be directly used for the G_{TDM} for the final visualization.

2.2 Change Frequency Heatmap (CFH)

Mariano et al. [24] presented a novel image-based representation, named Change Frequency Heatmap (CFH), which encodes the frequency of occurrence of temporal patterns associated with multivariate numerical data.

The CFH computation algorithm comprises three steps as illustrated in Figure 2.3: encode temporal data, compute temporal binary pattern, and compute histogram. The first step is responsible for encoding the temporal multivariate data $\mathscr{S} = \{X_1, X_2, ..., X_n\}$, which is a set with n elements $X_i =< x_{i,1}, x_{i,1}, \dots, x_{i,m} >$ with m dimensions. The output is a stack of matrices $\mathscr{M}$, where $\mathscr{M} =< M_1, M_2, \dots, M_T >$, and M_t is an $n \times m$ numerical matrix at timestamp $t \in [1, T]$ composed of n lines and m columns. Any element m_t of the matrix M_t represents the pattern information at the corresponding position in the space $S(n, m)$ and timestamp t. The space $S(n, m) =< (1, 1), (1, 2), \dots, (1, m), \dots, (n, 1), (n, 2), \dots, (n, m) >$ represents the discretization grid of a continuous 2D space. The second module computes temporal binary patterns by employing customized functions that encode temporal change profiles associated with the different matrix cells. This step results in a new stack of matrices $\mathscr{D}$, where $\mathscr{D} =< D_1, D_2, \dots, D_{T-1} >$ and and D_t is an $n \times m$ numerical matrix at timestamp $t \in$[1,T-1] composed of n lines and m columns. Therefore, any cell d_t in D_t is the same position as m_t in M_t. Equation 2.5 defines one example of encoding function, where number 1 means that there was a change across consecutive neighbor timestamps, while number 0 means that no temporal changes occurred.

$$d_t = \begin{cases} 1, & m_t \neq m_{t+1} \\ 0, & \text{otherwise} \end{cases} \tag{2.5}$$

During the last step, patterns of interest p in the d_t sequences of numbers '0's or '1's (e.g., $p_0 = 0$, $p_1 = 010$, $p_2 = 0110$, and $p_3 = 01110$) are utilized. The change presentation $d = d_1 d_2 d_3 ... d_t ... d_{T-1}$

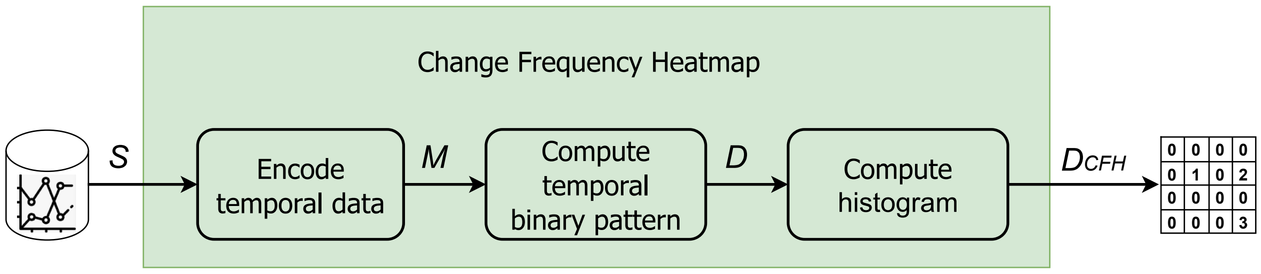


Figure 2.3: Change Frequency Heatmap (CFH) algorithm pipeline.

where $\omega_{x,y}$ is a sub part of d, x and y are the indexes of staring and ending timestamp. For example, $\omega_{2,4} = d_2 d_3 d_4$. If $l = length(p)$, $y = l - 1 + x$.

CFH_h counts the number of occurrences of the pattern of interest (binary pattern) in any element h of the matrix $\mathscr{S}_{CFH}$, as defined in Equation 2.6:

$$CFH_h = \sum_{x=1}^{T-l} \mathbb{1}\{\omega_{x,y} = p \mid \omega_{x,y} \in \Omega\{d\}\} \tag{2.6}$$

where $1 \le x < y \le T - 1$, $y = l - 1 + x$, and $\Omega\{d\}$ is the set of all sub parts of d .

We could consider that CFH_h is a pure matrix calculation algorithm to encode a pixel change pattern of a stack of matrices and output a binary motion histogram matrix. This method, therefore, provides a way to analyze temporal changes across sequences of matrices (e.g., images). As CFH_h encodes the relevant pattern changes in a matrix (grid), a heatmap is often utilized for their visualization.

Figure 2.4 introduces one running example in a 2D space. The values here have no actual meaning. For real applications, it could be any interested indicators like source POI label value, the cost value, etc. In the figure, the temporal data $\mathscr{S}$ includes a 4×4 discretization grid for a 2D space. The metric function calculates the change pattern (Equation 2.5) for neighbor matrices Ms and outputs into a new stack of matrices $\mathscr{D}$. For example, D_1 is computed based on M_1 and M_2. The pixel value of the final result matrix $\mathscr{D}_{CFH}$ how many times the pattern occurs. For instance, $\mathscr{D}_{CFH}(0)$ has the number of '0' for each pixel and $\mathscr{D}_{CFH}(010)$ counts the pattern change '010.'

2.3 Image-Foresting Transform (IFT)

Image-Foresting Transform (IFT) is a graph-based approach to the design of image processing operations. The considerable efficiency has been proved and its use has been demonstrated in several image analysis techniques and applications [19]. IFT solves image partition problems

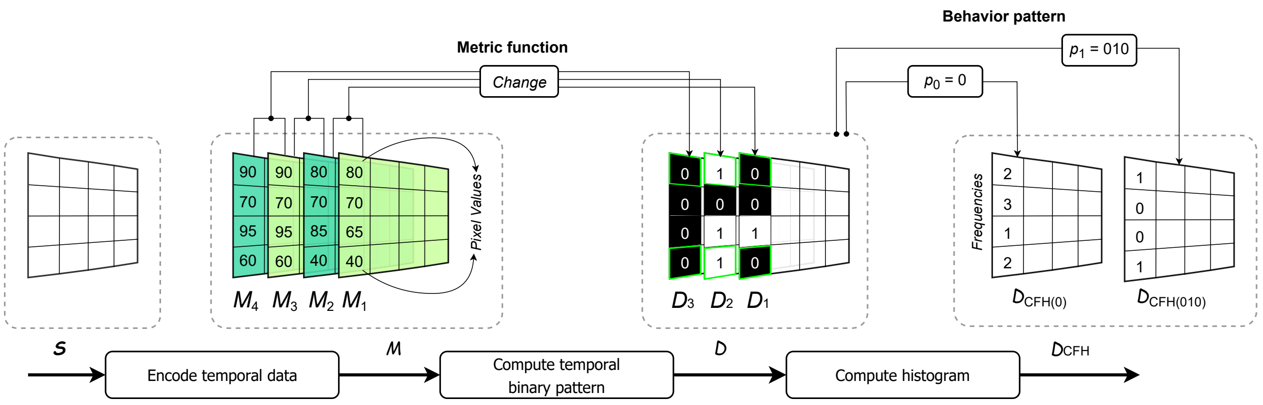


Figure 2.4: A running example for CFH. The temporal data $\mathscr{S}$ includes a 4×4 discretization grid for one limited 2D space. The metric function calculates the change pattern as Equation 2.5 for neighbor matrices Ms and outputs into a new stack of matrices $\mathscr{D}$. For example, D_1 is computed on M_1 and M_2. The pixel value of the final result matrix $\mathscr{D}_{CFH}$ encodes how many times a pattern of interest occurs. For instance, $\mathscr{D}_{CFH}(0)$ has the number of '0' for each pixel and $\mathscr{D}_{CFH}(010)$ counts the pattern change '010.' The figure drawing is adapted from [25].

in a graph-based formulation, in which an image is seen as a graph where pixels are vertices and edges are defined based on relations among pixels. It receives a seed set S as input and produces as output a minimum-cost path forest. For pixel-based image partition, the graph is often defined based on an Euclidean adjacency relation A. Two pixels (vertices) are neighbour if their Euclidean distance is below a threshold. The cost of a path depends on the local image properties along the path as color, gradient, or pixel position [29].

In our research, the IFT is modeled to compute an Euclidean Distance Transform that provides a label propagation model, as illustrated in Figure 2.5. The input is an image I with the seed set pixels. The output is a forest F with three images: the predecessor map P, the cost map C, and the root map R. For each pixel, the value of the predecessor map P represents the predecessor pixel index in the optimum path connecting the nearest seed to it. The value of the cost map C is the cost of the optimum path. The intensities found in the root map R encode the nearest seed index.

The IFT is computed according to Algorithm 1 below, which refers to an Euclidean Distance Transform [19, 29, 30]. The algorithm starts by setting zero to the cost intensity for the seed pixels and the infinity for the rest (Lines 2-3). Then, it adds all the seeds into the priority queue Q and marks the seeds as the roots with the index value -1. That means that there is no predecessor for these pixels. The cost of the seeds are 0 because there is no distance difference for themselves. Next, the algorithm iterates over all pixels in Q, from the ones with the lowest cost to the ones with the largest (Lines 4-16). For a target pixel p, the algorithm computes the distance between the root pixel $R(p)$ and any arbitrary pixel q in the neighborhood of p (defined

Algorithm 1: IFT algorithm

```
1  Auxiliary Data structures: A priority queue $Q$.
   Data: An image $I$, a set $S$ of seed pixels in $I$, and an Euclidean adjacency relation $A$.
   Result: A forest $F$ with root map $R$, the corresponding cost map $C$, and predecessor
           map P.
2  For all pixels $p$ of image $I$, set $C(p) \leftarrow +\infty$;
3  For all $p \in S$, set $P(p) \leftarrow nil$, $R(p) \leftarrow p$, $C(p) \leftarrow 0$, and insert $p$ in $Q$;
4  while $Q$ is not empty do
5      Remove from $Q$ a pixel $p = (x_p, y_p)$ such that $C(p)$ is minimum;
6      foreach pixel $q = (x_q, y_q)$ such that $q \in A(p)$ and $C(q) > C(p)$ do
7          Set $C' \leftarrow (x_q - x_{R(p)})^2 + (y_q - y_{R(p)})^2$, where $R(p) = (x_{R(p)}, y_{R(p)})$ is root pixel of $p$;
8          if $C' < C(q)$ then
9              if $C(q) \neq +\infty$ then
10                 Remove $q$ from $Q$;
11             end
12             Set $P(q) \leftarrow p$, $C(q) \leftarrow C'$, $R(q) \leftarrow R(p)$;
13             Insert $q$ in $Q$;
14         end
15     end
16 end
```

by an Euclidean adjacency relation A – Line 7). The Euclidean adjacency relation A is often implemented using the 8-connected adjacency, which means the closest pixels around a pixel of interest from 8 directions. When the distance is shorter, the predecessor index, the cost value, and the seed index of this pixel q are updated (Line 12). Finally, the algorithm propagates the cost intensity by inserting the neighbour pixels to the priority queue (Line 13). With this algorithm, a label propagation is performed.

Falcao et al. [19] demonstrated that with the use of a proper data structure in the implementation of the priority queue Q, the computational complexity of the IFT algorithm is $O(n)$, where n is number of pixels in the input image I.

One 5×5 image with two seeds is the input of the running example shown in Figure 2.5. The blue seed (index 6) and the red seed (index 18) are marked as pixel values 1 while the other non-seed pixels have value 0. After the calculation, the root map R has the corresponding blue and red regions. The cost map C has the square of the Euclidean distance along the path from each pixel to the belonging seed. The predecessor map reflects the path with the connectivity among the pixels. Figure 2.6 provides the visualization of this example. The blue and red dots in the figure (on the right) are two seeds. The forest is illustrated with the predecessor map by the arrows, the cost map by numbers, and the root map by the seed colors (on the left).

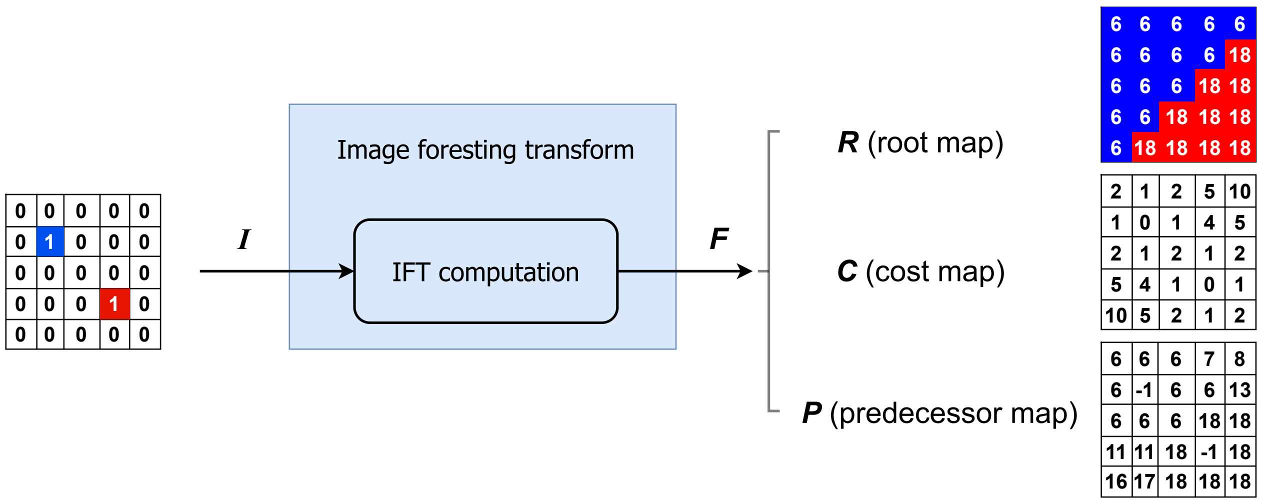


Figure 2.5: Image-Foresting Transform (IFT) algorithm pipeline.

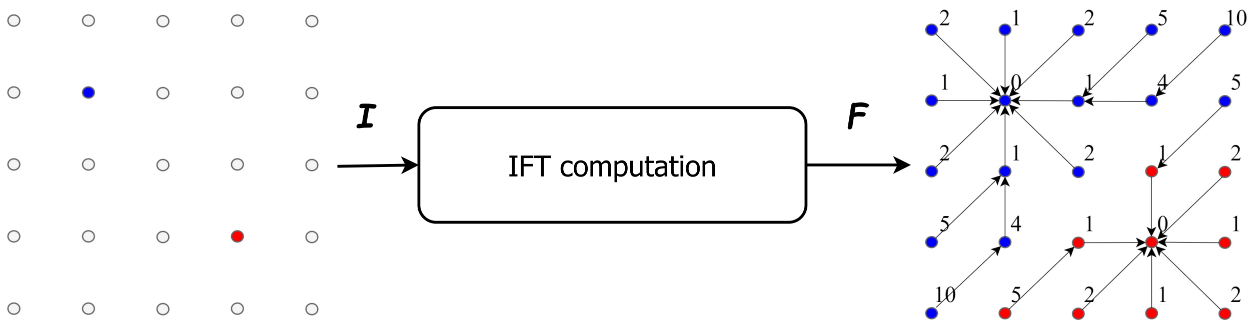


Figure 2.6: A running example for IFT. The two seeds (blue and red) in the image lead to a partition of the space into two regions. The result also includes information about cost and the predecessor.

Chapter 3

Literature Review

This chapter provides an overview of related work connected with our research. It includes visualization based on density maps and temporal change visualization. The concepts Topology Density Map (TDM) and Change Frequency Heatmap (CFH) mentioned have been introduced in Chapter 2.

3.1 Visualization based on Density Maps

Hogräfer et al. [12] provided an overview of map-based visualization approaches. According to them, density-based field schematization is an effective technique for aggregating representatives for local regions and displaying continuous scalar field data in one 2D space with map rendering. Density map visualization is one promising approach among density-based filed schematization to construct the visual coding of field data by density value, emphasizing geographic accuracy over visualization.

Due to the effectiveness in supporting space-oriented assessments, density maps have been successfully applied to urban analysis, especially concerning services of transporting people and goods, etc. Danese et al. [31] explored the application of density maps for the seismic risk analysis to improve the civil protection planning. Xie et al. [15] presented a new density estimation method for traffic accidents and validated the visual effects with real datasets. Scheepens et al. [13, 14] proposed a novel framework of using density maps to visualize multiple attributes by multivariate trajectories and validated it in vessel traffic conditions analyses. More recently, Delso et al. [32] evaluated the density map application for the impact of the obstacles for the pedestrian walkability. Ren et al. [16], in turn, designed an interactive visual analytic system on density map technique to demonstrate the effectiveness in air pollution distribution assessment.

An effective density map computation method reflects the attribute value distribution by the density intensity. Because of the data observation limited on the nodes and edges of the topological network, the computation of density values for each position in a 2D space is the most critical process of density map visualization. This process is called density estimation. Various methods for density estimation have been proposed. For example, Borruso [33] validated Quatic Kernel Density Estimation (KDE) method in urban immigrant population from the spatial distribution and tendency aspect. Krisp et al. [34] presented an adaptive directed KDE (AD-KDE) to recognize the underlying dynamics of the vehicles for traffic condition analysis. Nie et al. [35] designed a kernel density method called NKDE-GLINCS, which integrates Network-constrained Kernel Density Estimation (NKDE) and Network-constrained Getis-Ord Gi* (GLINCS), to detect the road segment anormal status. Yuan et al. [36] proposed a new Quad-tree-based Fast and Adaptive KDE (QFA-KDE) algorithm to compute the aggregation patterns more efficiently.

In the context of urban analysis, these density estimation methods may be distinguished in two categories. One type, referred as planar KDE refers to the estimation of the density value based on the Euclidean distance between positions without any constrained network. The typical applications includes the trail analysis of vessels [13, 14], trains [37] and flights [38]. The other type, named as NKDE, emphasizes the density estimation along a constrained network. It is widely adapted for the road-constrained events analysis like traffic conditions, etc.

Topology Density Map (TDM) [17] (described in Section 2.1) proposed by Feng et al., is a novel method integrating the advantages of planar KDE and NKDE. It utilized real traffic datasets and road network to validate its effectiveness of density estimation in 2D space as well as 1D network. Compared with other research and existing methods [39, 40, 41] regarding spatiotemporal data analysis, TDM provides an intuitive visualization for supporting decision making.

In our research, we improve the computation efficiency of TDM when estimating the density field from 1D network to 2D space. Furthermore, we explore a new visualization method to encoding and visualizing temporal changes results of TDMs. Those algorithms are validated in the context of mobility applications. Both formulations aim to leverage the added-value of using density maps by decision makers during the exploration of ST datasets.

3.2 Temporal Change Visualization

In the spatiotemporal context, temporal changes represent the attribute value changes over time, while spatial changes related to variations observed in distinct locations [42]. Recently, Fang et al. [43] presented a comprehensive survey of methods for time series data visualization. Their work divided most of the temporal change visualization methods into two categories. One is the visualization of time attributes, such as Spiral diagram, Calendar view, ThemeRiver view, Dynamic visualization, etc. The other is the visualization of high-dimensional time series data, which involves Parallel coordinate methods, ThemeRiver methods, etc. However, all of them focus on the visualization of the attributes change over time, not detecting the change patterns for multiple attributes simultaneously.

With the increasing collection of urban context data, it is challenging to analyze the temporal aspects associated with spatial data. One promising direction is to analyze and visualize the temporal changes in an explicit way. Krukowicz et al. [44] provided a comprehensive analysis of animal-vehicle road crashes by the temporal analysis with Calendar view and KDE of the accidents number. Liang et al. [45], in turn, explored temporal changes of population in a geographical area using an eigen decomposition method. Ziwen et al. [46] focused their work on the spatial and temporal patterns analysis for tourist source market using the travel demand index pattern evolution method. Many other methods have been proposed to represent temporal changes in the literature [20, 21, 22, 23], but their applications are limited to specific scenarios and applications, i.e., they are not generic enough to be tailored to other applications.

In the context of urban data, Zheng et al. [47] summarized the existing visualization techniques from temporal, spatial, and other aspects. One of their focus was the data exploration and pattern interpretation, which are highly relevant for decision making. According to them, for the multiple attribute visualization, pixel-based techniques interpreted by a matrix form is one of the most popular techniques. However, these techniques have pose challenges when their use requires filtering some patterns of interest explicitly for all the pixels simultaneously.

On the other hand, popular density estimation methods, such as KDE, have been extended to handle the temporal dimension. The spatiotemporal kernel density estimation (STKDE) method is an example [48]. A predictive hotspot mapping to represent the risk factor by considering the temporal dimension is a sample application of this type of density estimations [49]. As far as we know, none of them is suitable to visualize the temporal changes for TDMs.

To encode the temporal changes of multivariate attributes, Mariano et al. [24] have proposed a promising method Change Frequency Heatmap (CFH) – see Section 2.2. It may utilize ST datasets as a stack of matrices to produce a matrix of recording the occurrences of the interested change patterns for each position and has been validated in plant phenology analysis. It has not been utilized in the visualization of time-related urban data. In this work, we investigate its use in combination with topology density maps. The main motivation relies on the fact that the computation of CFH is essentially a matrix-based operation.

One of the most popular methods used for temporal representation is Space-Time Cube (STC), originally proposed by Hägerstrand [50] and represented by Kraak [51]. STC is a descriptive way for temporal data visualization using a 3D cube, whose one dimension represents time. This visualization strategy has been validated on many kinds of datasets, especially in urban contexts [6, 11, 39, 40, 41]. Compared with CFH heatmap, this visual effect provides one dimension for the temporal attribute. However, it is more suitable for the sparse distribution network or nodes, but messy for all the pixels on the geographic map.

Furthermore, Bach et. al [52, 53] presented all possibilities to transform this 3D STC cube into a 2D plane for the visualization. This means all temporal data visualizations method can be mapping to one kind of transformation. For instance, time flattening, collapsing the space-time cube along its time axis, represents merging all time slices into a single 2D image. It could be a effective visualization approach for temporal changes of spatial data. This inspires us to transform the temporal changes into a novel visualization.

Our research work aims to integrate CFH to encode temporal changes of the results of TDM with application-oriented customized metrics and behavior patterns. The traditional visualization of CFH is the heatmap, we explored to reflect the CFH result in the 3D map view with density maps. It is easier to understand trends and temporal pattern from a geographic location in urban analysis instead of an individual in plant phenology.

Chapter 4

IFT-based Topology Density Map (ITDM)

This chapter introduces IFT-based Topology Density Map (ITDM), a new method based on the Image-Foresting Transform (IFT) to compute Topology Density Map.

4.1 Overview

The ITDM computation follows the same pipeline adopted by Feng et al. [17] to compute a Topology Density Map (TDM), but for steps ⓔ and ⓕ (highlighted in Figure 4.1). In our formulation for step ⓔ, the IFT algorithm is used to compute the Euclidean Distance Transform that partitions the 2D space according the proximity of points to vertices. Next, information about the partition (label map produced by IFT) and proximity (cost map) is used to estimate the final density map (step ⓕ). Figure 4.2 illustrates the visual result by computing TDM using the same example presented Section 2.1. The performance analysis of this algorithm is conducted in Section 6.2.

This chapter addresses research question 1, which refers to how to use IFT to compute TDM.

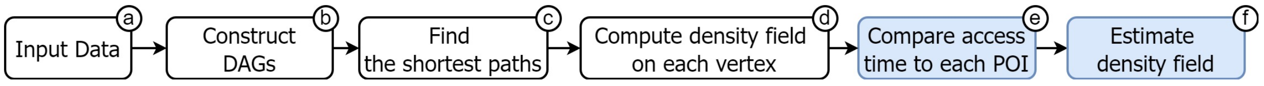


Figure 4.1: ITDM computation pipeline. It is based on modifications (highlighted in blue) on the TDM computation pipeline proposed by Feng et al. [17] (described in Section 2.1).

In the following, the proposed formulations for computing steps ⓔ and ⓕ are detailed.

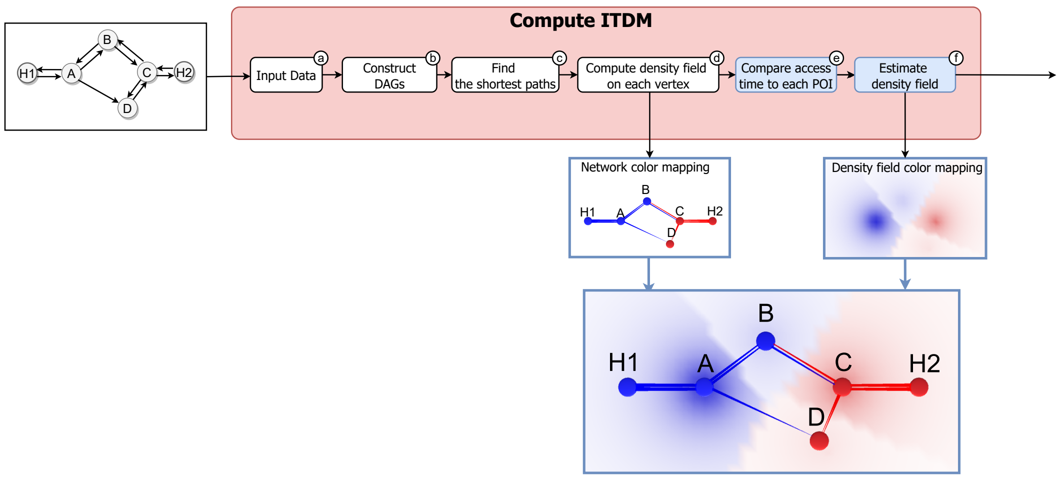


Figure 4.2: ITDM visual effects for an ITDM running example. The network data G includes two POI nodes ($H1$ and $H2$) and four non-POI nodes (A, B, C, and D).

4.2 IFT-based Computation of Access Time to Each POI

Figure 4.3 presents the main steps associated with the computation of the access time of any point in a 2D space to POIs. Three steps are considered: vertex mapping, IFT computation, and access time computation. Vertex mapping refers to encoding the input graph into an image that will be used as input of the IFT algorithm, i.e., it converts the graph to an image. In this step, the nodes in the graph are mapped to seed pixels in the image. Section 4.2.1 provides more details about the mapping process. As presented in Section 2.1, the accessibility data include the shortest path cost from the nearest POI as well as the POI label. It is the input of computing access time to each POI (step ⓔ). After the completion on 1D road network, the TDM algorithm compares the access time from each POI to any arbitrary point P, which is defined according to the Euclidean distance of the straight connection between P and the nodes. In this chapter, we propose a formulation to compute this Euclidean distance and find the closest node for each point P on the 2D space by using the IFT algorithm. Section 4.2.2 presents how this formulation works. The IFT computation produces a root map and a distance map that are used for the access time computation. This step is described in Section 4.2.3.

4.2.1 Vertex Mapping

The IFT algorithm maps an image to a forest based on the propagation of the influence zones defined by different seeds. The vertex mapping step concerns the creation of the IFT input image given the graph associated with the density field on each vertex. Let $G = (V, E)$ be the this graph, where V is a set of nodes and E is a set of edges. Recall that each node $v \in V$ has a position in the 2D space (x_v, y_v). Let I be an image with dimensions $W \times H$.

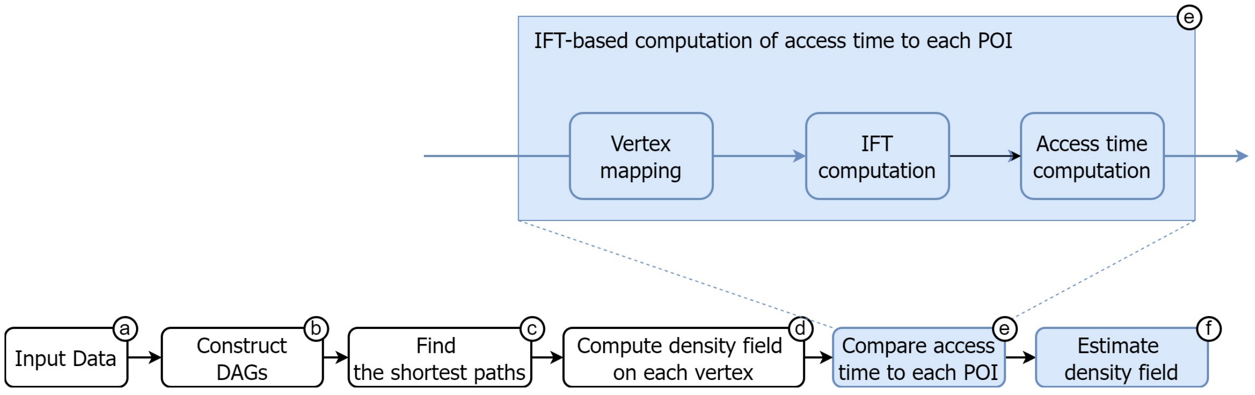


Figure 4.3: Pipeline for the IFT-based computation of access time to POIs.

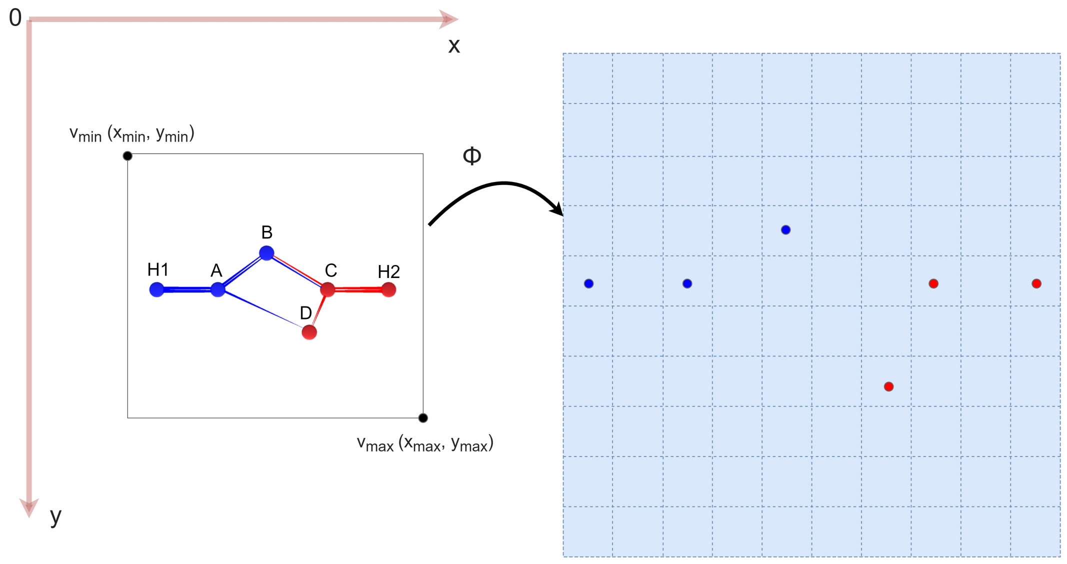


Figure 4.4: Illustration of the vertex mapping step.

Figure 4.4 illustrates the use of a mapping function ϕ to map vertices to an image. The resolution of the output image is defined according to the minimum distance among two vertices. The goal is to map vertices to different pixels, i.e., for each node vertex $v \in V$ there will be a pixel $p \in I$. Each pixel p associated with a node vertex v is taken as a seed of the seed set S used in the IFT algorithm.

The vertices $v_{min}(x_{min}, y_{min})$ and $v_{max}(x_{max}, y_{max})$ are two corners in our targeted 2D space with marked coordinate system. The point p is calculated by Equation 4.1 with the given v. It computes the scale parameter K_{px} first, which is critical to keep the same scale for x and y axis to avoid the distortion during the calculation. Next, we use K_{px} to calculate x and y value based on the difference between v and w_{min}, and the floor function.

$$p = \Phi(v) = (\lfloor \frac{x_v - x_{min}}{K_{px}} \rfloor, \lfloor \frac{y_v - y_{min}}{K_{px}} \rfloor) \tag{4.1}$$

where

$$K_{px} = \frac{x_{max} - x_{min}}{W-1} = \frac{y_{max} - y_{min}}{H-1} \tag{4.2}$$

There are also non-seed pixels p in the image I. For any pixel p with position (x_p, y_p) in the image by IFT, there exist one corresponding point P in the 2D space by the TDM algorithm. For each Point P, there is a vertex v_P, computed by Equation 4.3.

$$v_P = \Phi^{-1}(P) = (x_p \times K_{px} + x_{min}, y_p \times K_{px} + y_{min}) \tag{4.3}$$

Two factors are worthy to be handled with care. The first is the distance concept difference between TDM and IFT. The $d(v_i, v_j)$ is to calculate the distance between two vertices v_i and v_j on the 2D planar surface while $d(p_i, p_j)$ is the Euclidean distance between two pixels p_i and p_j for the image. The $d(v_i, v_j)$ is the geographical distance and it could be scaled from the Euclidean distance $d(p_i, p_j)$ directly.

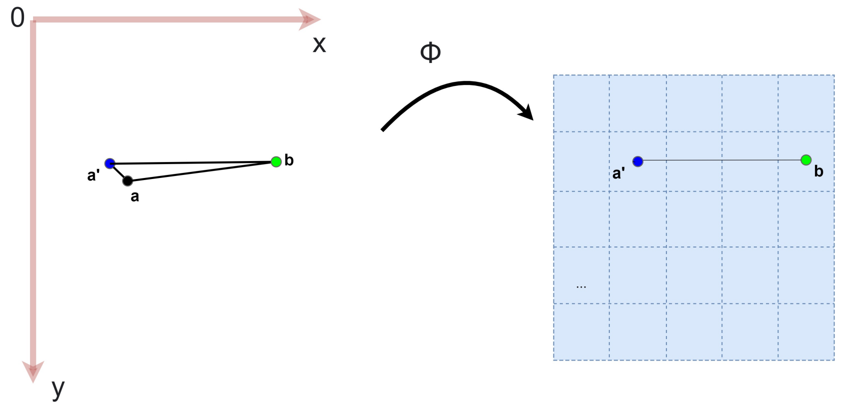


Figure 4.5: The approximate position calculation for the nodes.

The other factor is the approximate position calculation for the nodes. As illustrated in Figure 4.5, there is one vertex a (black) that represents a node in the 2D space. It may be mapped to a close position in the discretized space like a' (blue) in the vertex mapping procedure. Vertex b is not a node. However, the approximate distance $d(a', b) \neq d(a, b)$, which is the real one. When $d(a', b) \gg d(a', a)$, the distance $d(a', a)$ could be ignored. For the nearby vertices of a, the distance $d(a', b)$ is close to $d(a', a)$ and the error is larger. There are two ways to overcome this limitation. The first one is to increase the resolution and decrease the distance $d(a', a)$. The other method is to change the IFT algorithm. For instance, we could map a position to the image by Equation 4.1 without the floor function. This pixel position value could be float number to replace the integer, which is the approximated pixel position. This pixel can not be shown in the real image, but it is feasible to use this float position in the execution of the IFT algorithm. Then, this solution would eliminate the distance $d(a', a)$. In our research, we use the first strategy.

4.2.2 IFT Computation

This step concerns the execution of the IFT algorithm using the image I and the seed set S defined in the vertex mapping step. The IFT will partition the image into regions defined according to the Euclidean distance of pixels to the seeds. The IFT will produce a root map, a cost map, and a predecessor map.

As Section 4.2.1 discussed, the seeds in IFT are the nodes in TDM. The definition of seed pixels in the input image relies on assigning 1 as their values. With regard to the three IFT outputs, the predecessor map is not necessary for TDM. From now on, we will not refer to this map. The root map R and the cost map C are kept for the access time computation.

Figure 4.6 illustrates the computation of the IFT algorithm for the input graph presented on the right. The grey-scale images in this Figure are PGM P5 format [1]. The known non-POI nodes A, B, C, and D are taken as the seed pixels (marked white color) in the image (background color) in the first step vertex mapping. The discretization of 2D space are 145 × 109 pixels, which is also the size of input image I. After the IFT call, the root map, the color map, the predecessor map are created. For example, the root map R reflects four regions (influence zones) belonging to the four nodes given and the color map C shows the Euclidean distance values increase from the center because the nodes are in the center region. R and C are used to calculate the access time later.

4.2.3 Access Time Computation

Before computing access time, we need to determine how to estimate a density field. For that, we use Equation 2.1 in step ⓕ. Equations 4.4 and 4.5 below define how this density map is computed based on a simple variable substitution of Equations 2.1 and 2.2. In the equations, $F_c(POI_{nr}, P)$ is the cost from POI_{nr} to any arbitrary point P on the 2D planar surface. In this research, the cost represents the access time. Therefore, $F_c(POI_{nr}, P)$ is the **access time** value we need compute. $d(N_{nr}, P)$ is the distance and s_{avg} is the average speed for the direct connection between N_{nr} and P. The other parameters were introduced in Section 2.1 and are not emphasized here since they have no impact the the use of IFT to compute topology density maps.

[1] http://netpbm.sourceforge.net/doc/pgm.html (As of May 2022).

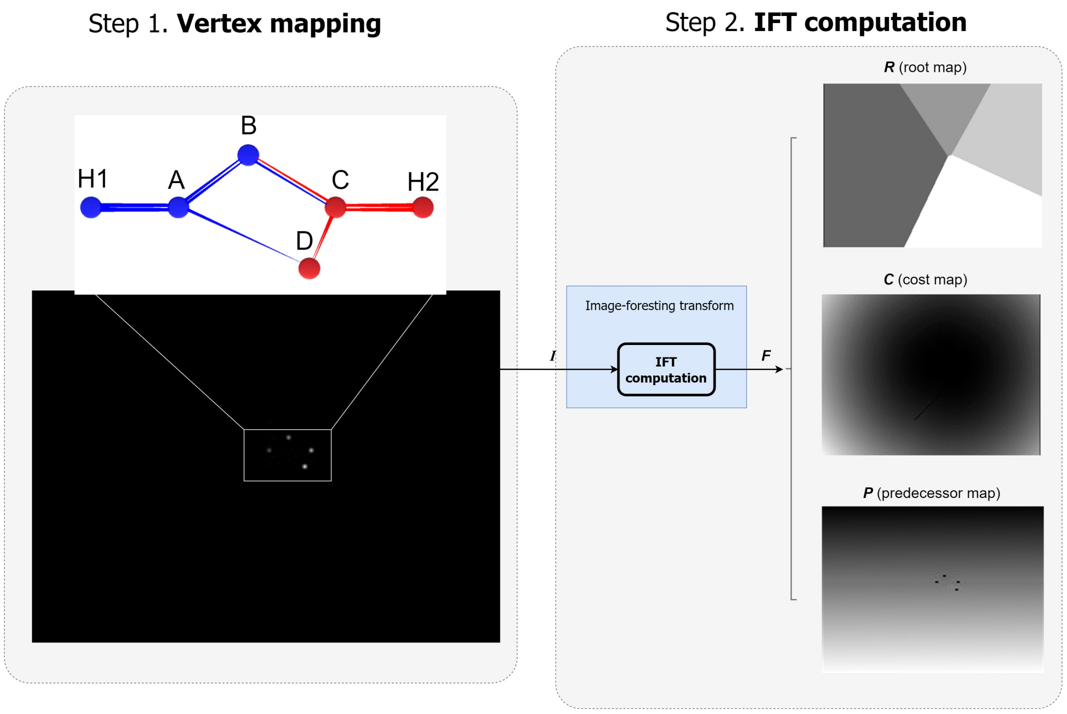

Step 1. **Vertex mapping**
H1
A
B
C
D
H2
Step 2. **IFT computation**
Image-foresting transform
IFT computation
I
F
R (root map)
C (cost map)
P (predecessor map)

Figure 4.6: Illustration of the IFT computation step.

$$F_c(POI_{nr}, P) = F_c(POI_{nr}, N_{nr}) + \frac{d(N_{nr}, P)}{s_{avg}} \tag{4.4}$$

$$\lambda(P|POI_{nr}, G) = \frac{1}{r} F_k\left(\frac{(F_c(POI_{nr}, P))^{-\alpha}}{r}\right) \tag{4.5}$$

For any pixel p of the image I by IFT, N_{nr} is the root map value $R(p)$ and $d(N_{nr}, P)$ is calculated by the cost map value $C(p)$ and scale parameter K_{px} by Equation 4.6. Therefore, the access time $F_c(POI_{nr}, P)$ can be computed based on the root map and the cost map of IFT. We construct the access time values for all pixels of the image I as the access time map F_c.

$$d(N_{nr}, P) = K_{px} \times \sqrt{C(p)} \tag{4.6}$$

The use of the IFT opens the opportunity to perform analysis of changes on density maps over time. Assuming that the input network may change over time, those changes would lead to different TDMs that could be computed efficiently using the IFT. We explore this scenario in Chapter 5.

4.3 Estimate Density Field

Different from the original TDM algorithm, in our formulation, the density field estimation (step ⓕ) will produce a density map Λ and a label map L. Those maps will be used for the temporal change computation in Chapter 5.

If only the root map by IFT is used to calculate the label map, this formulation is referred to as option 1 for ITDM. If both the root map and cost map produced by IFT are used to calculate the label map and density map, this is referred to as option 2 for ITDM.

Algorithm 2 outlines the main steps for the density field estimation. It iterates over all pixels in the image I (Lines 2-15). For ITDM option 2, it starts by creating a density map Λ with the access time map F_c (Line 4). Next, it finds the corresponding node vertex by the root map R (Line 5). Finally, it produces the source POI node for the given node N_{nr} (Line 6) using the accessibility data G_{cost}, which includes all accessibility information for each non-POI node.

Algorithm 2: Estimate density field

```
1  Auxiliary Data functions: TDM_Λ() and TDM_L() are two functions separately to compute
     the density map and label map value by TDM for the given pixel $p$.
   Data: Access time map $F_c$, Accessibility data $G_{cost}$
   Result: Density Map $\Lambda$, Label Map $L$
2  foreach pixel $p$ in image $I$ do
3      if ITDM option 2 then
4          Set $\Lambda(p) \leftarrow \frac{1}{r} F_k(\frac{(F_c(p))^{-\alpha}}{r})$(Eq. 4.5);
5          Set $N_{nr} \leftarrow R(p)$;
6          Set $L(p) \leftarrow$ the source POI node of node $N_{nr}$ by $G_{cost}$;
7      else if ITDM option 1 then
8          Set $\Lambda(p) \leftarrow \mathrm{TDM}_{\Lambda}(p)$;
9          Set $N_{nr} \leftarrow R(p)$;
10         Set $L(p) \leftarrow$ the source POI node of node $N_{nr}$ by $G_{cost}$;
11     else
12         Set $\Lambda(p) \leftarrow \mathrm{TDM}_{\Lambda}(p)$;
13         Set $L(p) \leftarrow \mathrm{TDM}_{L}(p)$;
14     end
15 end
```

Figure 4.7 illustrates the visual effects about the running example by TDM and ITDM. The results may be different from the original TDM. The definition of the suitable estimation option may consider requirements of the target application. In the example, a visualization based on ITDM option 2 is illustrated.

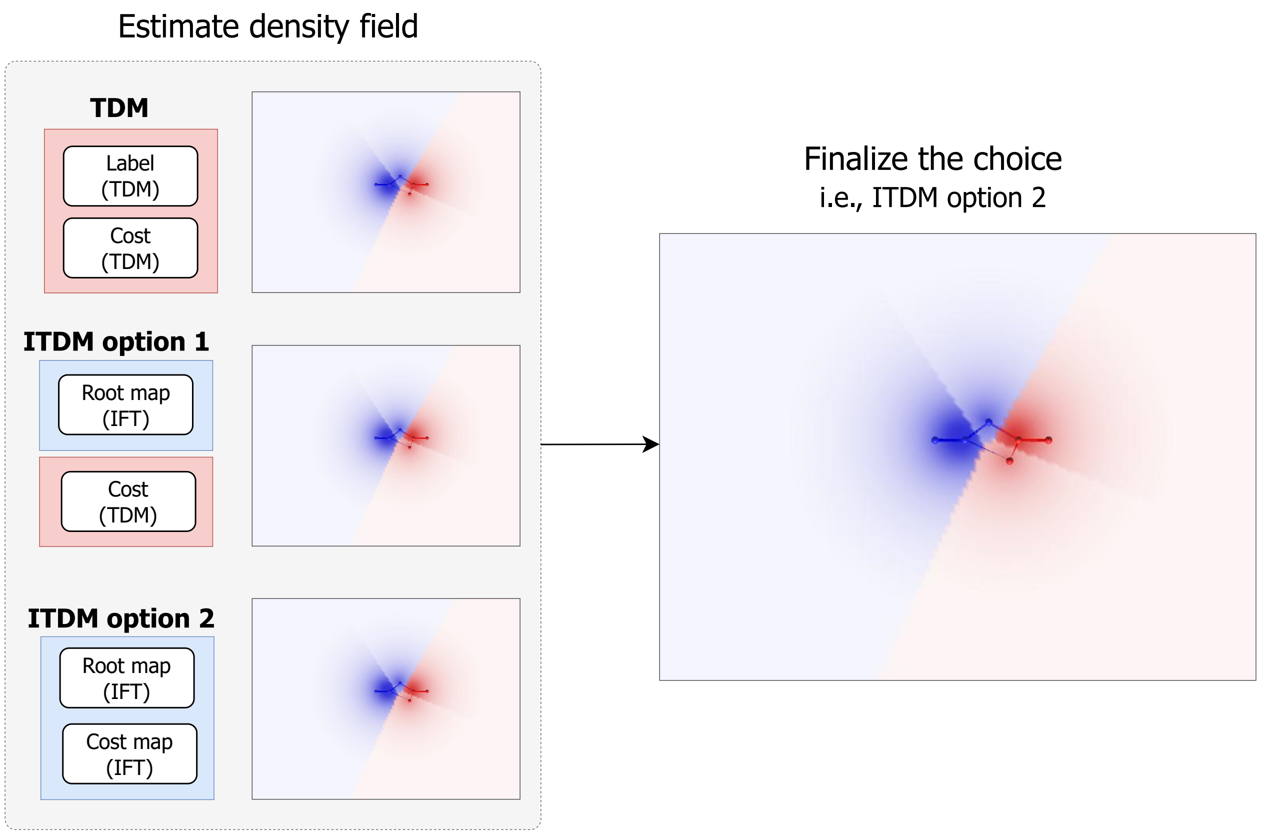


Figure 4.7: Visual effects of density fields for TDM and ITDM.

4.4 Computational Complexity

In the original TDM algorithm [17], the Euclidean distance computation is based on the computation of the distance of each pixel in the planar space to each node in the topology. Let N and n be the number of nodes in the topology and the number of pixels in the 2D space, respectively. The computational complexity of this operation is therefore $O(N \times n)$. For large values of N, the original algorithm [17] is much more costly than ITDM. Recall from Section 2.3, that the computational complexity of IFT is $O(n)$.

Chapter 5

Temporal Topology Density Map (TTDM)

This chapter describes the steps for the computation of the Temporal Topology Density Map (TTDM), a new algorithm to encode and visualize temporal changes in topology density maps. Changes are encoded by Change Frequency Heatmap (CFH) and are visually represented in the output maps produced by the IFT-based Topology Density Map (ITDM) algorithm. In this chapter, we address research question 2, which concerns how to use CFH to encode and visualize temporal changes on TDM.

5.1 Overview

Figure 5.1 illustrates the pipeline. The input data is a stack of graphs. Two main branches use this stack of graphs as input. The first one goes through the Compute Representative ITDM component and produces two outputs. They are the network accessibility data for the 1D network and the density map with the label map for the 2D space. Section 5.2 describes how it works. The other branch goes through the component Encode Temporal Changes and creates a change frequency map (matrix). Section 5.3 provides the details of this component. These three outputs of these branches are utilized to construct the final 3D visual effects by the module Compute the Visual Representation, which is presented in Section 5.4.

The input data of TTDM is a set of graphs $\mathcal{G} = \{G_1, G_2, \dots, G_T\}$ for timestamps $1, 2, \dots, T$. $\mathcal{G}$ encodes temporal changes in terms of edge costs. The network data $G_i(V, E) \in \mathcal{G}$ is the same as explained in Section 2.1. A $cost$ is associated with each edge, representing the dynamic temporal weight value. To incorporate temporal aspects in the current method, which is the main goal of this research work, we define the possibility of variations among the edge values. This means that the cost between each two nodes can dynamically change with time, and consequently, the nearest POI for each node may also change.

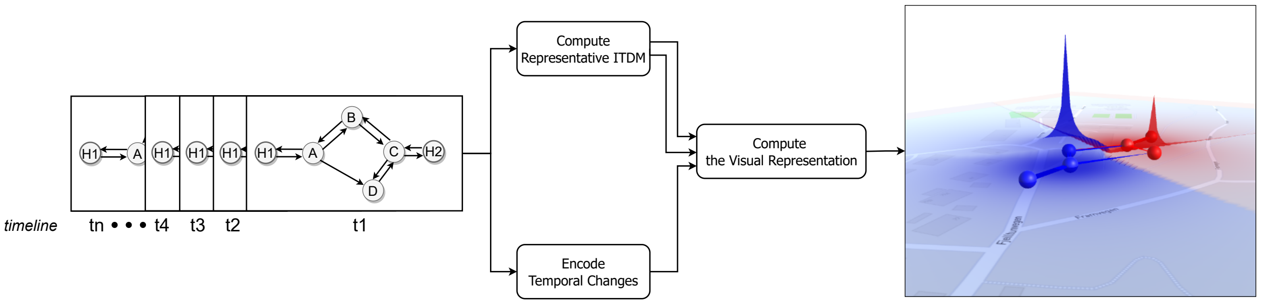
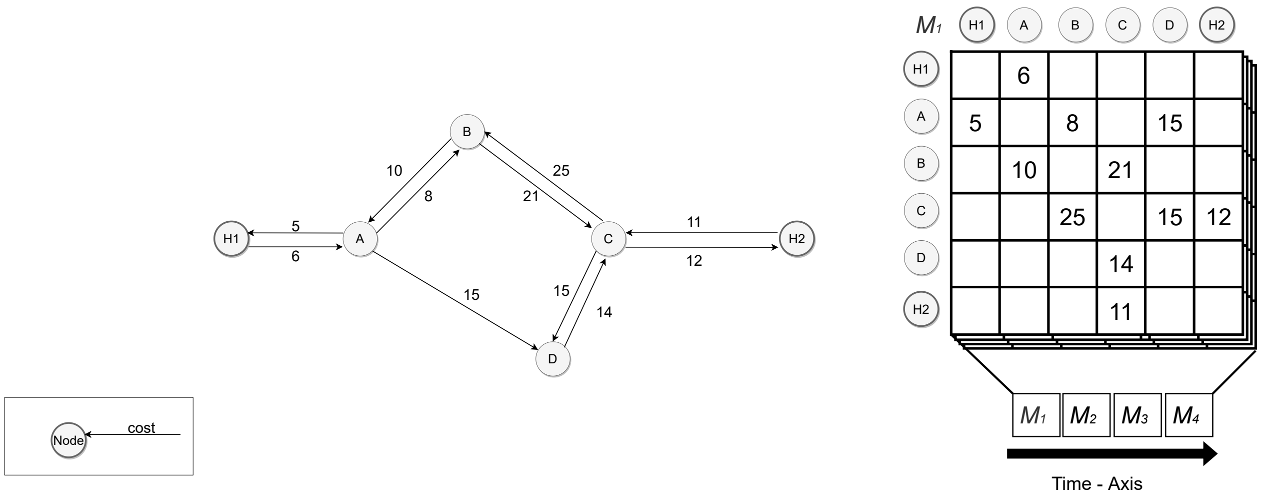

timeline
tn • • • t4
t3
t2
t1
H1
A
B
C
D
H2
Compute Representative ITDM
Compute the Visual Representation
Encode Temporal Changes

Figure 5.1: TTDM algorithm pipeline in concept.



M_1	H1	A	B	C	D	H2
H1		6				
A	5		8		15	
B		10		21		
C			25		15	12
D				14		
H2				11		

Figure 5.2: A running example for TTDM. Matrix M_1 (right) encodes the graph represented initial condition for the working example (left). $M_1 \sim M_4$ are the corresponding matrices for the four timestamps in sequence.

Adjacency matrices can be used to represent the edge costs of all graphs $G_i \in \mathcal{G}$. Let $\mathcal{M}$ be a set of matrices such that $\mathcal{M} =< M_1, M_2, \ldots, M_t, \ldots, M_T >$, where M_t is an $s \times s$ matrix at timestamp $t \in [1, T]$ composed of s lines and s columns. s is the number of nodes.

Figure 5.2 provides the running example for this chapter. It integrates a four timestamps temporal changes on the same example in Chapter 4. It is composed of $V = \{H1, A, B, C, D, H2\}$ and $E = \{e_1, e_2, ..., e_{11}\}$, where the edge is defined as $e_1 = (H1, A)$. In this case, the stack of matrices $\mathcal{M} =< M_1, M_2, M_3, M_4 >$ with the node number $s = 6$ and the size of the timestamps $T = 4$. According to our assumed temporal changes, matrix M_1 refers to the graph at timestamp t_1 illustrated in Figure 5.2. The empty elements in M_1 are filled out with zeroes for the diagonal and infinity for the other positions. The same applies for the other three matrices M_2, M_3, and M_4.

5.2 Compute Representative IFT-based Topology Density Map (ITDM)

This step aims to compute a graph to serve as a representative of the whole set $\mathcal{G}$. This representative graph (G_{rd}) will be used as input to the ITDM algorithm. The goal is to produce maps to be used for the 1D network and 2D space visualization. As Figure 5.3 illustrates, with the input $\mathcal{M}$, the module works as a filter that process all graphs in $\mathcal{G}$ using the input function f_{rd}. The default function is a weighted computation based on edge values as defined in Equation 5.1.

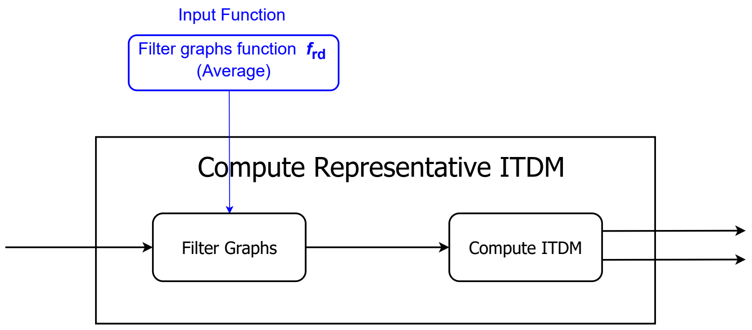

Input Function
Filter graphs function f_{rd} (Average)
Compute Representative ITDM
Filter Graphs
Compute ITDM

Figure 5.3: Compute Representative IFT-based Topology Density Map (ITDM) pipeline.

$$M_{rd} = \frac{1}{T}\sum_{t=1}^{T} w_t M_t \tag{5.1}$$

where M_{rd} is a matrix which encodes the representative graph G_{rd}, w_t are weights and $\frac{1}{T}\sum_{t=1}^{T} w_t = 1$. The default weighting scheme implemented assigns the same weight value for all matrices. In our example, that means $w_1 = w_2 = w_3 = w_4 = 1$.

The other module (Compute ITDM) has been described in Chapter 4. It computes the shortest path cost and the source POI label for each node. The cost and label information can then be used to determine the accessibility of nodes (first output). Also, this module also outputs the cost and the label maps that will be used later to produce the final visual representation related to the TTDM.

5.3 Encode Temporal Changes

As Figure 5.4 illustrates, to encode temporal changes, ITDMs for each timestamp matrix in $\mathcal{M}$ need to be computed. That step produces sequences of density maps and label maps. The module Compute CFH utilizes the density maps or the label maps as input. It then uses a metric f_m and behavior pattern S_{bp} to create a change frequency map, which reflects how many times the behavior pattern occurs for each position of the map. In the following sections, these two steps are detailed.

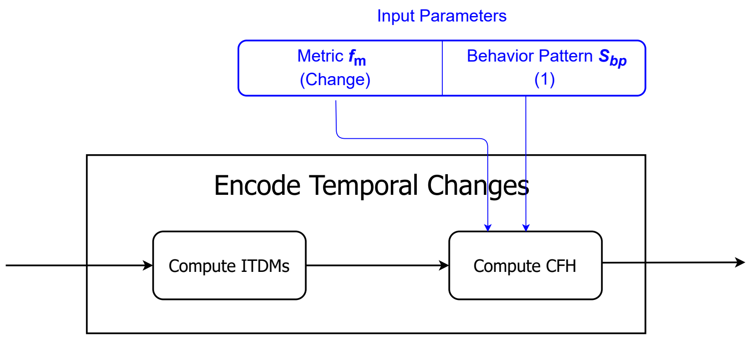


Figure 5.4: Encode temporal changes pipeline.

5.3.1 Compute IFT-based Topology Density Map (ITDM)s

This step concerns the computation of the IFT-based Topology Density Map (ITDM) for each timestamp t, i.e., it computes IFT-based Topology Density Map (ITDM) for each given input graph defined in terms of its edge cost matrix M_t ($t \in [1, T]$).

Running the ITDM algorithm for the matrices set $\mathscr{M}$ leads to creation of two sets: a set of labels $Label_{nr}$ and a set of costs $Cost_{nr}$. These two sets define the accessibility data set $< G_{cost} >$. After the density field estimation (Section 4.3), the algorithm creates a set of density maps $< \Lambda >$ and label maps $< L >$. The nodes in TDM are mapped as the seed pixels in IFT. The temporal changes of seed pixels represent the temporal changes of the pixels in corresponding regions. Therefore, the analysis of node temporal changes represented by $Label_{nr}$ is more direct.

Figure 5.5 illustrates the computation of multiple ITDMs. The upper half part of the shows the temporal variation in the matrices $\mathscr{M} =< M_1, M_2, M_3, M_4 >$ as the input. The rest of Figure 5.5 provides the computation result for the sample graph data. The cells highlighted in green refer to those changed between two consecutive timestamps. The path costs from POI nodes $(H1, H2)$ are calculated by the Dijkstra algorithm for directional graph. After the minimum calculation of the path costs from each POI for each nodes, $Label_{nr}$ and $Cost_{nr}$ are created, which form the shortest path costs information (G_{cost}). When $t = 1$, the non-POI node B has shortest path $(H1 \rightarrow A \rightarrow B)$ cost value 14 $(6+8)$ from the POI node $H1$. With the increase of $e_3(A, B)$ cost and decrease of $e_7(C, B)$ at $t = 2$, the shortest path for non-POI node B becomes $(H2 \rightarrow C \rightarrow B)$ cost value 21 $(11 + 10)$ from the POI node $H2$, which has lower cost than the other path $(H1 \rightarrow A \rightarrow B)$ from POI node $H1$ with cost value 36 $(6 + 30)$. The temporal variation affects the accessibility associated with different regions. The output is a set of G_{cost} for different timestamps. This set includes the accessibility data for the ***nodes*** of graph. The density maps

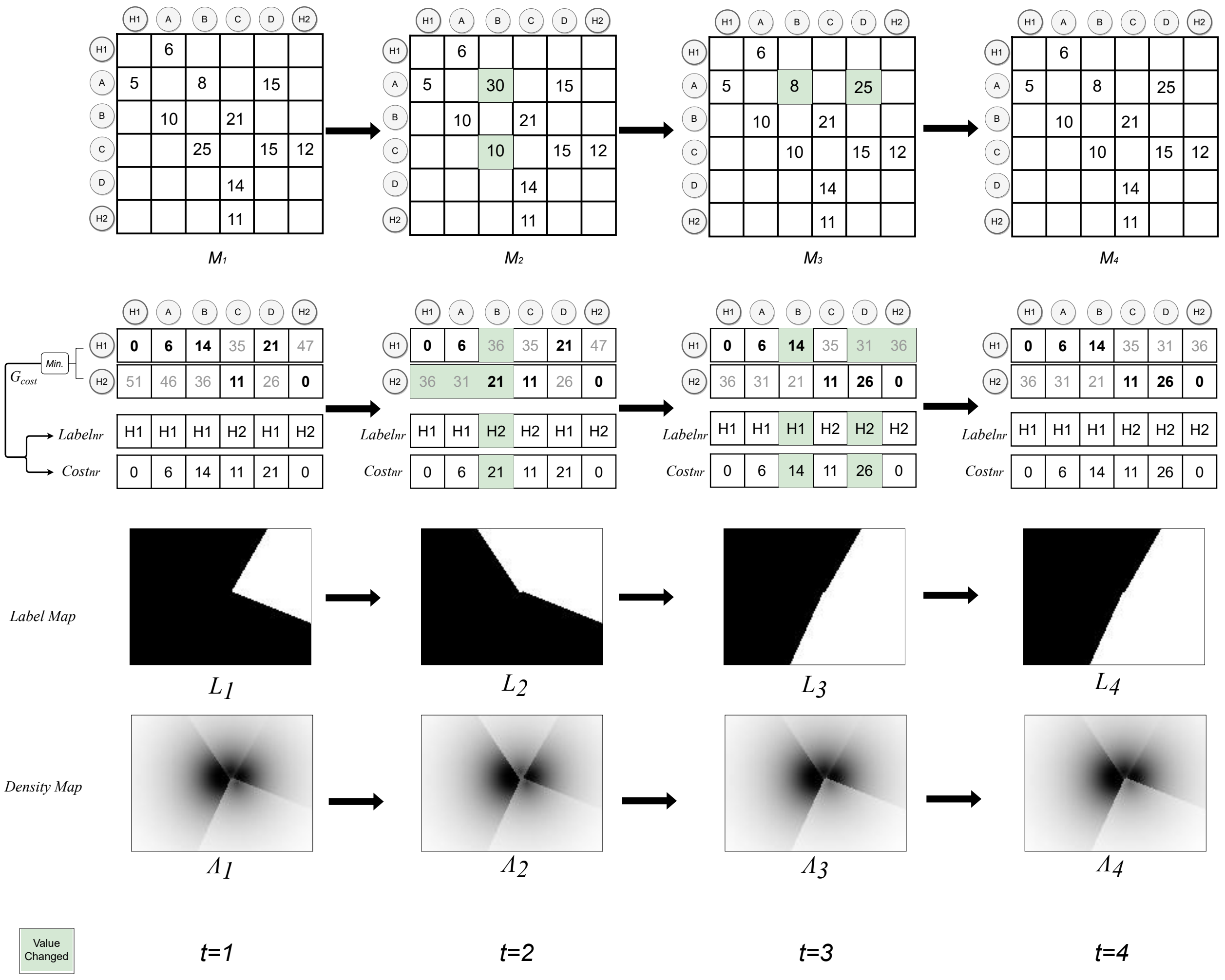


Figure 5.5: An example of the computation of multiple ITDMs. It reflects the temporal changes of the sample graph data when $T = 4$.

$< \Lambda >$ and label maps $< L >$ are also illustrated in the figure.

5.3.2 Compute Change Frequency Heatmap (CFH)

This step concerns the computation of the Change Frequency Heatmap, as described in Section 2.2. This component receives a metric function that defines the change pattern of interest as parameter. Another parameter is the change binary pattern.

Figure 5.6 illustrates this computation process. The input is a set of temporal maps. They can be density maps or label maps. After the compute temporal binary pattern with the metric function f_m, the algorithm creates a group of change binary maps. Finally, it outputs one change frequency map with the count number of the given behavior pattern.

Algorithm 3 outlines the steps of temporal change encoding. The first step is the construction

Algorithm 3: Encode temporal changes.

```
1  Auxiliary Data structures: The binary change map set $D_c$ for saving the binary pattern
     representation.
2  Auxiliary Data functions: $IsChanged()$ is a function to return whether the two input
     elements are changed or not within the difference by the threshold value $T_r$, the
     function $index()$ gets the element index for the map.
   Data: Temporal change map set $M_c$, the number of timestamps $T$, the binary pattern
         string $S_{bp}$, the metric function $f_m$ is $IsChanged()$
   Result: Change Frequency Map $T_f$
3  foreach element $m$ in $M_c$ do
4      foreach timestamp t in (1,T-1) do
5          if IsChanged($m_t$,$m_{t+1}$, $T_r$) then
6              $d_t \leftarrow 1$;
7          else
8              $d_t \leftarrow 0$;
9          end
10     end
11 end
12 $p = S_{bp}$;
13 foreach element $d$ in $D_c$ do
14     $T_f(index(d)) \leftarrow \sum_{x=1}^{T-l} \mathbb{1}\{\omega_{x,y} = p \mid \omega_{x,y} \in \Omega\{d\}\}$(Eq. 2.6);
15 end
```

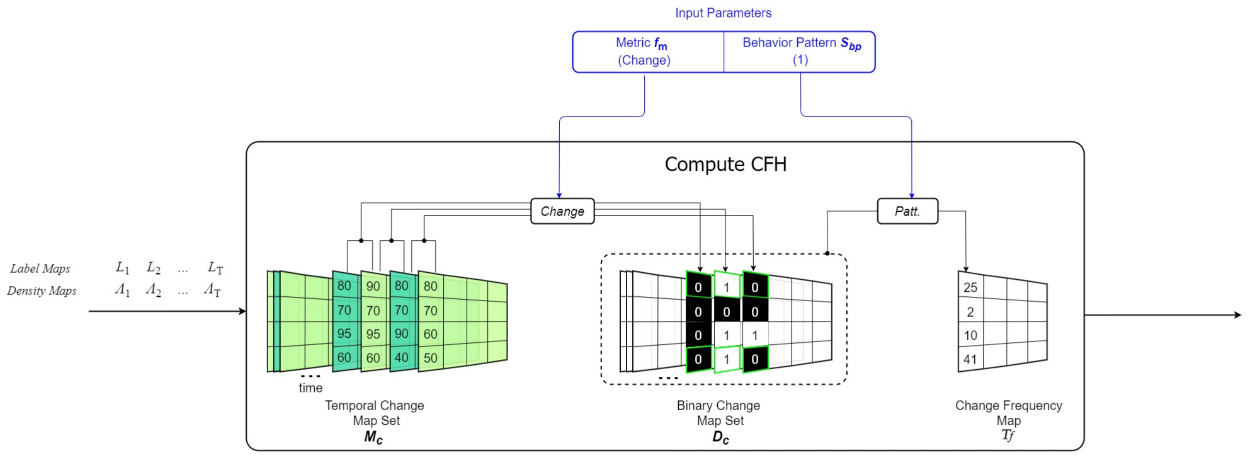
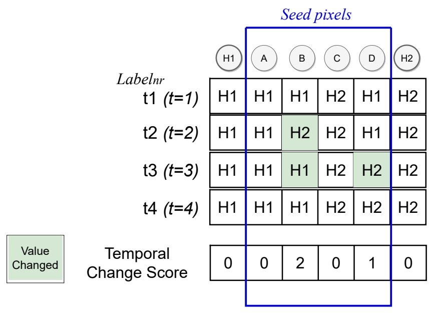

Input Parameters
Metric f_m (Change)
Behavior Pattern S_{bp} (1)
Compute CFH
Change
Patt.
Label Maps L_1 L_2 ... L_T
Density Maps Λ_1 Λ_2 ... Λ_T
80 90 80 80
70 70 70 70
95 95 90 60
60 60 40 50
time
Temporal Change Map Set M_c
0 1 0
0 0 0
0 1 1
0 1 0
Binary Change Map Set D_c
25
2
10
41
Change Frequency Map T_f

Figure 5.6: Illustration of the compute Change Frequency Heatmap (CFH) step.

of binary change maps (Lines 3-11). Next, the algorithm computes the change frequency map (Lines 13-15). The output change frequency map is used for the visual representation.

Figure 5.7 provides a scenario related to temporal changes observed for nodes in the working example when the binary pattern string S_{bp} is "1." The figure also represents the temporal changes for the seed pixels in the label maps $< L >$ when we set the label maps as the temporal change



Label$_{nr}$	H1	A	B	C	D	H2
t1 (t=1)	H1	H1	H1	H2	H1	H2
t2 (t=2)	H1	H1	H2	H2	H1	H2
t3 (t=3)	H1	H1	H1	H2	H2	H2
t4 (t=4)	H1	H1	H1	H2	H2	H2
Temporal Change Score	0	0	2	0	1	0

Figure 5.7: Illustration of temporal changes for the nodes in an example.

maps for computing the change frequency map. In the illustration, changes refer to nodes (also seed pixels) A, B, C, and D (non-POI nodes), and $H1$ and $H2$ (POI nodes). The variations of source POI for nodes, $Label_{nr}$, are represented in this figure (see the Temporal Change Score row related to label maps). It was produced by the module Compute ITDMs (Section 5.2).

Figure 5.8 shows a more detailed example of the temporal changes with label maps and colored nodes. In the example, the center region of the label maps is highlighted. The corresponding graph is presented in the middle of the figure for different timestamps. We assume that the color of $H1$ is blue and the color of $H2$ is red. At timestamp 1, the point of interest $H2$ is the closest to node C, and $H1$ is the closest to other nodes. At timestamp 2, the nearest point of interest for node B changes from $H1$ to $H2$. At timestamp 3, node B is more accessible from $H1$ again, while node D falls into the $H2$ group. Finally, at timestamp 4, there are no changes in the nearest point of interest among the nodes. If we encode these changes into numbers, there were two changes found in the status of node B and one change in the status of node D. Therefore, the temporal variation value for node B is two and for node D is one.

5.4 Compute the Visual Representation

The module Compute the Visual Representation provides the resulting visualization related to the execution the TTDM algorithm. As Figure 5.9 shows, it computes 1D network with the module (Network Color Mapping) and 2D space with the module (Density Field Color Mapping). Both of them are the outputs produced by the module Compute Representative ITDM (Section 5.2). Moreover, it utilizes the output of the module Encode Temporal Changes (Section 5.3) to calculate a height map by the module (Height Mapping). The last module, Visual Integrator, interpolates the heights for the vertices in 3D mesh with the given interpolation scale m and integrates all visual layers into a visual 3D world. The final visualization combines the maps related to temporal changes and the representative density field map density field map simultaneously. The following sections describe these modules.

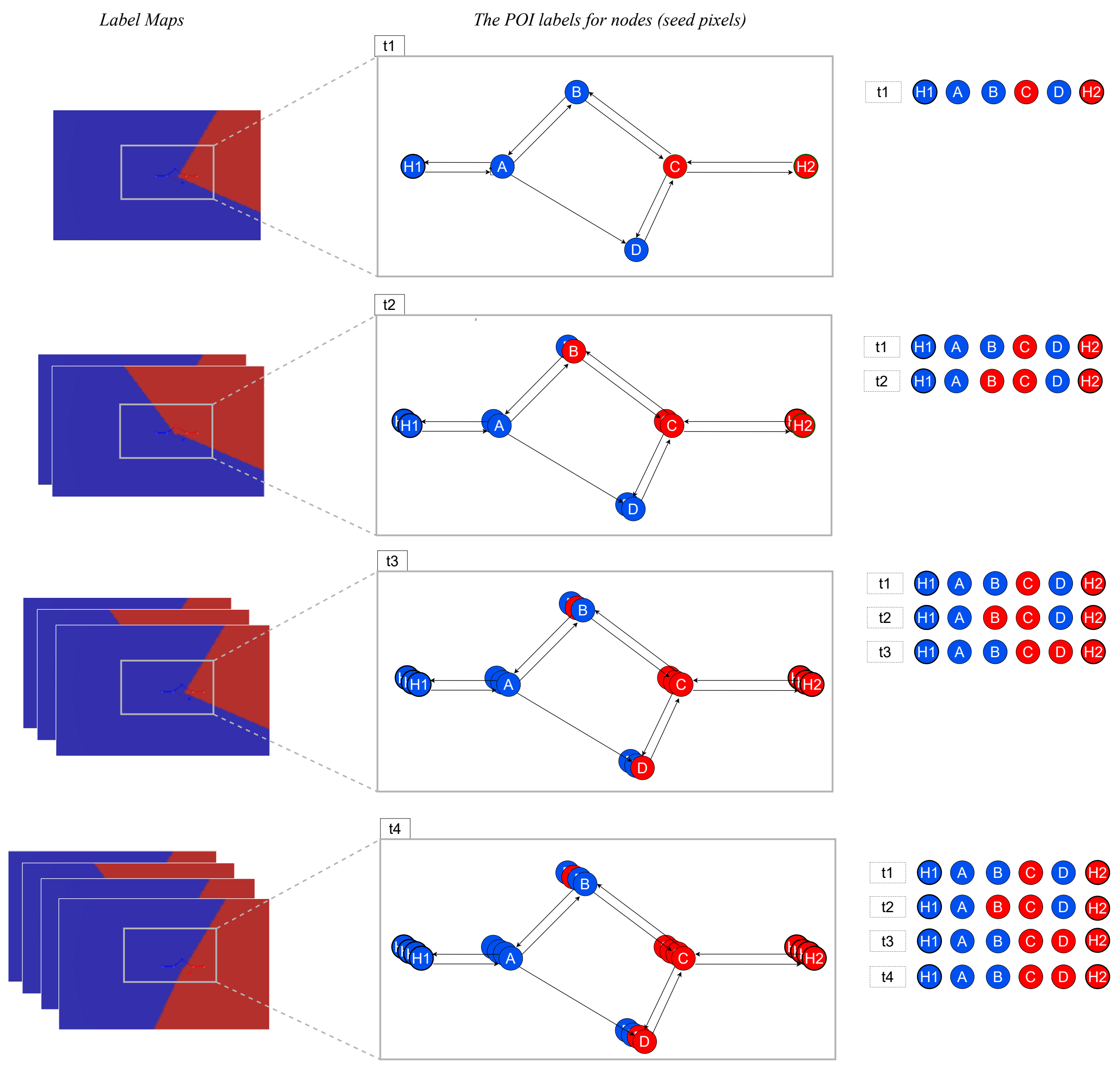


Figure 5.8: An example of temporal variation mapping.

5.4.1 Network Color Mapping

The network color mapping aims the creation of a visual representation of the 1D road network with colored nodes and tapered edges. Algorithm 4 outlines the key steps for network color mapping. The first step is to draw all nodes with the associated POI color (Lines 2-6). The other step is to draw all edges (Lines 7-19), which relies on computing the normalized cost values along edges (function $Norm$ in the algorithm) of the path cost separately for the start and end nodes first (Lines 8-16). After that, it finishes the drawing of the edge considering the different width values along the edge (Lines 17-18). The color of the edge is the same as the start node

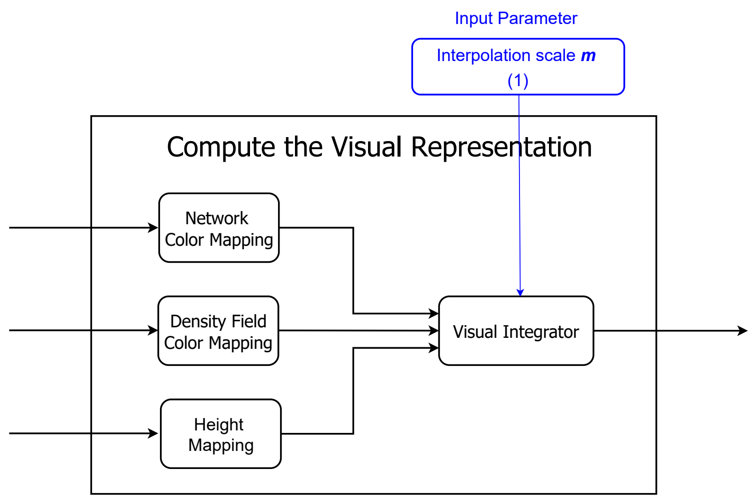

Input Parameter
Interpolation scale *m* (1)
Compute the Visual Representation
Network Color Mapping
Density Field Color Mapping
Height Mapping
Visual Integrator

Figure 5.9: Compute the Visual Representation pipeline.

color (Line 13).

Figure 5.10 presents the network color mapping with an example. Once finished the computation of the DrawNode function for all nodes, nodes A and B are blue, as their nearest POI node is node $H1$. The red nodes C and D are closer to $H2$. Next, the edge is computed started from $e_1(H1, A)$, $e_2(A, H1), \ldots, e_{11}(H2, C)$. For instance, $e_4(A, D)$ is blue because the start node D is blue. The cost value for A is less than D from the POI node $H1$, which means A has higher accessibility from POI node. Therefore the width of the line for e_4 is decreasing from the start node A to the end node D. Similarly, it is easy to follow the blue edge $e_6(B, C)$ and the red edge $e_7(C, B)$, a case of bi-directional edges with different colors.

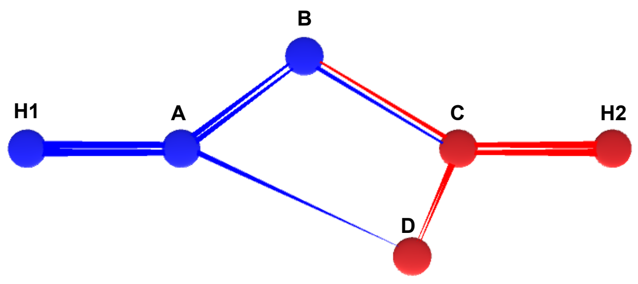


Figure 5.10: An example of network color mapping.

5.4.2 Density Field Color Mapping

Density field color mapping utilizes the density map Λ and the label map L to create the output color map L_c. Algorithm 5 outlines the key steps for density field color mapping calculation. For

Algorithm 4: Network color mapping

```
1  Auxiliary Data functions: Norm() is a function to return linear normalization result of
     given node path cost value in the cost set of $G_{cost}$. DrawNode() and DrawEdge() are two
     functions to draw node and edge with given parameters on Network Map $G_c$.
   Data: Accessibility data $G_{cost}$
   Result: Network Map $G_c$ includes colored 1D road network
2  foreach vertex $v$ in $V$ do
3      Set $P \leftarrow v$;
4      Set $c_r \leftarrow$ the color of node $P$ by $G_{cost}$;
5      DrawNode($v, c_r$);
6  end
7  foreach edge $e$ in $E$ do
8      Set $v_x \leftarrow$ the start node of $e$;
9      Set $v_y \leftarrow$ the end node of $e$;
10     Set $P1 \leftarrow v_x$;
11     Set $F_c(POI_{nr}, P1) \leftarrow$ the path cost of node $P1$ from belonging POI node by $G_{cost}$;
12     Set $s_{P1} \leftarrow 1 - Norm(F_c(POI_{nr}, P1))$;
13     Set $clr \leftarrow$ the color of node $P1$ by $G_{cost}$;
14     Set $P2 \leftarrow v_y$;
15     Set $F_c(POI_{nr}, P2) \leftarrow$ the path cost of node $P2$ from belonging POI node by $G_{cost}$;
16     Set $s_{P2} \leftarrow 1 - Norm(F_c(POI_{nr}, P2))$;
17     Set the proper edge width scaling factor $K_w$ (defaults to 1);
18     DrawEdge(e, $s_{P1}, s_{P2}, clr, K_w$);
19 end
```

each pixel in color map L_C (Lines 2-6), the final color value (Lines 5) with transparency value α_{color} (Lines 4) is calculated by the density estimated value λ (Equation 2.1 [17]) and the closest POI node color POI_{Color} (Lines 3).

Figure 5.11 provides the only result of density field color mapping and the integration with network color mapping. It is calculated on the average cost result among the timestamps. Therefore, we could find the color consistency of ***1D network*** (network color mapping) and ***2D space*** (density field color mapping). The region defined by $ABCD$ is easily accessible through $H1$ and $H2$ while the region $(H1A)$ is only affected by $H1$.

5.4.3 Height Mapping

The TTDM algorithm may encode temporal variations related to the costs between the nodes or the accessibility of nodes to the POIs. As discussed in Section 5.3, those temporal variations are encoded in a change frequency map. If we choose the label maps as the input for encoding the temporal changes, the change frequency map reflects the source POI label changes for each

Algorithm 5: Density field color mapping

```
1 Auxiliary Data functions: Norm() is a function to return linear normalization result of
    given density field value in the whole values set of Density Map $\Lambda$.
  Data: Density Map $\Lambda$, Label Map $L$
  Result: Color Map $L_C$
2 foreach pixel $p$ in $L_C$ do
3     Set $POI_{Color} \leftarrow$ the color of POI node $L(p)$;
4     Set $\alpha_{color} \leftarrow 1 - Norm(\Lambda(p))$;
5     Set $L_C(p) \leftarrow (POI_{color}, \alpha_{color})$;
6 end
```

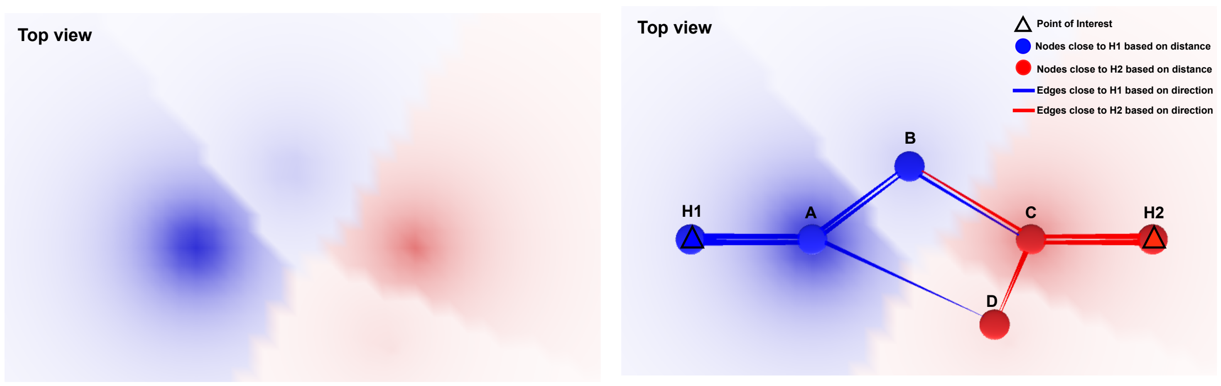

Top view
Top view
Point of Interest
Nodes close to H1 based on distance
Nodes close to H2 based on distance
Edges close to H1 based on direction
Edges close to H2 based on direction
B
H1
A
C
H2
D

Figure 5.11: An example of density field color mapping. The density field color mapping result (left) and the integrated result with the network color mapping (right).

pixel. The height mapping module aims to compute a height value for each pixel p. This height value of the pixel p is calculated based on its Euclidean distance to the nearest non-POI node using Equation 5.2.

$$H(p) = K \times \frac{T_f(p)}{1 + K_{px} \times \sqrt{C(p)}} \tag{5.2}$$

where p is a pixel in the height map, K is the scaling factor, $T_f(p)$ is the change frequency map value in the same pixel position p, $C(p)$ is the cost map value for pixel p. In fact, $K_{px} \times \sqrt{C(p)}$ is the distance $d(N_{nr}, P)$ between the point P and the nearest non-POI node N_{nr} as Equation 4.6. Algorithm 6 outlines the key steps.

Figure 5.12 is the height mapping result for the sample graph, with $H1$ and $H2$ representing the POI nodes. The black marked circles represent the nodes with more frequent temporal changes. The pixels closed to the nodes B ($T_f(B) = 2$) and C ($T_f(C) = 1$) have the higher change frequency map values, which are proportional to the elevations there. It also provides the visual effects integrated with ***1D network*** (network mapping) and ***3D mesh*** (height mapping).

Algorithm 6: Height Mapping

```
  Data: Change Frequency Map $T_f$, Cost Map $C$
  Result: Height Map $H$
1 foreach pixel $p$ in $H$ do
2     Set the proper scaling factor K (defaults to 1000);
3     Set $H(p) \leftarrow K \times \frac{T_f(p)}{1+K_{px} \times \sqrt{C(p)}}$ (Eq. 5.2);
4 end
```

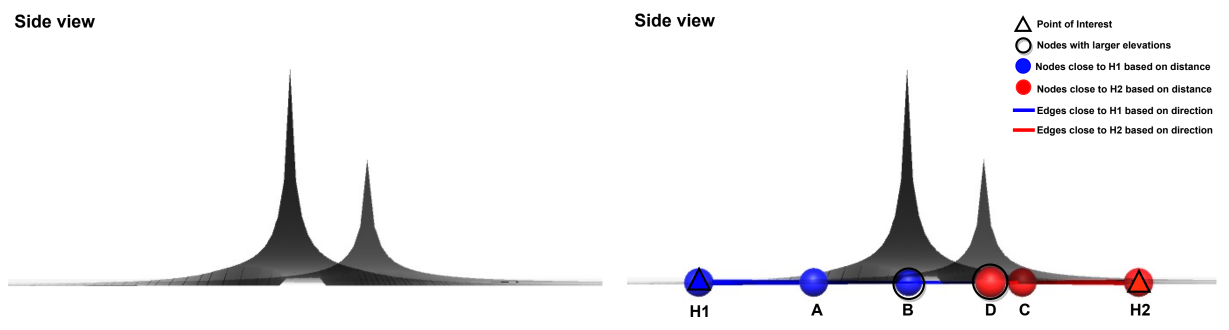

Side view
Side view
Point of Interest
Nodes with larger elevations
Nodes close to H1 based on distance
Nodes close to H2 based on distance
Edges close to H1 based on direction
Edges close to H2 based on direction
H1
A
B
D
C
H2

Figure 5.12: The height mapping result (left) and the integrated result with the network color mapping (right).

5.4.4 Visual Integrator

The Visual Integrator module includes the encoding of height values in a higher resolution map by mesh interpolation and the integration of the network and density field color mapping results. Mesh vertex contains x, y, and z coordinates and may contain a vector normal, a color value, and texture coordinates in 3D space. In our computation, we use local 3D coordinates (x, y, z) to represent a mesh vertex directly. Considering the most widely used coordinate system, local 3D coordinates (x, y, z) is chosen to implement the design while y is the elevation, x is the longitude, and z is the latitude. Mesh interpolation is a method to compute a new set mesh vertices based on an existing 3D mesh associated with the height map. In the proposed integration process, the scale of the final visual structure is defined by means of a parameter m, referred to from now on as "interpolation scale." The number of vertices in 3D mesh determines the complexity of calculation, which finally defines how much details are encoded in the visualization.

For the existing 3D mesh, let V_{orig} be a set of mesh vertices composed of s_v rows and s_v columns mesh vertices, s_v is the number of mesh vertices. The result after mesh interpolation is a new set of mesh vertices W composed of s_w rows and s_w columns mesh vertices. The required 3D mesh may be created by the W directly. The relation between s_w and s_v is defined in Equation 5.3

Algorithm 7: Mesh interpolation

```
1  Auxiliary Data functions: the function $index()$ gets the vertex index in the vertices set;
   Data: Original mesh vertices set $V_{orig}$, interpolation scale $m$
   Result: Interpolated mesh vertices set $W$
2  Set $s_w = s_v + (m-1) \times (s_v - 1)$;
3  initialize $W$ with $s_w$ rows and columns;
4  foreach $w$ in $W$ do
5      Set $i_r \leftarrow index(w)/s_w, i_c \leftarrow index(w)\%s_w$;
6      Set $i_{1r} \leftarrow i_r/m, i_{1c} \leftarrow i_c/m$;
7      Set $i_{2r} \leftarrow i_{1r}+1, i_{2c} \leftarrow i_{1c}+1$;
8      Set $v_1 \leftarrow V_{orig}(i_{1r}, i_{1c}), v_2 \leftarrow V_{orig}(i_{1r}, i_{2c}), v_3 \leftarrow V_{orig}(i_{2r}, i_{2c}), v_4 \leftarrow V_{orig}(i_{2r}, i_{1c})$
         considering the border condition $s_w$;
9      Set $k_r \leftarrow \frac{i_r - mi_{1r}}{m}, k_c \leftarrow \frac{i_c - mi_{1c}}{m}, k_h \leftarrow \frac{\sqrt{(i_r - mi_{1r})^2 + (i_c - mi_{1c})^2}}{\sqrt{2}m}$;
10     Set $n_z \leftarrow$ z value of $(v_4 - v_1) \times k_r + v_1$;
11     Set $n_x \leftarrow$ x value of $(v_2 - v_1) \times k_c + v_1$;
12     Set $n_y \leftarrow$ y value of $(v_3 - v_1) \times k_h + v_1$;
13     Set $w \leftarrow (n_x, n_y, n_z)$
14 end
```

$$s_w = s_v + (m-1) \times (s_v - 1) \tag{5.3}$$

where m is the interpolation scale.

Figure 5.13 provides an example of mesh interpolation. In this example, we assume the the project area of this 3D mesh is a 2D map (one scalar field) that includes the sample graph in step 1. In the example, V_{orig} is composed of 100 mesh vertices $(v_{1\_1}, v_{1\_2}, \ldots, v_{10\_10})$. They are the white background points given according to map accuracy in step 2. This number of mesh vertices is assumed as the highest default precision of the sample scalar field in our implementation, i.e., 10×10 is the highest grid size. After that, m is defined for mesh interpolation using 3D interpolation. The elevation value y of each mesh vertex is given by the height map H.

If $m = 2$, the total number of mesh vertices will increase to 361 according to the Equation 5.3, $(10 + (m-1) \times (10-1))^2 = 19^2 = 361$. Then, the interpolated vertices set W, which includes 361 mesh vertices (261 mesh vertices are new, and they are highlighted in grey from step 3). These mesh vertices are $(w_{1\_1}, w_{1\_2}, \ldots, w_{19\_19})$. If we resample the cells, more cells could be included in step 4. There are fewer area sizes of new cells after the finish of mesh interpolation. if $m = 1$, $W = V_{orig}$.

Algorithm 7 outlines the calculation progress of interpolation based on Figure 5.14. For each loop, the mesh vertices v_1, v_2, v_3, and v_4 are found in V_{orig}, and the mesh vertex w is the output

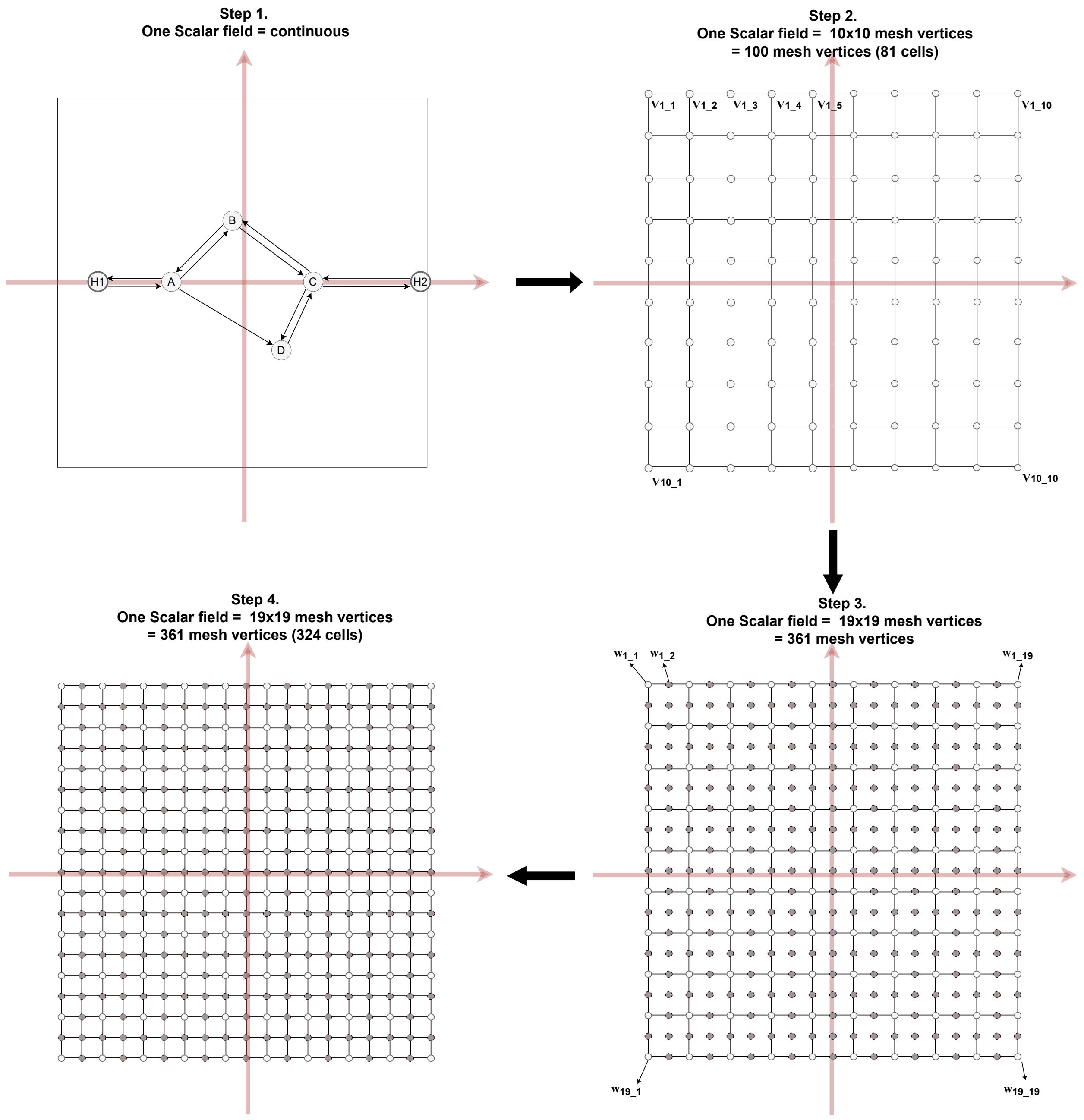

Step 1.
One Scalar field = continuous
H1
A
B
C
D
H2
Step 2.
One Scalar field = 10x10 mesh vertices
= 100 mesh vertices (81 cells)
V1_1
V1_2
V1_3
V1_4
V1_5
V1_10
V10_1
V10_10
Step 3.
One Scalar field = 19x19 mesh vertices
= 361 mesh vertices
W1_1
W1_2
W1_19
W19_1
W19_19
Step 4.
One Scalar field = 19x19 mesh vertices
= 361 mesh vertices (324 cells)

Figure 5.13: An example of mesh interpolation. Mesh interpolation steps for $m = 2$ with changing mesh vertices in one sample scalar field.

with the calculated result based on these four mesh vertices and m (Lines 11-13).

Figure 5.15 presents the visual results of the module visual integrator for the sample graph. it combines multiple 3D elements layers in one 3D space. For instance, we set 1D network layer, 2D space layer and 3D mesh layer correspondingly represents the visual effect of the modules network color mapping, density field color mapping and height mapping. Furthermore, the

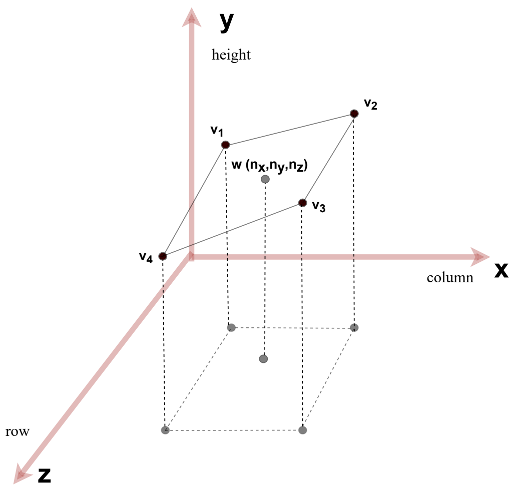


Figure 5.14: Illustration of mesh interpolation procedures.

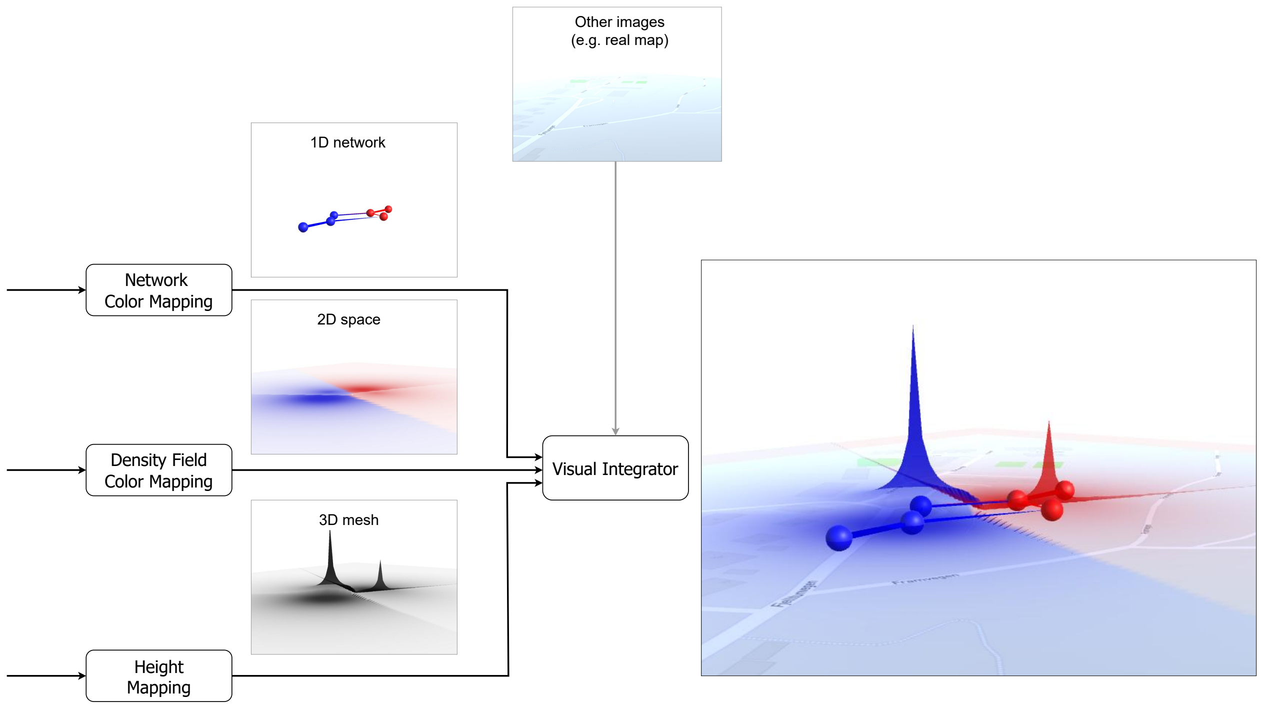


Figure 5.15: The visual integrator example with the multiple layers.

visual integrator could also display some auxiliary information in other images layer like real map.

Chapter 6

Validation

This chapter addresses the validation of the proposed algorithms: ITDM and TTDM. Section 6.1 introduces implementation aspects covering used technologies and developed prototypes. Section 6.2 addresses research question 3, which concerns the performance (quantitative) and qualitative assessment of TTDM. Section 6.3 focuses on research question 4, describing two case studies that employ the developed algorithms in mobility-related analyses.

6.1 Implementation Aspects

This section overviews the main technologies employed in the implementation of the proposed algorithms, as well as the prototypes created.

6.1.1 Overview of Used Technologies

Two prototypes were designed and implemented. The first one, a desktop software prototype, aims to support the performance and qualitative assessment of TTDM. This prototype allows algorithm computation analysis according to different parameter settings. The second one, a web-based software prototype, aims to support the analyses of diverse visual layouts associated with the two case studies considered in our study.

Figure 6.1 illustrates the main technologies employed in the implementation of these two software prototypes. The Python programming language and the OSMnx package [54][1] were utilized to download and export geospatial data from OpenStreetMap.[2] The python module is responsible for encoding the target network obtained from OpenStreetMap (Label 1) to a comma-

[1]https://github.com/gboeing/osmnx (As of May 2022).
[2]https://www.openstreetmap.org/ (As of May 2022).

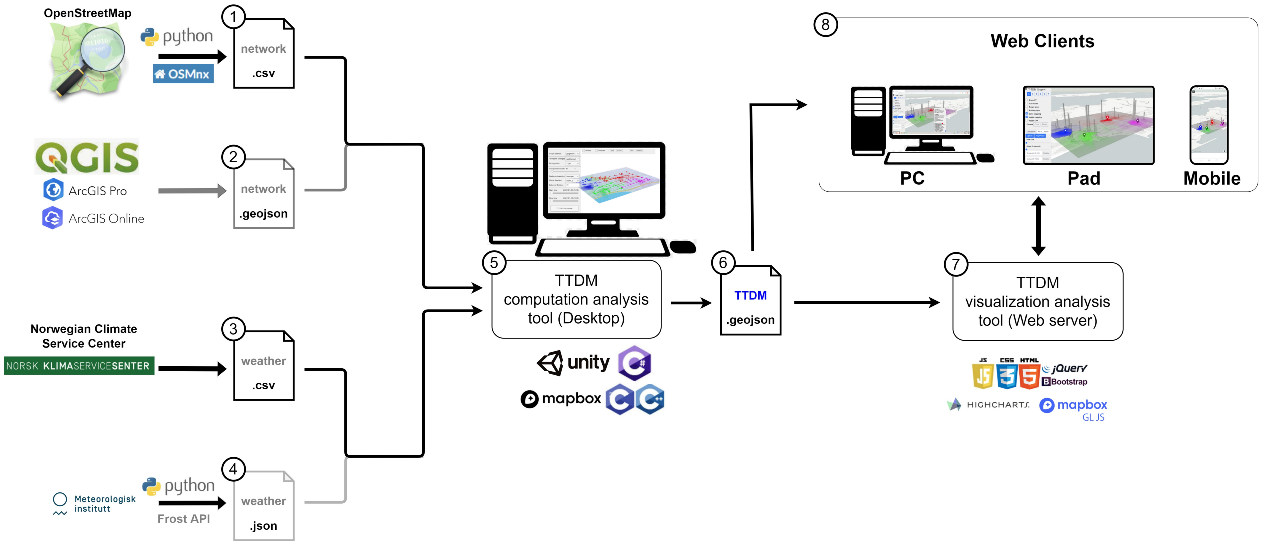


Figure 6.1: Overview of the different technologies used in the implementation of the prototypes.

separated values (CSV) file. The alternative supporting format (Label 2) of network data is GeoJSON[3], which is a geospatial data interchange format based on JavaScript Object Notation (JSON). The free Open Source software QGIS[4] and the commercial software ArcGIS (Pro, Online)[5] are the main popular tools to create and edit GeoJSON files. Similarly, there are also two additional formats (CSV and JSON) used for loading weather data from two Norwegian providers: Norwegian Climate Service Center[6] (3 in the figure) and Meteorologisk Institutt – Frost Application Programming Interface (API)[7] (4).

The TTDM computation analysis tool (Desktop) (Label 5) is a software prototype that contains an implementation of the TTDM algorithm in Unity[8] with C# script and Mapbox Software Development Kit (SDK).[9] The input graph network is encoded into two file formats (CSV, GeoJSON). This prototype computes edge costs based on weather data recorded in CSV and JSON. The prototype also integrates an IFT Dynamic Link Library (DLL) implemented based on the IFT C source code package.[10] The software provides an approach to execute the TTDM algorithm on selected datasets with customized parameters. It supports 3D visualization as well as saving the algorithm computation result as a GeoJSON format file (Label 6).

TTDM visualization analysis tool (Web server) (Label 7) is a software prototype to visualize the

[3]https://geojson.org/ (As of May 2022).
[4]https://qgis.org/en/site/ (As of May 2022).
[5]https://www.arcgis.com/ (As of May 2022).
[6]https://seklima.met.no/ (As of May 2022).
[7]https://frost.met.no/ (As of May 2022).
[8]https://unity.com/ (As of May 2022).
[9]https://www.mapbox.com/unity (As of May 2022)
[10]https://github.com/tvspina/ift-demo (As of May 2022).

TTDM computation results on the web. It is mainly implemented using Javascript, Mapbox GLJS library,[11] and Bootstrap.[12] This prototype supports the assessment of generated visual structures with a more user-friendly user interface (UI). Different kinds of web clients (Label 8) are expected to access this web server. Those clients allow users to upload a TTDM GeoJSON file (Label 6) and compare associated visual results.

6.1.2 Overview of Prototypes

This section describes the main features of the developed prototypes.

TTDM Computation Analysis

The software prototype TTDM computation analysis tool (Desktop) is designed to support computation analysis. Its main functionalities are:

1. Integration of an IFT C source code package in the TTDM computation.
2. Execution of algorithms to support performance and qualitative assessment and download of results.
3. Execution of the TTDM algorithm on a selected graph dataset with different parameters and visualization of results in a 3D view.
4. Saving of TTDM computation results as a GeoJSON file for visualization assessment.

Figure 6.2 provides an overview of the implementation components. The external resources include files or interfaces needed for the implementation. The functions are programmed in C# script codes, and visual structures are created by means of visual objects in Unity. Mapbox API (Label 1) is called by the Mapbox SDK (Label 4) to construct a Mapbox street map Layer (Label 9). The network and weather data files (Label 2) are the data source for the core function in the TTDM computation (Label 5). This algorithm needs to call the IFT algorithm available in the created IFT DLL file. The TTDM computation module uses the configuration defined in the user interface (Label 11) to update the TTDM Visual Layers (Label 10) (Section 5.4.4). Other features refer to exporting TTDM .geojson file (Label 6) and performance assessment (Label 7). Finally, label and density maps created during the computation can be exported (Label 8).

Figure 6.3 presents a screenshot of the User Interface of the TTDM computation analysis tool (Desktop). On the left region, there is a menu composed of five panels (Labels 1-5). This menu allows the definition of the configuration of the parameters before the TTDM computation. The users may select the graph dataset, the method for encoding temporal changes, the weather

[11] `https://www.mapbox.com/mapbox-gljs` (As of May 2022).
[12] `https://getbootstrap.com/` (As of May 2022).

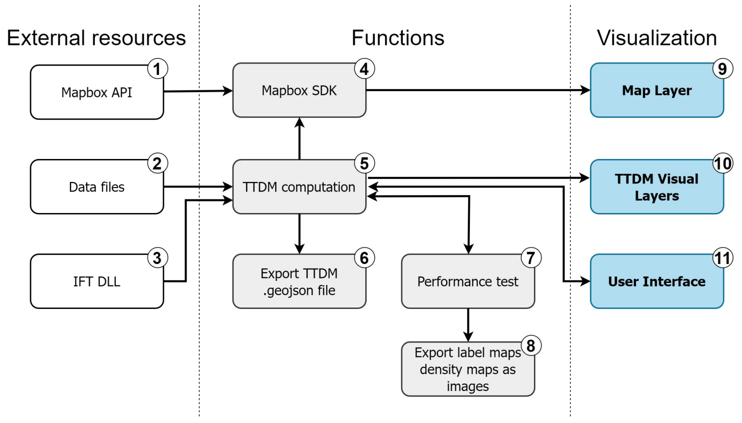


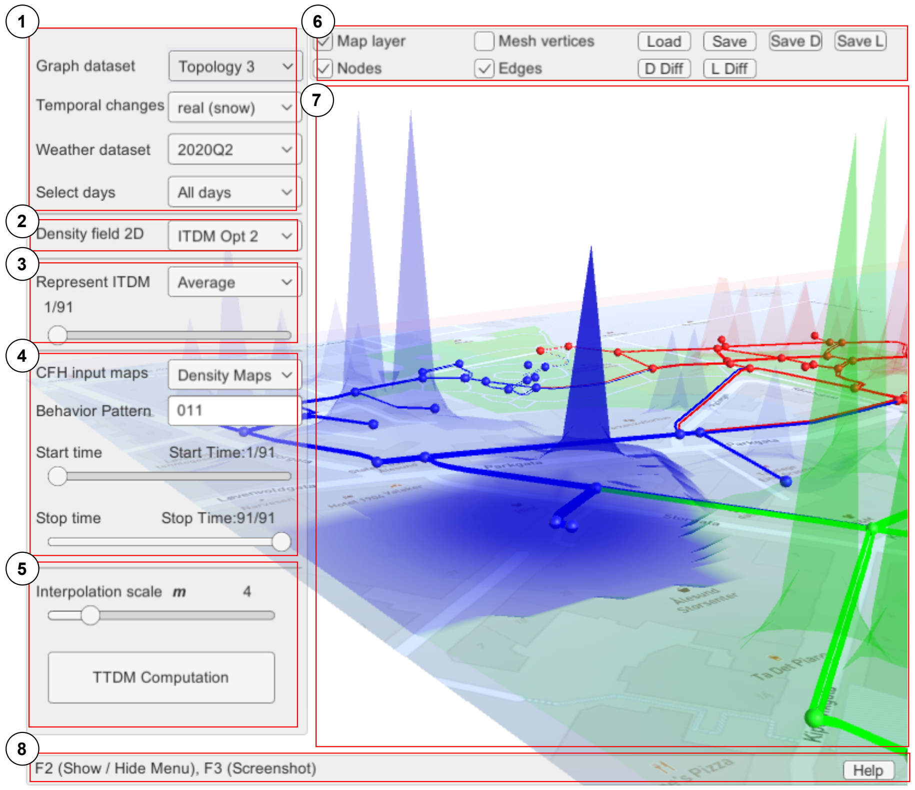
Figure 6.2: The implementation architecture of the TTDM computation analysis tool (Desktop).

dataset, and the daily data filter (e.g., "all days," "weekdays," "weekends") in the first Panel (Label 1). There are three choices in the drop-down menu for the temporal change encoding methods. Available options include to import predefined temporal changes, create random temporal changes, and compute the simulated temporal changes based on real weather data (e.g., in the current version, snow data). The second one (Label 2) provides three choices to estimate density field on 2D planar surface TTDM with running TDM, ITDM option 1 and option 2 (Section 4.3). It is also possible to use a slider to select any timestamp ITDM as representative. Another available option refers to the definition of the representative based on an average function (Section 5.2) on the third panel (Label 3). The fourth panel (Label 4) includes the parameter configuration related to the computation of a change frequency heatmap (Section 5.3). It allows to choose density or label maps as the input of the CFH algorithm, the change binary pattern, and the time range (by the sliders). The TTDM computation (Label 5) will be started after the choice of the interpolation scale. The top-right menu (Label 6) includes the display control of the visualization components: loading and saving of the TTDM computation result, saving density maps and label maps as figures, and executing the performance test. In the center region (Label 7), there is an area to display the 3D visualized results with the fly control camera by the mouse. At the bottom (Label 8), it shows some hot keys information and one help button.

TTDM Visualization Analysis

The software prototype TTDM visualization analysis tool (Web server) is designed for the visualization analysis. Its main functionalities are::

1. Decoding of the TTDM computation result (GeoJSON file) for the visualization analysis.
2. Filtering the temporal changes with customized parameters.
3. Visualizing all data in the geographic coordinates system.



Graph dataset: Topology 3
Temporal changes: real (snow)
Weather dataset: 2020Q2
Select days: All days
Density field 2D: ITDM Opt 2
Represent ITDM: Average
1/91
CFH input maps: Density Maps
Behavior Pattern: 011
Start time — Start Time:1/91
Stop time — Stop Time:91/91
Interpolation scale *m* 4
F2 (Show / Hide Menu), F3 (Screenshot)

Figure 6.3: Screenshot of the TTDM computation analysis tool (Desktop) created for TTDM computation analysis.

4. Providing a more user-friendly interface with some features, such as the object track function by mouse hover and click, the playback function of the ITDM for each timestamp, the camera positions synchronization, etc.

Figure 6.4 shows an overview of its implementation. Mapbox API (Label 1) is accessed by the Mapbox GLJS (Label 3) to create all visual layers (Label 5). The TTDM GeoJSON file (Label 2) is uploaded for the temporal change customized filter (Label 4). This filter implements the main functions based on the settings defined in user interface (Label 6) and exports the data to visual layers by Mapbox GLJS.

Figure 6.5 presents a screenshot of the user interface of the TTDM visualization analysis tool (Web server) accessed by a web browser. There is one toggle menu (Label 1) on the top-left. It includes two main tab pages (Label 2) “Load” and “Layers.” The “Load” page (Label 3a) allows the users to choose a remote TTDM GeoJSON file on the list directly or one local file to upload before the visualization analysis. It also supports the playback function with the sliders for the selected timestamp data and the waiting time for the load of each timestamp. Different options are available for encoding temporal changes, such as the selection of CFH input data (density map or label map), the optional metric functions (change, increase, decrease), the threshold

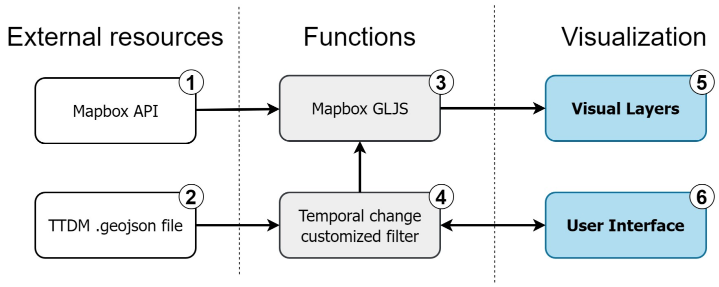
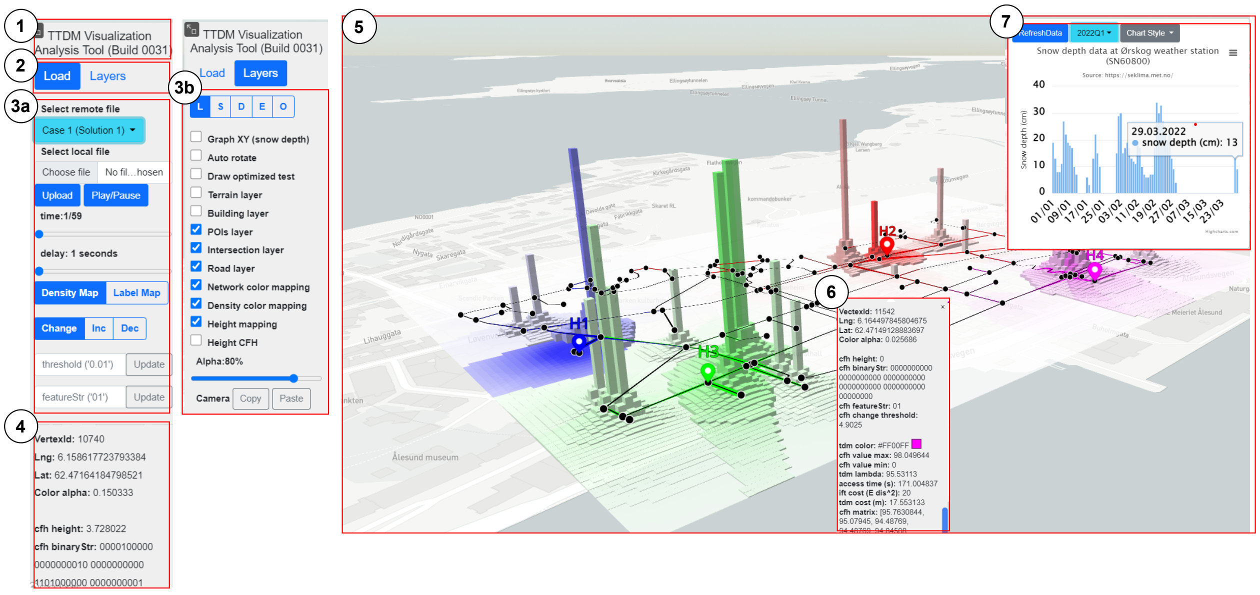


Figure 6.4: The implementation architecture of the TTDM visualization analysis tool (Web server).



Select remote file
Case 1 (Solution 1)
Select local file
Choose file No fil…hosen
time:1/59
delay: 1 seconds
threshold ('0.01')
featureStr ('01')

☐ Graph XY (snow depth)
☐ Auto rotate
☐ Draw optimized test
☐ Terrain layer
☐ Building layer
☑ POIs layer
☑ Intersection layer
☑ Road layer
☑ Network color mapping
☑ Density color mapping
☑ Height mapping
☐ Height CFH
Alpha:80%

VertexId: 10740
Lng: 6.158617723793384
Lat: 62.47164184798521
Color alpha: 0.150333

cfh height: 3.728022
cfh binaryStr: 0000100000 0000000010 0000000000 1101000000 0000000001

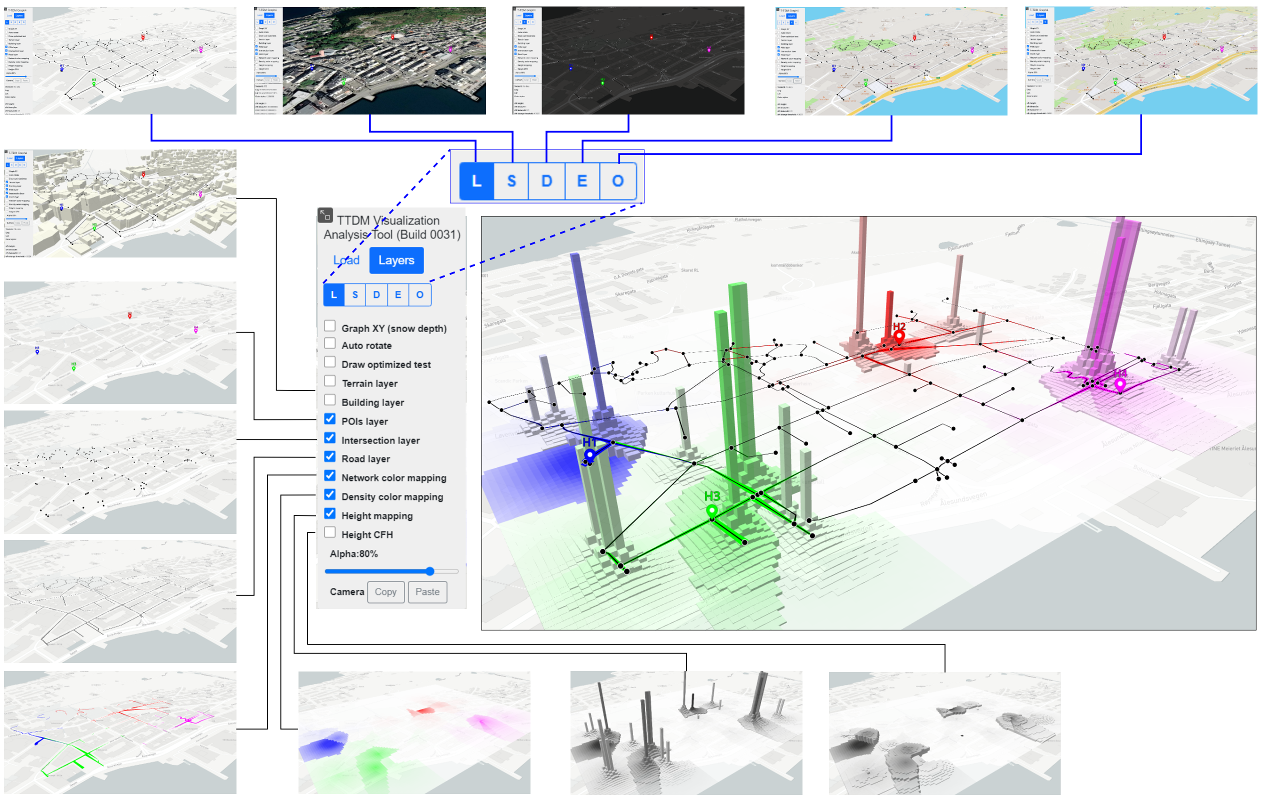
VectexId: 11542
Lng: 6.164497845804675
Lat: 62.47149128883697
Color alpha: 0.025686

cfh height: 0
cfh binaryStr: 0000000000 0000000000 0000000000 0000000000 0000000000 00000000
cfh featureStr: 01
cfh change threshold: 4.9025

tdm color: #FF00FF 🟪
cfh value max: 98.049644
cfh value min: 0
tdm lambda: 95.53113
access time (s): 171.004837
ift cost (E dis^2): 20
tdm cost (m): 17.553133
cfh matrix: [95.7630844, 95.07945, 94.48769,

Figure 6.5: Screenshot of the TTDM visualization analysis tool (Web server) created for TTDM visualization analysis.

for define binary maps, and the change binary pattern. The “Layers” page (Label 3b) provides more options for advanced visualization analysis. It includes layers to control the different map layer views provided by Mapbox GLJS, components of the visual integrator (Section 5.4.4). In addition, some features like the camera position synchronization by clicking copy and paste button and the transparency alpha adjustment are also provided on this page. The information on the object information panel (Label 4) is updated when the mouse moves on the map in the center main region (Label 5). This center main region displays the final visualization result on the map, supporting pan and zoom operations with the mouse. The object screenshot panel (Label 6) is activated after the left-click of the mouse. This panel makes it easier to compare the data of one marked position with another one shown in the object information panel (Label 4). The snow depth data may also be shown with various charts styles if the user activates the option "Graph XY (snow depth)" on the “Layers” page (Label 3b).



TTDM Visualization Analysis Tool (Build 0031)

Load Layers

L S D E O

- [ ] Graph XY (snow depth)
- [ ] Auto rotate
- [ ] Draw optimized test
- [ ] Terrain layer
- [ ] Building layer
- [✓] POIs layer
- [✓] Intersection layer
- [✓] Road layer
- [✓] Network color mapping
- [✓] Density color mapping
- [✓] Height mapping
- [ ] Height CFH

Alpha:80%

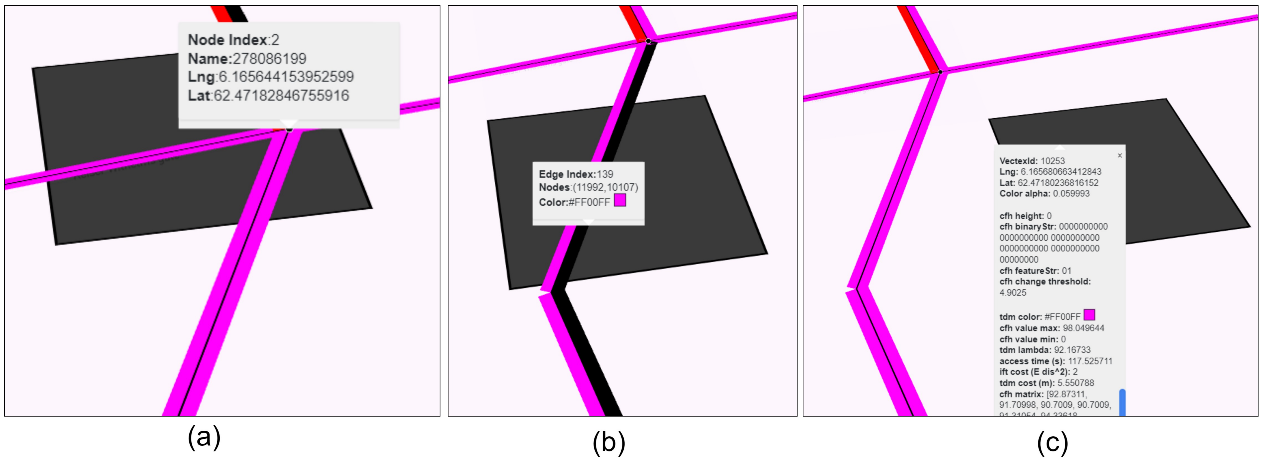
Camera Copy Paste

Figure 6.6: Visual effects of the default example after the initialization.

Figure 6.6 illustrates the visual effects of one example with the default configuration. For instance, the map style options include light (L), satellite (S), dark (D), street (E), and outdoor (O). All of them are provided by Mapbox GLJS. The default configuration is the light map style. It also supports the display of the terrain and building data as the additional features with selected map style. Next, three data layers (POIs layer, intersections layer, and roads layer) will responsively show POI nodes, intersections (nodes), and roads (edges). Here we used labels in the interface of the mobility applications instead of the algorithm to support future user studies with domain experts. At last, the three main modules (network color mapping, density color mapping, height mapping) of the visual integrator can be configured. Also, it is possible to choose alternatives to display the CFH result using height information. The center region of the figure contains the visual result related to the integration of these selected multiple visual layers. Here, we used the extrusion of polygon[13] to create the 3D bars array instead of 3D mesh as an alternative approach to visualize temporal changes. Compared with 3D mesh, it is easier to recognize the elevations accurately in our designed aspect.

Figure 6.7 is one example of the object track function. It tracks and displays the object hovered by the mouse. The priority from high to low is node, edge, and point. In TTDM GeoJSON

[13]https://docs.mapbox.com/mapbox-gl-js/example/3d-extrusion-floorplan/ (As of May 2022).



Node Index:2
Name:278086199
Lng:6.165644153952599
Lat:62.47182846755916

Edge Index:139
Nodes:(11992,10107)
Color:#FF00FF 🟪

VectexId: 10253
Lng: 6.165680663412843
Lat: 62.47180236816152
Color alpha: 0.059993

cfh height: 0
cfh binaryStr: 0000000000 0000000000 0000000000 0000000000 0000000000 00000000
cfh featureStr: 01
cfh change threshold: 4.9025

tdm color: #FF00FF 🟪
cfh value max: 98.049644
cfh value min: 0
tdm lambda: 92.16733
access time (s): 117.525711
ift cost (E dis^2): 2
tdm cost (m): 5.550788
cfh matrix: [92.87311, 91.70998, 90.7009, 90.7009,

Figure 6.7: Visual effects of the object track function for node (a), edge (b), and point (c).

file, all information is only saved in the geographic coordinates. TTDM visualization analysis tool (Web server) is a prototype of visualizing and analyze the TTDM computation result using GeoJSON format. It means it is also feasible to visualize the TTDM computation result by the programming script in any software that supports GeoJSON.

6.2 Performance and Qualitative Assessment

This section addresses research question 3, which concerns the quantitative and qualitative assessment of algorithms. To address this research question, two experiments were conducted:

1. Assessment of the efficiency of ITDM in terms of computation time;
2. Assessment of produced density maps and label maps when compared with the algorithm proposed by Feng et al. [17].

Section 6.2.1 describes the evaluation protocols in detail, while Section 6.2.2 presents and discusses obtained results.

6.2.1 Evaluation Protocol

Evaluated methods: With regard the efficiency assessment, we compare the computational time of three approaches: TDM [17] and our two formulations (ITDM option 1 and ITDM option 2) – see Section 4.3 for more details. Each algorithm was run five times and we report the average time and the standard deviation. The assessment includes different scenarios, such as distinct resolution scales, temporal changes for various time lengths, different networks, and diverse number of POI nodes. The assessment refers to the computation of TTDMs.

The qualitative assessment relies on the comparison of label and density maps with the ones produced with the original algorithm [17]. For the label maps, we directly compute whether the POI labels are different for each pixel when using ITDM or TDM [17].

For density maps, the indicator (Percentage difference) $\%D$ is used to compare ITDM results with the TDM result. The percentage difference $\%D$ for any pixel p in the map is the absolute value of the difference divided by the original value times 100 as Equation 6.1. Λ_{TDM} and Λ_{ITDM} are the density maps created by TDM and ITDM algorithms, respectively. The visual effects are encoded into a PGM P5 image[14]. It displays the gray scale colors with the maximum value (white color) and the minimum one (black color) for each pixel. The same format and method are also suitable for the label maps.

$$\%D(p) = \left| \frac{\Lambda_{TDM}(p) - \Lambda_{ITDM}(p)}{\Lambda_{TDM}(p)} \right| \times 100 \tag{6.1}$$

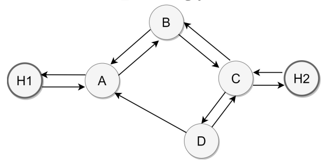
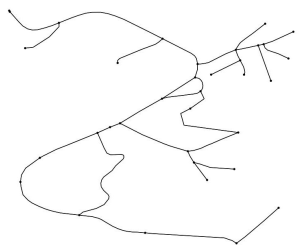
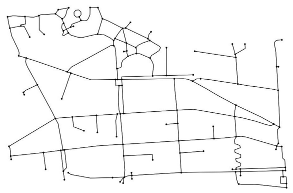
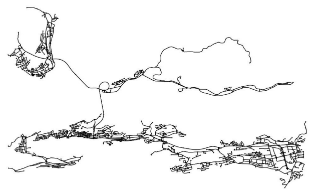
Datasets: Four topologies (road network) are utilized as in the performed assessments. Table 6.1 presents their main attributes as well as a mini graph view.

Topology 1 is the simplest one and contains 6 nodes and 11 edges. This topology is also the primary running example discussed in the previous chapters. Therefore, it has no corresponding actual network type and geographic region coordinates. The other three topologies are real networks with different numbers of nodes and edges. There are no predefined POI nodes in the performance assessment. Topologies 2 and 3 refer to geographic regions associated with a walkable road in the Ålesund centre. Topology 4, in turn, is composed of the main drive roads and intersections in the Ålesund municipality, Norway. Topology 1 was encoded as a default sample using a Unity C# script. The other three graphs were downloaded from OpenStreetMap first and encoded into a CSV file (Section 6.1).

Dataset Preprocessing: We implemented the dataset preprocessing using Python and OSMnx package. The created script downloads the drive and walk network data for a given bounding geographic regions. Each intersection includes the name, longitude, and latitude attributes. Intersection points will be processed as one node of TTDM algorithm. Every road involves the road length, max speed limitation, intersections, etc. A road will be processed as one edge. Considering the ideal traffic condition and lack of real traffic data for most roads in Ålesund, we assume that vehicles could reach the maximum speed limited by the roads. Recall that the edge cost is the access time. For the drive network, the access time is defined as the shortest driving time on the road, which is the max speed limitation divided by the road length. For the walk road network, the edge cost is the pedestrian average walk speed (5 km/h) divided by the road

[14]`http://netpbm.sourceforge.net/doc/pgm.html` (As of May 2022).

Table 6.1: The road networks considered in the conducted validation. Coordinates are encoded in degrees.

Name and Topology	Type	# Nodes	# Edges[a]	Mobility application	Geo region (North, South, East, West)
Topology 1 	default	6	11	-	-
Topology 2 	real	35	74	walk	62.4772180°, 62.4730731°, 6.1866435°, 6.1942393°
Topology 3 	real	130	320 (324)	walk	62.4736239°, 62.4710819°, 6.1582230°, 6.1675138°
Topology 4 	real	1553	3383 (3427)	drive	62.5262490°, 62.4526394°, 6.0807169°, 6.3748647°

[a] The value with brackets is the edge number in the raw dataset.

length. Random temporal changes are used for the simulation of access time variations in the experiments.

Configuration setting: To simplify the implementation with visual integrator, we set the vertex number of the mesh on Mapbox SDK is the same as the image size in IFT calling. All parameters

Table 6.2: The different resolution scales.

Scale m	**Mesh size**	**Image size**	**Number of pixels**
1	36×27	37×28	1K
4	144×81	145×82	12K
10	360×270	361×271	98K
20	720×540	721×541	390K

related to the algorithm resolution scale m are defined as presented in Table 6.2.

Machine: The experiments runs on a Lenovo Xiaoxin Air 14 (2021) 2.4GHz i5-1135G7 with a Nvidia GeForce MX450 graphs card and 16GB RAM memory.

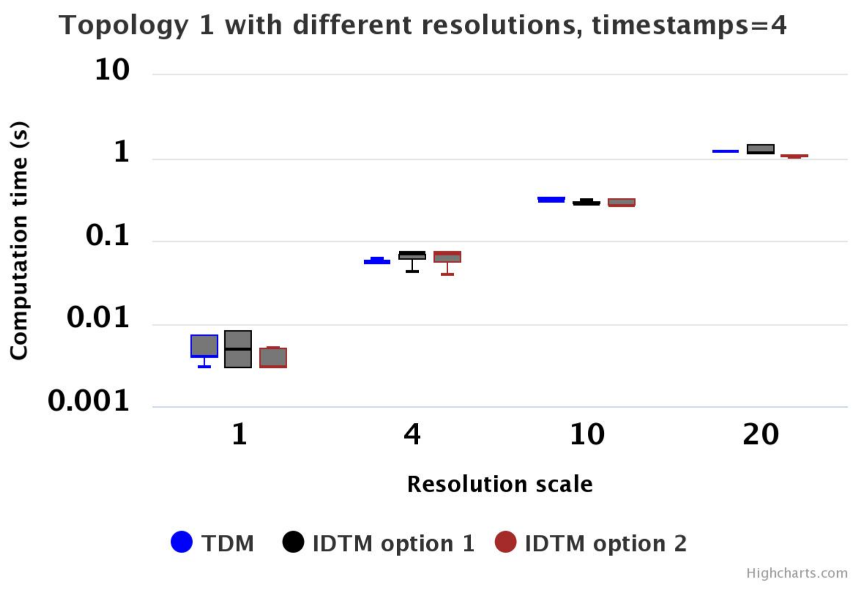
6.2.2 Results and Discussion

Efficiency Assessment: Figure 6.8 shows the result running on the road network Topology 1(refer to Table 6.1 for the definition) with different resolution scale (Figure 6.8a) and different timestamps (Figure 6.8b). Recall that computation time is represented in a logarithmic scale.

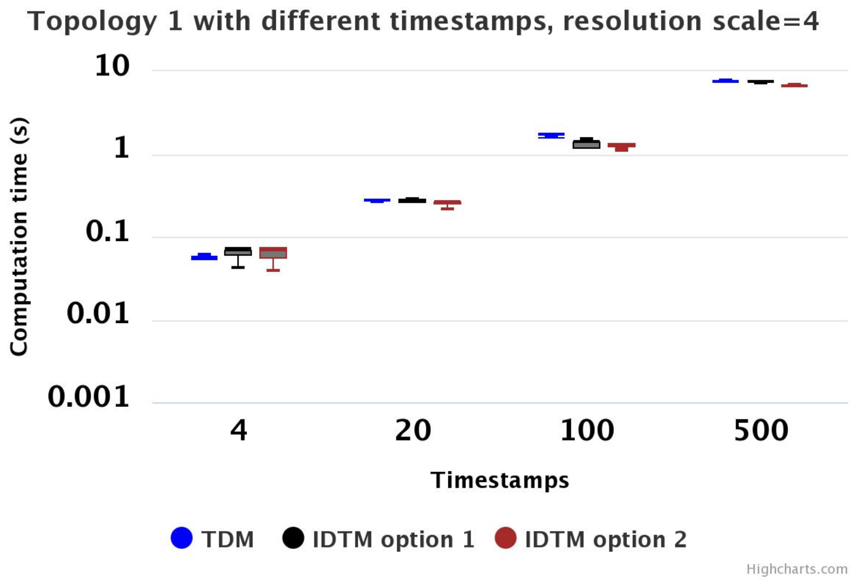
Topology 1 is the simplest network in Table 6.1. For this topology, TDM, ITDM option 1, and ITDM option 2 spend similar computation time and have less deviation as the resolution scale increases. ITDM option 1 and ITDM option 2 are a bit faster than TDM when the data time series is longer. The resolution scale m is set to 4 for the remaining experiments.

Figure 6.9 presents the average computation time of TTDM with 20, 100, 500 timestamps with random temporal changes. Reported results refer to the use of Topologies 2 (Figure 6.9a) and 3 (Figure 6.9b). The definition of Topology 2 and Topology 3 are found in Table 6.1. ITDM option 1 and ITDM option 2 are significantly faster than TDM for these two topologies. The worst case is Topology 2 with 20 timestamps, and the computation time of ITDM options is at least **3.1** times faster than the one of TDM.

To evaluate the performance of the algorithms for different numbers of POIs (from 3 to 100), we compute TTDM on Topology 3 (130 nodes) and Topology 4 (1553 nodes)(refer to Table 6.1 for their definitions) with 100 timestamps. Results are reported in Figure 6.10. ITDM option 1 (black bars) and ITDM option 2 (brown bars) lead to stable and less computation time than TDM (blue bars) for these two topologies. The number of non-POI nodes for Topology 3 is close to the number of POIs for networks when we randomly select 50 and 100 POIs. Figure 6.10a shows that the TDM (blue bars) is less costly when the number of POIs increases. However, when the number of nodes is much larger than the number of POIs as in Topology 4, TDM in TTDM shows the stable computation time for different numbers of POIs (see blue bars) in Figure 6.10b.



(a)



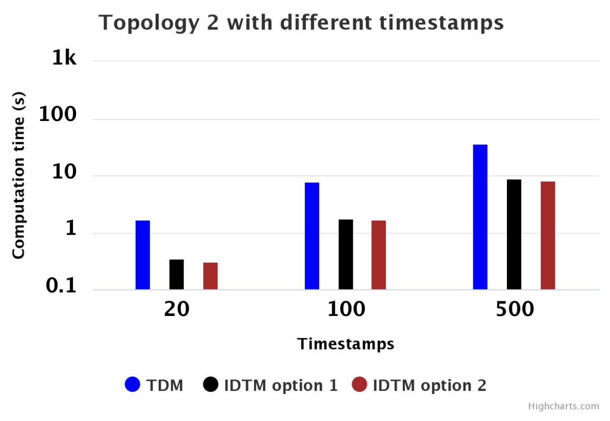
(b)

Figure 6.8: TTDM computation time on Topology 1. Figure (a) is the result with the same four timestamps and Figure (b) reflects the computation time for various timestamps in the same resolution scale.

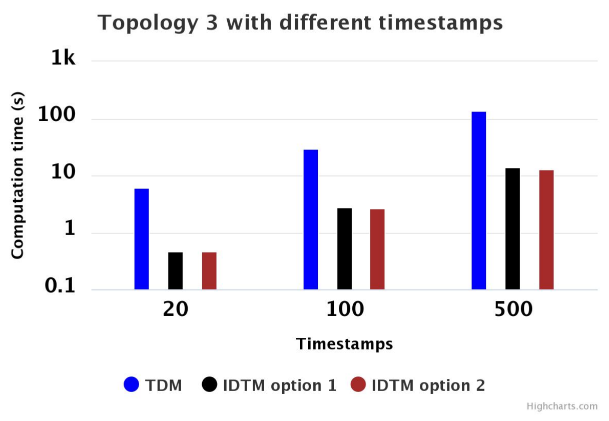
In summary, performed experiments show that:

1. The network complexity and the number of POI nodes greatly affect the computation time.

2. ITDM option 1 and ITDM option 2 have similar efficiency, and ITDM option 2 is a bit faster than ITDM option 1. For simple networks, computation time of the three options

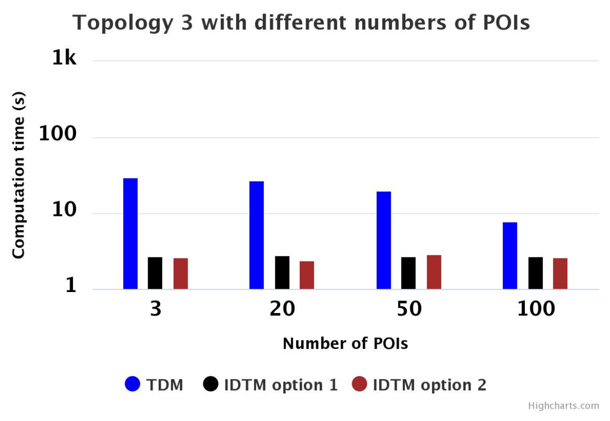


(a)

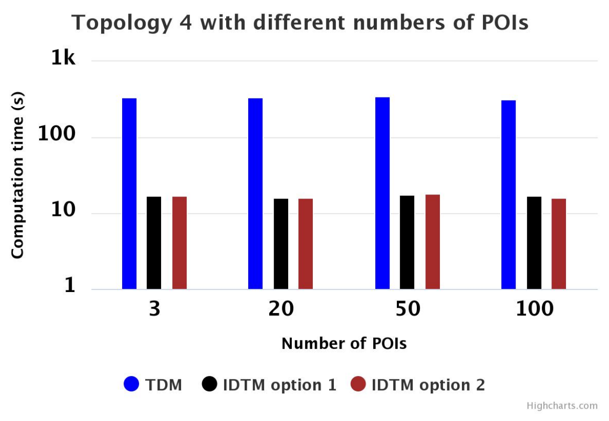


(b)

Figure 6.9: TTDM computation time on Topology 2 and Topology 3 with different timestamps.



(a)



(b)

Figure 6.10: TTDM computation time on the Topology 3 and Topology 4 with different numbers of POIs.

are almost the same even for long data time series.

3. ITDM option 1 and ITDM option 2 are much faster (at least **3** times) for complex networks, especially when long data time series are considered.

Qualitative Assessment: The label and density maps are the two main outputs for ITDM and TDM. Again, the assessment considers the two variations of the proposed algorithm. Both ITDM option 1 and ITDM option 2 use the IFT root map to compute the label map. ITDM option 2 computes the density map based on IFT cost map. The two simpler networks, Topology 1 and Topology 2, are used in our assessment (Section 6.2.1) as they make it easier to visualize the differences with the maps produced by the TDM algorithm [17].

Visual results are presented in Figure 6.11. As we can observe, there are four regions in the

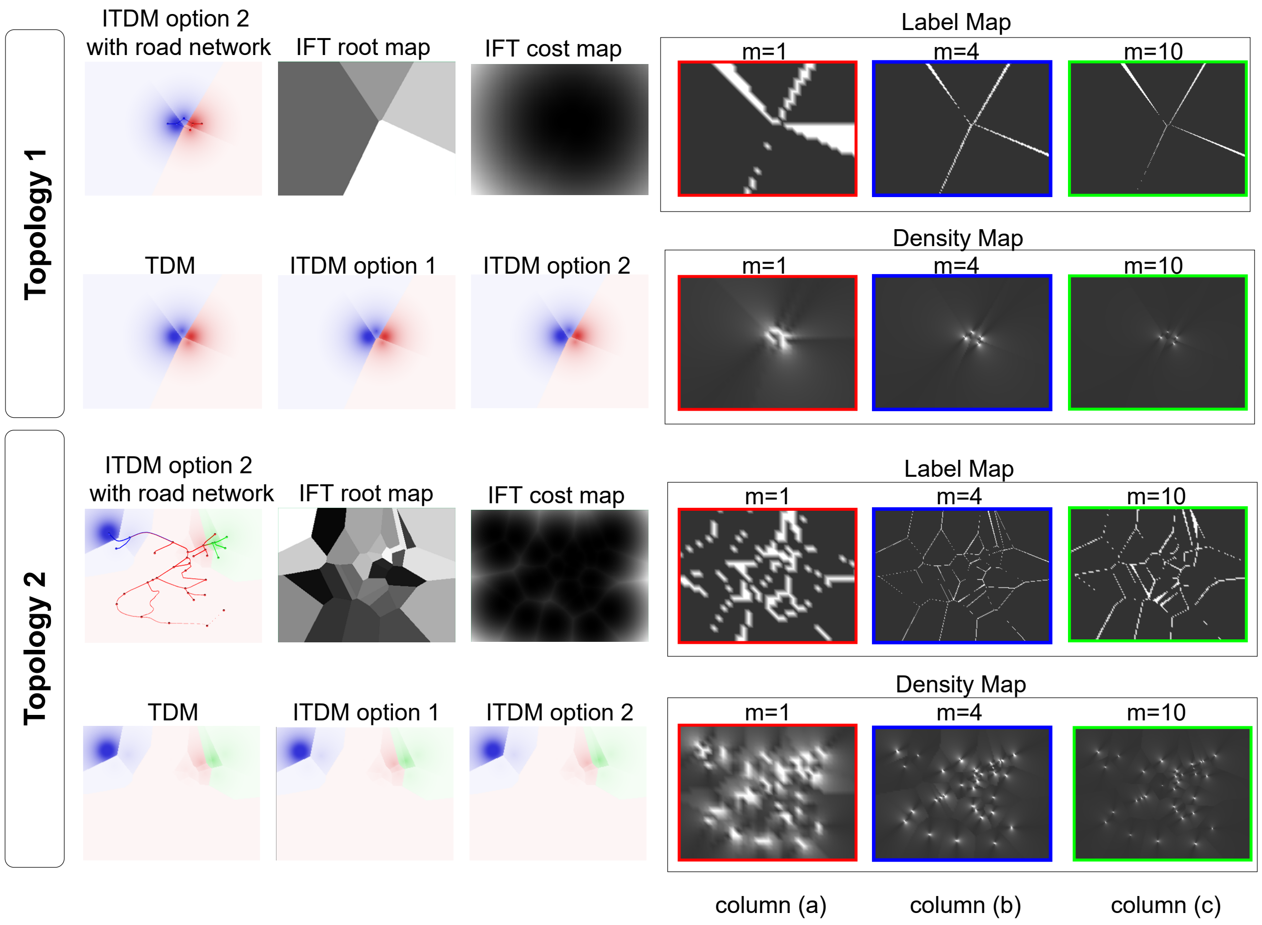


Figure 6.11: Value difference of label maps and density maps on Topology 1 and Topology 2 with different resolution scales.

IFT root map for Topology 1 because there are four non-POI nodes (two were chosen as POI nodes). For the label maps, there are less white color pixels (different label values for each pixel position) with the increasing resolution scale. The difference of the results computed by $m = 1$ (column (a), highlighted with red borders) and $m = 4$ (column (b), blue borders) are much obvious than the change between $m = 4$ and $m = 10$ (column (c), green borders). Meanwhile, the visual results for TDM, ITDM option 1, and ITDM option 2 are almost the same. As our analysis of the approximate position calculation for the nodes in Section 4.2.1, the value difference of distance is larger for regions close to nodes. Since the density map value is computed based on this distance, the changes also happen in the same area. For Topology 2, the results follow the same trend.

In short, the qualitative assessment can be summarized as follows:

1. Changes in label maps are more frequent on the borders and around pixels closer to seed regions.

2. Differences in density maps are more evident in regions close to nodes.

3. Increasing the resolution scale leads to a decrease in the percentage difference. According to the results, we recommend the use of a resolution scale equal to 4 (column (b), blue borders in Figure 6.11), as no significant differences are observed when $m \geq 4$.

6.3 Case Studies

This section presents compelling case studies concerning the use of TTDM in two different mobility-based applications. Section 6.3.1 introduces the datasets used in the case studies. Section 6.3.2 describes the first application, which refers to improve walkability of relevant regions in Ålesund, Norway. The second one (Section 6.3.3) concerns to speedup response time of emergency services.

6.3.1 Datasets

Real datasets: Topology 3, introduced in Table 6.1, is selected as our road network for Case Study 1. We retrieve four public service places as the POI nodes: $H1$, $H2$, $H3$, $H4$ (for more details refer to Table A.1).

For Case Study 2, we choose Topology 4 as the road network and three locations ($H101$, $H102$, $H103$), associated with emergency services facilities in Ålesund, Norway, selected as POI nodes (Table A.2).

For both cases, we used the snow depth history data at the Ørskog weather station (SN60800) from the Norwegian Climate Service Center.[15] It covers the time period from 01.01.2020 to 31.03.2022 and is separated into nine quarters (2020Q1-2022Q1). Any quarterly dataset can be filtered according to weekdays (Monday, Tuesday, Wednesday, and Thursday), weekends (Friday, Saturday, and Sunday), or all days (Monday - Sunday). By default, the data related to all days in 2022Q1 works as our primary data source.

Simulated data: Our research assumes that the average moving speeds of the people and vehicle vary linearly with the snow depth on the road, following the assumption of Wang et al. [55]. Therefore, when the snow depth is greater, the edge cost (time) will increase with the slower average speed.

Among the nearby weather stations having snow data in Ålesund municipality, the Ørskog weather station is the closest one to the Ålesund center. Therefore, we assume the snow depth linearly

[15] https://seklima.met.no/ (As of May 2022).

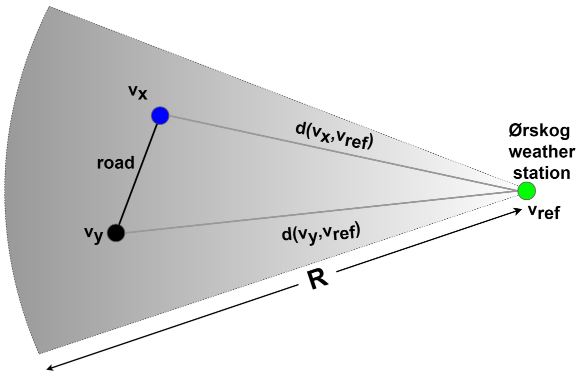


Figure 6.12: The assumption of the snow depth on the roads.

Table 6.3: Parameters for simulated temporal changes by snow depth.

Case name	Network	Reference speed (km/h) S_{ref}	Speed decreasing rate (km/(h · cm)) k_r	Min speed (km/h) S_{min}
Case Study 1	walk	5	0.1	2
Case Study 2	drive	$0.7 \times S_{max\_road}$	2.5	5

declines until 0 cm along the radius from the Ørskog weather station (highlighted as the white color) to the coast (the grey color), as illustrated in Figure 6.12.

Let the ordered pair (v_x, v_y) be an edge representing a road (the black solid line with the blue point v_x and the black one v_y). The vertex in the position of the Ørskog weather station is v_{ref} (the green point) and the function d is the geographic distance between two positions. If the snow depth at the Ørskog weather station is D_{ref}, we could assume that the snow depth D_{snow} on this road will be computed as defined in Equation 6.2:

$$D_{snow} = D_{ref} \times \frac{d(v_x, v_{ref}) + d(v_y, v_{ref})}{2R} \tag{6.2}$$

The average moving speed on this road S_{avg} is calculated by Equation 6.3, where S_{ref} is the reference speed, k_r is the speed decreasing rate assumed, and S_{min} is the slowest speed.

$$S_{avg} = \max(S_{ref} - k_r D_{snow}, S_{min}) \tag{6.3}$$

The affection radius R is set as 50km. S_{max_road} is the max vehicle speed allowed on each road. Table 6.3 summarizes the parameter settings for the different case studies.

The cost of a road will be computed by the distance divided by the average moving speed S_{avg}, which will construct the temporal changes of the network based on the snow data.

6.3.2 Case Study 1: Walkabaility Analysis

This case refers to mobility analyses towards improving walkability of relevant regions of a city. Specifically in this case, we assess how weather conditions affect the walkability in different areas of the city. Walkability is a representation of how easy is move in a path and thus We assume that the most walkable path is the one with the lowest access time, i.e., the path with the largest values in density map. We proposed that people will choose shorter path witch is particular important for people with reduce mobility such as elderly or mobility impaired persons.

Furthermore, the high temporal changing score for one location, which can be encoded in a change frequency map (Algorithm 3), means that the walkability of that region changes very frequently. In this case, people may change their destination frequently or at least they will hesitate to choose the desired destination. In this scenario, urban planners would prefer to keep the smaller values of change frequency map (less temporal changes) when looking for the optimal position for a new shopping centre for example.

Figure 6.13 details the target area considered in the case. There are two existing indoor public service places considered as POI nodes: $H1$ (highlighted in blue) and $H2$ (red). Planners may decide to retrofit some malls to build a new public service center and try to select a suitable location from two options. In the figure, they refer to $H3$ (green) and $H4$ (purple) as POI nodes. Therefore, they will choose one solution from Solution 1 ($H1$, $H2$, $H3$) and Solution 2 ($H1$, $H2$, $H4$). Furthermore, they are more concerned about the impact in the sample area (emphasized with the blue dotted line border in the top-left corner). The three small regions in this area are Region ① ($N2$,$N3$,$N4$,$N5$,$N1$), Region ②($N2$,$N6$,$N7$,$N3$) and Region ③($N2$,$N8$,$N9$,$N6$). Regions ① and ② are close to $H1$ and $N3$. The geographic information about these marked nodes $N1$-$N9$ are available in Table A.1.

The rest of this section is organized as below. The overall visual effects will be presented with all POIs ($H1$, $H2$, $H3$, $H4$) at first. After that, we discuss the result of TTDM after computing the representative ITDM (Section 5.2), which reflects the average walkability affected by the weather condition in 2022 Quarter 1. Next, TTDM temporal change (Section 5.3) results with different binary pattern string will be analyzed. Finally, we discuss different analysis scenarios related to the comparison of the results after filtering the daily data or select different time periods.

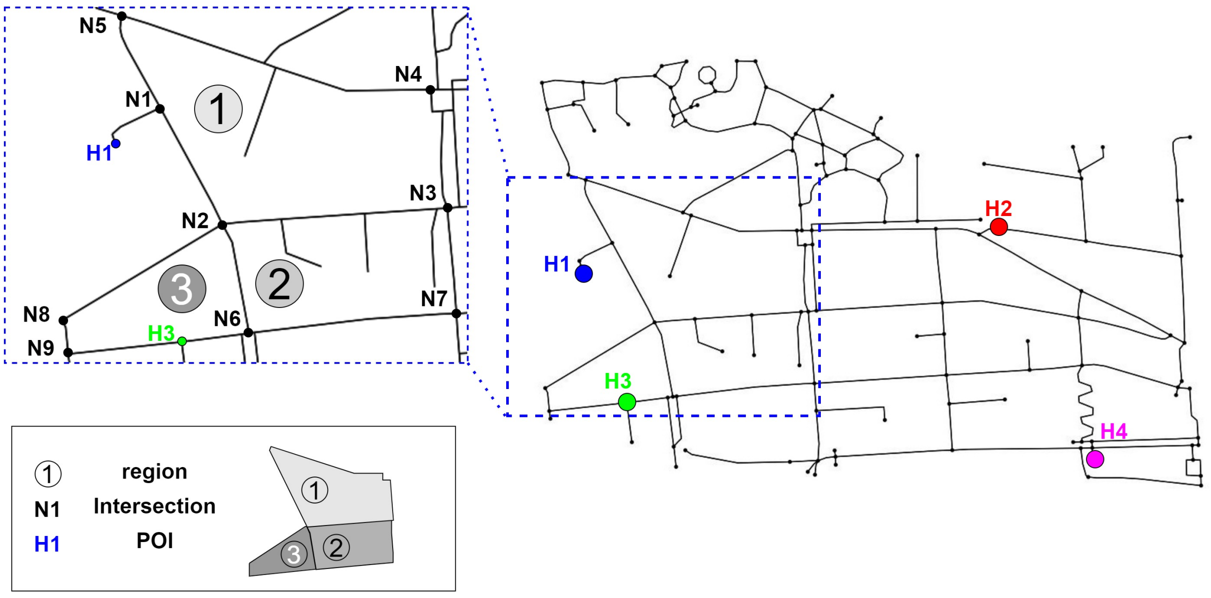
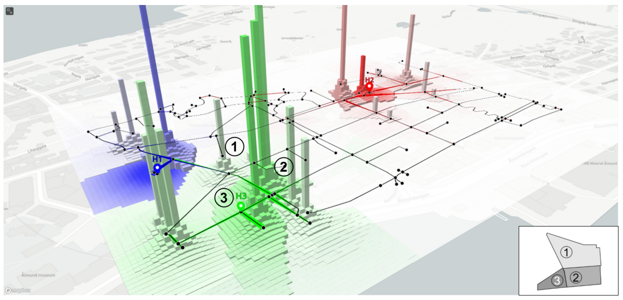
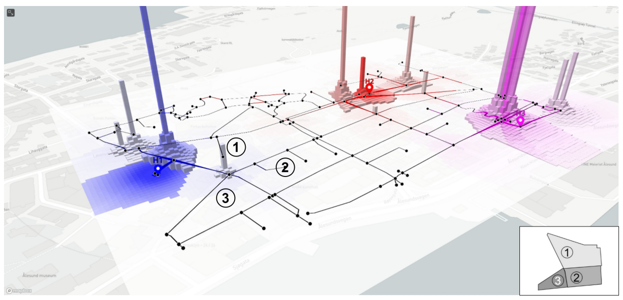


Figure 6.13: The interested sample regions in the Case Study 1. There are three small regions (① - ③) separated by the node $N2$ and neighbour nodes.



(a) Solution 1($H1$, $H2$, $H3$)

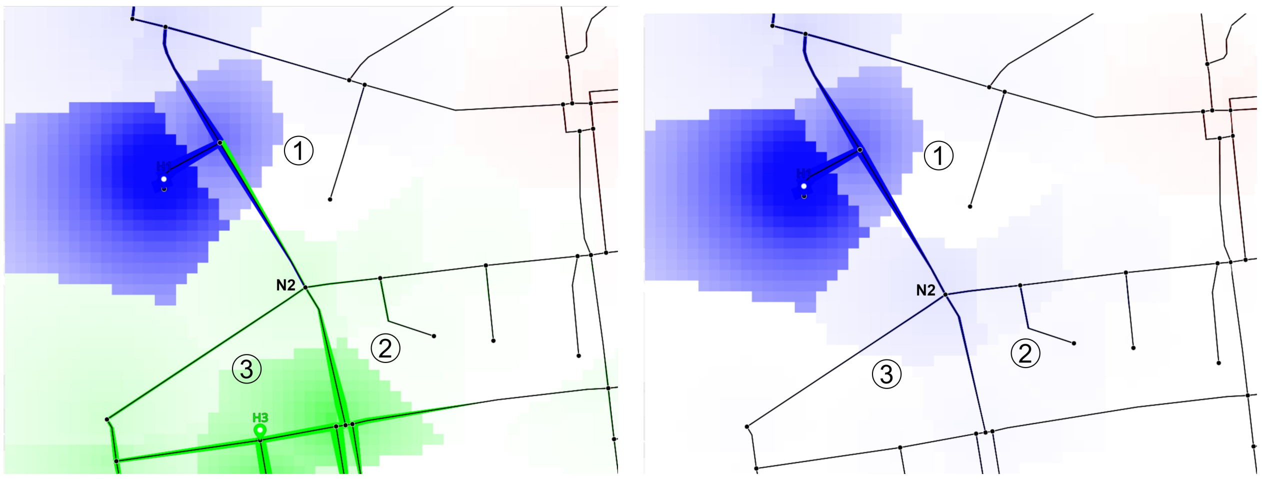


(b) Solution 2($H1$, $H2$, $H4$)

Figure 6.14: Overall visual effects for the two solutions

Overall visual effects

Figure 6.14 presents the overall visual effects for the two solutions when using density maps for temporal changes with the binary pattern string "01." The differences in color and height concentrate on the regions that are close to POI node $H3$ (green) and $H4$ (pink). For example, the density values in Regions ①,②,③ in Solution 1 (Figure 6.14a) are significant larger than Solution 2 (Figure 6.14b). That means the population in these regions could have shorter access time when the new center is built on $H3$ instead of $H4$. Therefore, Solution 1 is more suitable in general.



(a) solution 1($H1$, $H2$, $H3$)

(b) solution 2($H1$, $H2$, $H4$)

Figure 6.15: The results of the sample regions after computing the representative ITDM for the two solutions.

Computing a representative ITDM

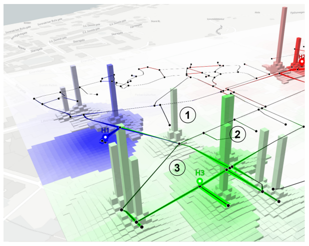
Figure 6.15 shows the results of computing representative ITDM for the defined sample regions. Region ③, which has the most pixels highlighted in green, is only affected by $H3$. This means the people in Region ③ has the highest walkability to the new public service center in location $H3$. Furthermore, the pixels and edges close to node $N2$ has the green color in Solution 1, which means people there will probably choose to walk to $H3$. These visual layouts further support the claim that Solution ① is more suitable.

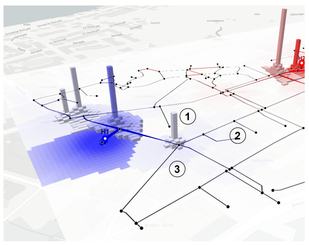
Encoding of temporal changes

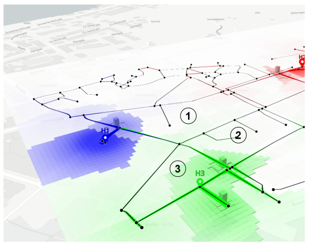
Figure 6.16 presents the TTDM layouts for two binary string patterns (“010” and “01110”). Figures 6.16a and 6.16b show more frequent temporal changes of pattern “010” compared to Figures 6.16c and 6.16d. This means the weather changed the walkability more frequently in only one day compared to three continuous days.

Suggestions for the decision maker – filtering by dates

With the use of filtering the daily data or comparing the results with different time periods, TTDM could support the choice of time period to operate the new public service center. For instance, the weekends in 2022Q1 have less temporal changes compared to the weekdays and all days in Figure 6.17. As Figure 6.18 shows, there are more frequent temporal changes in 2022Q1 among all nine quarterly time periods. There are almost no impact on walkability in some quarters caused by the weather data because there is nearly no snow in these quarters. In short,


(a) Pattern "010" Solution 1 ($H1$, $H2$, $H3$)


(b) Pattern "010" Solution 2 ($H1$, $H2$, $H4$)


(c) Pattern "01110" Solution 1 ($H1$, $H2$, $H3$)

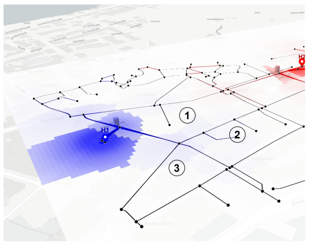

(d) Pattern "01110" Solution 2 ($H1$, $H2$, $H4$)

Figure 6.16: The TTDM results related to the encoding of temporal changes for two solutions considering different binary patterns.

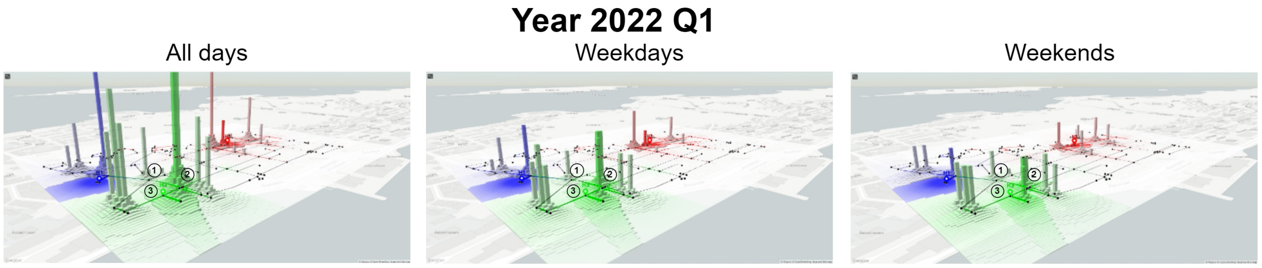


Figure 6.17: The comparison of filtering the daily data for different types of day.

according to this case, it is suggested to decide for $H3$ as new position of service center. If only considering the snow depth's impact, it is not a good idea to start the service in 2022Q1, especially for business days.

6.3.3 Case Study 2: Emergency Services

In this case, the objective is to speed up the response times in emergencies in Ålesund, Norway. Ålesund, one of the most beautiful city in Norway, is often selected as a destination for big cruise ships. These cruise ships stay in the city for one or two days and can bring from 2000 to 5000 passengers. During summer, Ålesund is visited often for more than one cruise and this number are expected to increase in the next few years and extend to the winter season. This large amount of visits have the potential to increase the demand of emergency services like ambulances, fire departments and/or police. Since Ålesund is an small city, the emergency services are limited and thus there is a risk of collapsing these services. Planning tools to arrange the tight resources of emergency services are therefore critical to secure their response time.

In the study of this case, we choose an example of the ambulances services offers in the Sentrum area of Ålesund. Figure 6.19, shows three health services providing ambulances. They are $H101$ (highlighted in blue), $H102$ (red) and $H103$ (green). $H101$ and $H102$ are close to the interested downtown area (latitude from 62.4691749° to 62.4729876° and longitude from 6.1496012°

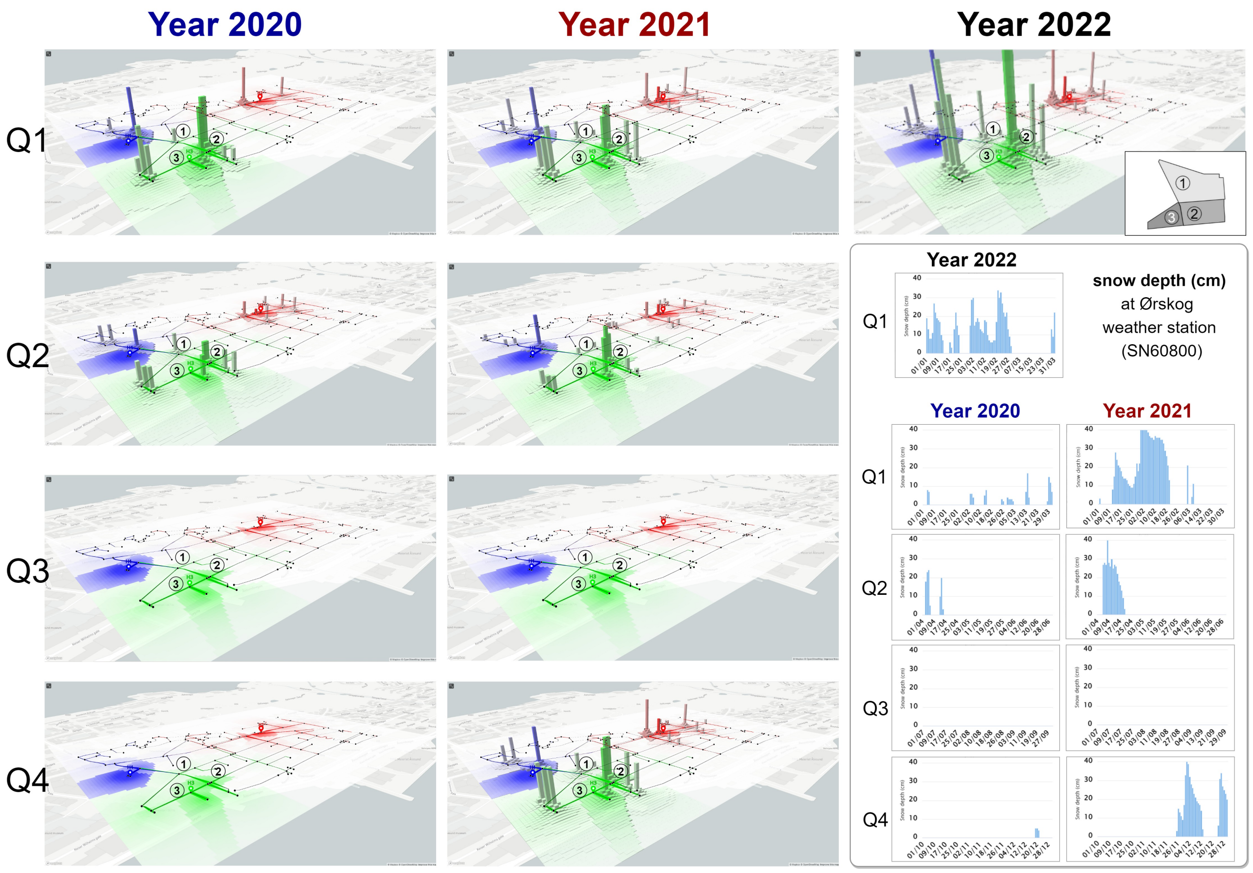


Figure 6.18: The comparison of various quarterly time periods with the binary pattern "01."

to 6.1605936°). From a planning perspective, city planners and emergency coordinators would need to know whether they should provide more resources at the time than one or more cruise ships arrive in Ålesund. In this case, the chosen areas of interest include Region ④ ($N101$, $N102$, $N103$) and Region ⑤ ($N102$, $N104$, $N105$). Region ④ is the port position for the cruise ships and Region ⑤ is a bus terminal nearby. The decreasing pattern of density value directly reflects the increasing access time. Therefore, discovering the recurring patterns of increasing access time in the regions of interest means that the pattern decreases with density maps.The results are presented as the same procedures as Case Study 1: overall visual effects, compute representative ITDM, encode temporal changes and suggestions for the decision maker.

Overall visual effects

Figure 6.20 presents the overall visual effects for the whole road network and the area of interest, in this case the downtown area (zoom in), when using density maps for temporal changes with the binary pattern string "01." Even though the whole road network is large, we could selected any small geographic area like downtown to compute the TTDM. It was not necessary to cover the location of the POI nodes in our computation targeted area for the density field estimation.

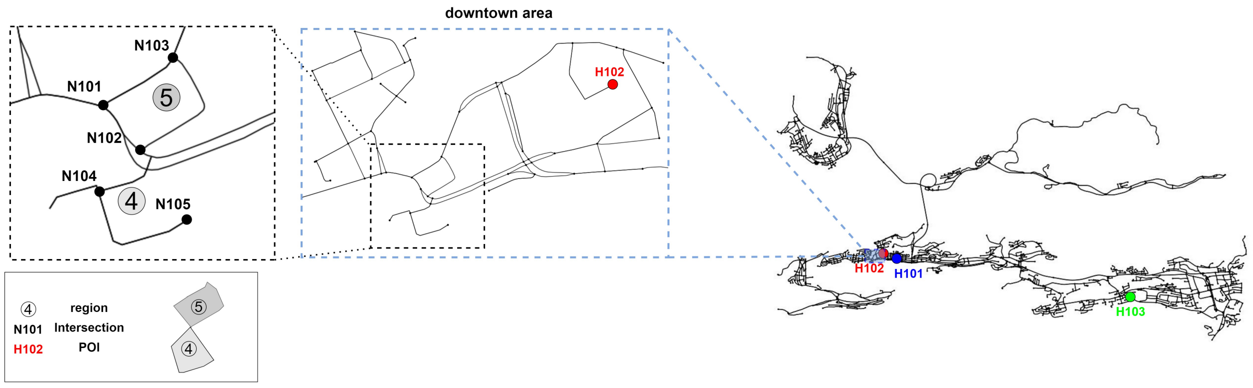
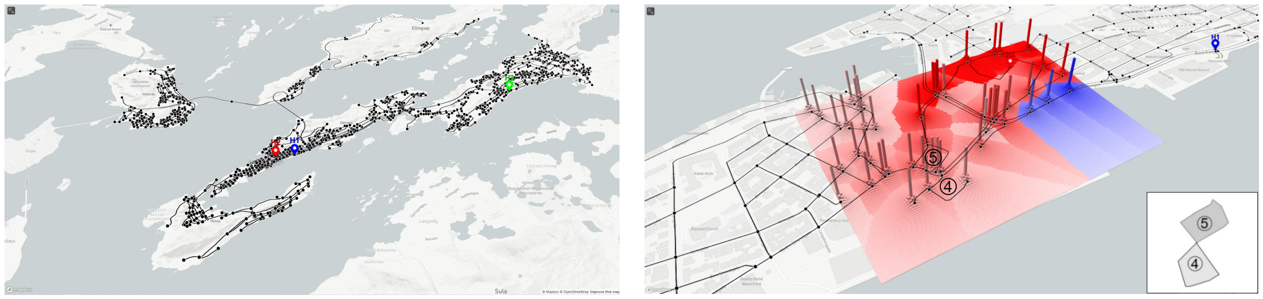


Figure 6.19: The interested sample regions in the Case Study 2. There are two small regions (④, ⑤) separated by the node $N102$ and neighbour nodes.



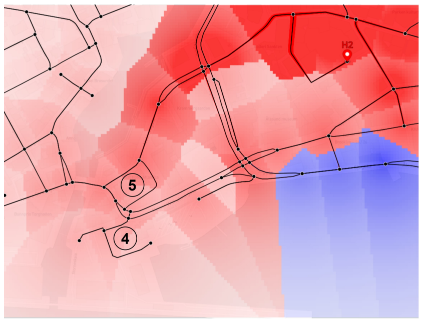
(a) the whole road network (b) the downtown area of interest (zoom in)

Figure 6.20: Overall visual effects for Case Study 2.

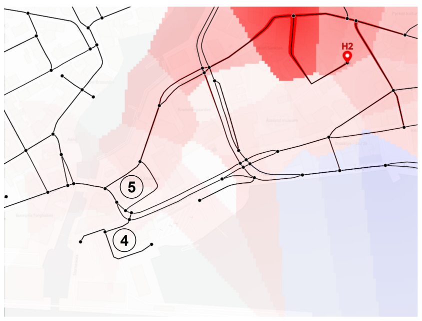
Figure 6.20b shows that Region ④ is closer to the emergency service center $H102$ with red color. This represents that it is only necessary for $H102$ to arrange the resources in Region ④. Furthermore, Region ⑤ has more red pixels, and it means the access time from $H102$ to Region ⑤ is shorter than to Region ④.

Computing a representative ITDM

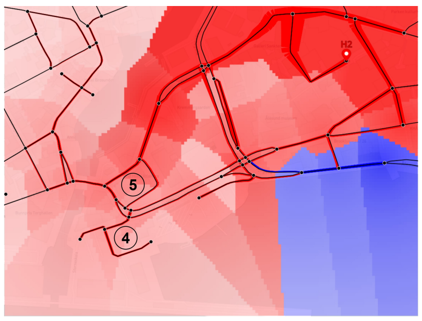
For the defined sample regions, Figure 6.21 presents the default result (average) after computing a representative ITDM as well as the ITDM result for the first five days. For all of them, Region ⑤ has shorter access time from $H102$ than Region ④ because Region ⑤ has larger density value with red color. If we compare the t1-t5 results, the density value for Region ④ continuously increases from t1 to t3, then keep the same at t4 and decreases at t5. Since the obvious temporal changes of density values in these Regions happen among the timestamps, it is more valuable for the encoding these variations before the further analysis.



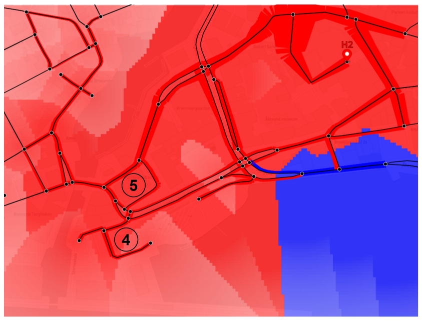
(a) average



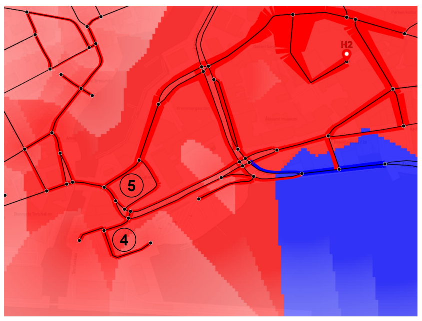
(b) t1 (01.01.2022)



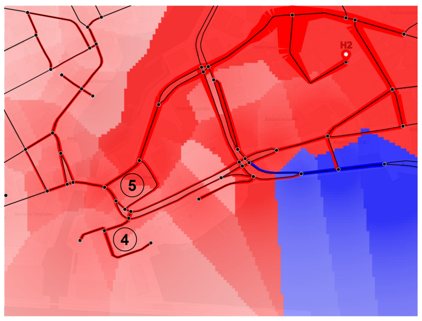
(c) t2 (02.01.2022)



(d) t3 (03.01.2022)



(e) t4 (04.01.2022)



(f) t5 (05.01.2022)

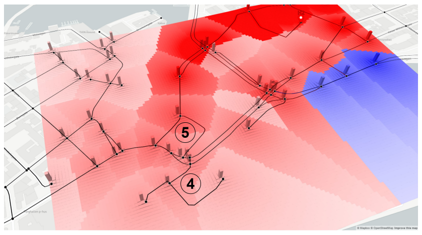
Figure 6.21: The results of computing a representative ITDM for the average and interested timestamps (t1-t5) with different weighted computation (Section 5.2).

Encoding of temporal changes

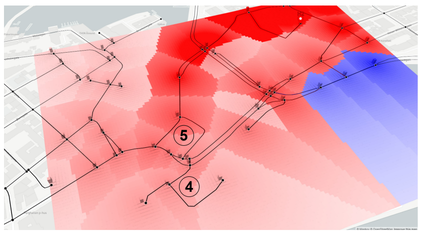
Figure 6.22 shows the TTDM results for different binary patterns. The number of "1" in the binary pattern string means the number of the consecutive days with the increasing access time. For example, "01110" represents the access time increasing pattern lasts three days. Figure 6.22c has the highest elevations for the targeted regions. This means the weather increases the access time more frequently on three successive days among all options (Pattern "010" - Pattern "01111110"). Therefore, planners could be advised to prepare more ambulance resources to speed up the response time during the 2022Q1 if the cruise ships stay for three days.

Suggestions for the decision maker – filtering by date

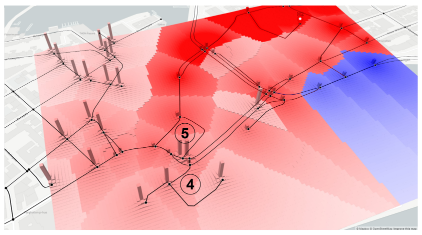
It is also feasible to select a time period for Case Study 2 by comparing TTDM with data filter and various time period data like Case Study 1 (Section 6.3.2). Furthermore, there is also one intuitive suggestion. Since Region ⑤ is better than Region ④, maybe it could be suggested to decision makers to arrange or suggest the patient moving from Region ④ to Region ⑤ when waiting for the coming of the ambulance.



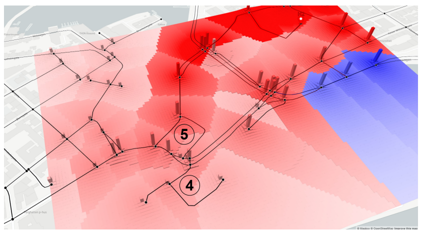
(a) Pattern “010”



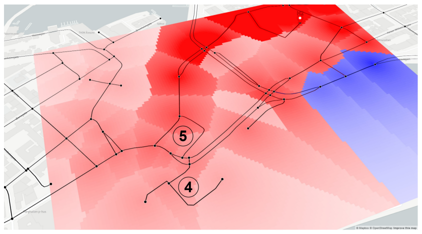
(b) Pattern “0110”



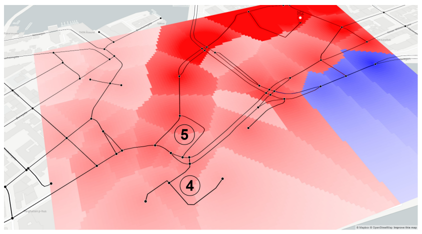
(c) Pattern “01110”



(d) Pattern “011110”



(e) Pattern “0111110”



(f) Pattern “01111110”

Figure 6.22: The results of encoding temporal changes with different binary pattern strings.

Chapter 7

Conclusions

This chapter describes the main contributions of this research work considering the proposed research questions (Section 7.1). Section 7.2 indicates directions for possible future work.

7.1 Contributions

The main objective of our research work is to encode and visualize temporal changes of topology density maps. It includes the design, implementation and validation of the proposed solution.

We first introduced a new method, IFT-based Topology Density Map (ITDM) (Chapter 4), which utilizes the Image-Foresting Transform (IFT) to compute Topology Density Map (TDM). The use of IFT improves the efficiency of TDM in the access time computation and in the estimation of density field. The analysis of the experiments in Section 6.2 provides more detailed evidence. Based on IFT-based Topology Density Map (ITDM), we also proposed a new algorithm, Temporal Topology Density Map (TTDM) (Chapter 5), to intuitively encoding and representing temporal variations associated with topology density map. The proposed solution explores Change Frequency Heatmap (CFH) that registers the occurrence frequency of change patterns of interest.

TTDM provides the possibility for the analysis of temporal changes associated with density maps along the network, an open gap in the literature. It starts with a stack of graphs. Each graph is a network with changing edge costs. After computing the representative ITDM, the method encodes interested temporal variation and visualize all final results in a 3D space.

The developed algorithms were embedded in two software prototypes. One prototype focuses on the TTDM computation analysis, including the support for the assessment of different configuration settings. The other addresses the visualization analysis itself. Both of them served as the tools for the conducted validation, involving performance and qualitative assessments. The proposed methods were also validated in two compelling case studies related to urban planning activities for the assessment of walkability and emergency services analysis on real road networks. The source code could be downloaded here[1,2].

In the following, we detail how each raised research question was addressed.

- **RQ1:** How to compute Topology Density Map (TDM) using the Image-Foresting Transform (IFT)?

  To compute Topology Density Map (TDM) using the Image-Foresting Transform (IFT), we proposed a new method, IFT-based Topology Density Map (ITDM) (Chapter 4). It optimizes the computation of access time to each POI with three steps (vertex mapping, IFT computation, and access time computation). The outputs of the algorithm, density and label maps, can then be utilized for the density field estimation with different options.

  One main limitation of TDM implementation [17] refers to the its lack of efficiency. With the increase of the network complexity, the most time-consuming step of TDM is to compute the access time from each POI to an arbitrary point on a 2D planar surface, which needs the iteration for all the nodes from each point position. The use of IFT provides a completely reverse way for computing Euclidean distances. It computes the Euclidean distance from the nodes, which are taken as input seeds, and finalizes with the partition of the whole image according to their influence zones. The results can then be utilized directly for the access time computation.

- **RQ2:** How to encode and visualize temporal changes on Topology Density Map (TDM) using Change Frequency Heatmap (CFH)?

  To encode and visualize temporal changes on Topology Density Map (TDM) using Change Frequency Heatmap (CFH), we designed a new visualization method Temporal Topology Density Map (TTDM) (Chapter 5). It is designed on the top of ITDM. The algorithm receives a sequence of graphs as input. Two main modules are designed to process these graphs and produce the inputs for the final module, the visual integrator. One module, compute representative ITDM, constructs a 1D network and 2D space visual layout. The other module encodes the temporal changes for the density maps and label maps by computing ITDMs. This result of this module is a height map, used to construct the final visual effects in 3D space with mesh interpolation module.

  The use of CFH allows the encoding of change binary patterns associated with changes over time. The input of CFH could be the density or label maps. This means the output

[1]https://github.com/felando1984/WebTTDM (As of May 2022).
[2]https://github.com/felando1984/TTDM (As of May 2022).

of CFH could reflect the temporal patterns for each geographic location according to different needs. Furthermore, the metric function and binary pattern string can be properly defined depending on the target application.

- **RQ3:** Would the use of Image-Foresting Transform (IFT) lead to a more efficient computation of Topology Density Map (TDM)? To what extent is the ITDM different from the TDM?

  Experiments conducted in Section 6.2 addressed this question. Experiments considered the performance and qualitative assessment with the comparison among TDM, ITDM option 1 and 2 from three aspects: computation time, map value difference, and computation result of the output file. Furthermore, the conducted experiments allowed the assessment of different parameters, such as resolution scale, size of network, the length of time, etc.

  We simulated random temporal changes on different size of networks to explore the suggested value of parameters. The resolutions scale m is advised to be equal to 4. Results show that ITDM is at least 3 times faster than TDM for the complex network with 100 timestamps. Moreover, with the visualization of value difference and theoretical analysis, it is more evident that the difference of visual affects between ITDM and TDM could be ignored when resolutions scale m is larger than 4.

- **RQ4:** Would the use of Temporal Topology Density Map (TTDM) be effective for analyzing changes over time in mobility-related applications?

  The proposed algorithms were validated in the context of two case studies related to mobility-related applications (discussed in Section 6.3). One referred to the walkability analysis on a walk network (small region, slower speed), while the other concerned the analysis of emergency service response time on a drive network (large region, faster speed). The weather condition was utilized to simulate the temporal change of access time with the assumed relation between moving speed and snow depth. In all cases, the use of TTDM is illustrated in the context of analyzing changes over time. The analysis included the overall visual effects, the encoding of temporal changes, and the impact of different date filtering schemes.

  Furthermore, we embedded the developed algorithms into two software prototypes. Those prototypes served as a toolbox to support our performance analysis of algorithms and case studies. One prototype focuses on the computation analysis of TTDM, while the other emphasizes the visualization analysis. GeoJSON[3] format is a standard for exchanging geospatial data. Both prototypes utilize this format to save/load the TTDM computation result,

[3]`https://geojson.org/` (As of May 2022).

which provides more flexibility.

7.2 Future Work

The conducted research opens up opportunities for future work in several directions. This section covers some of the envisioned research venues.

- Wrap-up the software prototype as a package or data service: In our research, the visualization analysis explores the computation resources of the local browser. If the algorithm could be wrapped up as a package for further development or provided as a data service, most of computation work could be realized in the server end point. In this case, only computation results would be sent to the browser (the client end point). The envisioned implementation could rely on technologies, such as ArcGIS API for Python,[4], Mapbox tiling service [5] Python library for QGIS,[6] etc.

- Explore more applications of TTDM for diverse analysis scenarios involving topological structure: One promising application would be the construction of sensor monitoring systems whose spatial distribution relies on a wired or wireless communication network. In this kind of system, the network includes data collection points, such as sensors, which are connected to the data centers. The sensors represent the non-POI nodes while the data centers serve as the POIs. The edge cost can be the communication delay time. In this application, the goal may be set to find out an appropriate configuration with suggested geographic positions of these data centers to minimize the access time of the sensor data. The temporal changes of delay time may reflect the stability of data communication within the network. These sensor networks could be time delay-sensitive scenarios like underwater acoustic sensor networks [56], smart grid-connected power system [57], nonlinear time-delay system [58], etc.

- Investigation on suitable domain-specific CFH configuration: Since the configuration of CFH, including the definition of the metric function and interested binary patterns, is application-dependent, it would be worthy to save or package them as libraries, such as NetworkX[7] and OSMnx [54][8] for future use. For instance, the configuration used in the two case studies provided in this research could serve as the first two templates for mobility applications. The users could benefit from them in the assessment of the impact of using different metric functions and patterns. Furthermore, once we have enough tem-

[4]https://developers.arcgis.com/python/ (As of May 2022).
[5]https://www.mapbox.com/mts (As of May 2022).
[6]https://docs.qgis.org/2.8/en/docs/pyqgis_developer_cookbook/intro.html (As of May 2022).
[7]https://networkx.org/ (As of May 2022).
[8]https://github.com/gboeing/osmnx (As of May 2022).

plates, they could be use to create training data sets. Machine learning methods, such as convolutional neural network (CNN) [59], could then be used to predict the ideal binary patterns for the related applications. It could greatly save the time to automatically determine patterns of interest and relevant behavior patterns, especially for less experienced users.

- Improvement of prototypes based on user studies: There are two software prototypes implemented for the computation and visualization analysis. Due to the time constraints, our validation only includes performance and qualitative assessment as well as discussion regarding the use of algorithms in two case studies. User studies, involving relevant stakeholders (e.g., urban planners), would contribute to collect expectations, suggestions, and impressions regarding algorithms and prototypes. Lessons learned could be explored in the inclusion of new features in the developed tools.

- More accurate density estimation with temporal changes in big cities: We used weather snow data to simulate the temporal changes based on the Ålesund's road networks in our case studies. In our case studies, the snow depth in one available weather station is the only aspect we assumed to affect the temporal changes in average moving speed on the roads. In the real world, it is possible to use the historical traffic data of roads and vehicles directly as real temporal changes once the data of sensors are ready. Until June 2022, there are limited historical traffic data of roads open to the public in Norway, which only covered some main roads in big cities.[9] In the future, it will be feasible to cover most of the roads in some cities. If we could consider more weather factors (like wind, temperature, etc) and have access to more data from more weather stations, temporal topology density maps could be assessed in more complex analysis scenarios.

[9]`https://www.vegvesen.no/trafikkdata/start/kart` (As of May 2022).

Appendix A

The detailed information of the nodes in case studies

Table A.1: The information of the nodes in the Case Study 1

Short name	Type	Description	Geo position (Latitude, Longitude)
H1	POI node	Ålesund Storsenter[1], Intersection 278087398	62.4723013°, 6.1589338°
H2	POI node	Bybadet[2], Intersection 7204337168	62.4726449°, 6.1644474°
H3	POI node	Hole Kjøtt AS[3], Intersection 8714559173	62.4714755°, 6.1596182°
H4	POI node	Master Mat[4], Intersection 7379970801	62.4711440°, 6.1657920°
N1	non-POI node	Intersection 278085830	62.4724539°, 6.1593316°
N2	non-POI node	Intersection 7379970863	62.4719770°, 6.1599410°
N3	non-POI node	Intersection 7379970513	62.4720850°, 6.1619762°
N4	non-POI node	Intersection 6286725867	62.4725801°, 6.1617773°
N5	non-POI node	Intersection 7379971301	62.4728340°, 6.1589430°
N6	non-POI node	Intersection 277936275	62.4715250°, 6.1602270°
N7	non-POI node	Intersection 7379970517	62.4716450°, 6.1621000°
N8	non-POI node	Intersection 6286725881	62.4715435°, 6.1585261°
N9	non-POI node	Intersection 6286725882	62.4714036°, 6.1585932°

[1] https://alesundstorsenter.no/ (As of May 2022).
[2] https://bybadet.no/ (As of May 2022).
[3] http://www.xn--holekjtt-b5a.no/ (As of May 2022).
[4] https://mastermat.no/ (As of May 2022).
[5] https://www.aleris.no/alesund/ (As of May 2022).
[6] https://statensbarnehus.no/ (As of May 2022).
[7] https://helse-mr.no/ (As of May 2022).

Table A.2: The information of the nodes in the Case Study 2

Short name	**Type**	**Description**	**Geo position (Latitude, Longitude)**
H101	POI node	Aleris Ålesund[5], Intersection 7379970095	62.4706440°, 6.1678710°
H102	POI node	Barnehuset Ålesund[6], Intersection 278085706	62.4723160°, 6.1587080°
H103	POI node	Ålesund Sykehus AS[7], Intersection 7389963213	62.4626490°, 6.3070280°
N101	non-POI node	Intersection 275605058	62.4707690°, 6.1522390°
N102	non-POI node	Intersection 7379970425	62.4704970°, 6.1527830°
N103	non-POI node	Intersection 7379971152	62.4711012°, 6.1531716°
N104	non-POI node	Intersection 7379971028	62.4702331°, 6.1522363°
N105	non-POI node	Intersection 282272324	62.4700806°, 6.1534794°

Bibliography

[1] Benjamin B Tumolo et al. “Toward spatio-temporal delineation of positive interactions in ecology”. In: *Ecology and Evolution* 10.17 (2020), pp. 9026–9036.

[2] Deepika Mann et al. “Spatio-temporal variations in landscape ecological risk related to road network in the Central Himalaya”. In: *Human and Ecological Risk Assessment: An International Journal* 27.2 (2021), pp. 289–306.

[3] Hye Won Lee, Kon Joon Bhang, and Seok Soon Park. “Effective visualization for the spatiotemporal trend analysis of the water quality in the Nakdong River of Korea”. In: *Ecological Informatics* 5.4 (2010), pp. 281–292.

[4] David Fawkner-Corbett et al. “Spatiotemporal analysis of human intestinal development at single-cell resolution”. In: *Cell* 184.3 (2021), pp. 810–826.

[5] Asako Sakaue-Sawano et al. “Visualizing spatiotemporal dynamics of multicellular cell-cycle progression”. In: *Cell* 132.3 (2008), pp. 487–498.

[6] Yuchuan Du, Fuwen Deng, and Feixiong Liao. “A model framework for discovering the spatio-temporal usage patterns of public free-floating bike-sharing system”. In: *Transportation Research Part C: Emerging Technologies* 103 (2019), pp. 39–55.

[7] Zhiguang Zhou et al. “Visual exploration of urban functions via spatio-temporal taxi OD data”. In: *Journal of Visual Languages & Computing* 48 (2018), pp. 169–177.

[8] Prikash N Meetei et al. “Spatio-temporal analysis of snow cover and effect of terrain attributes in the Upper Ganga River Basin, central Himalaya”. In: *Geocarto International* (2020), pp. 1–21.

[9] Xinyue Zhong et al. “Spatiotemporal variability of snow cover timing and duration over the Eurasian continent during 1966–2012”. In: *Science of the Total Environment* 750 (2021), p. 141670.

[10] Tianqi Xia et al. “Measuring spatio-temporal accessibility to emergency medical services through big GPS data”. In: *Health & place* 56 (2019), pp. 53–62.

[11] Chunbao Mo et al. "An analysis of spatiotemporal pattern for COIVD-19 in China based on space-time cube". In: *Journal of Medical Virology* 92.9 (2020), pp. 1587–1595.

[12] Marius Hogräfer, Magnus Heitzler, and Hans-Jörg Schulz. "The State of the Art in Map-Like Visualization". In: 39.3 (2020), pp. 647–674.

[13] Roeland Scheepens et al. "Interactive visualization of multivariate trajectory data with density maps". In: *2011 IEEE Pacific Visualization Symposium*. 2011, pp. 147–154.

[14] Roeland Scheepens et al. "Composite Density Maps for Multivariate Trajectories". In: *IEEE Transactions on Visualization and Computer Graphics* 17.12 (2011), pp. 2518–2527.

[15] Zhixiao Xie and Jun Yan. "Kernel density estimation of traffic accidents in a network space". In: *Computers, environment and urban systems* 32.5 (2008), pp. 396–406.

[16] Ke Ren et al. "Visual Analytics of Air Pollution Propagation Through Dynamic Network Analysis". In: *IEEE Access* 8 (2020), pp. 205289–205306.

[17] Zezheng Feng et al. "Topology density map for urban data visualization and analysis". In: *IEEE transactions on visualization and computer graphics* 27.2 (2020), pp. 828–838.

[18] Simon J. Sheather. "Density Estimation". In: *Statistical Science* 19.4 (2004), pp. 588–597.

[19] Alexandre X Falcao, Jorge Stolfi, and Roberto de Alencar Lotufo. "The image foresting transform: Theory, algorithms, and applications". In: *IEEE transactions on pattern analysis and machine intelligence* 26.1 (2004), pp. 19–29.

[20] Wolfgang Steiner, Friedrich Leisch, and Klaus Hackländer. "A review on the temporal pattern of deer–vehicle accidents: Impact of seasonal, diurnal and lunar effects in cervids". In: *Accident Analysis & Prevention* 66 (2014), pp. 168–181.

[21] Yeran Sun et al. "Analyzing human activities through volunteered geographic information: Using Flickr to analyze spatial and temporal pattern of tourist accommodation". In: *Progress in location-based services*. Springer, 2013, pp. 57–69.

[22] Tao Yang et al. "Regional frequency analysis and spatio-temporal pattern characterization of rainfall extremes in the Pearl River Basin, China". In: *Journal of Hydrology* 380.3-4 (2010), pp. 386–405.

[23] Andrew Borrie, Gudberg K Jonsson, and Magnus S Magnusson. "Temporal pattern analysis and its applicability in sport: an explanation and exemplar data". In: *Journal of sports sciences* 20.10 (2002), pp. 845–852.

[24] Greice Mariano et al. "Change Frequency Heatmaps for Temporal Multivariate Phenological Data Analysis". In: *IEEE 13th International Conference on e-Science*. Oct. 2017, pp. 305–314.

[25] Ewerton Silva et al. “A Change-Driven Image Foveation Approach for Tracking Plant Phenology”. In: *Remote Sensing* 12.9 (Apr. 2020), p. 1409. ISSN: 2072-4292.

[26] Alexandre Esteves Almeida. “Time Series Components and Breakpoints in Remote Sensing Image Analysis”. MA thesis. University of Campinas, Brazil, 2017.

[27] Nathália Menini Cardoso dos Santos. “Tools and Study Cases for the Characterization of Remote Sensing Time Series”. MA thesis. University of Campinas, Brazil, 2019.

[28] D. Peeters and I. Thomas. “Distance predicting functions and applied location-allocation models.” In: *Journal of Geographical Systems* 2 (2000), pp. 167–184.

[29] Ricardo da S. Torres, Alexandre X. Falcão, and Luciano da Fontoura Costa. “A graph-based approach for multiscale shape analysis”. In: *Pattern Recognition* 37.6 (2004), pp. 1163–1174. ISSN: 0031-3203.

[30] Ricardo Fabbri et al. “2D Euclidean distance transform algorithms: A comparative survey”. In: *ACM Computing Surveys (CSUR)* 40.1 (2008), pp. 1–44.

[31] Maria Danese, Maurizio Lazzari, and Beniamino Murgante. “Kernel density estimation methods for a geostatistical approach in seismic risk analysis: The case study of potenza hilltop town (Southern italy)”. In: *International Conference on Computational Science and Its Applications*. Springer. 2008, pp. 415–429.

[32] Javier Delso, Belén Martín, and Emilio Ortega. “A new procedure using network analysis and kernel density estimations to evaluate the effect of urban configurations on pedestrian mobility. The case study of Vitoria–Gasteiz”. In: *Journal of transport geography* 67 (2018), pp. 61–72.

[33] Giuseppe Borruso. “Geographical analysis of foreign immigration and spatial patterns in urban areas: Density estimation and spatial segregation”. In: *International Conference on Computational Science and Its Applications*. Springer. 2008, pp. 459–474.

[34] Jukka M Krisp, Stefan Peters, and Masria Mustafa. “Application of an adaptive and directed kernel density estimation (AD-KDE) for the visual analysis of traffic data”. In: *Online Proceedings GeoViz2011, Hamburg, Germany* (2011), pp. 9–11.

[35] Ke Nie et al. “A Network-Constrained Integrated Method for Detecting Spatial Cluster and Risk Location of Traffic Crash: A Case Study from Wuhan, China”. In: *Sustainability* 2015 (Mar. 2015), pp. 2662–2677.

[36] Kunxiaojia Yuan et al. “A quad-tree-based fast and adaptive Kernel Density Estimation algorithm for heat-map generation”. In: *International Journal of Geographical Information Science* 33.12 (2019), pp. 2455–2476.

[37] Tallys Gustavo Martins et al. “Visualizing the structure of urban mobility with bundling: A case study of the city of São Paulo”. In: *Anais do IV Workshop de Computação Urbana*. SBC. 2020, pp. 178–191.

[38] Christophe Hurter et al. “Bundled visualization of dynamicgraph and trail data”. In: *IEEE transactions on visualization and computer graphics* 20.8 (2013), pp. 1141–1157.

[39] Jing He et al. “Variable-Based Spatiotemporal Trajectory Data Visualization Illustrated”. In: *IEEE Access* 7 (2019), pp. 143646–143672.

[40] Jing He et al. “Diverse Visualization Techniques and Methods of Moving-Object-Trajectory Data: A Review”. In: *ISPRS International Journal of Geo-Information* 8 (Jan. 2019), p. 63.

[41] Mattias Persson. “A Survey of Methods for Visualizing Spatio-temporal Data”. MA thesis. Linköping University, Media and Information Technology, 2020, p. 62.

[42] Bo Huang and Magesh Chandramouli. “Spatio-temporal object modeling”. In: *Handbook of Research on Geoinformatics*. IGI Global, 2009, pp. 137–143.

[43] Yujie Fang, Hui Xu, and Jie Jiang. “A survey of time series data visualization research”. In: *IOP Conference Series: Materials Science and Engineering*. Vol. 782. 2. IOP Publishing. 2020, p. 022013.

[44] Tomasz Krukowicz, Krzysztof Firląg, and Paweł Chrobot. “Spatiotemporal analysis of road crashes with animals in Poland”. In: *Sustainability* 14.3 (2022), p. 1253.

[45] Tianwen Liang, Huan Liu, and Zheng Zhang. “Understanding Spatial and Temporal Change Patterns of Population in Urban Areas Using Mobile Phone Data”. In: *E3S Web of Conferences*. Vol. 145. EDP Sciences. 2020, p. 02007.

[46] Wang Ziwen et al. “Spatial and Temporal Patterns of Tourist Source Market Emissiveness: A Study of Shanghai, China”. In: *Geospatial Data Analytics and Urban Applications*. Springer, 2022, pp. 121–138.

[47] Yixian Zheng et al. “Visual analytics in urban computing: An overview”. In: *IEEE Transactions on Big Data* 2.3 (2016), pp. 276–296.

[48] Chris Brunsdon, Jonathan Corcoran, and Gary Higgs. “Visualising space and time in crime patterns: A comparison of methods”. In: *Computers, environment and urban systems* 31.1 (2007), pp. 52–75.

[49] Yujie Hu et al. “A Spatio-Temporal Kernel Density Estimation Framework for Predictive Crime Hotspot Mapping and Evaluation”. In: *ArXiv* abs/2006.00272 (2020).

[50] Torsten Ilägcrstrand. “What about people in regional science?” In: *Papers of the Regional Science Association*. Vol. 24. 1970.

[51] Menno-Jan Kraak. “The space-time cube revisited from a geovisualization perspective”. In: *Proc. 21st International Cartographic Conference*. Citeseer. 2003, pp. 1988–1996.

[52] Benjamin Bach et al. “A Review of Temporal Data Visualizations Based on Space-Time Cube Operations”. In: *EuroVis*. 2014.

[53] Benjamin Bach et al. “A Descriptive Framework for Temporal Data Visualizations Based on Generalized Space-Time Cubes”. In: *Computer Graphics Forum* 36 (2017).

[54] Geoff Boeing. “OSMnx: New methods for acquiring, constructing, analyzing, and visualizing complex street networks”. In: *Computers, Environment and Urban Systems* 65 (2017), pp. 126–139.

[55] Jing Wang and Haotian Liu. “Snow removal resource location and allocation optimization for urban road network recovery: a resilience perspective”. In: *Journal of Ambient Intelligence and Humanized Computing* 10.1 (2019), pp. 395–408.

[56] Chuan Lin et al. “A scheme for delay-sensitive spatiotemporal routing in SDN-enabled underwater acoustic sensor networks”. In: *IEEE Transactions on Vehicular Technology* 68.9 (2019), pp. 9280–9292.

[57] Guannan Lou et al. “Distributed secondary voltage control in islanded microgrids with consideration of communication network and time delays”. In: *IEEE Transactions on Smart Grid* 11.5 (2020), pp. 3702–3715.

[58] Lifeng Ma et al. “Distributed filtering for nonlinear time-delay systems over sensor networks subject to multiplicative link noises and switching topology”. In: *International Journal of Robust and Nonlinear Control* 29.10 (2019), pp. 2941–2959.

[59] Sushreeta Tripathy and Rishabh Singh. “Convolutional Neural Network: An Overview and Application in Image Classification”. In: *Proceedings of Third International Conference on Sustainable Computing*. Springer. 2022, pp. 145–153.