Graph-theoretic data modeling with application to neuromorphology, activity recognition and event detection

---

A Dissertation

Presented to

The faculty of the School of Engineering and Applied Science

University of Virginia

In partial fulfillment

of the requirements for the degree

Doctor of Philosophy (Electrical and Computer Engineering)

by

Tamal Batabyal

April 2019

Approval Sheet

This dissertation is submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy (Electrical and Computer Engineering)

Author: Tamal Batabyal

This dissertation has been read and approved by the examining committee:

Scott T. Acton, Dissertation Adviser

Zongli Lin, Committee Chair

Stephen G. Wilson, Committee Member

Laura Barnes, Committee Member

Daniel S. Weller, Committee Member

Barry Condron, Committee Member

Accepted for the School of Engineering and Applied Science:

Dean, School of Engineering and Applied Science

April 2019

Acknowledgement

I would like to express my deepest gratitude to my advisor, Dr. Scott T. Acton to give me the opportunity and freedom to pursue my research. He provided support and encouragement during my graduate studies, especially while pursuing out-of-the-box solutions of problems. I would like to thank my committee members, Dr. Zongli Lin, Dr. Stephen G. Wilson, Dr. Laura Barnes, Dr. Dan Weller and Dr. Barry Condron for their time and consideration. My sincere thanks to my recommenders - Dr. Scott T. Acton, Dr. Dan Weller, Dr. Andrea Vaccari, and Dr. Barry Condron for their important career advice during my post-doctoral applications. I would also like to thank my Master's advisor, Dr. Dipti Prasad Mukherjee, for his help and support.

Thanks are given to the members of VIVA Laboratory for their help and encouragement. I will definitely miss discussing research problems and socializing with them. Sincere thanks to Dr. Suvadip Mukherjee, Dr. Rituparna Sarkar, Dr. Shruba Gangopadhyay, Dr. Anirudhha Dutta for their advice and suggestions regarding my career opportunities. Friends have been a big part in my life during my graduate studies. I would like to acknowledge Neel Samanta, Justin Cavanaugh, Ivan Shabalin, Tiffany Ly, Paul Bonczek, Magdalena Woińska, Anna Borowska, Rachel Smith, Bassem Tossun, Sourav Maji, Deepak Kosanam and others. Special thanks to Izabela Hawro for her hospitality and generosity. I would like to mention Megan Evans and Deiziane Viane for their limitless enthusiasm in the squash court. Also, special thanks to the salsa club at University of Virginia.

My sincere thanks to Archan Ray and Kaushik Chakraborty for having weirdest and funniest discussions since we met at ISI Kolkata. I would also like to acknowledge Nilanjan Datta, Avik Chakraborti, Subhabrata Samajdar, and Sanjay Bhattacharjee for their support while I was at ISI Kolkata.

Lastly, nothing can I achieve without my parents, Samar Kumar Batabyal and Suvra Batabyal. I simply have no words to describe their continuous support, love, affection and sacrifice.

Abstract

Recently the world is witnessing an explosion of data, demanding scientific and application-specific models for data analysis. Unarguably, data is the new oil. It is well-known that a graph is the most abstract and discrete representation of data that originated from real world problems. These problems range from the seven bridges of Königsberg in 1736 to monolithic integrated circuits to biology, food web, internet, transportation, computer systems, social networks and countless others. Depending on the problem at hand, such complex data can be broadly classified into two groups. The *first* group accounts for problems where the data inherit distinct and invariant graph structures due to several factors that include the nature of data acquisition, data sampling, and intrinsically discrete structures of the data sources. A noteworthy example is 3D reconstructed neuron cells, where each neuron emulates a tree-structure branching topology. The *second* group of problems involves datasets that are deficient of fixed graph structures. Unlike the first category, the graph in this case should either be estimated based on optimization criteria or be initialized using some ad hoc constraints. Image or video based data is one example of this category, where the graph could be drawn from the pixels, the patches, or the frames depending on the nature of the problem.

The primary objective of this thesis is to investigate the above two categories of graph structured data with varying degrees of structural complexity in specific problems. We perform novel image analysis, build graph models, develop graph theoretic tools and scalable algorithms for two major purposes - informatics and categorization. In the first group, we consider activity recognition and neuromorphology. In the second group, we consider geomorphological event detection and event monitoring from videos.

In activity recognition, we use 3D reconstructed skeleton models of subjects performing a predefined set of activities. In such a model, the number of vertices and edges are fixed, connoting low structural complexity. We leverage such simple structures and propose Unified Graph Signal Processing (UGraSP) to integrate activity recognition and person identification using the same graph features. With additional graph structures, UGraSP is improved to make Unified Graph based Activity Detection (UGrAD) to encode the activity dynamics in terms of graph features. The central aspect of this thesis is the graph theoretic modeling of neuronal arbors. Feature-based representation and categorization of neurons, a topic that correlates with neuronal functionality, is still an open problem

in neuromorphology. We use 3D reconstructed neurons (fixed graph) from stained images, where the number of vertices is not fixed for every neuron but the edges between the vertices are structurally fixed. We propose NeuroPath2Path that utilizes path-based modeling of neuron anatomy and provides a visualization tool for the continuous deformation between a pair of neurons. NeuroPath2Path offers several advantages. Decomposition of a neuron into paths can be viewed as an assembly of individual circuits from the terminals to the soma, integrating semi-local features that act as path descriptors. Next, instead of subgraph matching, NeuroPath2Path provides a full-graph matching algorithm. The matching algorithm presents several biological factors, including fractality and decaying importance of features along the path. NeuroPath2Path also precisely investigates the feasibility of algorithmic constraints on the structural repertoire of neuronal arbors, thereby enforcing criteria, such as hierarchy mismatch. NeuroPath2Path can be extended to two major domains - morphological analysis and structural transformation of microglia cells, and in progressive degradation of neuronal paths in neurodegenerative diseases. In addition, the knowledge of neuronal functions, such as potentiation and co-adaptive spiking can be translated to neural networks in order to achieve better performance and emulate neurons by way of neural networks.

In geomorphological event detection, we are interested in three major application areas, which are road health estimation, sinkhole detection and monitoring, and rock-slope fault detection, from InSAR (Interferometric Synthetic Aperture Radar) data. The structural complexity of the data is elevated compared to the previous two problems. It is because there is no predefined connectivity among the vertices. We first model the data with a graph using K-nearest neighborhood approach. Then, we develop Laplacian Weighted Covariance (LaWeCo) for spatial localization of geomorphological events. Later, using a temporal bipartite graph model and iterative prior estimation, we propose DDT to detect as well as monitor such events.

Simultaneous detection and tracking of events from videos is another challenging problem with more complexity of graph structures. Here, there are multiple options for the selection of vertices and edges. By blending the recursive estimation of a spatial graph and temporal graphs of patches over time into a dictionary learning framework, we propose Graph based Dictionary learning for Event Detection (GraDED) to accomplish both the tasks. Lastly, we consider the problem of accelerating the convergence of LMS filters.

The problem apparently does not require graph modeling of data and combinatorial feature extraction. However, the imposition of a graph structure to the data is shown to improve the convergence of LMS.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Data is the new oil, fueling the advancement of scientific and application-specific data analysis [10, 11]. By way of complex data analysis, the scientific community diligently attempts to uncover the underlying principles of natural phenomena [4, 5, 12–21]. The reliability of such systematic analysis depends on suitable modeling of the data. However, significant variation in data and several other factors that are associated with the curation of such data, even for similar applications, precludes the development of a universal data model. As such, we need application-specific categorization of data models.

Among the existing tools for data modeling, graph based models serve as the most *abstract* and *discrete* representation of the data [22, 23]. In addition to using the raw data, a graph model also encodes the relationship between the extracted features or variables [24]. This relationship plays important roles in many tasks, including classification, regression, and inferences, which are instrumental in developing informatics. Graph encompasses three broad domains of literature, which are combinatorial graph theory, machine learning based graphical models and graph signal processing.

By imposing some constraints *a priori*, machine learning based models, such as probabilistic graphical model [12], conditional random fields [25] estimate the hidden relationship between the observed, latent, or both types of random variables. These

models are adopted for inference and classification related objectives. Combinatorial graph theory [23, 26] seeks answers behind the existence, enumeration, and genesis of structures and performs optimization of certain objectives on the structures. When the underlying graph is defined, graph signal processing [24] considers the data as the signal that resides on the graph, and accordingly augments classical signal processing tools to the graph domain. Graph filtering, manifold regularization, graph Fourier transform are few examples of graph signal processing tools. This thesis exploits the combinatorial part of structures that model the data in terms of a graph. In short, we leverage the morphology and geometry of structures [13] that are either present in the data or assigned in an abstract fashion in order to fulfill some objectives. Additionally, we utilize signal processing on the graphs whenever the tools are needed.

Apart from the existing literature, graphs also provide a set of advantages in terms of *minimalistic* representation. An example of activity recognition from a video dataset would be appropriate in this context. A typical frame in that video contains tens of thousands of pixels, incurring significant computational and storage loads. However, it is a well-known fact that our visual perception only tracks a specified number of *key* locations on a human body while the person is performing an activity. 3D reconstruction via skeletonization tools, such as Kinect, DensePose, OpenPose provide suitable rendering of the input videos to files containing the locations of joints of a human body. The skeleton then serves as the graph for downstream analysis. This conversion drastically reduces the storage load and enables on-line activity recognition. For example, UTKinect-Action3D dataset used Kinect with Windows SDK Beta and registered 10 actions which are performed by 10 subjects at 30fps. The RGB image has the resolution of 480×640 and the depth image has 320×240. The size of RGB and Depth videos are 1.79GB and 367MB respectively. Compared to the video data, the 3D reconstructed skeletonized dataset has the file size of only 3.3 MB, representing significant reduction of input data

size. The added advantage is that downstream analysis using graph dataset does not suffer from the problems that are associated with images and image processing tools. Research areas other than activity recognition vastly benefit from such modality conversion. In this thesis, we use

- 3.6 million 3D persistent scatterers from 62 ($AOI1$) and 28 ($AOI2$) InSAR images. Each scatterer is provided with the measured displacements over 232 time instances and geographical locations.
- 3D reconstructed neuron cells (.swc files) from stained images. The online repository, neuromorpho.org [27] contains thousands of such reconstructed neuron cells from different brain regions of several species.
- Kinect captured 3D skeletonized datasets.

In general, graph modeling of a problem can be constructed by identifying or defining the structure in terms of three parameters - vertices or nodes, edges or links, and edge weights. Owing to the nature of data acquisition or the data source, a lot of problems contain the graph structure inherently. Such problems carry a subtle dichotomy in their inherited structures based on whether the structure can be uniquely defined or not. Examples, such as 3D reconstructed neurons and Kinect-skeletonized humans, lend themselves to inherit unique graph structures, whereas the graph structures in images and videos can be defined based on the problem demands. In an image, a single pixel, a patch, or the entire image can be accounted as a vertex, denoting the scalewise change in the description of the vertex. Even in case of neurons, an entire neuron can be regarded as a vertex when we consider the Connectome.

Another important consideration is the existence of inherited connections. Kinect-generated skeletons and 3D traced neurons have defined connections. A .swc file lists how to connect the sampled locations (vertices) to reconstruct the neuron. In an image, such

connections are not uniquely defined. Ambiguous definition of scale and connectivity leads to two different practices of graph modeling.

The *first* group models the problems with graph structures that are unique or invariant. Lattice structures, polymer structures, neuronal and glial tree structures, genealogy, and 3D reconstructed human skeletons fall into this category. To meet certain objectives, a set of morphological, geometrical and combinatorial features is extracted from the structures. In this thesis, we primarily considers the datasets involving neurons and human activity recognition, with a major emphasis on neuromorphology as the application of graph theoretic features are vastly unexplored here.

The *second* group involves problems that have structured datasets without defined connectivity. For such problems, we need to first estimate the underlying graph structure or employ some ad hoc criteria (such as K nearest neighborhood) to connect the vertices. Rest of the analysis follow the similar paths to that of the first group. We experiment with InSAR dataset to identify and track geomorphological events (road health assessment, sinkhole monitoring, and rock-slope fault monitoring) at different scales. The scatterers (vertices) in the dataset are connected using K-nearest neighbor approach. Next, we consider graph modeling of videos from car-mounted cameras in order to track hazardous events. Lastly, we take a stream of input data, which have ill-conditioned autocorrelation matrix, and present an algorithm that recursively estimates the optimal relationship between data points. The data points are not graph structured. However, the imposition of a graph structure on the data is shown to improve the condition number, facilitating the convergence of LMS based filters.

There is an enormous variation in the complexity in the two groups of problems. The 3D skeleton obtained from Kinect for activity recognition is fairly simple, and does not contain significant structural variation. The number of joints that are identified by Kinect is fixed along with the connections among them. Besides activity recognition, the minor

structural variation in terms of joint-joint distances is utilized for person identification from videos where each frame contains only one person.

The neurons with no fixed number of sampled locations have enormous morphological and geometrical variation. This demands an entirely different modeling strategy. Our proposed modeling is not confined to neuron classification only. The model also helps build up neuroinformatics, and can be translated to the analysis of cell types (glia cells, astrocytes) with ramified branches.

The structural complexity increases when the data is captured without any predefined connections between the data points. InSAR data is one example of that category, where the similar algorithms to those are developed for neuromorphology and activity recognition can not be applied directly. Different connections among data points in InSAR lead to different results for the same analysis. As the physical connections between data points are not defined, we aim to explore the *functional* topology instead of the physical topology. It means that functionally correlated regions are densely connected without attempting to model the physical connections.

The complexity is elevated in case of videos, where we use two entirely different graphs. Each frame of the video is partitioned into blocks (patches or sub-volumess of a video) to construct a graph between the blocks. To track the event in time, the assembly of each block at all time points is used to make a temporal graph. The spatial graph is fully connected with the weight values that are determined by interblock coherence measure. The temporal graph is sequential and the weights are recursively updated.

Finally, we consider a stream of data (Markov input) where the problem is to improve the condition number of the matrix. This problem is more complex than the previous problems in terms of graph structures. Here, the graph between data points is not predefined. In addition, unlike Kinect, InSAR, and neurons, this problem does not demand any

morphological or geometrical features to be extracted from the graph if the graph is constructed. However, imposition of a graph provides a transformation that helps improve the condition number of the input autocorrelation matrix. It shows that the assignment of graph to a problem is not limited to extracting the combinatorial features only.

1.1 Objectives and contributions

The primary objective of this thesis is to model data with graph by leveraging the combinatorial graph theory and graph signal processing, where they are needed. Given such objective, we attempt to answer the following questions.

- How to provide graph-theoretic framework that can be utilized for several tasks, including classification, informatics, and tracking.
- (In case of biology) How to incorporate biological information in terms of graph features. In a quest of principles from data, how to separate algorithmic constraints from biological fact.
- How to incorporate combinatorial graph structures, such as complementary graph, bipartite graph, walk, path, line graph to enhance the discriminatory power of feature descriptors.

In response to those queries, the following contributions are summarized below.

Contribution 1:

In neuromorphology, the application of graph features and the potential of graph model are largely unexplored. We begin our experimentation with graph models based on the scale of the features. Akin to the bigram [28] model in text or speech processing, we first propose distribution based model that captures the conditional statistics of bifurcation

angle, fragmentation, and spatial density. Reading the neuron model analogous to a text or a speech sequence is the contribution here. Despite relatively good performance, we seek to find a model for visualization, and to incorporate biological rationale behind the feature selection. We devise NeuroSoL that capture morphological and geometrical features at local scale of vertices and edges. To the best of our knowledge, we are the first to introduce the concept of *structural relevance* and *connectivity-wise relevance* of a vertex in NeuroSoL, resulting in a one to one mapping of vertices between a pair of neurons.

Next, we improve our approach by incorporating path models of neuronal graph. The proposed path based algorithm is called ElasticPath2Path, which is again improved in NeuroPath2Path (Neurop2P in short). NeuroPath2Path offers several advantages. Decomposition of a neuron into paths can be viewed as an assembly of individual circuits from the terminals to the soma, integrating semi-local features that act as path descriptors. Next, instead of subgraph matching, NeuroPath2Path does not leave a single path unassigned, culminating in a full-graph matching algorithm. The matching algorithm presents the notion of relative fractality, path correspondences and incorporates physiological factors, such as decaying importance of features along the path, exploratory and competitive behavior for resource exploitation. NeuroPath2Path also precisely investigates the feasibility of algorithmic constraints on the structural repertoire of neuronal arbors, and thereby enforcing criteria, such as hierarchy mismatch. During classification, NeuroPath2Path delivers resilience to the class imbalance problem. Apart from classification, NeuroPath2Path can be augmented in two major domains - morphological analysis and structural transformation of microglia cells, and in progressive degradation of neuronal paths in neurodegenerative diseases. However, the best advantage of ElasticPath2Path and NeuroPath2Path is the visualization of deformation of paths between two neurons, which validates the path correspondence. The deformation between

a pair of neurons can also be tuned to explore neurogenesis, a well-known problem in cell-differentiation.

Contribution 2:

It is observed that the analysis including feature extraction using only one graph structure is not sufficiently discriminatory enough to perform tasks. Conventional methods perform quantitative analysis only on the intrinsic or estimated graph structure of a problem. In contrast, we use a series of structures that are derived from the original graph structure (intrinsic or estimated) of the problem. Examples of such derived structures that we use are complementary graph, line graph, bipartite graph, walks, structure graph, connectivity graph, and linear graph.

An example of neuron would suffice to explain the observation that we mention at the beginning of this contribution. A 3D reconstructed neuron possesses an intrinsic tree graph structure. By definition, a tree does not contain a loop. Let us consider another graph that has the same number of locations as of the neuron, but has edges which are not present in the neuron. We call it the complementary tree. By constraining the analysis of neuroanatomy only to the tree, we would not harness the information that is present in the complementary tree. Typically a neuron is referenced, meaning that each location has a 3D coordinate. By construction, each edge in the tree has a weight value that is the Euclidean distance between the locations that form the edge. This way the neuron is modeled as a weighted tree. Now, if we concern about the *existence* of the edge, we would consider the edge weight as 1 (if present in the tree) or 0 (if not present in the tree). If we focus on *importance* in terms of the length of the edge, we would consider the Euclidean distance. Therefore, we obtain four different structures when we combine the *existence* and *importance* of edges with the *tree* and *complementary tree* graphs. One can also extract paths within these structures, and develop hop-based model where a hop is defined as the number of edges on a path. In summary, an ensemble of different structures

that are derived from the original graph which models the data generally encapsulates rich anatomical and geometrical information.

1.2 Thesis outline

The rest of the dissertation is organized as follows: In Chapter 2, we present brief descriptions of the tools that are required for our works. Along with the tools, we provide an overview of graph theory and graph signal processing. Chapter 3 is dedicated to the problems that own invariant graph structures. We present our works on neuromophology, human activity recognition and neural networks. Chapter 4 includes the problems having non-unique graph structures. We focus on geomorphological event detection, spatiotemporal event detection, and system precondition to describe how the underlying graphs are recursively estimated and how the graph tools are used in these cases. Finally, we conclude in Chapter 5 with discussion of the methods, their possible extensions and applications.

Chapter 2

Background

In this chapter, we present a brief background on graph theory and graph signal processing. For algorithms and detailed analysis in graph theory and graph signal processing, interested readers may follow [23] and [24] respectively. We also discuss about ArcGIS, a geographic information system for working with georeferenced maps and charts. We use its desktop version ArcGIS 10.2.1 to visualize and analyze SqueeSAR data.

2.1 Graph theory

A graph $\mathcal{G}$ can be defined by the triplet $(\mathcal{V}, \mathcal{E}, \mathcal{W})$, where $\mathcal{V}$ is the set of vertices, and $\mathcal{E}$ is the set of edges to which corresponds a set of weights. In particular $\mathcal{V} = \{v_1, v_2, \cdots, v_N\}$, $\mathcal{E} = \{e_{ij} | v_i, v_j \in \mathcal{V}\}$ and weight set $\mathcal{W} = \{w_{ij} \neq 0 \Leftrightarrow e_{ij} \in \mathcal{E}\}$. The the set of vertices directly connected to v_i, $\mathcal{N}_i = \{v_j | e_{ij} \in \mathcal{E}\}$, is the *neighborhood* of v_i whereas the *degree* of v_i, $d_i = \sum_{v_j \in \mathcal{N}_i} w_{ij}$, is the sum of the weights of edges connected to v_i. For each graph, it is possible to define the *complementary graph* $(\bar{\mathcal{G}})$ defined by triplet $(\bar{\mathcal{V}}, \bar{\mathcal{E}}, \bar{\mathcal{W}})$, where $\bar{\mathcal{E}} = \{e_{ij} | e_{ij} \notin \mathcal{E}\}$ and $\bar{\mathcal{V}} = \{v_i | e_{ij} \in \bar{\mathcal{E}}\}$. A *walk* of length k on $\mathcal{G}$ is an alternating sequence of vertices and edges, $(v_i, e_{i,i+1}, v_{i+1}, \cdots, e_{i+k-2,i+k-1}, v_{i+k-1})$, where v_i and v_{i+k-1} are the start and end points of the walk respectively. In DDT, we consider only length-2 walks.

A graph is said to be *simple* if the graph contains no multiple edges between any pair of vertices or any self-loop. A graph is called *undirected* if no edge has any directionality. A *bipartite* graph, also called a bigraph, partitions $\mathcal{V}$ into two non-empty sets such that no two graph vertices within the same set shares an edge. Given a graph $\mathcal{G} = \mathcal{G}(\mathcal{V}, \mathcal{E})$ and a set of labels $\mathcal{L}$, we can define *vertex labeling* as a *bijective* function $f : \mathcal{V} \to \mathcal{L}$ [29].

Graphs can be conveniently represented by a set of matrices: the *adjacency matrix* ($\boldsymbol{A}$), defined as $A(i,j) = w_{ij}$, the *degree matrix* ($\boldsymbol{D}$), defined as $D(i,i) = d_i$, and the *Laplacian matrix* $\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{A}$. The Laplacian can be interpreted as a difference operator when applied to a graph signal $\boldsymbol{s} \in \mathcal{R}^N$:

$$\boldsymbol{L}s_i = \sum_{v_j : e_{ij} \in \mathcal{E}} w_{ij}[s_i - s_j]$$

where, with s_i we indicate the value that $\boldsymbol{s}$ assumes on node v_i. The incidence matrix, $B \in \mathcal{R}^{N \times |\mathcal{E}|}$ of $\mathcal{G}$ is defined as $b_{ij} = 1$ *or* -1 where the edge j is incident to or emergent from the vertex i. Otherwise $b_{ij} = 0$. The graph Laplacian $\boldsymbol{L} \in \mathcal{R}^{N \times N}$, which is a symmetric positive-semidefinite matrix, can be given by $\boldsymbol{L} = BWB^T$, where W is a diagonal matrix containing $\boldsymbol{w}$. A graph is *connected* if there exists at least a path between each pair of vertices. An *incomplete* graph is a graph where at least a pair of vertices is not connected with an edge. For a tree graph with size N, the total number of edges is $N - 1$. The weighted adjacency matrix ($\in \mathbb{R}^{N \times N}$) of a graph is defined as $\{a_{ij} = w_{ij} | i, j \in \mathcal{V}\}$. The degree vector D_{vc}, which indicates the degree of each vertex, is given by $D_{vc} = A\mathbf{1}$, where $\mathbf{1}$ is an array of 1s. The structural adjacency matrix A_s and degree vector,D_{vs} are defined in a similar fashion by replacing each weight w_{ij} by 1.

Using the Laplacian [24], $\boldsymbol{L}$, the symmteric normalized Laplacian matrix is defined as $\boldsymbol{L}_n = \boldsymbol{D}^{-1/2}\boldsymbol{L}\boldsymbol{D}^{-1/2}$. The set of sorted eigenvalues of $\boldsymbol{L}_n$ ($\Gamma = \{\lambda_0, \lambda_1, \cdots\}$ where $\lambda_0 \leq \lambda_1 \leq \cdots$) is known as the *spectrum* of the normalized Laplacian and, together with the set of corresponding orthogonal eigenvectors, it bears resemblance to the Fourier

analysis in classical signal processing. The random walk normalized Laplacian is defined as $\tilde{\boldsymbol{L}} = \boldsymbol{D}^{-1}\boldsymbol{L}$. The positive Laplacian is given by the formula, $\boldsymbol{L}^{+} = \boldsymbol{D} + \boldsymbol{A}$. The cross-adjacency matrix, $\mathcal{B}_{n_1 \times n_2}$ of two disjoint neighborhoods $\mathcal{N}_i$ and $\mathcal{N}_j$ with size n_1 and n_2 is defined as $\mathcal{B}_{ij}$=w_{ij} with $i \in \mathcal{N}_i$ and $j \in \mathcal{N}_j$.

2.2 InSAR & ArcGIS

Synthetic Aperture Radar (SAR) is a remote sensing technology that uses coherent radar pulses to reconstruct 3D landscapes in terms of images. Each SAR image contains both amplitude and phase data of each location on the landscape. The phase difference between two SAR images, taken at different times or vantage points, generates an 'interferogram' (InSAR). The variation in the interferogram reveals important topographical information, such as the ground elevation of a location over time (in mm). To cancel the atmospheric effects on the ground dispacement measurements, an advanced technique, called PSInSAR (Permanent Scatterer InSAR) is used. Permanent Scatterers are geo-referenced, and are abundant in locations where the signal strength of the reflected radar signals is excellent (e.g. rocky outcrops, transmission towers, tall buildings containing shiny metallic coated surface). In the rural landscape, the density of PS is significantly low. To resolve this problem, an improvement of InSAR is made, and new algorithm is called SqueeSAR.

We experiment with several SqueeSAR datasets. For example, we obtain the SqueeSAR data that is registered on the Central Virginia and the area is centered over the town of Middlebrook. This area of interest (AOI) has a lot of topographical significance, containing 100 lane miles of interstate routes, 200 lane miles of primary and secondary routes, subsidence areas, 600 bridges, culverts and major transportation structures. The SqueeSAR dataset is constructed using 62 SAR images, acquired by the Cosmo-SkyMed (CSK) satellite from 29th August, 2011 to 24th November, 2014 with an interval of every 16 days (except a 232 days interval between 25th October, 2012 to 30th June, 2013). After

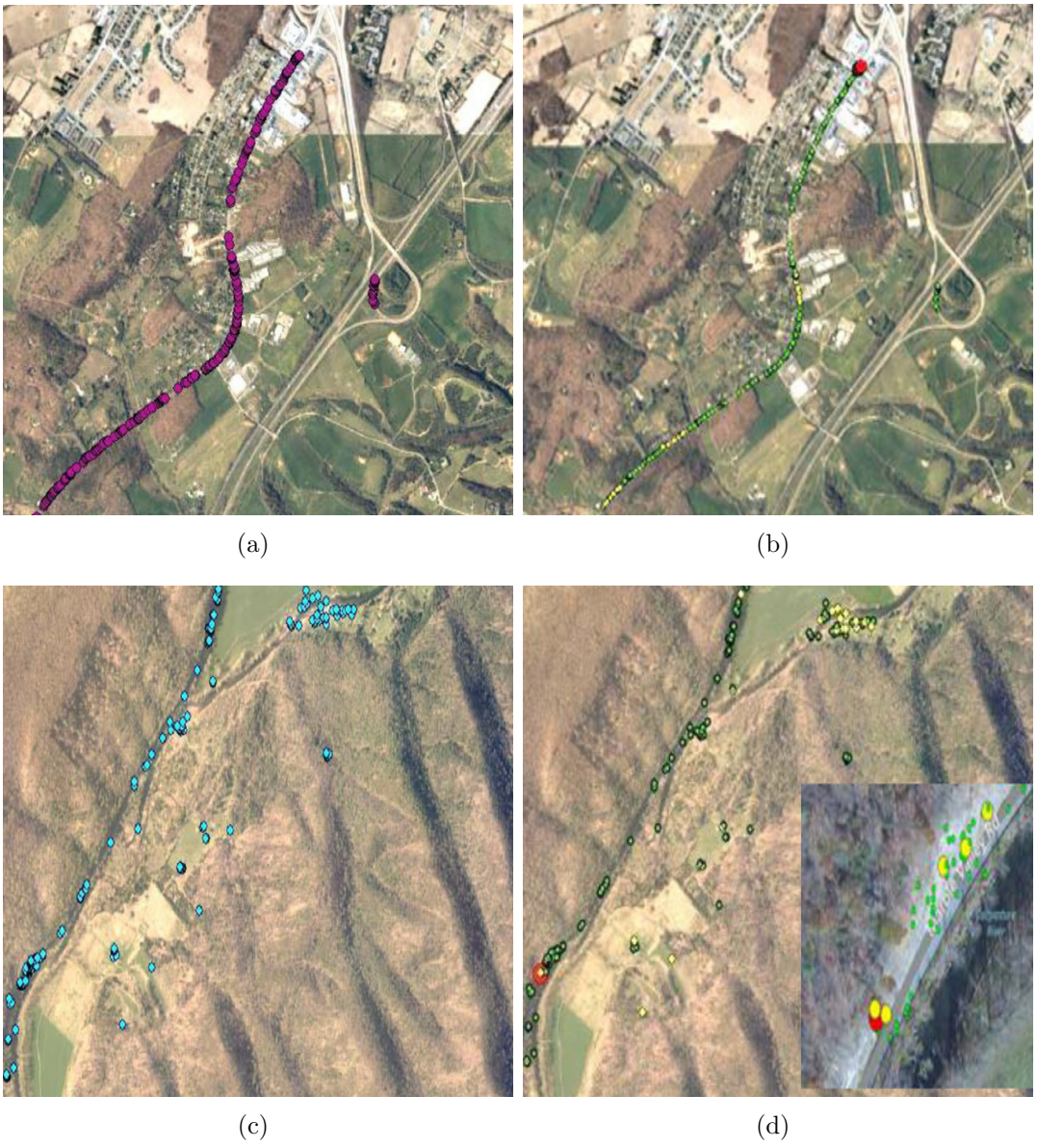
applying SqueeSAR algorithm, the resulting dataset contains approximately 800K scatterers with an average density of $497/km^2$. After invoking the SqueeSAR data as a layer to the ArcGIS, each PS will show the recorded displacements at the previously-mentioned time instances.

2.2.1 Coherent motion analysis toolbox

The coherent motion analysis toolbox is based on the Theil–Sen trend estimator and it was developed to detect regions, within the user-selected analysis area, showing indications of strong linear motion: grooves/ruts on a road/pavement, collapsing culverts, creeping slopes and, in general, any event expected to generate displacements with strong linear behavior in time. The Theil–Sen estimator is applied to the time series of the displacements to robustly evaluate the median of the displacement rates effectively reducing the contribution of potential contribution from the SAR additive and multiplicative noise. These rates are then checked against the user-selected warning and critical thresholds (given in mm/day) and each scatterer is colored accordingly.

2.2.2 Road smoothness analysis toolbox

Road smoothness index is directly related to the perception of riding quality. On scales, larger than typical IRI measurements and determined by the density of scatterers along the road, it is possible to define the normalized smoothness index (NSI) computed at each scatterer location and for each satellite acquisition time. This index is calculated using a graph theoretical approach and measures the smoothness of a signal defined over a graph (see Section 2.1). The hypothesis behind this approach is that regions of the road surface displaying large differential displacements between neighboring scatterers (large NSI) are prone to pavement deterioration. The difference in scale between the NSI and



(a) (b)

(c) (d)

FIGURE 2.1: Route-11: (a) permanent scatterers (PS) on top of a geo-referenced map, (b) the smoothness profile (NSI) of all the scatters. The red circle indicates extreme roughness that is detected by NSI, implying a possible event on the road surface. Route-600: (c) PS on top of a map, (d) impending rock-slope as shown in a dense region with Red and yellow marked scatterers. This is also confirmed by the field geologists.

IRI measurements means that direct correlation has only been observed for rare stretches of road with very high densities of permanent scatterers.

Chapter 3

Invariant or unique graph structure

Problems with invariant graph structures are ubiquitous in today's world. Traffic network [30], social network [31], gene-regulatory network [32, 33], neuron connectome [34] and numerous other applications contain invariant or unique graph structures. The term 'invariant' and 'unique' might lead to a lot of confusion, demanding some clarification. In the thesis, we use these terms to indicate the *existence of physical connections* among the vertices of a graph. For example, a 3D reconstructed neuron has a set of sampled locations as the vertices of the neuron graph. In addition, there exist physical connections among the vertices, which reconstructs the tree structure of the neuron. In Kinect based 3D reconstructed skeleton model, the sampled locations are joints and the connections emulate the body parts that connect the joints to form the skeleton. Two persons (or two people) connected via a friend request in Facebook is another example. These connections are unique in the sense that we do not need to estimate the connections based on some external criteria.

In contrast, the persistent scatters that are obtained from SqueeSAR dataset for geomorphological event detection have no predefined connections. We need to apply some external constraints to find the edges of the graph model in this scenario. Therefore, the connections are not unique. In image, the pixels form a uniform lattice and the

connections among them are undefined. The connections can be defined on different patterns such as 4 neighbors and 8 neighbors. In this case, even the vertices are non-unique in the sense that a pixel or a superpixel or a patch or even a frame could be regarded as vertices. A unique graph structure might be time varying, such as facebook graph. In summary, by invariant or unique, we account for the problems where the graph structures need not to be estimated.

We select two problems, which are human activity recognition and neuromorphology, which are modeled using invariant or unique graph structures. However, the structural complexity in the two problems are different. In human activity recognition, we use 3D reconstructed skeleton data, where the number of vertices and the edges among them are fixed, implying a graph model of low structural complexity. In such graphs, the only parameter that needs to be estimated is edge weight. Leveraging such simplicity in the model, other graph structures, such as bipartite graph, complementary graph, cliques can be derived with almost no computational overhead. In neuromorphology, we use 3D reconstructed neurons, where the number of vertices, which are traced locations, are not fixed but the connections among the vertices are defined. Due to the unequal number of vertices, algorithms that are developed using these datasets encounter subgraph isomorphism, inconsistent feature dimension and correspondence deficiency. To overcome these problems, strategies, such as same-hop walks, distribution of morphometrics, surjective correspondence maps, structural simplification (if applicable) are proved to be effective. We provide a through investigation and present our contribution in the neuromorphology section.

3.1 Activity recognition

The ability to identify a person and record, store, review and recognize the activities being performed is of key importance in several applications such as security and surveillance,

and elderly health care. Previous image-based (in contrast to joint location-based) studies include those in pose estimation [35], gait recognition [36], gesture recognition [37] and activity recognition [14, 38, 39]. Within the image-based framework, the activity recognition problem was further addressed by several methods, such as spatiotemporal interest points (STIPs) [40], histograms of oriented gradients (HOG) [41], and histograms of oriented flow (HOF) [42]. These methods aimed at the identification of relevant local features for which dynamic analysis was carried out via hidden Markov models (HMMs) [2], dynamic temporal warping (DTW) [43], and conditional random fields (CRFs) [44]. The task of person identification from multiple videos has also been widely addressed [45–47]. With the recent introduction of point cloud-based systems, the community interest was extended to the problem of person identification within these type of datasets [48, 49].

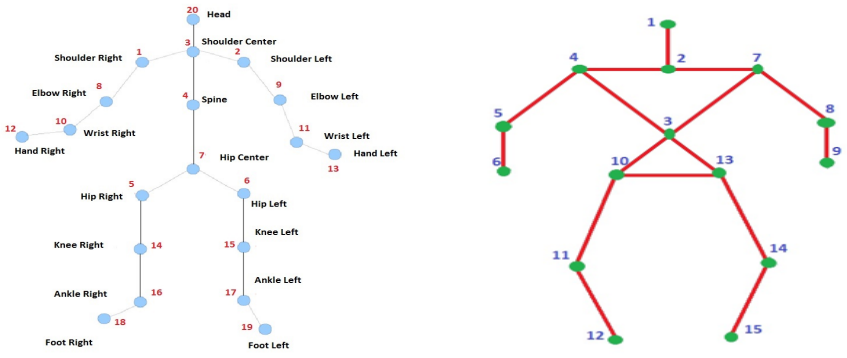
We propose a unified graph signal processing (UGraSP) technique that allows for simultaneous person identification and activity recognition from a single set of features.

3.1.1 Unified graph signal processing (UGraSP)

In our approach we take advantage of the point cloud representation, introduced by 3D sensors, to successfully extend person identification and activity recognition to datasets containing a wide variety of persons and activities. We achieve this by leveraging the graph theoretical formalism applied to the individual skeletal representation obtained from the imaging sensors where joints are represented as nodes and the edge connectivity follows the natural individual skeleton (Figure 3.1) with weights based on their Euclidean length.

3.1.1.1 Proposed method

In our framework, the 3D human skeleton is represented as a labeled set of N vertices, embedded in $\mathbb{R}^3$, with vertices represented by their spatial coordinates $v_i = (x_i, y_i, z_i)$ and



(a) (b)

FIGURE 3.1: A schematic representation of human skeleton with labeled joints captured by Kinect SDK for (a) MSR Action3D [1] and UTKinect [2] and (b) UCF action dataset [3].

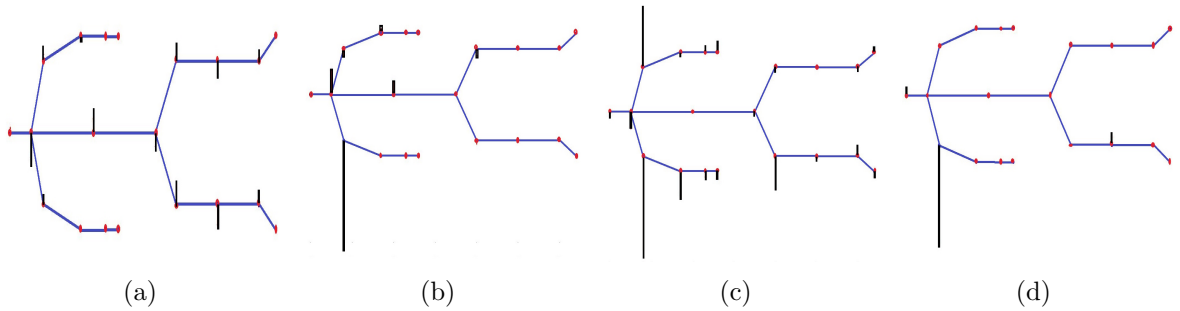
connected by a set of edges with weight dependent on the Euclidean distance between the connected vertices (figure 3.1). For each frame, we represent the joints coordinates in vectorized form $\boldsymbol{f}_g = [x_1, \ldots, x_N, y_1, \ldots, y_N, z_1, \ldots, z_N] \in \mathbb{R}^{3N}$.

3.1.1.2 Projection on extended Laplacian eigenvector basis

Once normalized, the orthogonal set of eigenvectors of the Laplacian constitutes a complete orthonormal basis that can be used to represent any graph signal $\boldsymbol{f} \in \mathbb{R}^N$. In order to project $\boldsymbol{f}_g$, we extended this approach to $\mathbb{R}^{3N}$ by creating the block diagonal matrix $\boldsymbol{U}_g = diag\{\boldsymbol{U}, \boldsymbol{U}, \boldsymbol{U}\}$. With this definition, we can leverage the *graph Fourier transform* [24] formalism and evaluate the transform $\hat{\boldsymbol{f}}_g$:

$$\hat{\boldsymbol{f}}_g = \boldsymbol{U}_g^T \boldsymbol{f}_g \tag{3.1}$$

Figure 3.2 shows an example of the transformed skeletal coordinate vectors $\hat{\boldsymbol{x}}$, $\hat{\boldsymbol{y}}$ and $\hat{\boldsymbol{z}}$ that form $\hat{\boldsymbol{f}}_g$ for a single frame of the video for the *high-throw* activity. We hypothesize that $\hat{\boldsymbol{f}}_g$ contains the signature of the skeleton of a specific person that can be utilized for activity recognition.



(a) (b) (c) (d)

FIGURE 3.2: Activity *high-throw*: (a) The eigenvector corresponding to maximum eigenvalue of $\boldsymbol{L}$ on the skeleton graph. Projected (b) X-axis, (c) Y-axis, and (d) Z-axis of all N=20 joints taken on a frame

3.1.1.3 Feature construction for activity recognition

A reliable feature descriptor for a specific activity is obtained from the vectorized representation of the covariance of $\hat{\boldsymbol{f}}_g$ evaluated over video subsets (*segments*). The choice of the length of these segments is relevant since segments that are too short will fail to capture significant variation whereas segments that are too long will not provide the temporal granularity required to identify the motion sequence within an activity. The total number of non-overlapped segments in a video can be obtained from $J = \lfloor S/\theta \rfloor$ where S is the number of frames within a video, θ is the length of each segment. Although we are considering only the first level of the temporal hierarchy proposed by [50], for each segment we construct simple and robust feature descriptor that, over the i^{th} segment of a video can be written as

$$\begin{aligned} \tilde{F}(i) &= \sum_{k=i\theta}^{(i+1)\theta-1} \boldsymbol{U}_g^T(\boldsymbol{f}_g^k - \bar{\boldsymbol{f}}_g)(\boldsymbol{f}_g^k - \bar{\boldsymbol{f}}_g)^T\boldsymbol{U}_g \\ &= \boldsymbol{U}_g^T C_i \boldsymbol{U}_g \qquad i \in \{1, 2, \cdots, J\} \end{aligned} \tag{3.2}$$

where C_i is the covariance matrix of $\boldsymbol{f}_g$ over the i^{th} segment of a video, and $\bar{\boldsymbol{f}}_g$ is the mean of coordinate vectors taking all the frames. The beginning frames of each video, where no activity is performed, are recognized using the low values of C_i and disregarded in our analysis.

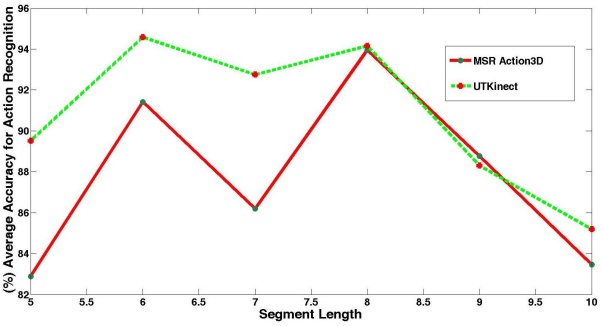


FIGURE 3.3: Average recognition accuracies of activities from [1] and [2] over different segment lengths.

3.1.1.4 Feature construction for person identification

It can be reasonably expected that the elements of the skeleton of a person will maintain their length during an activity performed at a fixed location. This property, captured by the adjacency matrix of the graph, is reinforced by the Laplacian. To correct for the changes in scale due to the location of a person within the field of view we normalized the Laplacian ($\boldsymbol{L}_N = \boldsymbol{D}^{-1/2}\boldsymbol{L}\boldsymbol{D}^{-1/2}$) and use its vectorized version ($\boldsymbol{L}_v$) as a feature in the training of a linear SVM. Since the measurements of the joints location are affected by noise, $\boldsymbol{L}_v$, for a specific person, will vary from activity to activity and trial to trial. Under the hypothesis that the noise introduced by the sensors is linear and additive, its effects can be minimized by averaging the vectorized Laplacians obtained from the initial frame of all the videos of a given person. The result is an average vectorized Laplacian $\boldsymbol{L}_E$, encoding each person's identity that, owe to the defined connectivity, which is also sparse. As an example, consider the skeleton in Figure 3.1(a). The 19 edges translate into $19 \times 2 = 38$ non-zero entries in the final $\boldsymbol{L}_E \in \mathbb{R}^{400}$. Because of this sparsity, we postulate that these features can be embedded in a lower-dimensional manifold.

Given M persons, we want to find the projection operator $\boldsymbol{P} : \mathbb{R}^{N^2} \to \mathbb{R}$, such that the projections $\boldsymbol{P}^T\boldsymbol{L}_E$ of the average Laplacians $\boldsymbol{L}_E$ are maximally separated on the real line. Let $\boldsymbol{L}_E^k$ and $\boldsymbol{L}_E^m$ be the average Laplacians for persons k and m respectively. Our objective is to maximize $(\boldsymbol{P}^T\boldsymbol{L}_E^k - \boldsymbol{P}^T\boldsymbol{L}_E^m)^2$ over all pairs $k, m \in \{1, 2, \cdots, J\}$ under the

constraint that $\sum_{i=1}^{N^2} P_i = 1$. Then the Lagrangian cost function $\chi(\boldsymbol{w})$ becomes,

$$\begin{aligned}
&\min_{\boldsymbol{P}\in\mathbb{R}^{N^2}} \chi(\boldsymbol{P}) = \\
&\min_{\boldsymbol{P}\in\mathbb{R}^{N^2}} \left[-\sum_{k<m}^{M} (\boldsymbol{P}^T \boldsymbol{L}_E^k - \boldsymbol{P}^T \boldsymbol{L}_E^m)^2 + \beta \left(\sum_{i=1}^{N^2} P_i - 1 \right) \right] = \\
&\min_{\boldsymbol{P}\in\mathbb{R}^{N^2}} \left[-\boldsymbol{P}^T \left(\sum_{c}^{M(M-1)/2} \check{\boldsymbol{L}}_c \check{\boldsymbol{L}}_c^{'} \right) \boldsymbol{P} + \beta\big(\boldsymbol{P}^T \mathbf{1} - 1\big) \right] = \\
&\min_{\boldsymbol{P}\in\mathbb{R}^{N^2}} \left[-\boldsymbol{P}^T \boldsymbol{A} \boldsymbol{P} + \beta\big(\boldsymbol{P}^T \mathbf{1} - 1\big) \right]
\end{aligned} \tag{3.3}$$

where, $\check{\boldsymbol{L}} = \big(\boldsymbol{L}_E^k - \boldsymbol{L}_E^m\big)$ for $m < k \in \{1, 2, \cdots, N^2\}$. Since $\boldsymbol{A} = \sum_c^{M(M-1)/2} \check{\boldsymbol{L}}_c \check{\boldsymbol{L}}_c^{'}$ is positive definite, the cost function is non-convex and it can be shown that the sub-optimal solution is given by

$$P_j = \frac{\sum_k A_{jk}^{-1}}{\sum_l \sum_m A_{lm}^{-1}}; \qquad j \in \{1, 2, \cdots, N^2\}. \tag{3.4}$$

Whenever a new vectorized average Laplacian ($\boldsymbol{L}_t$) is extracted from a frame, it can be projected on the real line as $\boldsymbol{P}^T \boldsymbol{L}_t$ where identification is carried out by finding the closest $\boldsymbol{L}_E$.

3.1.1.5 Results

We have tested our proposed methodology over three datasets: MSR Action3D [1], UTKinect [2] and UCF Kinect [3].

MSR Action3D [1]. This dataset contains twenty activities performed by 10 subjects 3 times each. The entire dataset has 557 activity video sequences of variable length. The data are captured at 15 frames/second. Each frame contains the (x, y, z) coordinates of 20 joints in a human skeleton captured by Kinect. We discard activity 20 due to inconsistent data in each video.

TABLE 3.1: Comparison of average accuracy of *UGraSP* vs. current methods for the MSR Action3D dataset [1].

Methods	Average accuracy(%)
Recurrent Neural Net [51]	42.50
Dynamic Temporal Warping [43]	54.00
Action Graph [15]	74.70
Hidden Markov Model [2]	78.97
Random Occupancy Pattern [52]	86.50
Actionlet Ensemble [53]	88.20
Cov3DJ [50]	90.53
UGraSP	**93.95**

TABLE 3.2: Average accuracy of each activity by our proposed method *UGraSP* using [1]. The overall accuracy is 93.95%

Activity	Acc(%)	Activity	Acc(%)
1 hnd high wave	100.00	2 hnd wave	100.00
horiz. wave	90.25	side boxing	100.00
hammer	88.50	bend	52.58
catch	90.00	fwd kick	100.00
fwd punch	100.00	side kick	90.00
high throw	100.00	jogging	100.00
draw $\mathcal{X}$	90.25	tennis swing	100.00
draw tick	92.50	tennis swerve	100.00
draw circle	100.00	golf swing	100.00
hand clap	100.00	SOA	94.50

UTKinect [2]. A stationary Kinect with SDK includes 10 activities performed by 10 subjects twice each. The activity videos are captured in RGB synchronized channel at a frame rate 30 frames/second. There are 10 subjects performing each of the 10 activities twice. In total, there are 6220 frames from 200 sample activity videos. The duration of each activity varies from 5 to 120 frames.

UCFKinect [3]. This dataset contains 16 different activities performed by 16 persons, 5 times each. The skeleton captured by Kinect consists of 15 joints, as shown in 3.1(b).

3.1.1.5.1 Activity recognition The analysis of activity is carried out with non-overlapping video segments. The sensitivity of the algorithm performance, with respect

TABLE 3.3: Activity-wise average accuracy (%) from UTKinect for [2] and *UGraSP*. The overall classification score is 94.58% using our method against 90.92% using [2].

Activity	[2]	UGraSP	Activity	[2]	UGraSP
walk	96.5	100.0	throw	59.0	77.3
sit	91.5	98.5	push	81.5	86.5
stand	93.5	94.5	pull	92.5	94.5
pick	97.5	94.5.0	wave	100.0	100.0
carry	97.5	100.0	clap	100.0	100.0

to segment length, is shown in Figure 3.3 for two datasets. The result indicates that lengths of segment $\theta = 8$ and $\theta = 6$ provide the best result for MSR Action3D and UTKinect datasets respectively. The overall accuracy over 19 activities are compared with existing methodologies and UGraSP outperforms all of them as shown in Table 3.8.

The accuracy with respect to individual activities is reported in Table 3.2. The classification accuracy for each activity is obtained by inserting features from (3.2) into a linear SVM with a Gaussian kernel of $\gamma = \frac{1}{\text{num of features}}$ and $C = 1$. The features for each activity are tested with a one-against-all approach, with a random partition of 90% for training and 10% for testing. The results for UTKinect and UCF Kinect [3] datasets compared with competing methods are shown in Table 3.3 and Table 3.4.

TABLE 3.4: Activity-wise average accuracy (%) from UCF Kinect for [3] and *UGraSP*. The overall accuracy using our method 97.75% against the competing method of 95.94%.

Activity	[3]	UGraSP	Activity	[3]	UGraSP
balance	97.5	100.0	vault	92.5	96.5
clmb ldr	93.8	97.2	run	97.5	100.0
clmb up	98.8	100.0	stp back	97.5	94.5
duck	100.0	100.0	stp front	97.5	100.0
hop	96.2	94.5	stp left	96.2	100.0
kick	98.8	100.0	stp right	98.8	100.0
leap	100.0	100.0	twist L	88.8	95.2
punch	95.0	97.2	twist R	86.2	88.9

3.1.1.5.2 Person identification For the datasets, UTKinect and MSR Action3D, we partition the feature sets in a 80:20 ratio as train:test. Training features are used to build up a SVM classifier with Gaussian kernel of $\gamma = \frac{1}{\text{num of features}}$ and $C = 1$. The recognition

accuracies are exhibited in Table 3.5 for UTKinect, Table 3.7 for MSR Action3D, and Table 3.6 for UCF Kinect datasets.

TABLE 3.5: Person identification average accuracy (%) from UTKinect [2] by *UGraSP*. The overall accuracy is 87.5%.

Person	UGraSP	Person	UGraSP	Person	UGraSP
P1	100	P4	75	P7	75
P2	100	P5	100	P8	75
P3	100	P6	75	P9	75
				P10	100

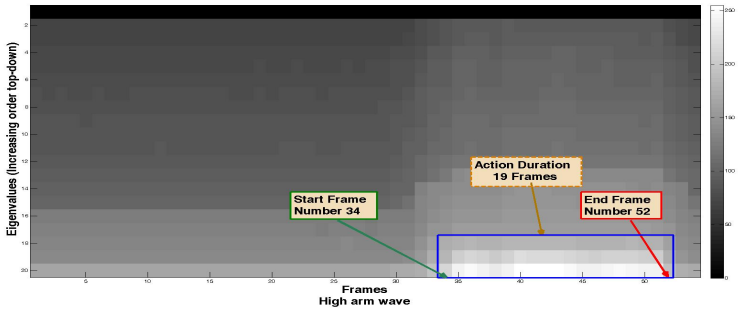
TABLE 3.6: Person identification average accuracy (%) from UCF Kinect [3] by *UGraSP*. The overall accuracy is 86.5%.

Person	UGraSP	Person	UGraSP	Person	UGraSP
P1	88.8	P6	88.8	P11	88.8
P2	90.0	P7	82.5	P12	87.5
P3	83.8	P8	85	P13	82.8
P4	83.8	P9	84.3	P14	88.8
P5	92.5	P10	92.5	P15	81.3
				P16	83.7

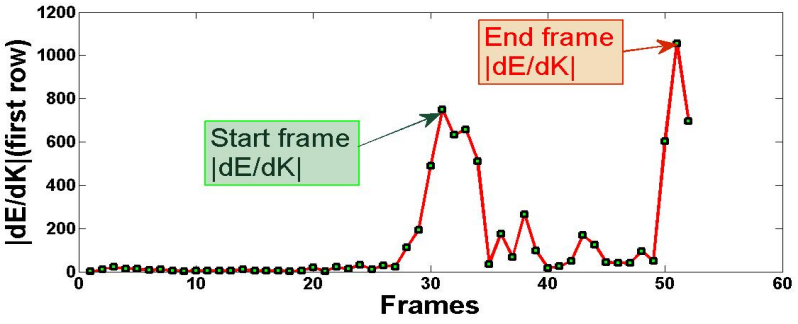
TABLE 3.7: Person identification average accuracy (%) from MSR Action3D [1] by *UGraSP*. The overall accuracy is 85.4%.

Person	UGraSP	Person	UGraSP	Person	UGraSP
P1	86	P4	94	P7	82
P2	88	P5	72	P8	90
P3	82	P6	92	P9	82
				P10	86

The performance of UGraSP depends on the fact that almost all the frames in a video are associated with a particular action that a subject in the video is performing. Realistically, in a video there might be more than one actions that are concatenated with a set of rest frames in between. Therefore, it is crucial to identify the action boundaries, and then to recognize actions using the frames within boundaries.



(a)



(b)

FIGURE 3.4: (a) Spectrogram of the complementary graph Laplacian for all the frames in the *high-arm-wave* activity video. (b) Value of $\partial E_\theta(k,0)/\partial k$ for the same video.

3.1.2 Unified graph based activity detection (UGrAD)

Activity boundary detection and activity recognition [39, 42] are two different problems that have been at the forefront of video analysis research for the last two decades. Boundary determination is the process of detecting the sequence of frames where a subject is performing an activity, whereas activity recognition deals with the determination of the action being performed. By improving boundary detection, the accuracy of activity recognition is increased since the feature descriptors, used in the recognition process, are computed using action frames [2,15,41,44,52,53]. While numerous approaches have been developed to tackle the challenge of a robust and reliable action determination based on a spatiotemporal image analysis framework [38], HOG [41], HOF [42], actionlets [53], covariance [37, 50, 54, 55], histograms [2], and SMIJ [56], none has considered accurate boundary detection to improve robustness and reliability of the activity discrimination.

TABLE 3.8: Comparison between UGrAD and other current methods average accuracy (MSR Action3D dataset [1].)

Method	Accuracy (%)
Actionlet Ensemble [53]	88.20
Lie Group [59]	89.48
Cov3DJ [50]	90.53
Covariance [54]	91.25
UGraSP [55]	93.95
UGrAD	**94.37**

Another challenge faced by activity recognition is the dynamic modeling of the temporal evolution of the detected feature descriptors. Authors have used covariance descriptors computed on video segments, leading to sensitivity to the chosen segment length [55], or descriptors depending on sets of subsequences linked to the activity cycle, resulting in a computationally intensive approach [50]. Others have used CRF [44], HMM [2], conditional restricted Boltzmann machines [57], recurrent neural networks [51] and latent variable [58] however, accurate parameters tuning and computational effort are often impediments to obtaining satisfactory results.

With the availability of skeleton-based datasets, as those acquired by several modern motion sensing devices, for each frame, subjects in the field of view are portrayed as a skeletal figure obtained by connecting the 3D locations of the detected joints using lines following a pattern resembling the human skeleton. In this work we will represent these skeletal figures as graphs and propose a unified approach that performs simultaneous activity recognition and activity boundary detection (UGrAD) using their complementary form. The *complementary* of a graph is a graph that connects all vertices not connected by edges in the original graph.

3.1.2.1 Action boundary determination

Let's consider an activity video with frames $\{f_1, f_2, \cdots, f_M\}$. Each f_i contains the set of N three-dimensional coordinates representing the location of each joint (vertex)

within the skeletal figure. For each frame, we can construct the complementary graphs $\{\bar{\mathcal{G}}_1, \bar{\mathcal{G}}_2, \cdots, \bar{\mathcal{G}}_M\}$ that, contrary to the expected stable behavior of the direct graph (representing the skeleton of the imaged subject), measures the dynamics of the activity since it connects vertices that are expected to be moving with respect to each other. We then evaluate the Laplacian $\bar{\boldsymbol{L}}_n$, normalized to account for the perspective foreshortening, and evaluate the set of eigenvalues $\bar{\Gamma}_i \in \mathbb{R}^{N\times 1}$. We use the latter to construct the *spectrogram* of $\bar{\boldsymbol{L}}_n$ ($\bar{\boldsymbol{\Gamma}} \in \mathbb{R}^{N\times M}$) where the columns are given by the $\bar{\Gamma}_i$. Fig.4.8(a) shows that, within the spectrogram, the presence of an activity can be identified by a few high order eigenvalues. Let us construct a window $\theta \in \mathbb{R}^{m\times s}$, where $m < N$ represents the number of top eigenvalues and $s \ll M$ the number of consecutive frames considered, in order to seize the transition of an activity. If we define with $\bar{\boldsymbol{\Gamma}}_c \in \mathbb{R}^{m\times M}$ the cropped subset of $\bar{\boldsymbol{\Gamma}}$ that includes only the top m eigenvalues, we can identify the set $\boldsymbol{F}_A$ of frames containing and action:

$$f_k \in \boldsymbol{F}_A \iff \left|\frac{\partial E_\theta(k,0)}{\partial k}\right| \geq T_0; 0 \leq k \leq M \tag{3.5}$$

where $E_\theta(k,0)$ represents the first row of $E_\theta = \bar{\boldsymbol{\Gamma}}_c^2 * \theta$. The partial derivative has a number of solutions indicating a set of peaks at different frame number k (Fig.3.4(b)). The largest peaks occur at the beginning and at the end of an activity. The selection of the threshold T_0 is carried out using an iterative approach, similar to the hysteresis in edge detection problems, until at least the minimum number of frames required for the polynomial fitting (see section 3.1.2.3) is selected. The process typically ends when the start and end frames lie on the opposite outer edges of the peaks in Fig.3.4(b), thus including the entire action.

3.1.2.2 Bipartite flow construction

Once $\boldsymbol{F}_A$ and the action boundaries are detected, we determine the temporal description of an activity by analyzing the evolution of each joint location. To achieve this, we

TABLE 3.9: Average accuracy of each activity by UGrAD using [1]. The overall accuracy is 94.37%

Activity	Acc. (%)	Activity	Acc. (%)
1 hnd high wave	100.00	2 hnd wave	98.5
horiz. wave	93.45	side boxing	100.00
hammer	90.5	bend	59.2
catch	92.25	fwd kick	100.00
fwd punch	100.00	side kick	91.5
high throw	100.00	jogging	100.00
draw $\mathcal{X}$	90.25	tennis swing	98.25
draw tick	90.00	tennis swerve	100.00
draw circle	100.00	golf swing	98.5
hand clap	100.00		

introduce the concept of *flow* of a graph between two consecutive frames. Consider two frames $0 < l < k < M_A$, where M_A is the number of frames in f_A, and the corresponding complementary graphs, $\bar{\mathcal{G}}_l$ and $\bar{\mathcal{G}}_k$. We define the *flow operator* as $\mathcal{F} : (\mathcal{G}_l, \mathcal{G}_k) \longrightarrow \mathcal{B}^{(l,k)} \in \mathbb{R}^{2N\times 2N}$, where $\mathcal{B}^{(l,k)}$, the *flow diagram*, is the adjacency matrix of the bipartite graph created between $\bar{\mathcal{G}}_l$ and $\bar{\mathcal{G}}_k$. To be more precise, if we take two vertices v_1 and v_2 of $\bar{\mathcal{G}}_l$ that are connected by the edge $e_{1,2}$ with weight $d^p_{1,2}$ equal to the Euclidean distance within the p-th frame, we define the flow between frames l and k as

$$\begin{aligned}
\mathcal{B}^{(k,l)}_{1,2} &= \mathcal{F}_{k,l}(v^k_1, v^l_2) = \mathbb{S}\frac{1-\exp[-a(d^l_{1,2}-d^k_{1,2})]}{1+\exp[-a(d^l_{1,2}-d^k_{1,2})]},\\
\mathcal{B}^{(k,l)}_{1,1} &= \mathcal{F}_{k,l}(v^k_1, v^l_1) = \mathcal{B}^{(k,l)}_{2,2} = \mathcal{F}_{k,l}\left(v^k_2, v^l_2\right) = 0,\\
\mathcal{B}^{(k,l)}_{2,1} &= \mathcal{F}_{k,l}(v^k_2, v^l_1) = -\mathcal{B}^{(k,l)}_{1,2} = -\mathcal{F}_{k,l}\left(v^k_1, v^l_2\right)
\end{aligned} \tag{3.6}$$

where $\mathbb{S} = sgn(d^l_{1,2} - d^k_{1,2})$, and $0 < a < 1$ is a scaling factor. This definition of flow naturally provides a way to construct the bipartite graph between $\bar{\mathcal{G}}_l$ and $\bar{\mathcal{G}}_k$. If the weights $d^p_{1,2}$ remain unchanged between two consecutive frames, the flow will be zero. The flow is constrained between 1 and -1 to normalize for the variations among subjects performing different trials of the same activity and the sum of all flows between l and k is zero. The idea behind our definition of flow is to imagine a subject performing an

TABLE 3.10: Average recognition accuracy (%) for each activity from UTKinect for [2] and UGrAD. The overall recognition accuracy is 95.1% for UGrAD and 90.92% for [2].

Activity	[2]	UGrAD	Activity	[2]	UGrAD
walk	96.5	97.5	throw	59.0	81.2
sit	91.5	97.25	push	81.5	89.25
stand	93.5	96.4	pull	92.5	96.0
pick	97.5	98.5	wave	100.0	100.0
carry	97.5	100.0	clap	100.0	98.5

activity as a closed system with no external force or energy applied. In this case, the flow is defined in such a way that the system behaves like a conservative field.

A typical flow during *high-arm-wave* activity is shown in Fig.3.5(a)-(c). It can be seen that the pattern in Fig.3.5(c) is oppositely aligned to Fig.3.5(a). This is because the hand moves away from the body at the onset of *high-arm-wave* and finally it moves close to the body at the end. The inversion of the flow reflects this motion. Fig. 3.5(d) shows an example of a flow diagram for the action *hand-clapping*.

3.1.2.3 Polynomial fitting

Each flow between two vertices constitutes a time-series which is hypothesized to have a unique pattern different for each action. We propose to capture such pattern by fitting Legendre polynomials. Thanks to the high level of symmetry in the flow definition (3.6), a single flow value is copied (with change of sign) in four entries in $\mathcal{B}_{l,k}$: (v_1^k, v_2^l), (v_2^k, v_1^l), (v_2^l, v_1^k), and (v_2^k, v_1^l) reducing the number of polynomials to fit.

The selection of the set of fitting polynomials is guided by some key observations regarding the flow, as given below.

1. There can be a constant flow across the frames in $\boldsymbol{F}_A$.

2. There can be accelerated motions across frames. (e.g. golf-swing)

3. If the initial and final positions of active joints are the same or the action is repetitive, the flow becomes oscillatory. (e.g. hand-clapping)

4. Normal human motion is typically smooth over the frames as the frame rate is usually

higher than the activity speed.

5. Same actions may take different length of frames depending on the person and the trial. Hence, the polynomial representation must consider frame length and interval.

Between the several potential candidates, observations (2)-(5) above prompted us to choose the Legendre polynomials of the first kind as defined in (3.7) for $|x| < 1$. Thanks to the orthogonality property they add the benefits of computational ease and stability to our algorithm.

$$P_n(x) = \sum_{k=0}^{n} x^k \binom{n}{k} \binom{\frac{n+k-1}{2}}{n}; \; n = 0, 1, \cdots \tag{3.7}$$

We assume that, excluding the symmetrical entries, each flow entry β_i (where $1 \leq i \leq \lambda = N(N-1)/2$) in $\mathcal{B}_{i,j}^{(k,l)}$ is independent of all the others thus reducing the computational complexity by requiring the fitting of Legendre polynomials to each entry separately. Since Legendre polynomials are defined in $-1 < x < 1$, in our discrete approximation, the interval is mapped to the M_A frames comprising the action becoming $x \in [-1, -1 + \frac{2}{M_A - 1}, \cdots, 1]$.

If we choose the maximum degree to be L, with $L + 1$ polynomials, we can write

$$F_x(\beta_i) = b_0 P_0(x) + b_1 P_1(x) + \cdots + b_L P_L(x) \tag{3.8}$$

with $b_k \in \mathbb{R}$, $-1 \leq x \leq 1$, and $L \leq M_A$. This can be rewritten in a matrix form as

$$\begin{bmatrix} F_{-1}(\beta_i) \\ \vdots \\ F_1(\beta_i) \end{bmatrix} = \begin{bmatrix} P_0(-1) & P_1(-1) & \cdots & P_L(-1) \\ \vdots & \vdots & \ddots & \vdots \\ P_0(1) & P_1(1) & \cdots & P_L(1) \end{bmatrix} \begin{bmatrix} b_0 \\ \vdots \\ b_L \end{bmatrix} \tag{3.9}$$

TABLE 3.11: Average recogntion accuracy (%) of each activity from UCF Kinect for [3] and UGrAD. The overall classification accuracy for UGrAD is 98.2% compared to of 95.94% for [3].

Activity	[3]	UGrAD	Activity	[3]	UGrAD
balance	97.5	96	vault	92.5	98.45
clmb ldr	93.8	95.7	run	97.5	100.0
clmb up	98.8	98.4	stp bck	97.5	99.2
duck	100.0	100.0	stp frnt	97.5	100.0
hop	96.2	97.2	stp lft	96.2	98.6
kick	98.8	98.25	stp rght	98.8	100.0
leap	100.0	100.0	twist L	88.8	97.75
punch	95.0	99.25	twist R	86.2	92.4

or, more compactly, $\boldsymbol{F}(\beta_i) = \boldsymbol{PB}$. The solution can be obtained with in a least-square fitting fashion evaluating

$$\boldsymbol{B} = (\boldsymbol{P}^T\boldsymbol{P})^{-1}\boldsymbol{P}^T\boldsymbol{F}(\beta_i) \tag{3.10}$$

Thanks to the orthogonality property of $P_n(x)$, $(\boldsymbol{P}^T\boldsymbol{P})$ can be simplified in the limit as follows

$$\boldsymbol{P}^T\boldsymbol{P} = \begin{bmatrix} \sum_x P_0^2(x) & \cdots & \sum_x P_L(x)P_0(x) \\ \vdots & \ddots & \vdots \\ \sum_x P_L(x)P_0(x) & \cdots & \sum_x P_L^2(x) \end{bmatrix}, \boldsymbol{P}^T\boldsymbol{P} \xrightarrow[M_A\to\infty]{} \begin{bmatrix} 2 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \frac{2}{2L+1} \end{bmatrix} \tag{3.11}$$

For each β_i, the $L+1$ coefficients are taken as features resulting in a feature matrix for an entire activity video of size $\boldsymbol{\Upsilon} \in \mathbb{R}^{\frac{N(N-1)}{2}\times(L+1)}$. The vectorized version of $\boldsymbol{\Upsilon}$ is used as a feature vector identifying a specific action. These vectors are used for classification of activities using an SVM classifier.

3.1.2.4 Datasets

We test our algorithm on the datasets given in Section 3.1.1.5.

3.1.2.5 Results

The experimental set up is kept fairly similar for all the datasets. The size of the window θ is set to $m = 3$ and $s = 2$. To compute the flow the scale factor a is taken as 0.02 for MSRAction3D and 0.09 for UTKinect and UCFKinect. The maximum degree of the Legendre polynomial is set to 6 resulting in 7 coefficients for β_i. The degree of the polynomial also sets the minimum number of frames required for the selection of the threshold T_0. After obtaining the set of feature descriptors $\boldsymbol{\Upsilon}$, an SVM with $\gamma = 1/(\text{num of features})$ and $C = 1$ is used to classify the activities. A 10-round cross-validation analysis, with a ration of 9:1 (train:test) is used with a *one-against-all* approach.

The overall accuracy of UGrAD for is compared with state-of-the-art techniques in Table 3.8 (MSRAction3D dataset). The results for MSRAction3D, UTKinect and UCFKinect are shown in Table 3.9, 3.10, and 3.11 respectively.

UGrAD, shows significant improvement in discriminating activities after properly identifying the activity boundaries. Although UGrAD performs 0.5% better than UGraSP in terms of the average accuracy, it can be seen from Table 3.9 and Table 3.2 that the recognition accuracy of bending and horizontal wave actions are improved. Among all the actions, bending is the most challenging to recognize.

In summary, the normalized flow takes care of person-specific and trial-to-trial variations in movements and reduces noise. The use of Legendre exhibits to capture constant-speed, accelerated, and oscillatory motion patterns which can be visualized in images depicting flow. Combining with our previous work [55], the salience of our work is found in the comprehensive solution to person identification, activity boundary detection and activity classification, in which the same graph-based features are used in all three tasks. The implementation is computationally tractable and can be embedded in surveillance appliances. Currently we are extending this work to detect the boundary and classify very fast-changing variable-speed activities.

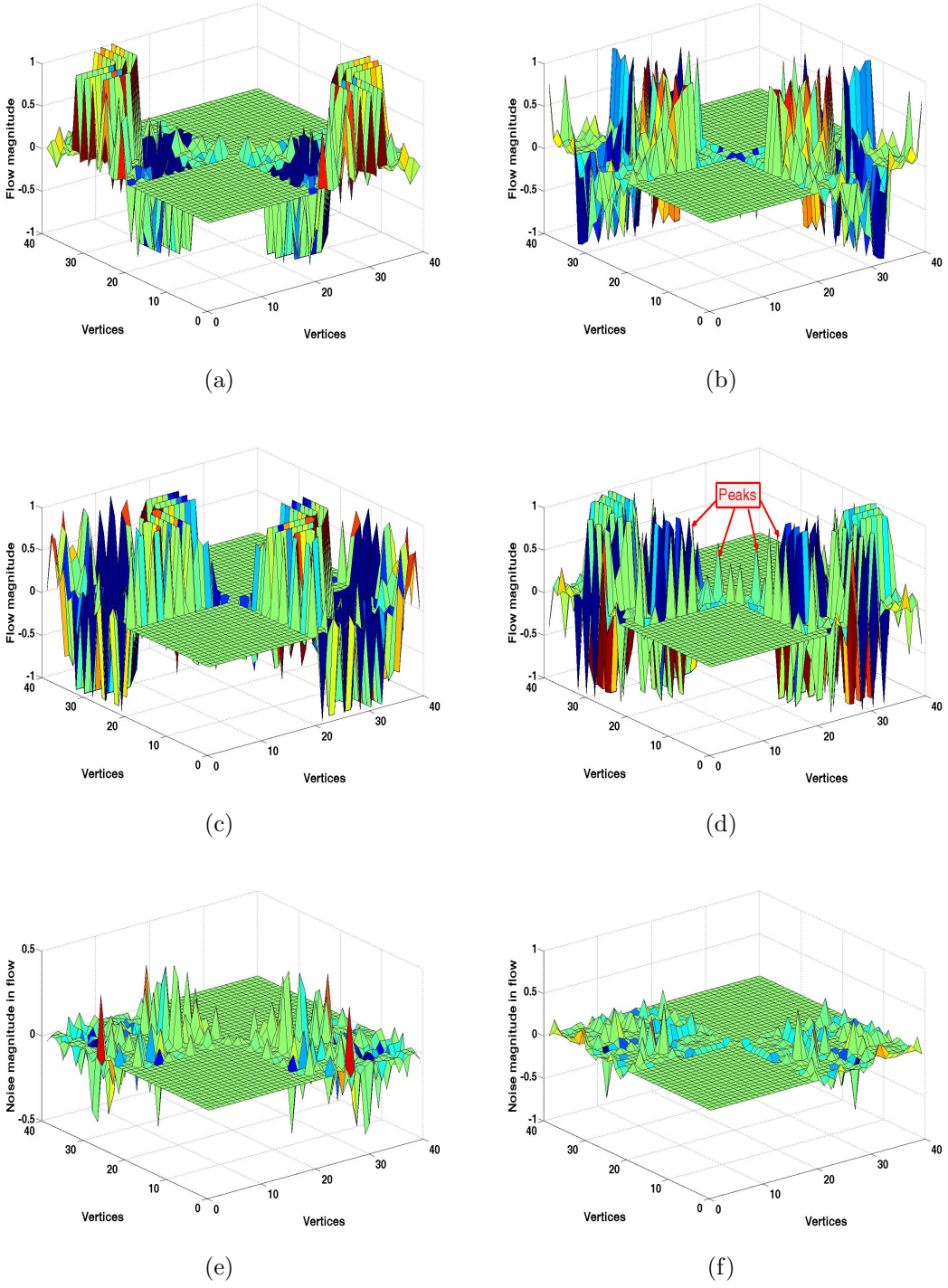


FIGURE 3.5: Activity flow diagrams ($\mathcal{B}_{i,j}^{(k,l)}$). Activity *high-arm-wave*: (a) activity is about to begin, (b) middle, and (c) end of the activity. (d) Middle frame of *hand-clapping*. (e)-(f) Typical start and end frames of activity videos.

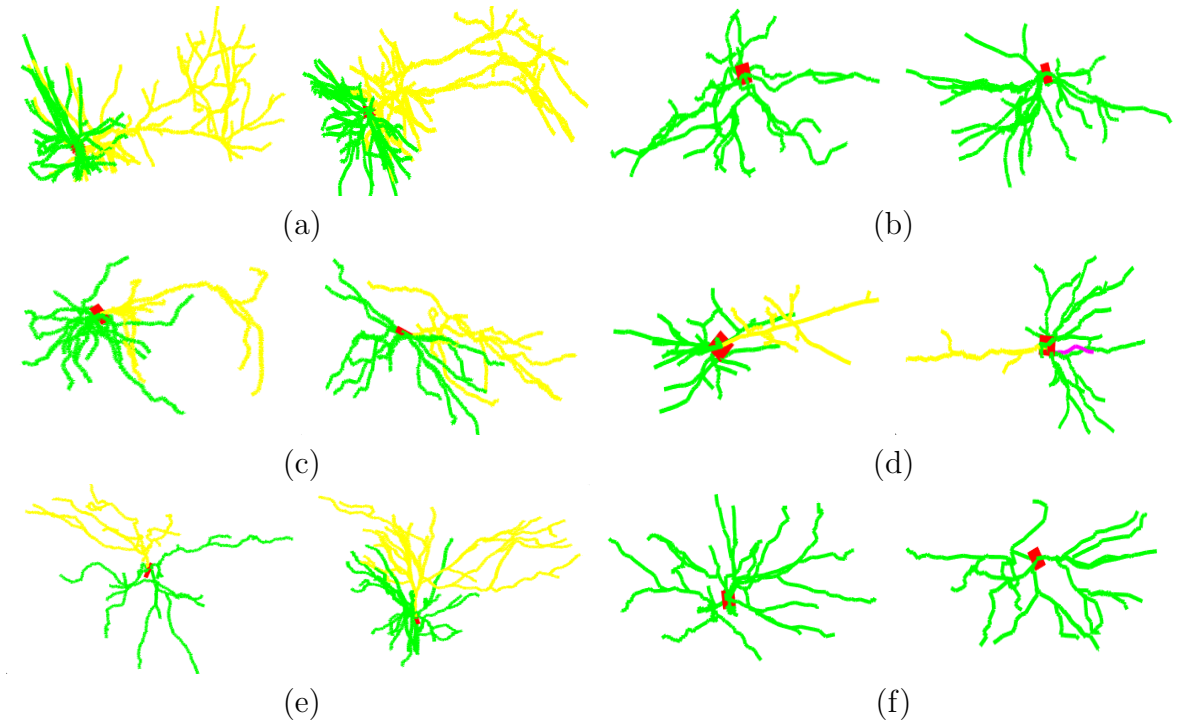


FIGURE 3.6: Pyramidal neurons from (a) primary motor cortex, (b) secondary motor cortex, (c) prefrontal cortex, (d) somatosensory cortex, (e) primary visual cortex, and (f) secondary visual cortex of the mouse. The dendritic branches in yellow are apical dendrites, and in green are basal dendrites. The red square in each cell is the soma, used as the designated root node in our analysis. This figure provides a glimpse of region-based arborial differences among pyramidal cells. Cells differ in size and volume, which are scaled for visualization.

3.2 Neuromorphology

Neurons process information by transmitting electrical signals via complex circuitry. The functionality of each neuron depends on a set of intrinsic factors, such as morphology, ionic channel density, gene expression, including the extrinsic ones, such as connectivity to other neurons [60, 61]. In 1899, Cajal [62], considered the founder of modern neuroscience, put forward his pioneering work on neuroanatomy with detailed, accurate, and meticulous illustrations, and posited that the shape of a neuron determines its functionality. Experimental results strongly support this idea. Inspired by this fundamental work, the study of neuromorphology primarily aims at analyzing and quantifying the complex shape and physiology of neurons in specific functional regions to identify relationships.

Neurons vary greatly in size, shape, and length. A major obstacle towards understanding the brain is the development of efficient ways to encode these shapes. The

anatomical and geometrical features of neurons of any cell-type, for example, pyramidal cells differ based on the regions in which the cells reside [60, 63]. Fig. 3.6 shows regional variation in the structure and geometry of dendritic arbors of pyramidal cells [64]. It is observed that the number of branches, length, surface area, and volume of apical dendrites is $4-9$ times larger for hippocampal than for cortical regions, whereas in terms of the same features of basal dendritic arbors, it is approximately 3 times [60]. Another source of variation stems from technical imprecision in measurements obtained while performing 3D reconstruction from image stacks using software tracing tools, such as Neurolucida [65]. Noise due to technical imprecision includes wide variations in the number of manually or semi-automatically traced 3D locations (approximately between 60 to $70,000$), the number of ramified branches and bifurcations by different tracers, and deletion of dendritic spines adversely affect the registration of neurons, and thereby induce error in morphological feature quantification. The skeletons of dendritic and axonal branches form a tree topology with a number of bifurcations. The bifurcations at successive stages help in a series of effective and unambiguous signal processing modules, such as active and passive signal propagation, integration, filter, attenuation, oscillation, and backpropagation [61, 66].

From the soma to the dendritic terminals of a neuron, the diameter of the dendritic shaft tapers [67, 68]. The increased diameter of a dendritic shaft near the soma is tailored to faster signal propagation to the soma compared to the dendritic tuft , which helps generate action potential in the soma. Several research works consider the branches in the proximity of soma are more important compared to the distant dendritic tuft and spines in the analysis of neuromorphology [69–71]. The length of dendritic branch segments shows similar behavior when propagating away from soma. For instance, the terminal segments are longer than the intermediate branch segments for basal dendrites in cortical pyramidal cells [63]. These observations support the Bayesian philosophy which is geared towards

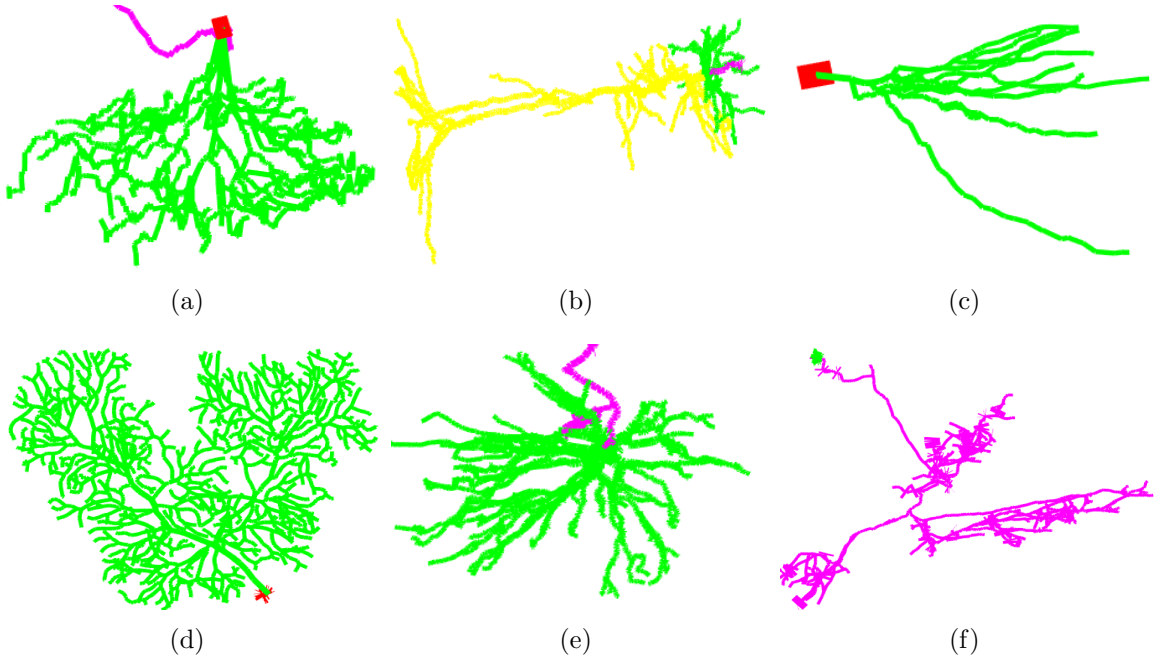


FIGURE 3.7: (a) A retinal ganglion cell from the inner plexiform layer of a 9 month-old adult mouse. The 3D reconstructed cell has 3938 3D locations, 50 bifurcations, 106 branches, 56 rooted paths, $255.52\mu m$ height, $4499.5\mu m$ diameter, $9061.14\mu m^3$ volume, $18,213.9\mu m^2$ surface area. (b) A 3D reconstructed (traced by Neuromantic [4]) pyramidal cell of an adult mouse having $24,868$ 3D locations, 95 bifurcations, 200 branches, 106 tips, $814.05\mu m$ height, $16341.8\mu m$ diameter, $12569\mu m^3$ volume, $24,825.9\mu m^2$ surface area. (c) A hippocampal granule cell (in the dentate gyrus) of a 9 month-old mouse is traced using Neurolucida. The 3D reconstructed neuron contains 414 3D locations with 7 bifurcations, 15 branches, 14 rooted paths, $98.33\mu m$ height, $443.18\mu m$ diameter, $3107.27\mu m^3$ volume, and $3255.33\mu m^2$ surface area. (d) A Purkinje cell in cerebellar cortex of a 28 day-old mouse. The reconstruction is performed by Neurolucida, containing 3187 3D locations. The neuron has 391 bifurcations, 783 branches, 393 tips, $184.35\mu m$ height, $4366.24\mu m$ diameter, $12,794.7\mu m^3$ volume, and $31,574.8\mu m^2$ surface area. (e) A motor cell in the spinal cord of a 10 day-old mouse, which is reconstructed with 5868 locations using Neurolucida tracer. The 3D traced neuron has 47 bifurcations, 103 branches, 58 tips, $415.6\mu m$ height, $784.158\mu m$ diameter, $4006.15\mu m^3$ volume, and $11,931.8\mu m^2$ surface area. (f) A long-axon projection neuron from the thalamus of a 6 months old mouse - it is traced by Large Volume Viewer(LVV) [5] with 2818 3D locations. The traced neuron has 169 bifurcations, 265 branches, $2731.69\mu m$ height, $55,742.44\mu m^3$ volume and $189979\mu m^2$ surface area. The color code is the following: yellow=apical dendrites, green = basal dendrites, magenta = axons, red = cell body/soma/root. The quantified statistics on the number of bifurcations and tips or rooted paths that are mentioned above are extracted from dendritic arbors of each cell-type.

the analysis of morphogenesis of neurons [71]. Functions such as synaptic boosting [72], coadaptive local spiking [73], and global spike amplification [74] suggest the use of other morphometrics to describe the structural aspects on the functions. For example, packing density of ramified branches and bifurcations of neuron potentially trigger intermittently co-adpative spiking.

The tree-type arbors of neurons and the availability of the inventory of digitally-traced 3D reconstructed neurons, Neuromorpho [75], provided significant momentum in the last decade for the quantitative and qualitative assessment of neuroanatomy via graph-based morphometrics. In Neuromorpho, the sequentially-aligned slices of microscopic images are registered and traced using software [76], such as Neurolucida [65] and Neuromantic [4], and the reconstructed images can then be processed through software, such as L-measure [77] to extract an extensive list of morphological metrics. On one hand, there are several research works dedicated to analyze the neuromorphology of specific cell types, such as basal dendrites of cortical pyramidal cells [63, 71], GABAergic interneuron cells [61] and others. These works account for region-specific variations in the physiology and anatomy of a neuron cell to establish the effect of certain functions on the structure. On the other hand, research efforts, such as blastneuron [78], neurosol [79],and TMD [69], attempt to extract model based features, which are catered to the need for automated classification of different neuronal cells. The motivation behind this avenue of research is that it is impossible to identify and categorize one trillion neuronal cells by adopting manual or even semi-automatic methods.

Towards achieving this automation, we *first* propose scalable morphometrics to account for primarily three morphological features, which are bifurcation angle, arbor's spatial extent, and branch-wise normalized fragmentation. These visually observed features are biologically relevant for different types of neurons. Our approach is completely automatic, coordinate independent, and does not involve any preprocessing steps, such as neuron alignment to a fixed coordinate frame. In addition, our morphometrics are independent of the size of a neuron, which excludes the possibility of of conventional problem of subgraph matching. Precisely, we compute the conditional distributions of the above mentioned three features, which serve as the feature descriptors of a neuron in our work, NeuroBFD (named NeuroBFD for neuron bifurcations, fragmentation and density).

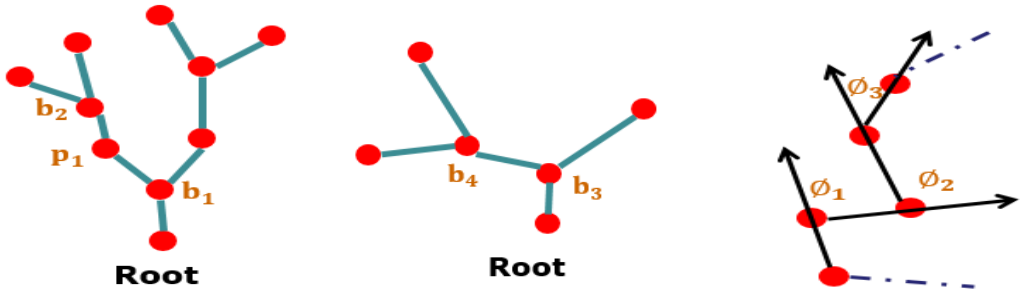
3.2.1 NeuroBFD

The rationale behind NeuroBFD is that we attempt to extract the parentnode-childnode statistics of the features, which is similar to the bigram statistics in natural language processing. Conventional research efforts extracted first degree statistics, such as mean diameter, in which local and discriminatory information is squashed inside a single statistical value. Instead, we show that this set of second-degree statistics constructs sparse and discriminatory features for neuron classification.

3.2.1.1 Feature construction

The feature construction for neuron classification is carried out on two stages - the extraction of a set of primary or raw features, and the determination of conditional distributions for each feature in the set. The primary features that we consider are bifurcation angle, fragmentation of each branch, and spatial density of sampled locations of each neuron. This set of features has visually significant variations in each neuron sample. For example, the bifurcation angle appears steeper, and the branches have 'wrinkles' in case of the motor neurons when compared to the ganglion neurons.

Bifurcation angle statistics: The distribution of bifurcation angle, can be computed for each neuron sample, and an empirical distribution of the bifurcation angle can be obtained by ensemble averaging all such distributions. However, it is evident that the arborization extent of a child node is dependent on its parent node in a neuron. For example, the bifurcation angle at a node guides the locations at which its subsequent child nodes sprawl their branches as explained in Fig. 3.8. This vital information appears to be missing in the empirical distribution. To incorporate such dependency of bifurcation angles, we develop a conditional distribution of bifurcation angle. In order to construct the distribution, the algorithm starts reading the location of each node from the *root* node. The set of bifurcation angles in order can be found by identifying the subset of vertices



(a) (b) (c)

FIGURE 3.8: (a) and (b) are two different neurons with designated root nodes. $b_k = k^{th}$ bifurcation. p = parent vertex. (a)p=immediate parent vertex of bifurcation b_2. b_1 is the immediate parent bifurcation vertex of b_2. Smaller angle at b_1 constricts the span of its child branches, especially at b_2. (b)larger angle at $b3$ helps b_4 to pan out. (c) describes how to measure branch fragmentation. $\phi_1, \phi_2 \geq \tau$, and $\phi_3 < \tau$.

for which the structural degree vector,D_{vs} at least equals 3. Let us denote this subset as $B_v = [p_1, p_2, ..., p_{N_v}]; N_v < N$. The bifurcation angles associated to the vertices in B_v are $\theta_v = [\theta_1, \theta_2, ..., \theta_{N_v}]$. Here, $\theta_k;\ k \in \{1, 2, ..., N_v\}$ is the angle between two child branch segments coming from p_k. We define the conditional bifurcation matrix $\Gamma \in \mathbb{R}^{S_1 \times S_1}$ by

$$\Gamma(i,j) = \begin{cases} \Gamma(i,j) + 1 & if\ \theta_i \sim \theta_j, \\ 0 & o.w., \end{cases} \tag{3.12}$$

where S_1 is the number of bins that equally divides the range $[0, 180]$. $\theta_i \sim \theta_j$ means that θ_i is the immediate parent bifurcation angle of θ_j, which is explained in Fig. 3.8. We claim that Γ is a discriminatory feature of a neuron. The example plots of Γ for one Pyramidal and one Purkinje cells are shown in Fig. 3.9.

TABLE 3.12: Confusion score

	Gang	Gran	Motor	Purk	Pyra
Ganglion	**85.7**	4.1	4.1	0	6.1
Granule	4.2	**81.7**	1.4	0	12.7
Motor	33.3	0	**66.7**	0	0
Purkinje	0	0	0	**100**	0
Pyramidal	1.4	3.3	3.2	1.2	**90.9**

Branch fragmentation: Fragmentation of each neuronal branches is one of the important features for neuron classification. For certain neuron cell types, the branch arbors

and their curvatures are smooth. Whereas, some cell types contain neurons with many wrinkles. For quantitative assessment of such branch wrinkles, we count the number of non-differentiable sampled locations at each branch. We set an arbitrary parameter τ as the threshold for angles between consecutive branch segments as explained in Fig. 3.8.

Let the set of leaf nodes (pendant vertices) of a neuron is Lf_v. Let the combined set of B_v and Lf_v be $\mathcal{V}_{bl}$. We define a fragmentation measure on each vertex in $\mathcal{V}_{bl}$. Let $v_k, v_m \in \mathcal{V}_{bl}$ be a leaf node and the parent bifurcation vertex (see Fig. 3.8) of the leaf node respectively. There are M points with $M-1$ segments in the branch $(v_k \longrightarrow v_m)$ of a neuron. There are $m_1 (< M)$ thresholded non-differentiable points in $(v_k \longrightarrow v_m)$. We define a normalized fragmentation score for v_k as $\chi = \frac{m_1}{M-1}$. By default, we set the normalized fragmentation score of the root vertex as unity. The normalized fragmentation matrix $\kappa \in \mathbb{R}^{S_2 \times S_2}$ of the branch is defined as

$$\kappa(i,j) = \begin{cases} \kappa(i,j) + 1 & if\ \chi_i \sim \chi_j, \\ 0 & o.w., \end{cases} \tag{3.13}$$

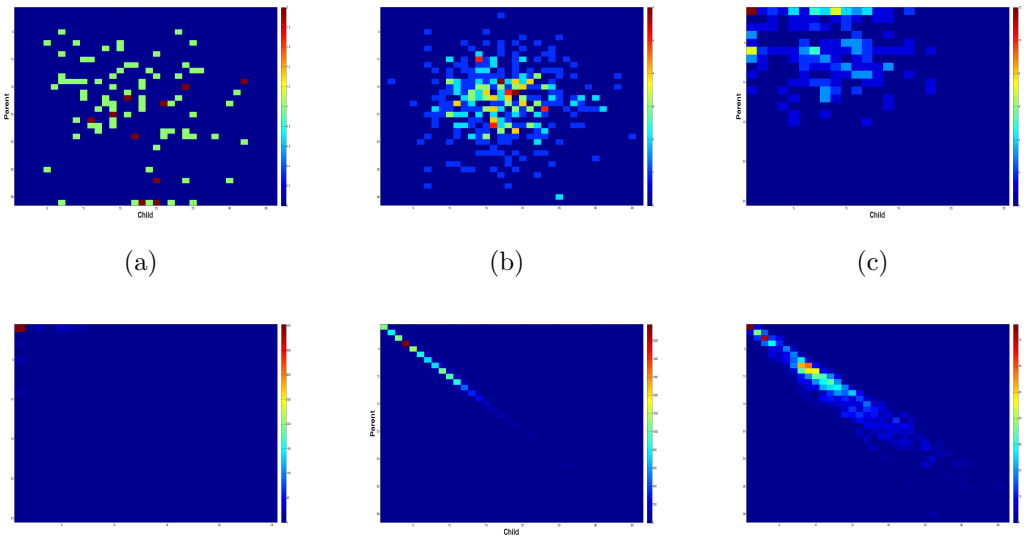
where S_2 is the number of bins for the range of normalized fragmentation score, which is $[0,\ 1]$. $\chi_i \sim \chi_j$ indicates that the vertex having normalized fragmentation score as χ_j is the parent bifurcation vertex of the one with the score as χ_i (Fig. 3.8). Examples of κ for a typical motor and purkinje cells are shown in 3.9(c) and (d) respectively. The motor cell exhibits significant fragmentation in each branch of the neuron. In contrast, the purkinje cell has comparatively minute fragmentation in branches.

Spatial density: The spatial density of a sampled location of a neuron is defined as the number of branches that pass through a unit volume centered at the location. In NeuroBFD, we exploit the complementary graph structure of a neuron to obtain a scale or depth based density measure. In graph terminology, the scale or depth of a vertex of a neuron (tree structure) is defined as the smallest number of branch segments between

the vertex and the root node. Let there are $m_\delta (> 1)$ vertices at a depth δ from the root node of a typical neuron. A subset of m_δ vertices generate $m_{\delta+1}$ child vertices at the next depth level $\delta + 1$. By the property of a tree, there are no edges within m_δ vertices. Similar argument holds for $m_{\delta+1}$ vertices. By construction, the complementary of the tree graph contains edges within and between the above mentioned sets of vertices. In NeuroBFD, we ignore the edges between two consecutive depth levels. Let d_δ and $d_{\delta+1}$ be the average distance between all pairs of m_δ and $m_{\delta+1}$ vertices respectively. A conditional distribution, $\Psi \in \mathbb{R}^{S_3 \times S_3}$ of all such $(d_\delta, d_{\delta+1})$ is a discriminatory feature descriptor for a cell type. Here, S_3 is the number of bins which encompass the range $[0,\ 1]$. As a convention, if at any depth, there is only one vertex, we set the average distance at that depth as zero. Fig. 3.9(e) and (f) show Ψ for a typical pyramidal and Purkinje cells respectively. The prominent diagonal of Ψ for the pyramidal cell implies that the branches and the sampled locations are largely in proximity. The off-diagonal entries in the Ψ for Purkinje indicates a significant spatial variation in the location of sampled points in the neuron. After computing Γ, κ, and Ψ, we vectorize each feature matrix, and then concatenate them to form the feature descriptor, $\Upsilon \in \mathbb{R}^{S_1^2+S_2^2+S_3^2}$ for a neuron. It can be noted from Fig. 3.9 that the features matrices are sparse as are the feature descriptor.

3.2.1.2 Results

We apply NeuroBFD on two different datasets containing digitally-traced neurons. The first dataset contains five cell types - ganglion, granule, motor, Purkinje, and pyramidal, which are taken from only one subject type, the mouse. The second dataset is taken from the work [79], which has pyramidal, motor, serotonergic, and ganglion as cell types taken from human, rat, cat, mouse and drosophila. We show the effectiveness of NeuroBFD by providing the confusion matrix for the first dataset. For the second dataset, we compare



(a) (b) (c)

(d) (e) (f)

FIGURE 3.9: Bifurcation matrix for (a) pyramidal and (b) Purkinje cells. Fragmentation for (c) motor and (d) Purkinje cells. Spatial density for (e) pyramidal and (f) Purkinje cells.

our results with [79] and [6]. In all the experiments, we set S_1, S_2, and S_3 as 36, 25, and 36 respectively. The threshold, τ for measuring the fragmentation is set as 15^0.

Dataset 1: This dataset contains a total number of 4434 neurons, out of which 1377 are ganglion, 1195 are granule, 116 are motor, 57 are Purkinje, and 1729 are pyramidal neurons. The training dataset contains 1288 (ganglion), 1124 (granule), 113 (motor), 54 (Purkinje), and 1563 (pyramidal) neuron samples with rest of the data as the test dataset. This 5-class classification is carried out by using support vector machines (SVM) with linear kernel. The linear kernel is suitable because each feature descriptor is sparse and of sufficiently high dimension ($\in \mathbb{R}^{3217}$). In our approach, we build 10 classifiers by taking each pair of cell types together. So there are 10 linear SVMs which are trained using the training dataset. We apply the majority voting algorithm to determine the predicted class of each test sample.

The confusion score is given in the following table 3.12. The overall accuracy is 85%. The accuracy for motor cells drops because it has similar morphology as of ganglion. In addition, the experimental test is carried out only on 3 samples of motor cells.

Dataset 2: The description of the dataset is provided in Table 3.14. We compare the results of NeuroBFD against NeuroSoL [79] and Path2Path [6]. The metrics for comparison against NeuroSoL and Path2Path are chosen as $\frac{avg.\ interclass\ similarity}{avg.\ intraclass\ similarity}$ and $\frac{avg.\ intraclass\ distance}{avg.\ interclass\ distance}$. The values of these metrics are expected to be smaller as smaller intraclass distance implies the same-class features are close together, and larger interclass distance implies the between class features are far apart. In NeuroBFD, we select the same metric as in Path2Path. The comparative result is provided in Table 3.13. The

TABLE 3.13: Pairwise interclass distances

	NeuroBFD	NeuroSoL	path2Path
Pyra-Motor	**0.18**	0.24	0.49
Motor-Seroto	**0.21**	0.35	0.57
Pyra-Seroto	**0.10**	0.12	0.47

results indicate that NeuroBFD outperforms the other two approaches

NeuroBFD adopts the conditional distribution based approach in which the one to one correspondence between the graph structure of a neuron and the distribution of its features can not be maintained. Instead of working in the transformed domain (in case of NeuroBFD, it is the distribution), we use the graph domain to construct anatomical features in terms of graph matrices.

3.2.2 Neuron solver using Laplacian (NeuroSoL)

We propose an automated algorithm, *NeuroSoL* to obtain a feasible solution by modeling a neuron as a graph with the sampled points on the neuron as vertex-set. In this framework, comparing the anatomy of neurons translates to graph matching. It is important to notice that graph matching is a computationally challenging problem. More specifically, graph isomorphism poses a computational bottleneck in performing shape matching of graphs given the same cardinality of vertex sets. The challenge becomes intensified for large scale graphs when the cardinalities of vertex sets are different leading to an exhaustive search of subgraph isomorphism.

A typical dataset of *registered* neurons contains the coordinates of locations or vertices of each neuron. The assignment of coordinates significantly reduces the possibility of isomorphic graphs. In order to seize the morphological variation, we extract two crucial features - structure and connectivity in terms of graph measures. In addition, the geometric variation is captured using the combination of original and complementary graphs of a neuron [80]. We claim that the assembly of sorted graph measures extracted from these two graphs of a neuron constitute a set of robust feature descriptors.

3.2.2.1 Vertex labeling

The graph model of a neuron in fig. 3.10(a) is shown in fig. 3.10(b) with labels on vertices and weights on edges. Given such a simple, connected, and weighted graph $\mathcal{G}$, labeled[1] by $f_{\mathbb{N}}$ it is possible to define a new labeling function f_I, by introducing a bijection (permutation) I such that $f_I = I(f_{\mathbb{N}}(\mathcal{V}))$. The various properties of the graph described by the matrices $\boldsymbol{A}$, $\boldsymbol{D}$, and $\boldsymbol{L}$, are invariant with respect to permutation of the vertex labeling. What we are aiming for, is to define a bijection f_I such that, given a graph, vertices are labeled with numbers based on the decreasing order of *relevance*.

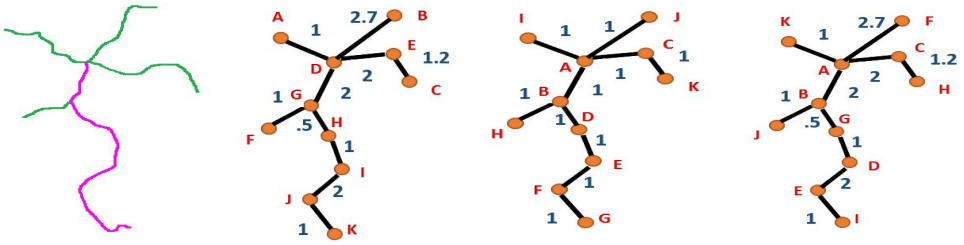


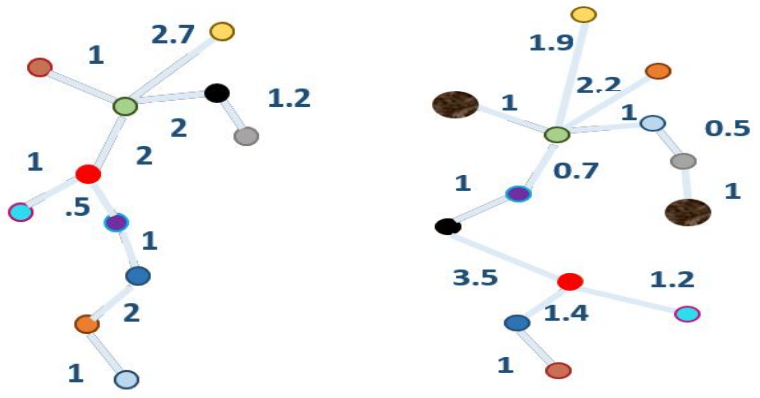
FIGURE 3.10: (a) A neuron sample, and (b) its graphical representation. (c) Sorted structure graph, and (d) connectivity graph.

When defining the concept of *relevance*, we look at the number of incident edges and the actual degree of each vertex. The former is a measure of structural relevance of a vertex to the graph. The actual degree, obtained by adding up the weights of incident edges, reflects the relevance of connectivity induced by the vertex to the graph. Fairly

[1]For the rest of this work, we will use the word *labeling* to indicate *vertex-labeling*.

large values of both for a vertex indicate that the vertex contributes significantly to the entire graph. On the other hand, small values of both imply that the vertex has low priority. This way of quantification of relevance is manifested in degree-based sorting in the decreasing order of relevance of vertices. As a result of sorting, the elements of structure and connectivity Laplacians are permuted. It is to note that sorting is a well-known candidate of the class of vertex-labeling functions.

To define vertex-labeling of structure Laplacian, first we derive the structure graph of a neuron by assigning unity-valued edge weights. Under this setting, the degree matrix is written as $\boldsymbol{D}_s$ and the corresponding Laplacian as $\boldsymbol{L}_s$. The connectivity graph is obtained by fixing each weight as the Euclidean distance between two vertices that form the edge. The derived degree and Laplacian matrices are denoted by $\boldsymbol{D}_c$, and $\boldsymbol{L}_c$ respectively. For remainder of the description, we use subscripts s and c as abbreviations for *structure* and *connectivity*.



(a) (b)

FIGURE 3.11: (a)-(b) Correspondence of relevance (shown in colors) between the connectivity graphs of two different neurons of two different sizes.

In order to describe sorting for structure and connectivity graphs, we introduce two sorting functions (vertex labeling) f_s and f_c. The rearranged Laplacians based on sorted

degree matrices respectively are given by

$$\boldsymbol{L}_{f_s} = \boldsymbol{L}_s\big(f_s o f_{\mathbb{N}}\big) \ , \ \boldsymbol{L}_{f_c} = \boldsymbol{L}_c\big(f_c o f_{\mathbb{N}}\big) \tag{3.14}$$

To couple structure with connectivity, we also define a 'mixed' sorting function $f_{c\leftarrow s}$. Often, while sorting the diagonal of $\boldsymbol{L}_c$, a subset of vertices may have same degree, which leads to an inconsistency in the assignment of relevance. To resolve this problem, the *order* of relevance of the subset in $\boldsymbol{D}_{f_s}$ is examined and then plugged into the connectivity Laplacian as shown in eq. 3.15.

$$\boldsymbol{D}_{f_{c\leftarrow s}} = \begin{cases} d_i > d_j & \text{Then } f_{c\leftarrow s}(i) \ \prec \ f_{c\leftarrow s}(j), \\ d_i = .. = d_j & \text{Check } v_i = .. = v_j \ in \ f_s \\ randomly & o.w. \end{cases} \tag{3.15}$$

The structure and connectivity graphs of fig.4.5(b) are shown in fig.4.5(c) and (d) respectively. It is important to note that the labels of vertices in fig.3.10(c)and(d) are different from fig.3.10(a). The labels are given in capital alphabets $\{A, B, ..., K\}$ with decreasing order of relevance i.e. $A \prec B \prec ... \prec K$. The connectivity graph is mixed-sorted. It is because to define the order of relevance between D and E in fig.4.5(b) (both of degree 3), the order is checked in the structure graph ($E \prec F$).

Fig. 3.11 exhibits our method to compare two neurons with different cardinalities using only connectivity graphs. The color of each vertex in the two graphs indicates the correspondence of relevance. It means that, for example, the green vertex of the left neuron is compared with that of right neuron in fig. 3.11. The two vertices of the comparatively larger circles in fig. 3.11(b) are left unmatched with fig. 3.11(a). This is why we introduce two *dummy* vertices in the feature descriptor of fig. 3.11(a) to make the

cardinality of vertex-sets equal. It is evident that this procedure automatically prunes leaves (vertices with structural degree 1) without involving any preprocessing step.

TABLE 3.14: Neurons used in our experiments

Archive	Animal	Region	Cell Type
Allman	Human	Neocortex	Pyramidal
Jacobs	Human	Neocortex	Pyramidal
De Koninck	Rat	Neocortex	Pyramidal
Brown	Rat	Neocortex	Pyramidal
Cameron	Cat	Spinal Cord	Motor
Fyffe	Cat	Spinal Cord	Motor
Lu Lichtman	Mouse	Brainstem	Motor
Chiang	Drosophila	Adult central	Serotonergic
Chiang	Drosophila	Optic lobe	Serotonergic
Chalupa	Mouse	Retina	Ganglion
Miller	Mouse	Retina	Ganglion

3.2.2.2 Feature extraction

The sorted structure Laplacian (SSL) and mixed sorted connectivity Laplacian (MSCL) of a graph capture the morphology of the neuron. In SSL, the set of vertices with degree 3 indicates the number of *bifurcations*. The sum of diagonal elements of SSL gives twice the total number of branches. Due to tree-structure of neurons, the top two path lengths from the highest degree vertex of SSL gives the height and depth of a neuron. MSCL reflects the deformed *topology* of a neuron, *total length*, *maximum path distance to root*. To capture the geometric features, Laplacians $(\bar{\boldsymbol{L}}_{f_s}, \bar{\boldsymbol{L}}_{f_{c\leftarrow s}})$ of complementary graph are added in the feature assembly. The final feature descriptor of a neuron with N sample points becomes

$$\Psi_N = \left[\boldsymbol{L}_{f_s}|\boldsymbol{L}_{f_{c\leftarrow s}}|\bar{\boldsymbol{L}}_{f_s}|\bar{\boldsymbol{L}}_{f_{c\leftarrow s}}\right] \tag{3.16}$$

To compare two neurons, the feature descriptors are first aligned by optimizing a cost function. This procedure is discussed next.

3.2.2.3 Optimization

Let us consider two neurons with vertex sets of cardinality N_1 and N_2 such that $N_1 = N_2 + M$ and $0 < M < min(N_1, N_2)$. The feature descriptors of the neurons are Ψ_{N_1} and Ψ_{N_2} respectively. To measure element-wise distance between Ψ_{N_1} and Ψ_{N_2}, Ψ_{N_2} needs to be zero-padded accounting for the rows (columns) of deficit vertices. Therefore, M zero rows (columns) are added to the bottom (right) of each Laplacian in Ψ_{N_2}. Let the zero-padded Ψ_{N_2} be $\Psi^0_{N_2}$.

However, Ψ_{N_1} and $\Psi^0_{N_2}$ still might be element-wise misaligned. The obvious reason is that when M zero rows (columns) are attached to the bottom (right) of Ψ_{N_2}, it is implicitly assumed that the dummy M vertices have least relevance. This might not be the case for Ψ_{N_1}. To resolve the problem, the locations (vertex numbers) of the zero rows (columns) are tweaked, which leads to a set of *four* vertex labeling functions $f^i_{2\leftarrow 1} : \mathcal{V}_{N_2} \cup \mathcal{V}_M \longrightarrow \mathcal{V}_{N_1}$ with $\mathcal{V}_M$ as the set of dummy vertices. For example, $i = 1$ is for $\boldsymbol{L}_{f_s}$ in eq. 3.16. The four labeling functions correspond to four Laplacians in Ψ_{N_2} respectively.

In a mathematical framework, the alignment problem searches for a bijective function $f_{2\leftarrow 1}$ between the vertex sets of $\mathcal{G}_{N_1}$ and $\mathcal{G}_{N_2}$ to maximize a cost function $\mathcal{D}$,

$$\hat{f}^i_{2\leftarrow 1} = \max_{f_{2\leftarrow 1}} \mathcal{D}\Big(\boldsymbol{L}^{N_1}_i, \boldsymbol{L}^{N_2}_i\Big). \tag{3.17}$$

In eq. 3.17, $\boldsymbol{L}^{N_1}_i$ is the i^{th} Laplacian of Ψ_{N_1}, and $\mathcal{D}$ is the cost function, given by

$$\begin{aligned} \Theta &= \Phi(\boldsymbol{L}^{N_1}_i) \odot \Phi(\boldsymbol{L}^{N_2}_i);\ \epsilon \in (0,1) \\ \mathcal{D}\Big(\boldsymbol{L}^{N_1}_i, \boldsymbol{L}^{N_2}_i\Big) &= \frac{\mathbf{1}^T \Theta \mathbf{1}}{||\big(\boldsymbol{L}^{N_1}_i - \boldsymbol{L}^{N_2}_i\big) \odot \Theta||^2_F + \epsilon} \end{aligned} \tag{3.18}$$

In eq. 3.18, Φ is a function which assigns unity to a non-zero element in a matrix. $\odot$ is element-wise matrix multiplication. $\mathbf{1}$ is a column vector of all 1s. The numerator

of the above cost function measures the number of non-zero entries in both matrices being aligned (overlapped) for a specific $f_{2\leftarrow 1}$. The denominator computes the distance between two Laplacians after applying $f_{2\leftarrow 1}$. Overall, the objective implies maximum alignment with minimum distance between a pair of Laplacians. It is assumed that no neuron is fully-connected. Otherwise, $||\bar{\boldsymbol{L}}_i^{N_1} - \bar{\boldsymbol{L}}_i^{N_2}||_F^2 = 0$ for the complementary of a fully-connected graph.

In order to achieve a score by comparing two classes (neuron-type, subject-type), C_i and C_j, we calculate the tuple of minimum separability, λ_1, and maximum separability, λ_2, between two classes as our similarity measure as follows:

$$\begin{aligned} \lambda_1 &= min_{x\in C_i, y\in C_j} \mathcal{D}(\Psi_{N_i}, \Psi_{N_j}) \\ \lambda_2 &= max_{x\in C_i, y\in C_j} \mathcal{D}(\Psi_{N_i}, \Psi_{N_j}) \end{aligned} \tag{3.19}$$

3.2.2.4 Results

We apply *NeuroSoL* to 4 classes of digitally-traced neurons taken from different test subjects [75] as shown in Table 3.14. The effectiveness of our proposed approach is demonstrated by presenting intra-subject variation and cell type specific variation. It is to notice that the separability measure in eq. 3.19 uses the cost function in eq. 3.18. Therefore, fairly large values of λ_1 and λ_2 in eq. 3.19 of two neurons implies that the neurons belong to the same class. On the other hand, two neurons with comparatively smaller values of λ_1 and λ_2 belong to different classes respectively. The neuron samples are manually selected from [75]. The maximum and minimum sizes of all the neurons used in our experiment are 4244 and 213 respectively.

The cell type test samples include four categories of neuron - Pyramidal, Motor, Serotonergic, and Ganglion. Before computing the separability scores, all the samples in each cell type are put together followed by a random selection of 15 neurons from each

cell type. This process is iterated 4 times and the separability scores are noted for each pair of cell types.

The cell type maximum and minimum separability values are given in Table 3.15. Due to space-constraint, the scores are reported by rounding off to nearest integers.

TABLE 3.15: Cell type minimum and maximum separability

	Pyra	Motor	Sero	Gangl
Pyra		$(37-101)$	$(11-58)$	$(41-109)$
Motor			$(61-73)$	$(67-112)$
Sero				$(12-22)$

To compute the subject-specific variability, subjects are taken from different archives. As an example, to measure the human-specific separations, Allman and Jacobs archives are considered to be two separate classes. Under this test setting, the intra-subject variations by taking human, cat, rat, and drosophila are reported in Table 3.16.

TABLE 3.16: Intra-subject separation

Human	Cat	Rat	Dros
$(302-377)$	$(168-237)$	$(318-571)$	$(147-213)$

It can be observed from Table 3.15 that the cell type classes - serotonergic and ganglion are well-separated as indicated by low values of separability scores. The large deviations in other pairs can be attributed to different subjects involved in the computation of separability. As an example, comparison between Pyramidal and Motor cell types involves four subjects - human, rat, cat, and mouse. For intra-subject variation from Table 3.16, Human and Rat show comparatively higher scores indicating class-specific similarity respectively.

NeuroSoL is a registration-independent approach because it incorporates the Euclidean distance between each pair of vertices as defined by the graph connectivity. The use of vertex relevance, and the combination of neuron graph and its complementary graph can capture the geometry of the neuron. It enables the visualization of one-one correspondence of vertices between a pair of neurons. However, NeuroSoL is plagued

with the variation of the number of sampled locations of all neurons. In short, NeuroSoL is cardinality dependent, implying the possibility of the subgraph isomorphism problem and triggering sub-optimality in its optimization step.

In summary, conventional state-of-the art methods [7, 78] for the classification of neurons can be broadly divided in two categories. Research in the $first$ category, which is supervised in nature, employs different feature extraction algorithms followed by suitable classifiers to obtain classification accuracy in percentage. The validation of the methods are performed by adopting a series of statistical tests. However, the significant variation in the neuron skeletons precludes the selection of the optimal set of morphometrics as features. Adoption of feature transformations, such as principal component analysis (PCA) or kernel transformations, may improve the classification accuracy. Nevertheless, these transformations obscures the identity of discriminating features as the transformed space is formed by linear or nonlinear composition of extracted features. In addition, the classification accuracy of categorization does not quite explain the physiological and structural differences between two neurons.

The *second* category follows mostly unsupervised approaches and attempts to compute pairwise distances between neurons. Authors in [81] used Fourier based shape descriptors to encode the global shape of a neuron, which however ignored the local features of the neuron arbor. Gillette [82, 83] performed a sequence alignment based algorithm for categorization by decomposing a neuron into a sequence of branches. The approach failed to consider geometric features. Blastneuron [78] adopted a mixed strategy. Using a supervised approach, the method first extracted 21 global morphological features and 13 moment invariant features to retrieve a set of targets that closely matches to each test neuron in terms of the anatomical structures. Each target is then RANSAC sampled [84] and aligned optimally to the test neuron, which outputs a distance value. This unsupervised routine decides the output category of the test neuron based on the

minimum distance criteria. The method involved initial pruning of branches and resampling of each neuron, which collectively alters the morphology statistics. Moreover, the retrieval accuracy of 233 projected neurons (PN) of Drosophila drops significantly to 39% as the number of potential candidates that are to be compared with the target increases. NeuroSoL [79] offered a graph-theoretic method which is free from registration and resampling. In spite of its appeal of using graph theory, the matrix alignment routine is NP-hard in nature, thereby producing suboptimal results. The problem of comparing a pair of neuron topologies can also be regarded as a graph kernel based similarity measure problem [85]. However, the rationale behind conventional graph kernels, such as the random walk kernel may be inconsistent with the morphological understanding of a neuron.

Instead of modeling a neuron as a generic graph, the neuron can be modeled as a specialized graph that contains a collection of rooted paths, where each path starts from the soma, called the root node, and ends up in a dendritic terminal. It is important to note that each path acts like an *atomic neuron*, as it contains the soma and a dendritic end to complete a circuit. Most of the synapses along a path will be nearer the soma than at the end of the path. It is convenient to think problems, such as synaptic plasticity as the evolution of a set of synapses over time along all the paths. During this evolution, there are birth, death and rearrangement of paths. Following the same logic, *quantifying the problem of distinguishing two neurons can be equivalently mapped as finding the cost of evolving a set of circuits optimally from one neuron to the other.*

Another relevant fact is that path based models [6,69,86] integrate both global (overall shape based approach) and local (vertex or sampled location based approach) features of neuron topology. Topological morphology descriptor (TMD) [69] aimed at solving the categorization problem, encoded the birth and death of path segments over time in a persistence diagram used as a barcode. The authors showed that TMD exhibits robustness to

erroneous 3D sampling and ambiguous branching when the neuron is reconstructed using two different tracing tools. However, the conversion of a discrete 3D reconstructed neuron to the persistence image space is irreversible and many-to-one. Based on the distance used to mark and quantify the birth and death of a branch or component of the neuronal tree, a single persistence image may correspond to multiple neurons. In addition, an appropriate distance measure between persistence diagrams is still unavailable. The work in Path2Path [6] shows potential to address the neuron cell categorization problem and can be extended to several other related problems, such as synaptogenesis, degeneration of neurons due to neurological diseases, and synaptic plasticity which can be studied by inspecting the path statistics. The work described in this article is motivated by the framework of Path2Path.

3.2.3 What is Path2Path and its variants?

The principle of Path2Path is based on finding the optimal correspondence between the paths of one neuron to that of the other using a proposed metric. It is an intuitive circuit-based approach that appeals to its electrical engineering inventors. In Path2Path, each sampled location on a path is endowed with 3D coordinate values and two features, *concurrence* and *hierarchy*. The concurrence value at each location denotes the number of paths from the soma to dendritic ends that visit that node. The hierarchy value at a location indicates the depth of the location from the soma in terms of the number of bifurcations between the point and the soma. The hierarchy value of a location counts the number of bifurcations one has to cross while traversing from the soma to that location. Using the 3D coordinates, concurrence, and hierarchy values of each location on a path, authors in Path2Path proposed an empirical metric that outputs a distance value between two paths. A path from a neuron corresponds to a path from another neuron if the distance between the paths is minimum over all the paths of the latter neuron.

This approach has several drawbacks. The Path2Path algorithm is dependent on the number of sampled locations of each path and the registration. The selection of the metric is arbitrary in a sense that the metric is null when two paths have the same set of concurrence values but different locations and hierarchy values. Therefore, it does not qualify the axioms of a metric. In addition, the proposed distance measure uses the Euclidean distance between two paths as a part of the distance computation routine, which favors the pair if they are aligned in proximity after registration. The algorithm of finding the correspondence is not one-to-one and it often leads to the degenerate case where all paths from one neuron are matched to only one path in the second neuron. The problem exacerbates when the number of samples in the two paths are unequal. One potential solution is to resample each path using a constant step [78], but may, unfortunately, eliminate the importance of the locations, such as curvature of a rooted path prior to resampling.

3.2.4 ElasticPath2Path

ElasticPath2Path [86] attempted to address the previously mentioned problems. It introduces a mid-point based resampling routine as opposed to constant-length resampling. To ensure one-to-one correspondence between a pair of paths from two different neurons, the Munkres algorithm [87] is employed. Most importantly, ElasticPath2Path envisages the problem of distinguishing two neurons as a continuous deformation between the corresponding paths of the neurons. Such homeomorphism is computed by applying the square root velocity function (SRVF) [9] to the Euclidean coordinates of each sample on a path. The visual deformation of the corresponding paths has an enormous impact in the validation of the path based on customized features and the proposed distance measure.

3.2.4.1 Neuron as a graph

We model a neuron with a simple, connected, undirected tree $\mathcal{G}$ with a designated root node. A neuron $\mathcal{G}$ with n dendritic terminals can be decomposed into n paths $f_i;\ \{1, 2, ..., n\}$. f_i is a continuous function for each path such that $f_i : [0, 1] \longrightarrow \mathcal{R}^3$ with $f(0) = (0, 0, 0)$. Let Γ be the set of all such f_is, which is a linear subspace of the classical Wiener space. For numerical computation, f_i is finitely sampled and each sample is treated as a vertex of the neuron graph.

The concept of *path concurrence*, C_{f_i} of a vertex originates from counting the number of times a given vertex is revisited in all f_is. If $C_{f_i}(t_s) = k;\ t_s \in [0, 1],\ k \in \mathcal{N}$, the vertex at t_s on the path f_i is shared among k paths of the neuron, $\mathcal{G}$. As given in [6] the path concurrence value can be mathematically represented by

$$\begin{aligned} C_{f_i}(t) = k;\ t \in [0, 1],\ j_1, j_2, \ldots, j_k \in \{1, 2, ..., n\} \\ f_i(t) = f_{j_1}([0, 1]) \cap f_{j_2}([0, 1]) \cap \cdots \cap f_{j_k}([0, 1]). \end{aligned} \tag{3.20}$$

After the computation of the path concurrence values at all the vertices, authors in [6] introduced the concept of path hierarchy values. The path hierarchy value of a vertex on a path f_i, $H_{f_i}(t_s)$ is the number of times the concurrent paths do not visit the vertex while traversing from the root node to the dendritic terminal of f_i.

In [6], each vertex on an arbitrary path, f_i therefore has 3D location, path concurrence value $C_{f_i}(t_s)$ and path hierarchy value $H_{f_i}(t_s)$ as shown in Fig.4.5. The distance metric between any two paths f_i and f_j was given in [6] as

$$D\big(f_i, f_j\big) = \int_0^1 \frac{|C_{f_i}(t) - C_{f_j}(t)||f_i(t) - f_j(t)|}{\lambda + \sqrt{H_{f_i}(t) H_{f_j}(t)}} dt. \tag{3.21}$$

In eq. (3.21), λ is a positive constant to avoid singularity.

3.2.4.2 Elastic morphing and SRVF

The metric in (3.21) contains the absolute difference term $|f_i(t) - f_j(t)|$, which is the Euclidean distance between the two paths. From the point of view of geometry, a path can be thought of as an open curve which starts from a designated root node, and ends at a given dendritic terminal. However, the submanifold $\subset \mathcal{L}^2([0,1], \mathcal{R}^3)$ consisting of all the open and closed curves is not Euclidean. To measure the geodesic distance between f_i and f_j, we need a suitable shape representation and a Riemannian metric.

It is shown in [9] that after the transformation of the space of curves by square root velocity (SRV) function, the space becomes Euclidean with the Euclidean distance acting as an elastic metric. The SRV of an arbitrary path f_i can be given by $q_i(t) = \dot{f}_i(t)/\sqrt{||\dot{f}_i(t)||};\ t \in [0,1]$. The *elasticity* comes from the fact that a curve can continuously deform from one shape to another like a rubber band. To perform such continuously elastic morphing, one needs to take care of scaling variability, rotation, translation, and reparametrization of curves.

***Translation**:* The SRV transformation inherently takes care of the translation factor.

***Scaling**:* To tackle the scaling variability, authors in [9] restrict the lengths of all the curves to unity, which transforms the flat Euclidean space to a sphere. Therefore, in order to retrieve the intermediate deformations from one curve to another, one needs to find and traverse the geodesic path on the hypersphere. This poses a problem in the path matching between neurons. The restriction of unit length significantly alters the morphology of the paths. This is because the paths from the root to the dendritic terminals of a neuron differ in lengths, creating the distinctive morphology of the neuron. In our work, we do not impose the restriction of unit length of a path.

***Rotation and Reparametrization**:* The rotation group, $SO(3)$ and the reparametrization group, Γ are compact Lie groups. Let M be the Euclidean manifold of all the open

curves after SRV transformation. The individual quotient spaces, $M/SO(3)$ and M/Γ are also submanifolds inheriting the Riemannian metric of M, which ensures that the quotient space, $M/\big(\Gamma \times SO(3)\big)$ is also a submanifold. Therefore, any arbitrary path $f_i \in M$ is, at first, subjected to rotation and reparametrization, if required, followed by the SRV transformation q_i to find an element in the quotient space. The registration of f_i with respect to f_j via rotation is performed by Kabsch algorithm [88], which registers two sets of coordinate vectors.

To introduce reparametrization, first note that the numerical implementation of (3.21) requires an equal number of vertices in f_i and f_j. However, in practice, the number of vertices differs significantly from path to path. In addition, the locations of the vertices are fairly nonuniform to account for the path fragmentation, *wiggliness* of path segments [7], which are essential structural characteristics of a neuron. It is evident that for two curves of arbitrary lengths, more samples (vertices) approximate (3.21) as an integral. The error in distance, $|f_i(t) - f_j(t)|$ between two curves decreases with the increase in the number of samples. Notice that the resampling of a path is a class of reparametrization function.

In contrast to the resampling routine in Path2Path, we keep the positions of the actual vertices on a path f_i fixed, and add other samples in between in an iteratively sequential fashion. Between two consecutive samples on a path, we insert the midpoint of the samples as a new point. This procedure retains the neuronal characteristics of each path. The concurrence, C_{f_i}, C_{f_j} and the hierarchy, H_{f_i}, H_{f_j} values are interpolated as $\tilde{C}_{f_i}, \tilde{C}_{f_j}$ and $\tilde{H}_{f_i}, \tilde{H}_{f_j}$ respectively.

Let the number of vertices of f_i and f_j are N_i and N_j. The number of samples in the resampling procedure is fixed as ρ. After reparamterization, the paths become $\tilde{f}_i$ and $\tilde{f}_j$, which are subjected to SRV function producing q_i and q_j respectively. The q_i is then

rotated as $\tilde{q}_i$ with respect to q_j for registration. The distance $D\big(f_i, f_j\big)$ between the two paths is computed by inserting $\tilde{q}_i$, q_j, $\tilde{C}_{f_i}, \tilde{C}_{f_j}$, $\tilde{H}_{f_i}$ and$\tilde{H}_{f_j}$ in (3.21).

3.2.4.3 Path-to-Path matching

In Path2Path [6], the matching algorithm is greedy, which has a serious drawback of singularity in which all paths in one neuron can be matched to only one path in the other neuron. We tackled the problem by defining a one-to-one job assignment problem. Let us consider two different neurons, $\mathcal{G}_1$ and $\mathcal{G}_2$ having the sets of paths as $P_1 = \{f_1^1, f_2^1, \dots f_{|P_1|}^1\}$ and $P_2 = \{f_1^2, f_2^2, \dots f_{|P_2|}^2\}$ with $|P_1| \leq |P_2|$ respectively. Here $|P|$ indicates the cardinality of the set P. The distance measure, $D(f_i^1, f_j^2)$ as defined in eq. (3.21) can be regarded as a cost between two paths f_i^1 and f_j^2. Let $C \in \mathcal{R}^{|P_1|\times|P_2|}$ be the matrix with $C(i,j) = D(f_i^1, f_j^2)$. The path-to-path matching problem between two neurons can be regarded as a variant of a job assignment problem. Here, $|P_1|$ is the number of workers and $|P_2|$ is the number of jobs that are to be assigned to the workers. As in most of the cases, $|P_1| \neq |P_2|$, the assignment problem is unbalanced. We append $(|P_2| - |P_1|)$ zero rows to the bottom of C as dummy workers. The optimal one-one job assignment is then performed using the Hungarian algorithm [87].

3.2.4.4 Datasets and Results

We apply Elastic Path2Path (ElasticP2P for short) on a dataset containing (.swc format) files of digitally-traced neurons taken from Neuromorpho.org. The dataset consists of five major cell types - pyramidal, granule, motor, purkinje, and ganglion. To restrict the model organism, we consider the murine neurons only. There are total 4434 neurons used in our study, out of which 1729 are pyramidal, 1195 are granule, 116 are motor, 57 are purkinje, and 1377 are ganglion. An example of the path correspondences in ElasticP2P in case of three major cell types, which are pyramidal, motor, and ganglion are exhibited in Fig. 3.12. The pyramidal, motor, and ganglion neuron samples contain 28, 35, and 14 rooted paths respectively. The costs of morphing from one neuron to another neuron

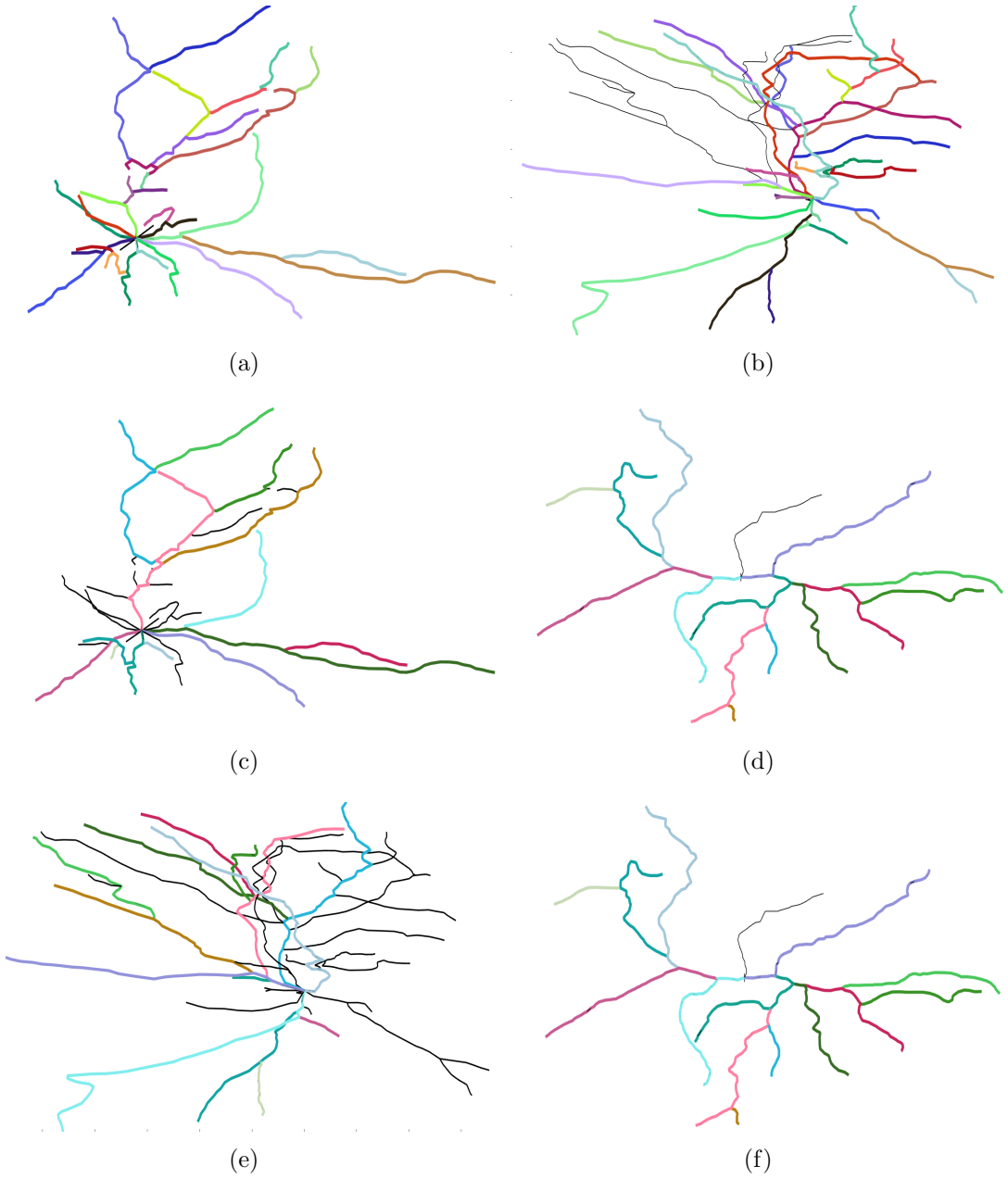


FIGURE 3.12: The path correspondences are shown in the corresponding colors in case of (a)-(b) pyramidal-motor, (c)-(d) pyramidal-ganglion, and (e)-(f) motor-ganglion cell types. The paths which are matched are depicted in thick colored lines. The paths which are left out (larger neurons) are shown in thin black lines.

are 478.67 (pyramidal-motor pair), 514.09 (pyramidal-ganglion pair), and 353.14 (motor-ganglion pair). We also check the consistency of ElasticP2P by computing distances between same samples, which turn out to be null.

Due to the variation in the cardinalities of the sets for the five cell types, we resort to unsupervised classification. The dataset is partitioned into a set for clustering and a

test set for determining the retrieval accuracy. We perform different levels of partitioning of the dataset to test the resilience of our approach over NeuroBFD and Path2Path. At each level, we partitioned the dataset five times randomly maintaining the same ratio between the size of the training and test dataset. The retrieval is carried out using majority vote rule. At each partition, for a candidate in the retrieval set, we compute the nearest 11 samples from the cluster set. The class which appears largest number of times out of 11 labels is assigned to the candidate neuron. We average the retrieval scores and show them in Fig. 3.13. In all the instances of this experiment, ρ is kept 100. The results suggest that ElasticP2P exhibits consistent performance over a wide range of

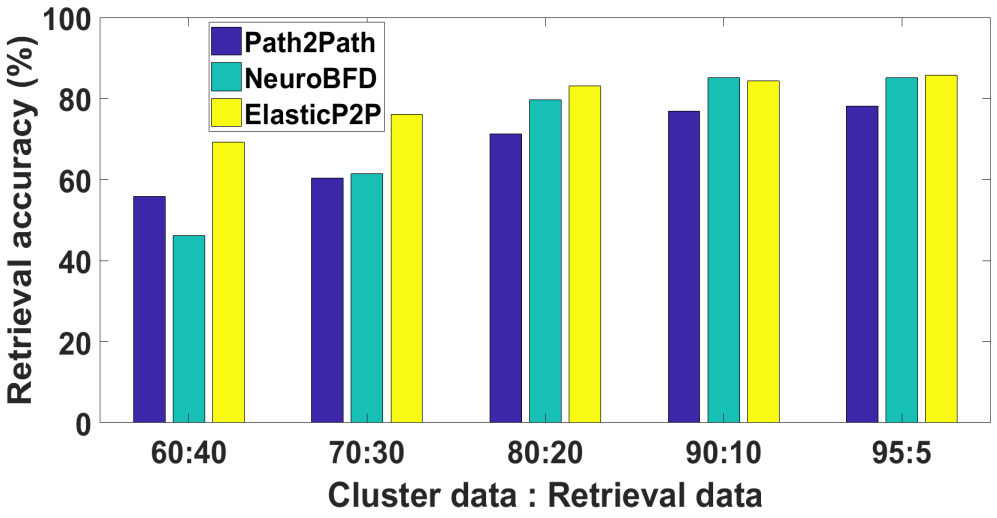


FIGURE 3.13: The retrieval performance (in %) of ElasticP2P against Path2Path [6] and NeuroBFD [7].

data partitioning compared to Path2Path and NeuroBFD. The classification method in NeuroBFD is supervised, which used a set of linear SVMs. With the gradual reduction in the amount of training data, the performance of NeuroBFD collapses as evident from Fig. 3.13. At the data ratio of 9 : 1, NeuroBFD performs marginally better (85.1%) than ElasticP2P (84.3%). The consistent improvement over Path2Path can be attributed to the insertion of the one-to-one path assignment routine and the rectified resampling routine in ElasticP2P.

The only hyperparameter of ElasticP2P is ρ, the number of samples after resampling. We demonstrate the effectiveness in terms of retrieval accuracy and computational demand in terms of execution time of our algorithm with different choices of ρ in Fig. 3.14. In Fig. 3.14, a pyramidal cell and a ganglion cell containing 28 rooted paths with 463

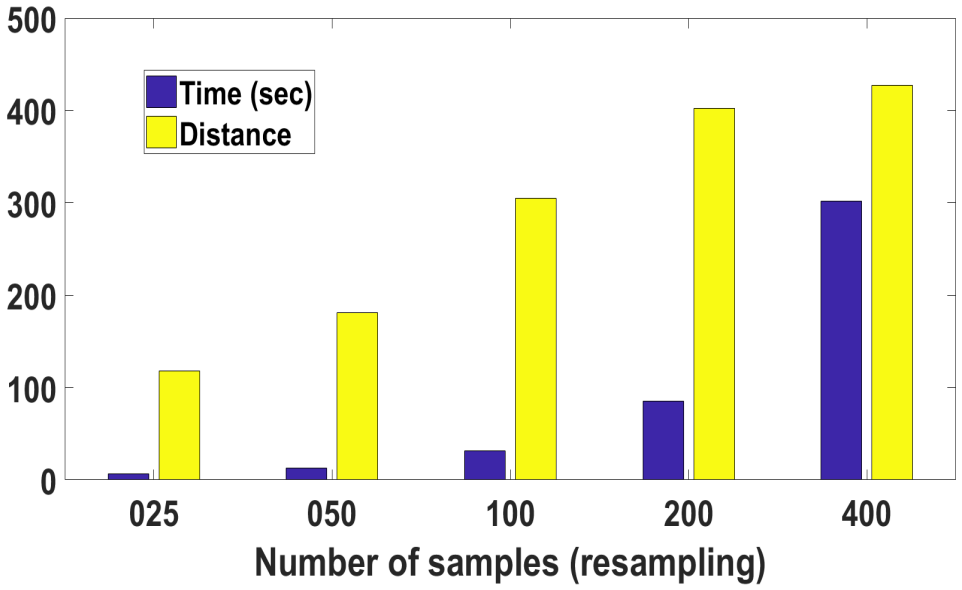


FIGURE 3.14: The plot shows the distance between a pyramidal and a ganglion neuron and the computational time (in seconds) when 25, 50, 100, 200 and 400 samples are taken per path.

3D locations and 40 paths with 2588 3D locations respectively are selected. The distance and computation time required as a function of the number of samples after resampling are shown.

ElasticP2P laid the foundation of the visualization as well as the classification of complex tree-structures. However, elasticPath2Path failed to address the problem where there is a significant difference in the number of paths between two neurons. As the correspondence is one-to-one, it asserts that elasticPath2Path performs subgraph matching. Both Path2Path and elasticPath2Path did not consider important anatomical morphometrics, such as bifurcation angles and partition asymmetry. Besides, elasticP2P is still registration-dependent, cardinality-dependent and contains no biology-driven rationale behind the selection of its distance function.

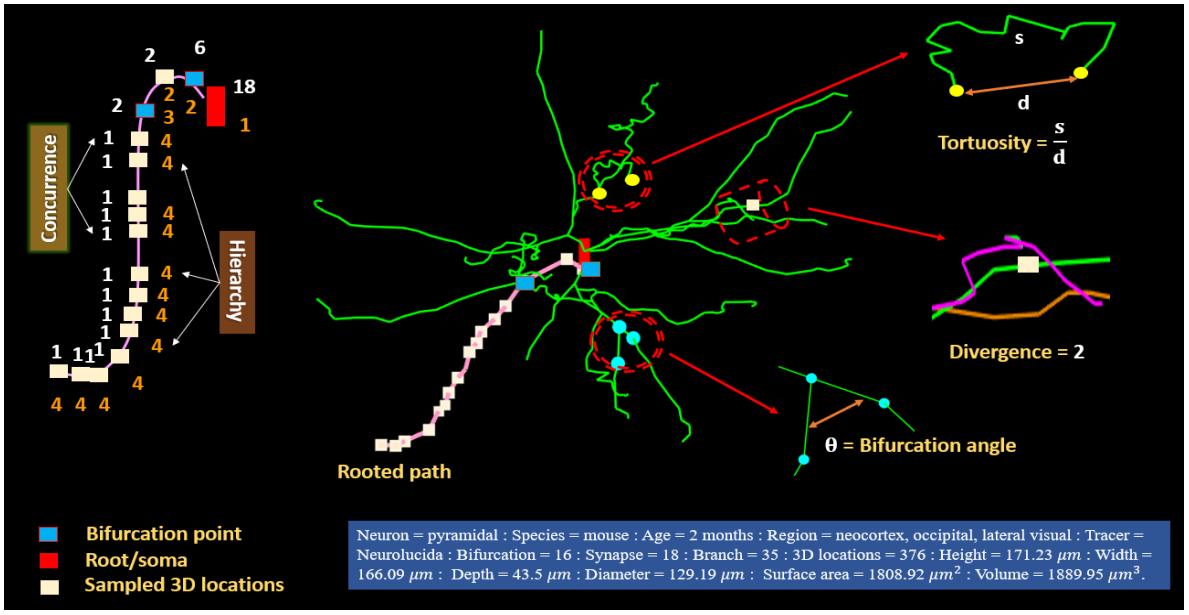


FIGURE 3.15: The figure shows a set of morphometrics that are used in our work. The metrics are numerically computed from a neocortical pyramidal cell with its relevant anatomical information provided at the bottom of the figure. A rooted path of the neuron (at the center) is shown (pink) with the 3D locations and bifurcation points, and the corresponding concurrence and hierarchy values are noted on the left hand side of the figure. 18 paths are originated from the soma, indicating 18 synaptic terminals. The immediate bifurcation point has a concurrence value of 6, because, including the current path, there are a total of 6 paths that end up in 6 different terminals. Concurrence values of the rest of the 3D locations are computed accordingly. It can be observed that there are four concurrence values that are marked on the chosen path, $\{1, 2, 6, 18\}$. The hierarchy values are obtained by sorting the indices of these four values in reverse order, which in turn describe the depths of locations with respect to the root. On the right hand side, three morphometrics on the selected paths are demonstrated, which are tortuosity, divergence, and bifurcation angle. Quantification of the wrinkle or tangle of a segment (tortuosity) existing between either two consecutive bifurcation points, or a bifurcation point and a following terminal is performed by measuring the curve length s, and the Euclidean distance between the start and end locations, d. Neocortical pyramidal cells show pronounced wrinkles in their branches. Divergence of a location entails competitive behavior. With a distance scale fixed beforehand, the number of branch segments that are in the immediate neighborhood of a location defines its divergence. Bifurcation angle is another important morphometric, which we measure by using the inner product between two vectors emanating from the bifurcation point. Wide bifurcation angles connote greater exploration of extracellular environment. More branches and smaller bifurcation angles, in general, lead to higher divergence. Neurons with higher divergence tend to have longer path lengths.

3.2.5 NeuroPath2Path

Key aspects of NeuroPath2Path

1) The inception of NeuroPath2Path comes from the realization of neuromorphogenesis and the self-similar phenotype of neuronal arbor. Since its birth from the soma, a path of a neuron has an *exploratory* attribute to collect external resources by the *minimal-length-maximal-routing* [34] strategy. Due to the parsimonious exploitation of intrinsic resources

(ion density, ATP and other electrophysiological items), the path, which fails to procure external resources, retracts. The exploratory attribute of a path can be expressed by the concurrence values at each sample point of a path. More paths imply more exploration. As the path matures, it has a *competitive* attribute [71, 89, 90] with respect to the other paths in its neighborhood in order to form a synapse. To account for competition, we count the number of paths in the proximity of each sampled location on a path and assign the count to that location.

2) The fractal dimension [60,91] of a neuronal arbor is considered one of the key morphometrics because the fan-out branches of a neuron bear self-similarity. In Path2Path and elasticPath2Path, the notion of matching the paths ignores this important feature. We extend the use of Munkres algorithm to perform one-to-one matching in a sequential fashion, which replicates the self-similar behavior.

3) As path features, we consider the bifurcation angle, partition asymmetry, and fragmentation score to each 3D location on a path. It is shown that the distribution of bifurcation angles in the basal dendrites of cortical pyramidal cells follows a Von Mises distribution [63]. An experimentally observed fact is that the mean bifurcation angle of branches ordered in a reversed fashion is discriminative for pyramidal cells in different cortical regions. However, the mean bifurcation angle of branches in standard order remains similar for the pyramidal cells. We take the standard ordering of branches, instead of the reverse order, to discriminate different neuronal cell types. Partition asymmetry [27, 60, 92] is another visually-significant morphometric. We use the *caulescence* measure as defined in [60] to account for the tree asymmetry.

4) We provide visualization of the continuous deformation between a pair of neurons and enumerate path similarity statistics to justify the correspondences between the paths. In contrast, conventional methods perform feature customization and extraction, and the classification, in supervised or unsupervised settings, depends on the abstract feature

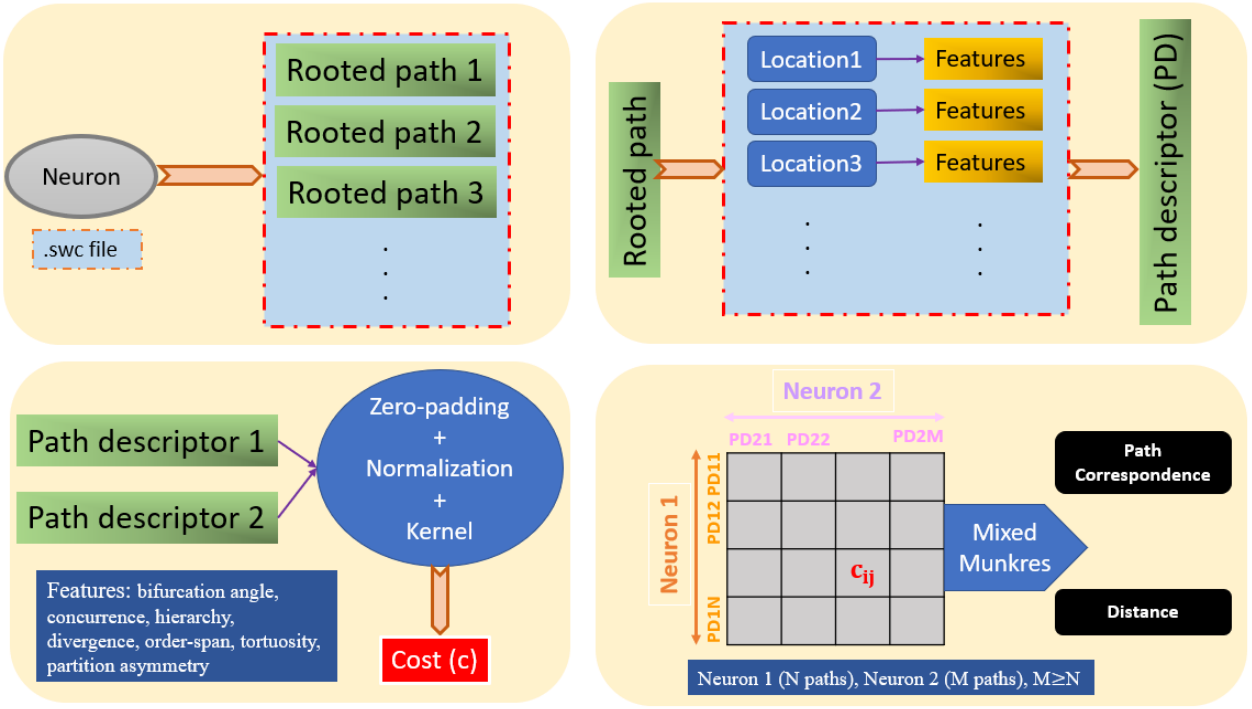


FIGURE 3.16: The figure presents a schematic representation of our proposed method and work flow. **[Top Left]:** An SWC [8] file encoding the 3D reconstruction of a neuron is read, and later the neuron is decomposed into an assembly of rooted paths. Each rooted path, spanning from the soma to a synaptic terminal, contains the 3D coordinates of each location traced on the path. **[Top Right]:** Each rooted path is subjected to feature extraction from each location on the path. The exhaustive list of the features that are used in our approach is given in the bottom left (blue box). We extract 7 features, implying that the path descriptor is a matrix of dimension (number-of-samples $\times$ 7).

space and the strength of the classifier. In those methods, the mapping between the space of 3D reconstructed neurons and the feature space is irreversible and abstract. Therefore, apart from the statistical quantification and analysis, it is ambiguous whether improved accuracy of the categorization stems from the trained classifier or the discriminating strength of the extracted features or both. In NeuroPath2Path, the classification problem is modeled as a variant of the transport problem. First, the correspondence of paths between a pair of neurons are decided in the feature space. Next, the correspondence is utilized to deform one neuron to the other. The distances computed between the paths and the deformation together justify the validity of the correspondence.

5) With suitable feature selection, NeuroP2P framework can be applied to perform morphological analysis of any cell type with ramified branching arbors, such as microglia and astrocytes. The continuum that is present in the evolution from one cell type to

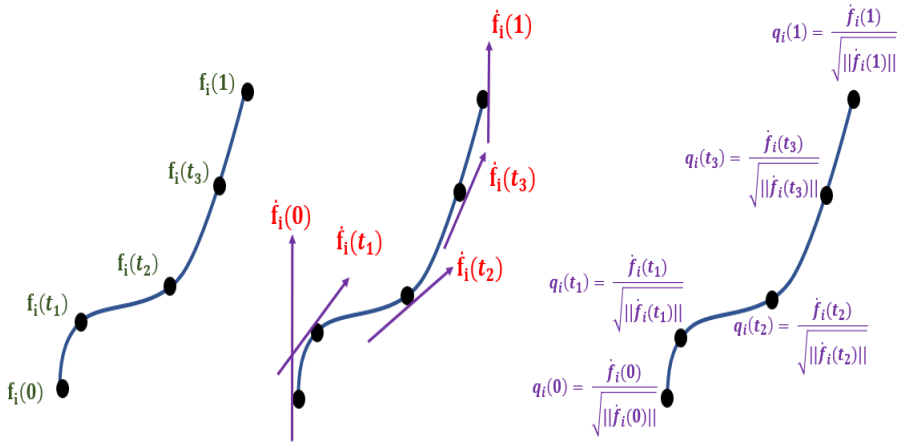


FIGURE 3.17: A schematic representation of the square root velocity function (q), which is computed at locations on an open curve. This function endows the curve with elasticity so that it can continuously deform (bend, stretch, shrink) to another curve. In a neuron, each rooted path can be modeled as an elastic open curve.

the other can be utilized in the analysis of cell differentiation. As an example, under certain constraints, the strategy of continuous morphing with branch splitting (explained later) can retrieve the intermediate states of a neuron cell while it evolves from a neural progenitor cell to its fully developed state. In short, NeuroP2P can serve as an effective tool for cell-specific informatics, which is not restricted to classification only.

3.2.5.1 Path modeling of a neuron

As mentioned in the introduction, a digitally-traced 3D sampled neuron can be modeled as a graph. Let the graph be represented by $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{W})$, where $\mathcal{V}$ is the number of 3D locations as vertices and $\mathcal{E}$ is the set of edges connecting the vertices with the corresponding weights $\mathcal{W}$ [23]. $\mathcal{G}$ is said to be simple if it does not contain multiple edges between any two vertices. A graph is called undirected if there is no preferred direction associated to an edge. A sequence of contiguous edges is called a path if no vertex and edge are repeated in that sequence. A path of length k has k number of edges or equivalently $(k+1)$ vertices. A sequence of contiguous edges is called a trail if no edge is repeated. If all the vertices except the start and the end of a trail are distinct, it is called a loop. A simple graph without a loop is termed as a tree. If the degree of each vertex is fixed, tree has the fastest growth by volume, hence smallest curvature [93]. A

graph is said to be single-connected if there exists at least one path between a pair of vertices. In case of a neuron, $\mathcal{G}$ is a *simple, undirected, weighted,* and *single-connected* tree.

A path can be considered as an open curve, $f_i(t),\ t \in [0,1]$, as defined in differential geometry. The cardinality of the set of vertices, or ,equivalently, the total number of 3D locations, is given by $|\mathcal{V}| = N$. Here, there are n dendritic terminals, which implies that the total number of paths rooted at the soma is n. Let Γ be the set containing all the paths $f_i,\ i \in \{1, 2, .., n\}$, which is a linear subspace of the classical Wiener space. Each path is sampled with the number of samples as ϕ with the sampled path denoted by $\tilde{f}_i$. We extract K features for each sample on $\tilde{f}_i$, which can be compactly given by the feature matrix for f_i as $\Theta_i \in \mathbb{R}^{\phi \times K}$. Let Θ be the ordered set containing the feature matrix for all the paths, $\Theta = \{\Theta_1, \Theta_2, ..., \Theta_n\}$, where Θ_i corresponds to the i^{th} path $\tilde{f}_i$. The *path model* of a neuron $\mathcal{G}$ can be mathematically represented as $H = \{\Gamma, \Theta, \mu\}$, where μ is a measure that we define in the next section. Note that we use the set of paths, Γ, as an ordered set which has a one-to-one correspondence with the elements in Θ. The standard branch order of a path, f_i, is defined as the order in which the locations of bifurcation on a path are visited from the root to the end of the path. Similarly, the reverse branch order is defined when the direction of traversal is reversed. For interclass comparison of neurons, we use the standard order. Whereas, for intraclass comparison, we follow the reverse order.

3.2.5.2 Proposed methodology

Our proposed method, which is sequential, scalable and modular, consists of four key stages as depicted in Fig. 3.16. In the first stage, centrally curated files of 3D-traced neurons in SWC format (or equivalent formats) are read and then preprocessed to extract only the dendritic arbors, including the soma. Several preprocessing modules, such as

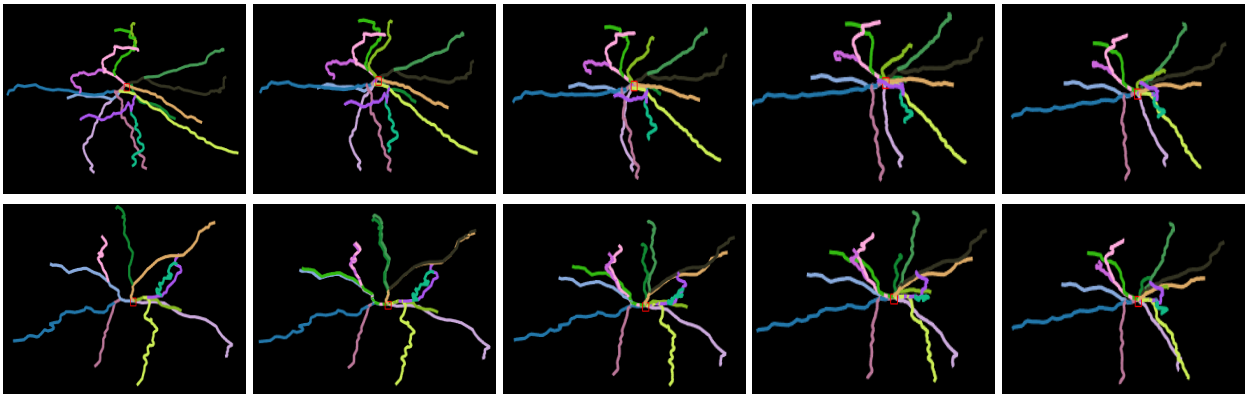


FIGURE 3.18: The figure depicts the evolution of 15 paths of one pyramidal neuron to 11 paths of another pyramidal neuron. Both the neurons are procured and curated from the neocortex (occipital, secondary visual and lateral visual) brain regions of 2 month-old mice. On the leftmost column, the top figure corresponds to the candidate neuron 1; the bottom figure is the target neuron 2. The evolution is represented in multiple arrays such that the ODD rows are read left-to-right and the EVEN rows are read right-to-left. The color associated with each path acts as a marker for the correspondence. At each intermediate step, the morphing of each path is calculated in the SRVF space [9] and then the path is projected back in the real 3D domain. In accordance with known properties of the SRVF, the SRVF takes care of the translation between paths. However for visual clarity, we intentionally allow the rotation of each path with respect to the root (soma) while the path advances towards merging with the target path. Prior to applying the SRVF, we reorganize the coordinates of each neuron in decreasing order of the ranges along the X, Y, and Z axes, implementing an in-place rotation of each neuron.

range-wise calibration, bifurcation location determination, and synaptic tip identification are employed to aid in preparing an assembly of rooted (soma) paths.

In the second stage, a set of features are extracted from each path, forming a feature descriptor of the path, $\Theta_i \forall i$. An exhaustive list of the features that are used in our method is provided in Section 3.2.5.2.1, and the systematic quantification of the features is provided in the Appendix. Notice that each path descriptor can be populated with additional structural and geometric features in order to perform fine-grained analysis.

The central aspect of the third stage is finding an appropriate cost function, as illustrated in Section 3.2.5.2.2. The cost function assimilates several anatomical features (such as segment length and bifurcation angle) and physiologically relevant factors (such as the competitive behavior, decaying anatomical importance of a path from the soma to synapse). A rigorous optimization framework is also formulated to find the relative contributions of such factors. In short, this stage delivers a distance measure between a pair of paths to the last stage.

With a distance measure between neuron paths in hand, this measure is augmented in the final stage as elucidated in section 3.2.5.2.3. We theoretically establish the correspondence of paths of a pair of neurons by repeatedly applying the Munkres algorithm. In contrast to the conventional approaches where the distance between neurons inherently accounts for *sub-graph* matching, we propose a full-tree matching algorithm. The repeated application of the Munkres algorithm reveals the fractal or self-similar nature of a pair of neurons. Equivalently, the following question may be posed: how many identical copies, taken together, of the first neuron can match with the second neuron, assuming the second neuron is much larger than the first one? Once the correspondence is found, neurons are diffeomorphically transformed to each other by morphing corresponding paths. This visual representation aids in justifying the correspondence of paths.

3.2.5.2.1 Feature extraction on a path We extract a set of discriminating features from each path $f_i \in \Gamma$ of H, which are bifurcation angle (b_i), concurrence (C_i), hierarchy (ξ_i), divergence (λ_i), segment length (β_i), tortuosity (κ_i), and partition asymmetry (α_i). Therefore, $\Theta_i = [b_i, C_i, \xi_i, \lambda_i, \beta_i, \kappa_i, \alpha_i] \in \mathbb{R}^{\phi \times 7}$. Each feature encodes a specific structural property of a neuronal arbor, as described in the Appendix. A schematic of different features along with the systematic quantification is shown in Fig.3.15.

3.2.5.2.2 Path alignment and path distance measure Given an unequal number of samples in a pair of paths, finding the appropriate distance between two paths or open curves is challenging. Due to the resampling bias imposed by a given tracer, in general, a path contains erroneous sampled locations which could alter the path statistics. For example, adding an extra leaf vertex changes the concurrence values of all the locations on a path. Unlike conventional approaches that used different resampling procedures, such as mid-point based resampling, RANSAC sampling, and spectral sampling, we use the help of the branch order as mentioned in section 3.2.5.1 for suboptimal alignment.

Consider two neurons, G_1 and G_2, with the corresponding path models given as H_1 and H_2, respectively. Let f and g be the two paths that are arbitrarily selected from Γ_1 and Γ_2, respectively. Without loss of generality, let us assume that f and g contain ϕ_1^b and ϕ_2^b, the number of locations from which the current paths bifurcate. In the case, where $\phi_1^b < \phi_2^b$, we append $(\phi_2^b - \phi_1^b)$ zeros at the end (standard branch order) or at the front (reverse branch order) of a feature vector on f.

Experimental evidence [63] suggests that the importance of a bifurcation location on a path decays as one travels the path from the soma to the synaptic end. We utilize this relative importance by way of hierarchy values of the bifurcation locations on a path. Let the sequential order of hierarchy values from the root to the terminal on f be $\xi_f = [\xi_1, \xi_2, ..., \xi_{\phi_1^b}]$. Using ξ_f, the k^{th} importance weight is given by $w_k = \frac{1}{\xi_k + \epsilon} / \sum_{j=1}^{\phi_1^b} \frac{1}{\xi_j + \epsilon}$. ϵ is introduced to avoid the indeterminate case. According to the hierarchy, it is obvious that $\xi_1 < \xi_2 < ... < \xi_{\phi_1^b}$. Thus, $w_1 > w_2 > ... > w_{\phi_1^b}$.

Let us consider a feature $\upsilon \in \{b, C, \lambda, \kappa, \beta, \alpha\}$. The values of the feature on the paths, f and g, are defined by

$$\begin{aligned} \upsilon^f &= [\upsilon_1^f, \upsilon_2^f, ..., \upsilon_{\phi_1^b}^f, \underbrace{0, .., 0}_{\phi_2^b - \phi_1^b}] \\ \upsilon^g &= [\upsilon_1^g, \upsilon_2^g, ..., \upsilon_{\phi_1^b}^g] \end{aligned} \tag{3.22}$$

The distance between υ^f and υ^g, weighted by the importance factor, is given by

$$d(\upsilon^{fg}) = \sqrt{\frac{1}{\phi_2^b} \sum_{k=1}^{\phi_2^b} w_k (\upsilon_k^f - \upsilon_k^g)^2} \tag{3.23}$$

This distance is computed for each $\upsilon \in \{b, C, \lambda, \beta, \kappa, \alpha\}$. The overall distance between the paths f and g can be expressed as a weighted average of individual distances.

$$\begin{aligned} \mu^{fg} &= \delta_1 d(b^{fg}) + \delta_2 d(C^{fg}) + \delta_3 d(\lambda^{fg}) \\ &\quad + \delta_4 d(\kappa^{fg}) + \delta_5 d(\beta^{fg}) + \delta_6 d(\alpha^{fg}). \end{aligned} \tag{3.24}$$

For simplicity, we take $\delta_i = \frac{1}{6} \forall i$ and consider the final distance as the intrinsic distance between the neurons. For classification, we determine δ through optimization using $maximizing - interclass - minimizing - intraclass$ distance strategy (See algorithm 2 in the Appendix). We term δ as the relative importance of features.

3.2.5.2.3 Path assignment and self-similarity Let the number of paths in H_1 be $|\Gamma_1| = n_1$. Similarly, for H_2, this value is $|\Gamma_2| = n_2$. Without loss of generality, let us assume $n_1 \leq n_2$. Using eq. 3.24, the cost matrix of paths between G_1 and G_2 becomes $\mathcal{D}$ ($\mathcal{D}_{ij} = \mu^{ij},\ i \in \Gamma_1, j \in \Gamma_2$). By applying an analogy for the path assignment as a job assignment problem with n_1 workers and n_2 jobs, we adopt the Munkres algorithm to find the optimal assignment of jobs to the workers from $\mathcal{D}$. In most cases, including inter- and intra-cellular neurons, the job assignment problem is an unbalanced $n_1 < n_2$. We append $(n_2 - n_1)$ zero rows to $\mathcal{D}$ to serve as dummy workers. ElasticPath2Path [86] employed this technique and resulted in an output of n_1 optimally matched paths between G_1 and G_2. However, this is essentially subgraph matching, which may lead to misclassification while dealing with two structurally similar, but different, cell types. For example, hippocampal CA3 pyramidal and cerebellar Purkinje cells have similar dendritic branch patterns, but significantly different number of paths. To resolve this problem we devise an algorithm, given in the Appendix, by applying Munkres algorithm repeatedly to obtain a full-tree matching. To meet such criterion, the algorithm gives n_2 pair of paths. Let the pair be $(\gamma_{11}, \gamma_{21}), ..., (\gamma_{1n_2}, \gamma_{2n_2})$, where $\gamma_{1i} \in \Gamma_1$ and $\gamma_{2j} \in \Gamma_2$. Recall that $n_1 < n_2$, which implies

that some of the γ_{1i} are repeated while forming the pair. Finally, the distance between G_1 an G_2 is given by

$$\chi_{G_1G_2} = \sum_{k=1}^{n_2} \mu^{\gamma_{1k}\gamma_{2k}} \tag{3.25}$$

Let $\lfloor \frac{n2}{n1} \rfloor = T$. Then, this procedure to find the correspondence is termed as $T-$regular matching, which in turn can be thought of T nearly self-similar structures akin to a fractal system. The detailed algorithm is provided in the Appendix. There are four modules that are sequentially executed in the algorithm. The first module mathematically deciphers the relatively self-similar anatomy of a larger neuron compared to a smaller one, yielding the number of copies of the smaller one needed to stitch together to approximately obtain the larger one. The routine runs for $\lfloor \frac{n_2}{n_1} \rfloor$ times, which indicates that each path in neuron 1 (containing n_1 paths), is matched with $\lfloor \frac{n_2}{n_1} \rfloor$ paths of neuron 2 (containing n_2 paths). Here $n_2 > n_1$.

The second module runs for the remaining unpaired paths of neuron 2. The assigned correspondence is added to the list of paired paths from the first module. However, not all the pairs are anatomically consistent. This is dictated by an internal constraint of Munkres algorithm, in which the assignment is carried out without replacement. In the Munkres algorithm, if one 'worker'(a path from neuron 1) is assigned a 'job' (a path from neuron 2), then the 'job' is not available for further assignment. Therefore, if the distance between two paths is significantly large, it demands further inspection whether the pair of paths is morphologically different to each other or the algorithmic constraint induces the large distance value. This motivates us to introduce the third module.

In the third module, we inspect the pair of paths having distances more than a threshold. The threshold is selected based on the skewness, median and standard deviation of the distance values. As mentioned earlier, in order to find the distance of a feature on two paths (eq. 3.23), we append zeros to the path having relatively fewer number of

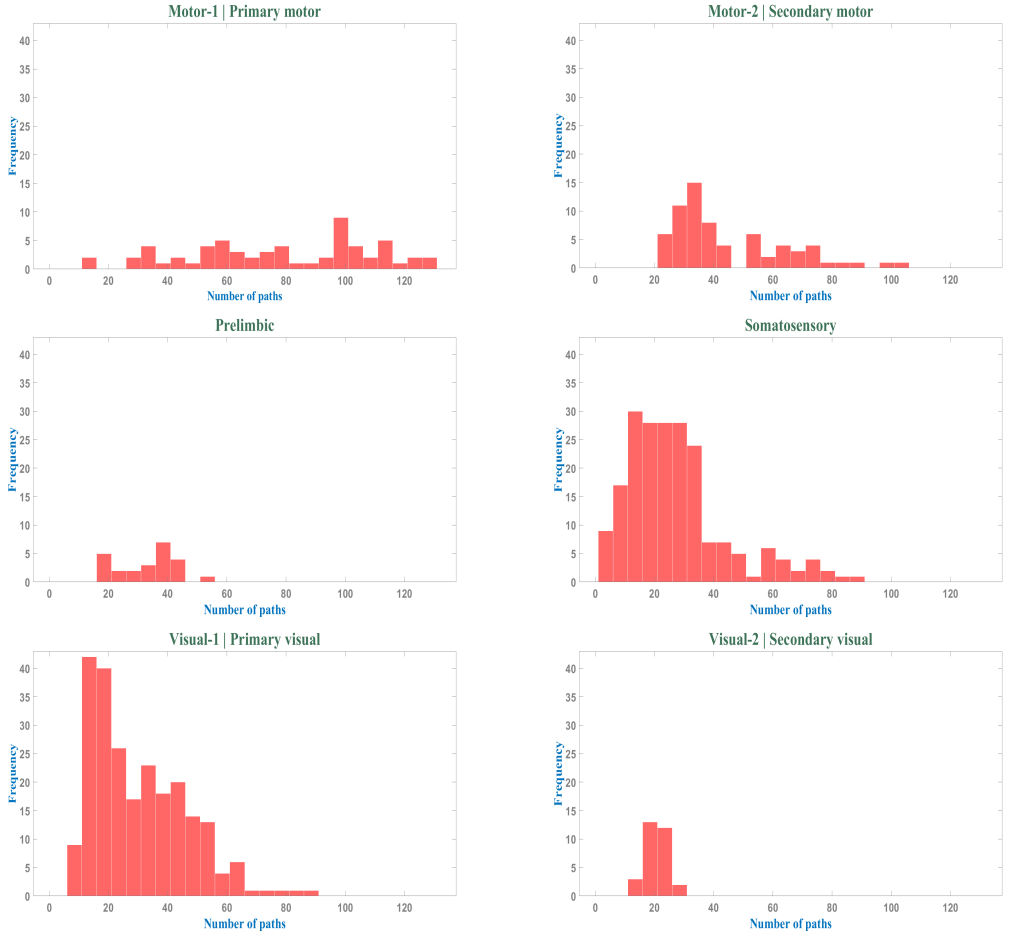


FIGURE 3.19: The figure shows the distribution of paths in each pyramidal cell category from Dataset 1. By glancing at the distribution profiles, a set of inferences can be drawn. The distribution of paths in primary motor is fairly uniform. For neurons from the somatosensory cortex and primary visual cortex, the histogram is right-skewed, indicating a majority of neurons with the number of paths lying in the range $[10, 40]$. The probability distribution of somatosensory pyramidal neurons resembles a right-skewed gamma distribution, and that of primary visual neurons closely follows an exponential distribution. The profiles of secondary visual and prelimbic neurons are poorly understood due to scarcity of samples. Most importantly, their distributions are entirely overlapped (within $[10, 40]$) in the region where the majority of primary visual and somatosensory neurons can be sampled. From the figure, it is evident that the number of paths alone is not sufficiently discriminatory.

locations than the other. The choice of traversal order dictates to which side the zeros are appended. Notice that more zeros lead to higher distance value between paths, and this happens only when there is significant mismatch in the highest level of hierarchy. This fact can be interpreted from the morphological viewpoint. A path with a large number of bifurcation locations (so, large maximum hierarchy value), called a central path of a neuron, exploits the environment of the neuron extensively when compared to path with

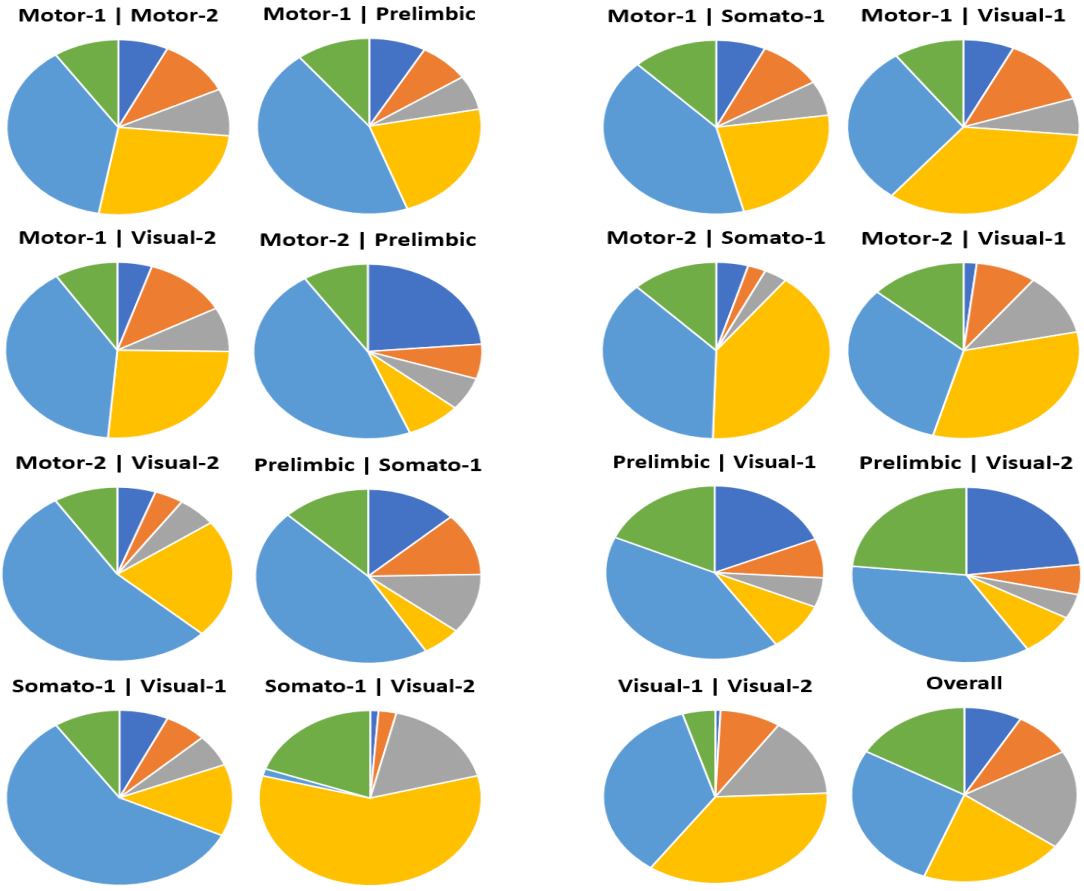


FIGURE 3.20: Relative importance δ for subsets of classes of Dataset 1. Each color corresponds to a specific feature and the area, subtended by the color in a pie chart, indicates its relative importance. By property, $\sum \delta = 1$, which implies a probability distribution. The color codes are as follows. **green**-divergence, **orange**-bifurcation angle, **gray**-partition asymmetry, **yellow**-concurrence, **deep blue**-tortuosity, **sky blue**-segment length. The pie charts taken together asserts a set of inferences. (1) The relative importance of features δ of all the classes (marked 'overall') somewhat follows a uniform distribution. (2) Segment length and concurrence are two predominant features when the pyramidal neurons from primary motor cortex (motor-1) are compared to the rest of the classes. (3) For the prelimbic class, divergence, tortuosity, and segment length appear to be most important. (4) δ for the somatosensory class toggles between two distributions with comparatively smaller and larger importance of concurrence.

fewer number of bifurcations. Unless otherwise required, a path with large hierarchy values should not be compared with a path with much smaller maximum hierarchy value. The highest level of hierarchy values of two paths are given by h_1 and h_2 with $h_1 < h_2$. We set a criteria that if $|h1 - h2| > \frac{max[h1,h2]}{2}$, we do not consider the distance between the pair, and opt for the best match in terms of minimum distance for each path of the pair separately. This is outlined in the reassignment module. The reassigned pairs are added to the list of paired paths serving as the list of correspondence.

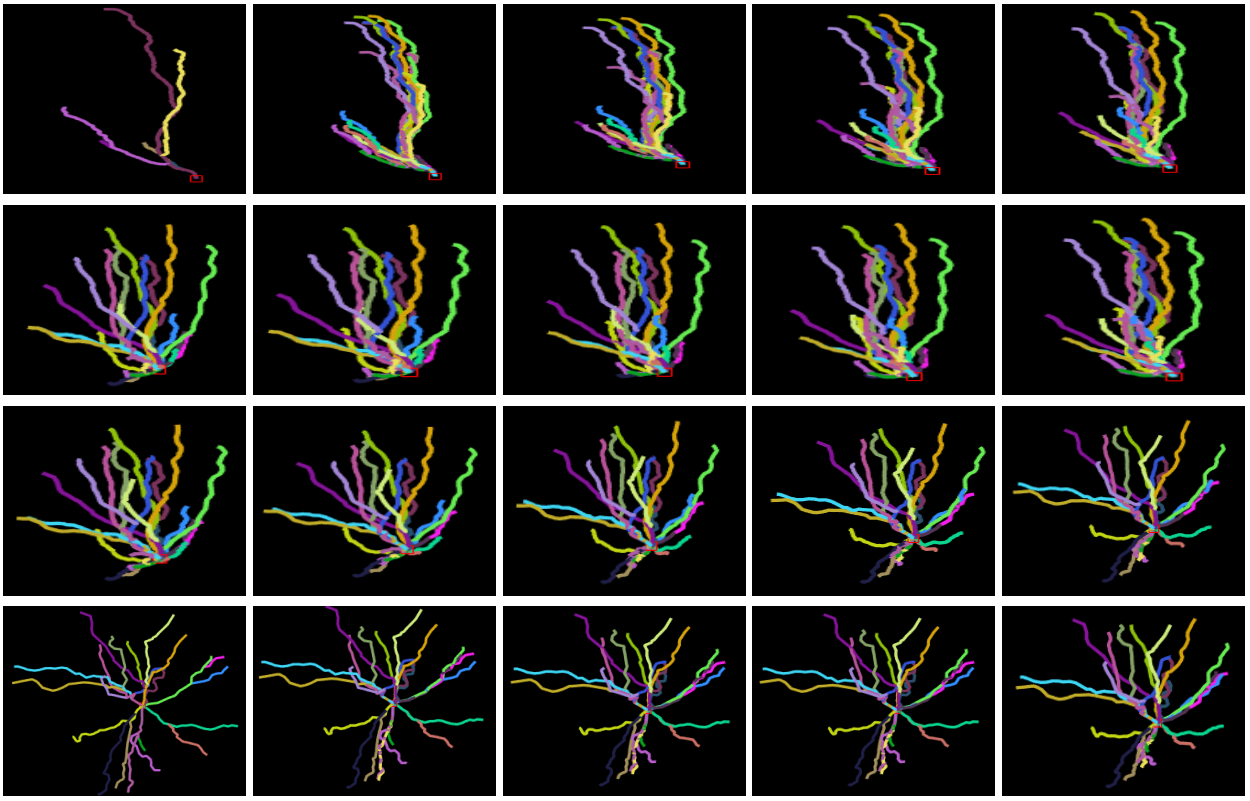


FIGURE 3.21: This gallery of images captures the progressive evolution of paths from a granule neuron to a pyramidal one. The granule neuron is procured from the hippocampus (dentate gyrus) of a 5 month-old mouse, containing 6 rooted paths. The pyramidal neuron is sampled from the neocortex (occipital lobe, secondary visual, lateral visual) of a 2 months old mouse, containing 22 paths. The evolution is represented in multiple arrays such that the ODD rows are read left-to-right and the EVEN rows are read right-to-left. In the first column, the top image is the granule cell and at the bottom is the pyramidal one. Structurally, the pyramidal neuron is larger than the granule one. However, they are properly scaled to fit for visualization.

3.2.5.2.4 Path morphing Once the correspondence of paths between neurons is established, it is imperative to know the structural similarity between the paths - *whether a pair of paths are structurally similar to each other, or the pair is structurally incoherent but the algorithm outputs such a pair due to its internal constraints.* This is achieved in two ways: with a visual representation by morphing the paths of one neuron to that of the other using an elastic framework, and by extracting path statistics. A rooted path of a neuron can be considered as an open curve as shown in Fig. 3.17 [9, 86]. Each location on the path can be considered as a function of a parameter,$t \in [0, 1]$. The square root velocity function (SRVF) that is applied on a location $f(t)$ is defined as $q(t) = \frac{\dot{f}(t)}{\sqrt{||\dot{f}(t)||}}$. For a pair of paths i and j, we obtain q_i and q_j, which assists in retrieving the intermediate

deformations as linear combinations of q_i and q_j given by $q_{ij}^n = q_i(1-n)+nq_j;\ n \in [0,1]$. n denotes the intermediate algorithmic time steps. Although the deformations are exhibited using the 3D coordinates of the locations of a path, the deformations can also be computed in the feature domain. An example of the continuous morphing process between two pyramidal neurons from the secondary visual cortex of the mouse is shown in Fig. 3.18. The 15 paths of the former neuron merge with 11 paths of the latter upon termination of the morphing process. This implies that more than one path of the first neuron have the same final destination path of the second neuron. It is noted that our algorithm does not consider the costs that are incurred by the merging or splitting of paths during progression. The assessment of such costs requires biophysical measurements of neurons, such as metabolic cost of merging or splitting of branches. Therefore, the cost between paths in eq. 3.24 is proportional to the cost of structural disparity instead of biophysical costs.

The prime question is: why do we need to inspect intermediate deformations? Statistical assessment of anatomical similarities between paths is *sufficient* to validate the correspondence that is obtained from the Munkres algorithm. However, to make the correspondence *necessary*, the intermediate deformations should comply with key cell-type characteristics [9]. So we use the SRVF framework to show the deformations so that any noticeable incoherence can be attributed to the feature selection, distance measurement, or both algorithms even though we might obtain improved classification accuracy in the end.

3.2.5.3 Datasets and results

We validate the approach on two datasets that are collected from a centrally curated on-line repository of 3D reconstructed neurons, Neuromorpho.org [75]. To demonstrate the strength of our approach, one dataset is compiled for intraclass and the other one for interclass analysis and comparison.

3.2.5.3.1 Dataset-1 (Intraclass) This dataset contains 3D-traced neurons from 6 distinct regions of the mouse neocortex. The regions with their cortical locations are visual-1 or primary visual (occipital), visual-2 or secondary visual (occipital), prelimbic (prefrontal), somato-1 or primary somatosensory (somatosensory), motor-1 or primary motor (frontal), and motor-2 or secondary motor (frontal).

We experiment with 62 neurons of motor-1, 68 neurons of motor-2, 24 neurons of prelimbic, 204 neurons of somato-1, 237 neurons of visual-1, and 30 visual-2 neurons with 625 neurons in total. The neurons vary widely in their morphological characteristics, such as the number of paths in each neuron. The histogram of paths corresponding to each category is shown in Fig. 3.19.

Next, we investigate the relative importance of each feature (mentioned in section 3.2.5.2.1) in terms of δ when comparing a set of classes. For space constraint, we provide δ values separately for each pair of classes and all the classes taken together. The relative importance is listed in Fig. 3.20 by a pool of pie charts. A set of class-specific inferences regarding the relative importance is enlisted in the figure description. The pie charts present a comprehensive view of feature strength. In practice, however, the values are required to report the distance between a pair of neurons. The values are reported in Table 3.17.

It is worthwhile to note that although there is significant variance in feature strength when all pairs are considered, the distribution approximates a uniform distribution when all classes are taken. This result endorses the selection of our features for all-class classification tasks. It is also important to mention that this framework can incorporate any set of path-specific features, not restricted to our selected features only.

In order to verify the consistency of the path correspondence obtained from Munkres algorithm (provided in section 3.2.5.2.3), we statistically evaluate each pair of paths

TABLE 3.17: Importance weight δ values for dataset-1. For space constraint, we provide feature-specific importance weight for classification in case of pairwise classes and all classes separately.

	Tortuo	Bifur-angle	Part-aym	Concur	Seg-len	Diverg
Motor1-Motor2	0.0729	0.1072	0.0876	0.2644	0.3794	0.0955
Motor1-Prelimbic	0.0820	0.0737	0.0631	0.2262	0.4483	0.1067
Motor1-Somato	0.0706	0.0929	0.0646	0.2328	0.4175	0.1218
Motor1-Visual1	0.0689	0.1230	0.0659	0.3317	0.2878	0.1227
Motor1-Visual2	0.0499	0.1212	0.0812	0.2612	0.3952	0.0912
Motor2-Prelimbic	0.2719	0.0737	0.0703	0.0899	0.5367	0.0484
Motor2-Somato	0.0452	0.0256	0.0328	0.4008	0.3730	0.1266
Motor2-Visual1	0.0167	0.0773	0.1041	0.2996	0.2969	0.2055
Motor2-Visual2	0.0545	0.0411	0.0564	0.2235	0.5489	0.0756
Prelimbic-Somato	0.1314	0.1168	0.1121	0.0568	0.4621	0.1187
Prelimbic-Visual1	0.2091	0.0821	0.0617	0.1012	0.4615	0.0795
Prelimbic-Visual2	0.2311	0.0551	0.0453	0.0790	0.3556	0.2339
Somato-Visual1	0.0693	0.0598	0.0584	0.1319	0.5816	0.0941
Somato-Visual2	0.0144	0.0310	0.2043	0.6963	0.0159	0.0382
Visual1-Visual2	0.0075	0.0880	0.1482	0.3551	0.3558	0.0473
Overall	0.0785	0.0807	0.1736	0.1956	0.2600	0.2076

in the correspondence list using pyramidal neurons from two different regions (the somatosensory cortex and secondary visual cortex.) The neuron from the somatosensory cortex (neuron-2) contains 28 rooted paths, while the other (neuron-1) has 11 rooted paths. Table 3.18 provides the exhaustive list of path correspondences, distances between corresponding paths, correspondences obtained by a competitive approach named ElasticPath2Path [86], and the best correspondences of the paths of neuron-2 with that of neuron-1. Notice that the best correspondence of a path f of neuron-2 is the path g of neuron-1, which yields minimum distance with f. Whereas, the Munkres algorithm works on the criteria where the sum of path distances (in our case 11 paths at a time) is minimized.

In Table 3.18, the two columns on the left enumerate the pair that consists of the path number of neuron-1 and that of neuron-2. A subset of paths of neuron-1 is repeated because neuron-2 (with 28) has more paths than neuron-1 (with 11). So from neuron-1 to neuron-2, the correspondence is a surjective mapping. This is in contrast with ElasticPath2Path, where the mapping is bijective and, as result of that, the algorithm

outputs only 11 pairs in the correspondence list. The rest of the $28 - 11 = 17$ paths are left unmatched, yielding a solution of the subgraph matching problem. The unmatched paths are marked with 'NA' in the fourth column.

The last column, tagged as the best match, identifies only $\{4, 5, 8, 9\}$ path indices out of 11 paths of neuron-1. Nevertheless, this best matching algorithm also elicits a potential solution for subgraph matching. There are certain extreme cases where all paths of one neuron are matched with only one path of the other neuron, posing *degenerate* solutions of the neuron matching problem. We mark the correspondences in yellow, where the results of our algorithm and best match coincide.

Recall that neuron-1 has 11 paths and neuron-2 contains 28 paths. Careful observation of the first column of the table suggests that the set of numbers $\{1, 2, ..., 11\}$ is repeated twice in the serial order followed by 6 path indices which are $\{9, 4, 10, 8, 2, 5\}$. Here, Munkres algorithm is applied thrice. Each time Munkres algorithm outputs 11 pairs of paths for correspondence. Therefore, the first two passes encompass $11 * 2 = 22$ pairs leaving $28 - 22 = 6$ paths of neuron-2 unassigned. Before applying the third pass, the cost matrix $\mathcal{D}$ is cropped with a dimension $\mathcal{R}^{11\times6}$. The cropped cost matrix is then transposed ($\mathcal{R}^{6\times11}$), zero-appended ($\mathcal{R}^{11\times11}$) and subjected to Munkres. The above observation also indicates that 2 self-similar copies of neuron-1 approximates neuron-2 in the sense of minimum path to path distance. Therefore, the relative fractal index of neuron-2 with respect to neuron-1 is $2\frac{6}{11}$ or 2.545.

The question is: can the arithmetic average (which is 0.92) of the 'Distance' column of Table 3.18 be regarded as the final distance between the neuron-1 and neuron-2? Unfortunately, it is not. The reason is explained in section 3.2.5.2.3 and reiterated briefly in the following sentences. After computing the correspondences (column-1 and column-2), we identify the defective sets of pairs for which there are significant differences in the hierarchy levels. The larger the difference, the larger the number of zeros that are

appended to each feature on the path, raising the chances of technical error in the final distance value. In the table, the defective pairs are emboldened with blue color. We delete these pairs and replace the correspondences of path indices 18 and 9 (neuron-2) with their best matches from neuron-1. Note that path indices 7 and 4 of neuron-1 have already been matched with other paths of neuron-2, which are $7 - -28$, $4 - -11$, and $4 - -12$. Therefore, those paths are not subjected to re-assignment. After inserting the best matches for the path indices 18 and 9 (which are 9 and 5 from neuron-1 respectively), the corresponding distance values are noted. This is described in the fourth routine, 'Reassignment' of algorithm 2. The final distance between the two neurons turns out as 0.90 (rounded off). The competitive approach, ElasticP2P produces a distance value of 0.67, which implies that the two neurons are more similar. This is discordant with the fact that the two neurons are sampled from two different regions and have two distinct arbor types. This disagreement can be explained due to subgraph matching nature of ElasticP2P. Neuron-1 with 11 paths is well-matched with a part of neuron-2. However, the rest 17 paths of neuron-2 are structurally dissimilar with neuron-1. In this case, our method, NeuroPath2Path performs significantly better in distinguishing two neurons in terms of distance owing to its full-graph matching property.

For classification, we compute the importance values δ from (3.24) and show them in Table 3.17. The importance values are applied to compute the distance between a pair of neurons. Using our distance function, we resort to the K nearest neighborhood classifier. We randomly partition the dataset into our training and test set using a constant ratio and rerun the experiment 5 times. The ratio that we maintain is 0.1 and 0.2 as train and test datasets. As the number of paths that a neuron has is a distinguishable feature for certain classes, we devise a strategy to test each neuron from the testing dataset. For a neuron with number of paths as n_P, we seek candidate neurons from the training set with the number of paths ranging in $[n_P - L, n_P + L]$. Overall, NeuroPath2Path contains

TABLE 3.18: Distance and correspondence between paths. The correspondences between the paths of neuron-1 and neuron-2 are enlisted in the first two columns. The numbers in yellow indicate that the correspondence obtained by NeuroP2P matches with the candidates of best correspondence in the sense of minimum distance. The pairs in blue are subjected for further verification because of large differences in the hierarchy values (Routine 4 in Algorithm 2).

Index (Neuron-1)	Index (Neuron-2)	Distance	ElasticP2P [86]	Best match
1	25	1.0595	1	9
2	2	0.8031	2	5
3	27	0.4530	3	5
4	12	0.7910	4	4
5	3	0.5571	5	5
6	26	0.6165	6	5
7	28	0.5690	7	5
8	5	0.5606	8	8
9	23	0.6271	9	9
10	13	0.7904	10	4
11	24	0.6096	11	5
1	7	1.4997	NA	8
2	1	0.9902	NA	9
3	15	1.1052	NA	5
4	11	0.9223	NA	9
5	6	0.5932	NA	5
6	17	1.7108	NA	9
7	**18**	1.2869	NA	9
8	20	0.8704	NA	9
9	4	0.7708	NA	5
10	14	0.9186	NA	5
11	10	1.1796	NA	9
9	8	0.8504	NA	5
4	**9**	1.2279	NA	5
10	16	1.1810	NA	5
8	19	1.1658	NA	9
2	21	1.1522	NA	5
5	22	0.8931	NA	5

two hyperparameters, K (number of nearest neighbors) and L. THis step is followed by the identical testing procedure while considering the interclass dataset. We fixed $L = 50$ for our experiments. As noted before, we adopt the reverse and standard branch orders for dataset-1 and dataset-2, respectively.

With the ratio of train and test as $8 : 2$, the confusion matrix of NeuroPath2Path for an instance of random partition of data is shown in Fig. 3.26. It can be seen that,

while NeuroPath2Path distinguishes Motor-1, Visual-1, and Visual-2 quite well, the class of Somato-1 is significantly misclassified with Motor-1, Motor-2, and Visual-1, leading to a decline in the classification score.

	Motor-1	Motor-2	Prelimbic	Somato-1	Visual-1	Visual-2
Motor-1	11	3	0	1	0	0
Motor-2	3	6	0	2	1	0
Prelimbic	0	1	3	2	0	0
Somato-1	3	3	0	15	5	0
Visual-1	3	3	0	1	22	1
Visual-2	0	0	0	1	0	4

FIGURE 3.22: Confusion matrix of an instance of classification using dataset-1. The overall accuracy is 66%. Here, we set $L = 50$ and $K = 3$.

Next, in Fig. 3.23, we illustrate the comparative performance of NeuroPath2Path against TMD and NeuroSoL. The train to test ratio is set at $8 : 2$. NeuroSoL shows an erratic behavior as K increases. TMD offers a consistent margin of classification accuracy per K. Here, at a given value of K, margin implies the difference between the maximum and minimum scores of 5 experiments which are independently instantiated by randomly partitioning the dataset with $8 : 2$ train:test ratio. Fig. 3.23 suggests that NeuroPath2Path achieves peak performance when K is set as 7, but with a noticeable margin. To scrutinize the performance of TMD and NeuroPath2Path, we routinely

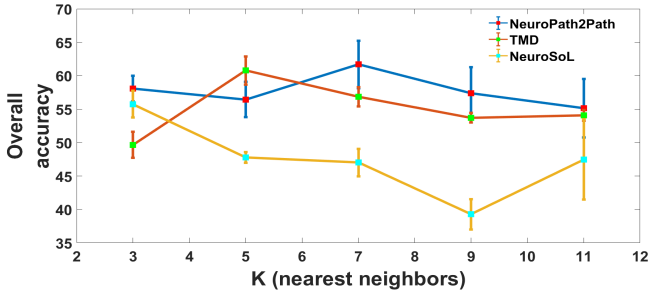


FIGURE 3.23: The figure shows comparative performance of NeuroPath2Path against TMD and NeuroSoL using different values of K in K-NN classifier. At each K, we perform 5 experiments for each of these methods and the associated scores are shown with the mean (colored square) and associated range of values.

inspect the class-wise retrieval accuracy, a crucial metric which is obscured in Fig. 3.23 due to the averaging effect. The result is shown in Fig. 3.24. In a majority of cases, despite

	Motor-1	Motor-2	Prelimbic	Somato-1	Visual-1	Visual-2
NeuroPath2Path	80%	50%	33%	50%	73.33%	80%
TMD+K-NN	58.33%	76.9%	0%	62.5%	63.8%	100%

FIGURE 3.24: This figure shows one typical instance of classwise retrieval accuracy of NeuroPath2Path and TMD. NeuroPath2Path maintains better classwise performance than TMD. We use 62 neurons of motor-1, 68 neurons of motor-2, 24 neurons of prelimbic, 204 neurons of somato-1, 237 neurons of visual-1, and 30 visual-2 neurons. It is evident that TMD is adversely affected by class imbalance.

comparable overall classification scores, TMD tends to be affected by class imbalance, leading to significantly poor accuracy for few classes.

3.2.5.3.2 Dataset-2 (Interclass) The second dataset consists of 3D-reconstructed neurons that are traced from five major cell types of the mouse: ganglion, granule, motor, Purkinje, and pyramidal. We experiment with an imbalanced pool of 500 ganglion cells, 490 granule cells, 95 motor cells, 208 purkinje cells, and 499 pyramidal cells, where the corresponding SWC files are obtained from the neuromorpho repository. The cell-specific distribution of paths is shown in Fig 3.25.

For classification, we compute the important weights δ of each features, and due to space constraints, the δ values are enumerated in Table 3.19 for pairwise classes and the case with all the classes taken together. The importance-weighted distance value, μ^{fg} in (3.24) is used to compute the distance of a pair of neurons. We empirically find that the nonlinear transformation of μ^{fg}, given by $\frac{1}{1+exp(-\mu^{fg})}$, yields an improved classification performance.

With a train:test ratio as $8:2$, one instance of the confusion matrix, obtained by NeuroPath2Path is provided in Fig.3.26. We demonstrate the effectiveness of NeuroPath2Path over two state-of-the-art approaches - Topological Morphological Descriptor (TMD) [69] and NeuroSoL [79]. For each value of K, we randomly partition the dataset 5 times maintaining a constant $9:1$ ratio between the train and test datasets. In short, for every K, we obtain 5 accuracy scores, which are plotted in Fig. 3.27.

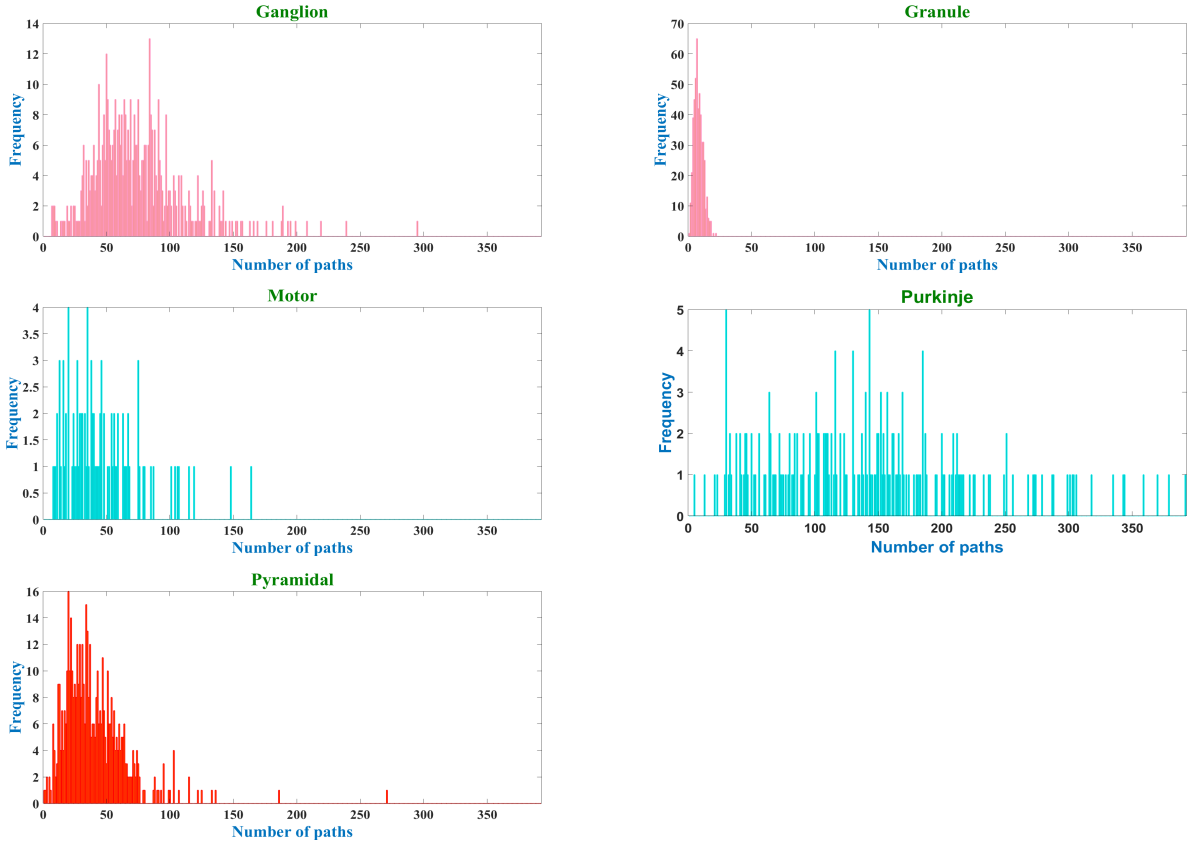


FIGURE 3.25: The figure shows the cell-specific distribution of the number of paths. It is observed that the distribution of paths in the case of Purkinje cells is approximately uniform. The remainder of the distributions are left-skewed.

TABLE 3.19: Importance weight δ values for dataset-2

	Tortuo	Bifur	Part-aym	Concur	Seg-len	Diverg
Ganglion-Granule	0.0278	0.0138	0.0241	0.2102	0.6727	0.0533
Ganglion-Motor	0.0464	0.0797	0.0918	0.1865	0.4730	0.1226
Ganglion-Purkinje	0.27	0.032	0.1437	0.1905	0.2270	0.1368
Ganglion-Pyramidal	0.0433	0.0001	0.0553	0.5121	0.3156	0.0809
Granule-Motor	0.0356	0.0491	0.0372	0.1449	0.6697	0.0636
Granule-Purkinje	0.0147	0.1218	0.1875	0.1073	0.5285	0.0402
Granule-Pyramidal	0.0453	0.0305	0.0372	0.2231	0.5390	0.1248
Motor-Purkinje	0.0094	0.4046	0.0556	0.0007	0.5167	0.0130
Motor-Pyramidal	0.0223	0.0737	0.0465	0.1275	0.6650	0.0650
Purkinje-Pyramidal	0.0219	0.2201	0.1088	0.1528	0.4223	0.0740
Overall	0.1304	0.0372	0.0417	0.1786	0.4489	0.1633

TMD appears to be very consistent in accuracy and range scores, achieving an accuracy of 85.02% when $K = 5$. However, while computing the confusion matrices of the classification scores obtained by TMD, we notice that in the majority of instances, the correct classification of motor cells is abnormally low and approaches 0% in some

	Ganglion	Granule	Motor	Purkinje	Pyramidal
Ganglion	43	1	1	2	10
Granule	0	50	0	0	0
Motor	2	0	10	0	1
Purkinje	1	2	1	25	4
Pyramidal	1	4	1	0	48

FIGURE 3.26: Confusion matrix of an instance of classification using Dataset-2. The overall accuracy is 85.02%. Here, we set $L = 50$ and $K = 9$. It can be seen from the matrix that one-fifth of ganglion cells are misclassified as pyramidal, leading to a decline in accuracy. However, granule cells are perfectly classified.

cases. It is important to notice that Dataset-2 has an imbalance in terms of the number of examples in each cell category, with motor cells containing the lowest (95) and ganglion cells containing the highest (500) number of examples. This fact is unobserved in Fig. 3.27 due to the averaging effect. We adopt the metric, class-wise accuracy of retrieval, and present the results in Fig. 3.28. It is evident that NeuroPath2Path exhibits strong resilience against the class imbalance problem.

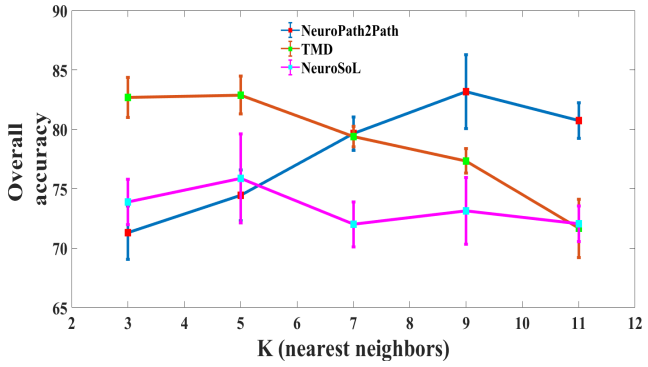


FIGURE 3.27: The figure shows comparative performance of NeuroPath2Path against TMD and NeuroSoL using different values of K in K-NN classifier. At each K, we perform 5 experiments for each of these methods and the associated scores are shown with the mean (colored square) and the range values. The profiles of TMD and NeuroPath2Path surprisingly appear to have opposite trends over K. NeuroPathPath hits the top accuracy of 86.2% when $K = 9$.

	Ganglion	Granule	Motor	Purkinje	Pyramidal
NeuroPath2Path	75.4%	100%	77%	75.7%	89%
TMD+K-NN	83%	95.9%	5%	87.8%	66%
TMD+SVM	85%	97%	15.7%	95%	78%

FIGURE 3.28: This figure shown the classwise retrieval accuracy of different methods including NeuroPath2Path. It is observed that by using TMD the retrieval accuracy of Motor cells shows minimal improvement when SVM is used. We use 500 ganglion cells, 490 granule cells, 95 motor cells, 208 purkinje cells, and 499 pyramidal cells. The imbalance in class adversely affects the classification accuracy. NeuroPath2Path maintains consistent class performance.

Similar to the train and test ratio of Dataset-1, we conduct experiments using the ratio of 0.1 and 0.2 separately. The classification scores are given in Fig. 3.29.

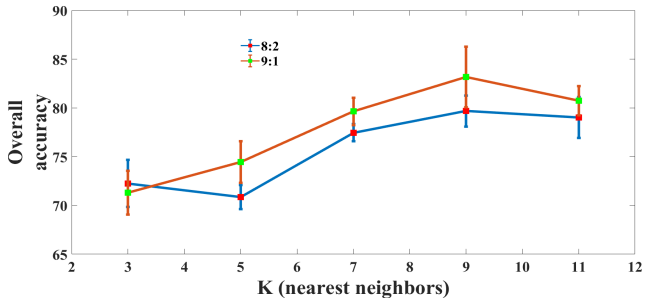


FIGURE 3.29: The performance of NeuroPath2Path on two different partitions, which are $9:1$ and $8:2$, of Dataset-2 is shown. $K = 9$ is found to be a suitable candidate of K-NN classifier.

Appendix

Description of features We extract a set of discriminating features on each path $f_i \in \Gamma$ of H, which are bifurcation angle (b_i), concurrence (C_i), hierarchy (ξ_i), divergence (λ_i), segment length (β_i), tortuosity (κ_i), and partition asymmetry (α_i). • *Bifurcation angle* is a key morphometric that dictates the span and the spatial volume of an arbor. It is hypothesized that the span of an arbor at each level of bifurcation depends on the bifurcation of its previous level [7,63,71], suggesting the influence of Bayesian philosophy. This organizational principle is utilized in several stochastic generative models [71] for the synthesis of specific neuron cell types. The sequence of bifurcation angles at bifurcation vertices located on a path of a neuron captures local geometry. For example, a sequence of non-increasing bifurcation angles from the root to the dendritic terminal of a path indicates the pyramidal shape geometry of the neuron. For a location with multifurcation, we use the maximum of the bifurcation angles computed using pairwise branches originated from that location towards the dendritic terminals.

• *Concurrence, hierarchy and divergence* encode the effect of phenomenological factors, which are exploration (ex. Purkinje fanning out rostrocaudally) and competition (ex. retinal ganglion cells), that contribute in the growth of a neuron. The definition of

concurrence and hierarchy are already given in section 3.2.3. The divergence of a location on a path, f_i is proportional to the repulsive force that the location experiences from its neighborhood path segments. Let C_{f_i} be the sequence of concurrence values of the path $f_i \in \Gamma$ when one visits the locations from the root to the dendritic terminal. As an open curve, each path can be parameterized with the parameter $t \in [0, 1]$. $C_{f_i}(t_s) = k;\ t_s \in [0, 1]$ indicates that $k(\leq |\Gamma|)$ paths share the location t_s on f_i. The divergence λ of a location $f_i(t_s)$ is defined as $\lambda(f_i(t_s)) = 1_{\{f_j \mid |f_j(t)-f_i(t_s)|\leq\epsilon, f_j\neq f_i, f_j\not\succ f_i\}}$. Here, 1 is the indicator function computing the number of such f_js which follow the conditions $|f_j(t) - f_i(t_s)| \leq \delta, f_j \neq f_i$ and $f_j \not\succ f_i$. The first condition implies that a location of f_j has to be in the ϵ neighborhood of f_i. $f_j \not\succ f_i$ indicates that the location of bifurcation at which f_j deviates from f_i does not appear after $f_i(t_s)$ on the path f_i.

• *Tortuosity and partition asymmetry* are two important anatomical features of a neuron. Tortuosity refers to the amount of 'zig-zag' or bending of a path. Let us take a segment on a path f_i as $f_i([t_1, t_2]);\ 0 \leq t_1 < t_2 \leq 1$. Let there be $m - 1$ locations in $[t_1, t_2]$. The tortuosity of the segment is defined as $\kappa = \frac{\sum_{j=1}^{m} ||f_i(t_{j+1})-f_i(t_j)||_2}{||f_i(t_2)-f_i(t_1)||_2}$ with $t_{m+1} = t_2$. Partition asymmetry accounts for how the size of a neuron tree varies within the neuron. We use a variant of caulescence, proposed in [60], as a measure of tree asymmetry. Caulescence at a bifurcation location is evaluated by way of $\alpha = \frac{|l-r|}{l+r}$, where l is the size of the left tree and r of the right tree of the bifurcation vertex. We define the size of a tree by the number of paths or equivalently the number of dendritic terminals. Note that the quantity $(l + r) + 1$ is the concurrence value of the bifurcation vertex.

Weight determination Let the combined distance vector containing the individual feature distances be $D^{fg} = [d(b^{fg})\ d(C^{fg})\ d(\lambda^{fg})\ \ d(\kappa^{fg})\ d(\beta^{fg})\ d(\alpha^{fg})]^T$. The corresponding unknown weight vector is $\delta = [\delta_1, ...\delta_6]$. While comparing two neurons of sizes N and M with $N <= M$, the distance computation after applying the Munkres algorithm repeatedly will produce M pairs of paths, indicating M such D^{fg}s. The desired

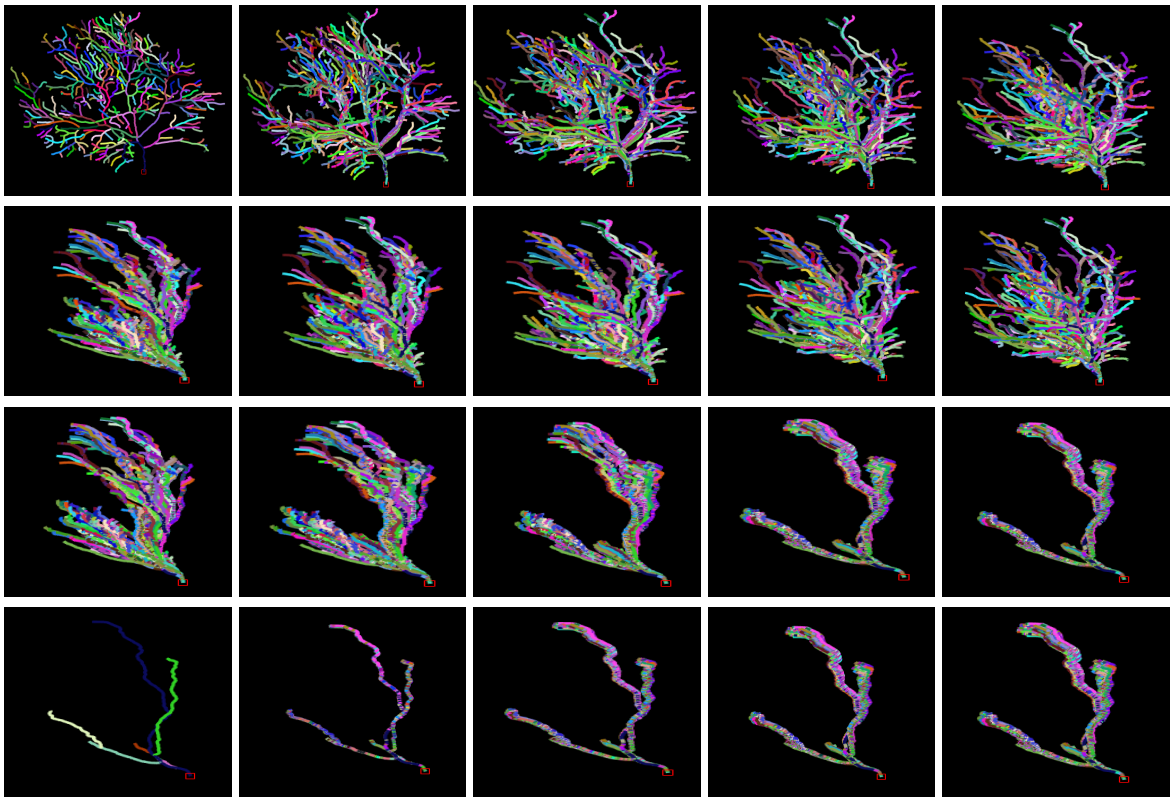


FIGURE 3.30: This gallery of images captures the progressive evolution of paths from a Purkinje neuron to a granule one. The granule neuron ($426.5\mu m^3$ volume) is procured from the hippocampus (dentate gyrus) of a 5 months old mouse, containing 6 rooted paths. The Purkinje cell ($13094\mu m^3$ volume) is sampled from the cerebellar cortex of a 35 day-old mouse, containing 304 paths. The evolution is represented in multiple arrays such that the ODD rows are read in the left-to-right and the EVEN rows are read in the right-to-left fashion. In the first column, the top image is the granule cell and the bottom shows the pyramidal cell. Volume-wise, the Purkinje neuron is significantly larger than the granule neuron. However, they are scaled for visualization.

characteristic of each component of δ is positivity. In addition, we enforce $\sum \delta_i = 1$, implying a probability estimate. δ_i thus indicates the relative importance of the feature υ_i.

We adopt the constrained *maximizing-interclass -minimizing-intraclass* distances strategy to find our desired δ. Mathematically,

$$\begin{aligned}\delta^{opt} &= -\arg\min_{\delta} \frac{1}{2}\delta^T\Big(\sum_{\substack{i,j=1\\i<j}}^{S}\sum_{k=1}^{N_i}\sum_{l=1}^{N_j}\sum_{z=1}^{M_{kl}} D^z(D^z)^T\Big)\delta \\ &\quad +\delta^T\Big(\sum_{i=1}^{S}\tau_i\sum_{\substack{k,l=1\\k\neq l}}^{N_i}\sum_{z}^{M_{kl}} D^z(D^z)^T\Big)\delta \\ &\quad -\omega_1 log\delta + \omega_2\big(\sum_{i=1}^{6}\delta_i - 1\big) \end{aligned} \tag{3.26}$$

The first term in the above equation encompasses all the distances between neurons from pairwise classes. The second term encodes the intraclass distances, implying the distances between neurons for each class. The third term enforces positivity of each weight δ_i. This is a logarithmic barrier penalty term that restricts the evolution of δ at intermediate iterations to the region where $\delta > \bar{0}$. The last term accounts for the probabilistic interpretation of δ. S is the number of classes.

Eq. 3.26 is solved by using gradient descent with fixed $\eta = 0.01$. The equation and its derivative can be simply written as,

$$\begin{aligned} L(\delta) &= -\frac{\delta^T\Pi\delta}{2} + \delta^T\sum_{i=1}^{S}\frac{\tau_i\Pi_i}{2}\delta - \omega_1 log\delta + \omega_2\big(\delta\mathbf{1} - 1\big) \\ \frac{dL}{d\delta} &= -\Pi\delta + \sum_{i=1}^{S}\tau_i\Pi_i\delta - \frac{\omega_1}{\delta} + \omega_2 \end{aligned} \tag{3.27}$$

We use this derivative term in the following algorithm 1 to obtain optimal δ.

Distance between neurons The algorithm to find distance between a pair of neurons consists of four stages - finding self-similarity (routine-1), remaining path assignment (routine-2), finding pairs with hierarchy mismatch (routine-3) and reassignment of the defective pairs (routine-4).

Algorithm 1: Find δ

Data: $\Pi, \Pi_1, \Pi_2, ..., \Pi_S$ (for all classes);
Initialization: $\delta_{cur}, \delta_{tmp}, \tau_1, \tau_2, ..., \tau_S, \omega_1, \omega_2$, Iter, tol, η;
while $iter < Iter$ **do**
  while $||\delta_{cur} - \delta_{tmp}||_2 < tol$ **do**
    $D \longleftarrow \sum_{k=1}^{S} \tau_k \Pi_k$;
    $Der \longleftarrow -\Pi\delta + D\delta - \frac{\omega_1}{\delta} + \omega_2$;
    $\delta_{curr} \longleftarrow \delta_{tmp}$;
    $\delta_{curr} \longleftarrow \delta_{curr}/(\delta_{curr}\mathbf{1})$;
    $\delta_{tmp} \longleftarrow \delta_{tmp} - \eta Der$;
  $iter \longleftarrow iter + 1$;
  $\omega_1 \longleftarrow \omega_1/2$;
  $\omega_2 \longleftarrow 2\omega_2$;
  $\tau_i \longleftarrow 1.1\tau_i \forall i$(more intraclass compaction);

Discussion:

NeuroPath2Path follows a graph-theoretic approach that utilizes path-based modeling of neuron anatomy and provides a visualization tool by way of a geometric model that aids in performing continuous deformation between two neurons. NeuroPath2Path offers several advantages. The decomposition of a neuron into paths can be viewed as an assembly of individual circuits from the terminals to the soma, integrating semi-local features that act as path descriptors. Next, instead of subgraph matching, NeuroPath2Path does not leave a single path unassigned, culminating in a full-graph matching algorithm. The matching algorithm presents the notion of relative fractality and path correspondences, and incorporates physiological factors, such as decaying importance of features along the path, exploratory and competitive behavior for resource exploitation.

NeuroPath2Path also precisely investigates the feasibility of algorithmic constraints (such as on Munkres algorithm) on the structural repertoire of neuronal arbors, and thereby enforcing criteria, such as hierarchy mismatch. During classification, NeuroPath2Path delivers resilience to the class imbalance problem. In the future, in order to explore the full potential of the approach beyond classification, we aim to augment NeuroPath2Path in two major domains - morphological analysis and structural transformation of microglia

Algorithm 2: Find χ

Data: $\mathcal{D}, n_1, n_2$ $(n_1 \leq n_2)$, Inf;
Output: pairList, dstSum
Initialization: $dstSum \longleftarrow 0$, $pairTmp \longleftarrow [1:n_2]^T$, $count \longleftarrow 1$, $pairList \longleftarrow [\,]$, $\mathcal{D}_c \longleftarrow \mathcal{D}$, $dstB \longleftarrow [\,]$;
Routine 1: (Self-similarity)
while $count \leq \lfloor \frac{n2}{n1} \rfloor$ **do**
  $\hat{\mathcal{D}} \longleftarrow \mathcal{D}$ (append rows with Inf);
  $pairL, dst \longleftarrow Munkres(\hat{\mathcal{D}})$;
  $dstSum \longleftarrow dstSum + dst$;
  $dstB \longleftarrow dstB \cup dst$;
  $pairTmp \longleftarrow pairTmp \setminus pairL[1:n_1]$ $((n_2 - n_1)$ dummy);
  $pairList \longleftarrow pairList \cup pairTmp[pairL[1:n_1]]$;
  $\mathcal{D} \longleftarrow \mathcal{D}[1:n_1, pairTmp]$;
  $count \longleftarrow count + 1$;

Routine 2: (Remaining $n_2 - \lfloor \frac{n2}{n1} \rfloor n_1$ paths) $T \longleftarrow n_2 - \lfloor \frac{n2}{n1} \rfloor n_1$;
$\mathcal{D} \longleftarrow \mathcal{D}^T$;
$\hat{\mathcal{D}} \longleftarrow \mathcal{D}$ (append rows with Inf);
$pairL, dst \longleftarrow Munkres(\hat{\mathcal{D}})$;
$dstSum \longleftarrow dstSum + dst$;
$dstB \longleftarrow dstB \cup dst$;
$pairList \longleftarrow pairList \cup pairTmp[pairL[1:T]]$;

Routine 3: (Hierarchy mismatch)
$dg \longleftarrow Hist[pairList[:,1]]$ $deg \longleftarrow Hist[pairList[:,2]]$ (second column);
$md \longleftarrow Median[dstB]$, $sd \longleftarrow SD[dstB]$;
$sk \longleftarrow Skewness[dstB]$;
if $sk > 0$ **then**
  $IX \longleftarrow where[dstB > md + sd]$
$misalign2 \longleftarrow [\,]$;
$listU \longleftarrow [\,]$;
for $k \leftarrow 1$ **to** $length[IX]$ **do**
  $pvt \longleftarrow pairList[k, 1:2]$;
  $h1 \longleftarrow H[pvt[1]]$, $h2 \longleftarrow H[pvt[2]]$(max hierarchy);
  if $|h1 - h2| > \frac{max[h1,h2]}{2}$ **then**
    if $deg[pvt[2]] < 2$ **then**
      $misalign2 \longleftarrow misalign2 \cup pvt[2]$;
      $misalign1 \longleftarrow misalign1 \cup pvt[1]$;
      $dstSum \longleftarrow dstSum - dstB[k]$;
      $listU \longleftarrow listU \cap pairList[k, :]$;
if $listU \neq [\,]$ **then**
  Del $pairList[listU]$

Routine 4: (Reassignment)
for $k = 1$ **to** $length[misalign2]$ **do**
    $col \longleftarrow misalign2[k]$;
    $row \longleftarrow \arg\min_{row} \mathcal{D}_c[:, col]$;
    $dstSum \longleftarrow dstSum + \mathcal{D}_c[row, col]$;
    $pairList \longleftarrow pairList \cup [row, col]$;
for $k = 1$ **to** $length[misalign1]$ **do**
    if $dg[misalign1[k]] > 1$ **then**
        $row \longleftarrow misalign1[k]$;
        $col \longleftarrow \arg\min_{col} \mathcal{D}_c[row, :]$;
        $dstSum \longleftarrow dstSum + \mathcal{D}_c[row, col]$;
        $pairList \longleftarrow pairList \cup [row, col]$;

cells, and in progressive degradation of neuronal paths in neurodegenerative diseases.

Chapter 4

Non-unique graph structure

In this section, we discuss some applications with non-unique graph structures. The absence of uniqueness can be attributed to two primary reasons, either fixed number of vertices without physical connections or no fixed vertices and edges, with the latter being more complex than the former. First, we experiment with SqueeSAR dataset, obtained from InSAR images, to identify and monitor three primary events, which are pavement rutting (significantly low-scale event), rock-slope fault (distance scale larger than pavement rutting), sinkhole formation and growth (large-scale event). SqueeSAR data belongs to the first category, where there are a fixed number of scatterers (vertices) without edges.

Event detection from videos falls into the second category, where the vertices and edges are ambiguous. It is a challenging problem to model such problems. To find global patterns, often features are extracted from each frame, and then they are observed over time. To extract the global trend of local patterns, such as in object tracking, researchers generally adopt patch or superpixel based approaches. To extract only local patterns, such as in segmentation, pixel based approaches are employed.

Lastly, we mention a problem regarding improvement of the condition number of the autocorrelation matrices of several classes of data, where the imposition of graph structure

help estimate the connectivity in the graph. This connectivity provides a transformation of the data, which eventually improves the condition number accelerating the convergence of an LMS filter.

4.1 Spatio-temporal event detection

The detection of active regions, which is defined as a predominating behavioral pattern present in data, poses a major challenge in big data. Also called *event detection*, this problem can be regarded as a variant of segmentation problem popular in image processing domain. Essentially, the goal of event detection is the identification of the *number* of active regions, with their precise *locations* and *extents*, present within a dataset. The uncertainty of finding a local perturbation increases if the points are non-uniformly sampled and sparse. Typical approaches aim at capturing the synchronized behavior of a locus of points; however, the extent of the perturbations might vary in size with time. In this work, we consider an application where the goal is the detection of active regions within ground displacement time series datasets collected by interferometric synthetic aperture radar (InSAR). We use these sparse, non-uniformly sampled InSAR measurements to detect geomorphic events such as sinkholes and rockslides.

The existing approaches can be divided in two broad categories - spatial domain techniques and spectral domain techniques. One of the spatial methods involves evaluating the average velocity from the time series of a point and detect behaviors inconsistent with global activities such as seasonal oscillation. This type of point-based approach often does not provide the extent of the anomaly. As an example, considering the displacement associated with a developing sinkhole, the points near the center will have a higher coefficient of variation compared to the points at the periphery. Methods based on the covariance between time series of points may lead to false-positives as they do not differentiate between the magnitudes of the displacements. In addition, the covariance between two very

distant points might be large due to a similar pattern of variation. Other location-based approaches [94, 95] suffer from storage overload and imprecision in locations, whereas location-free methods [96] assume the signal is either stationary or slowly-varying. On the other hand, spectral domain methods [16, 22, 97] decompose the signal in the space spanned by the eigenvectors of the Laplacian in order to identify significant trends in a signal. A recent work [16] using spectral domain technique used the Laplacian of and the difference between potential functions, analogous to Laplacian-of-Gaussians (LoG) and Difference-of-Gaussians (DoG) filters in image processing. A major bottleneck of these approaches is the computation of eigenvectors of the Laplacian matrix, especially in case of the large-scale graphs.

4.1.1 Laplacian weighted covariance (LaWeCo)

Following recent thrust to solve the problem by developing a scale-space approach [17], and spatial-domain template matching [18, 98], we propose a spatial-domain method, LaWeCo, using the Laplacian of a graph to couple neighborhood points combined with a scale-dependent metric that provides information regarding the strength and the extent of the neighborhood affected by the perturbation. The implementation of our algorithm utilizes a scale hierarchy that improves on computational effort by carefully updating the graph Laplacian and covariance matrix at each increasing scale level.

4.1.1.1 Methodology

Having defined the underlying graph framework, let us consider a dataset that can be represented as a graph $\mathcal{G}$ with N nodes. Let us also consider a graph signal $\{f_i(t_k)\}_{i=1,\ldots,N;k=1,\ldots,T}$ where the values are defined for each node v_i and for a set of time intervals t_k not necessarily uniformly sampled. When $\mathcal{G}$ can be embedded in a Banach space, it is possible to define the edge weights as function of the *distance* between the nodes. With the term

scale (s) we identify a region within a distance of s from any given node. The vertex set, $\mathcal{V}$ can be partitioned into M subsets ($\{\mathcal{V}_m^{(s)}\}_{m=1,\dots,M}$) based on the scale s:

$$\mathcal{V} = \bigcup_{m=1}^{M} \mathcal{V}_m^{(s)} \qquad N = \sum_{m=1}^{M} N_m^{(s)}$$

where $N_m^{(s)}$ is the *order* (number of nodes) of $\mathcal{V}_m^{(s)}$. For each partition we can evaluate the covariance matrix $\boldsymbol{C}^{(s)}$ between the time series of the graph nodes ($c_{ij} = \mathrm{cov}(f_i(t_k), f_j(t_k))$ for all the nodes within distance s from each other) and define the *effective variance* ($\sigma_e^{(s)}$) that reflects the strength of coherent activity for each node in the m^{th} neighborhood:

$$\sigma_e^{(s)}(v_i) = \frac{1}{2}\left(\sigma(v_i) + \frac{1}{d_i}\sum_{j\in\mathcal{N}_i^{(s)}} w_{ij}^{(s)}|c_{ji}^{(s)}|\right) \tag{4.1}$$

where $\sigma(v_i)$ is the variance associated with $f_i(t_k)$ and $c_{ji}^{(s)}$ is the indexed element of the covariance matrix. In order to reduce the contribution of covariances from distant nodes, the weights $w_{ij}^{(s)}$ are defined by a decreasing function of the distance between vertices v_i and v_j. In this work we used $w_{ij}^{(s)} = \frac{1}{s\sqrt{1+d_{ij}^2}}$, where d_{ij} is the Euclidean distance between nodes v_i and v_j. By using the graph theoretical framework, we can rewrite (4.1) as

$$\boldsymbol{\sigma}_e^{(s)} = Diag\left(|\tilde{\boldsymbol{L}}^{(s)}||\boldsymbol{C}^{(s)}|\right) \tag{4.2}$$

where $\tilde{\boldsymbol{L}}^{(s)}$ is the normalized Laplacian of the fully connected graph $\mathcal{G}_m^{(s)} = \mathcal{G}_m^{(s)}(\mathcal{V}_m^{(s)}, \mathcal{E}_m^{(s)})$.

The covariance between the time series of two nodes only reflects their linear association and is independent of their magnitudes. To reduce the potential false-positives that might be introduced by this lack of sensitivity to the signal's local power, we incorporate

the mean of the absolute values of the time series in our formulation:

$$\begin{aligned} \boldsymbol{\sigma}_e^{(s)} &= \boldsymbol{\mu}^{(s)T} Diag\left(|\tilde{\boldsymbol{L}}^{(s)}||\boldsymbol{C}^{(s)}|\right) \\ \mu_j^{(s)} &= \frac{\sum_{k=1}^{T}|f_j(t_k)|}{T} \quad \text{for} \quad j \in \mathcal{V}_m^{(s)}. \end{aligned} \tag{4.3}$$

By definition, the Laplacian only considers one-hop connectivity. This can be understood by remembering that it is defined by using degree and adjacency matrices that are constructed using only edges, their weights, and the neighborhood of individual nodes. Within this context, (4.3) is restricted to one-hop connectivity. This definition can be extended to a larger number of hops by using powers of the Laplacian $\tilde{\boldsymbol{L}}$: for k-hops, we can use $\tilde{\boldsymbol{L}}^k$. We can then define a series of different kernels depending on the desired length of the paths.

$$\mathcal{K}\left(\tilde{\boldsymbol{L}}\right) = \begin{cases} \tilde{\boldsymbol{L}}^k & \text{k-hop} \\ \sum_{l=1}^{P}\tilde{\boldsymbol{L}}^l & \text{upto P-hop} \\ \left(\mathbb{I}-\tilde{\boldsymbol{L}}\right)^{-1} & \text{upto } \infty\text{-hop} \end{cases} \tag{4.4}$$

The kernels in (4.4) can be used in (4.3) by replacing $\tilde{\boldsymbol{L}}^{(s)}$ with the one generated by the desired kernel. The inclusion of multiple hops enforces better coupling between points by following edges to transfer the effects of a node perturbation.

4.1.1.2 Multiscale formulation

Since we are interested not only in identifying the location of active regions, but also their size, we implement a scale-space analysis approach by subsequently partitioning the dataset in blocks of increasing size. For computational ease, we start by subdividing the dataset in total 2^Z grids of identical size, set depending on the application at hand, where Z identifies the starting *level*. For each of the blocks, a fully connected graph

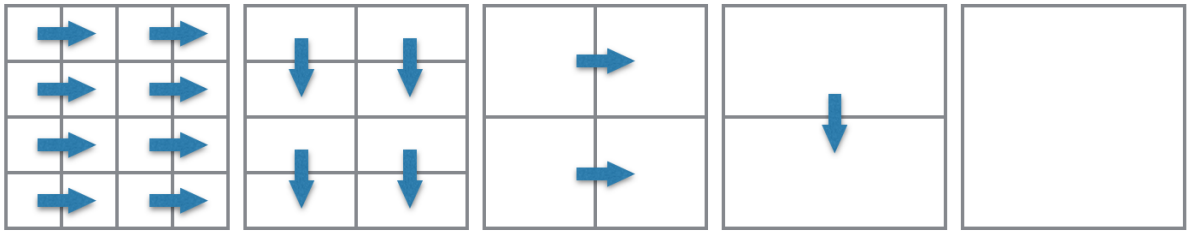


FIGURE 4.1: Multiscale block concatenation in LaWeCo. The grid indicates the blocks used for the analysis whereas the arrows indicate the direction in which the concatenation occurs before the following analysis step.

is generated considering only the nodes within the block, its Laplacian and covariance matrices evaluated, and the value of $\boldsymbol{\sigma}_e^{(s)}$ computed by defining s as the diagonal of the block. Once the first level is evaluated (smallest scale), the next level is obtained by concatenating each block with one of its neighbors *first-right-then-down* fashion as shown in Fig.4.1. Once the new blocks are defined, the previous evaluation of $\boldsymbol{\sigma}_e^{(s)}$ is repeated with now s defined as the length of the diagonal of the concatenated blocks. As an example, consider two blocks, $\mathcal{N}_1$ and $\mathcal{N}_2$, with n_1 and n_2 points respectively. The corresponding Laplacian, $\tilde{\boldsymbol{L}}_1$ and $\tilde{\boldsymbol{L}}_2$ are constructed using fully connected graphs. Let the covariance matrices be $\boldsymbol{C}_1$ and $\boldsymbol{C}_2$ respectively. To compute the next coarser level Laplacian L, the cross-adjacency matrix, $\mathcal{B}_{n_1\times n_2}$, between $\mathcal{N}_1$ and $\mathcal{N}_2$ is computed. Let the degree matrix formed by the row-sum of $\mathcal{B}$ be $D1 = D_{n_1\times n_1}$, and by the column-sum be $D2 = D_{n_2\times n_2}$. The Laplacian $L_{(n_1+n_2)\times(n_1+n_2)}$ of the concatenated blocks becomes

$$L_{(n_1+n_2)\times(n_1+n_2)} = \begin{pmatrix} \rho L1 + D1 & \mathcal{B} \\ \mathcal{B}^T & \rho L2 + D2 \end{pmatrix} \tag{4.5}$$

where $\rho = s_Z/s_{Z-1}$, the *scale-ratio*, is the ratio of the current scale s_Z to the next scale s_{Z-1}. Each edge weight is updated as $w_{ij}^{(s_{Z-1})} = \rho w_{ij}^{(s_Z)}$. The concatenation of blocks at every scale is expected to continue until the active region boundaries are reached. An

active region boundary is obtained when there is a fixed set of points showing significant $\boldsymbol{\sigma}_e^{(s)}$ irrespective of the increment in scale. The update procedure depicted in (4.5) significantly reduces the computational load, especially for large-scale graphs. A similar approach can be used to evaluate the covariance matrix of the concatenated block.

$$C_{(n_1+n_2)\times(n_1+n_2)} = \begin{pmatrix} C_{n_1\times n_1} & C_{n_1\times n_2} \\ C^T_{n_1\times n_2} & C_{n_2\times n_2} \end{pmatrix} \tag{4.6}$$

From (4.5) and (4.6), the calculations of $\mathcal{B}$, $C_{n_1\times n_2}$, and $\mathcal{K}$ are regarded as the computational cost at each scale except the initial one.

4.1.1.3 Dataset

To demonstrate the effectiveness of our approach, we implement it to detect active regions within large graphs derived from InSAR data. In particular we use spatiotemporal point cloud datasets obtained by a state-of-the-art technique known as SqueeSAR [99]. These datasets provide a non-uniformly (both in time and space) sampled measurement of Earth's surface deformations within the imaged area. Each point (node) is associated with a precise latitude and longitude. The time series of a point's displacements is provided and can be utilized to detect sharp changes, seasonal trends, and linear or nonlinear movements.

Dataset 1 - Route 600. The first dataset depicts an impending slope failure on Route 600 near the *Calfpasture river* around the location (11,213,601.181, 6,727,296.005) [1]. The dataset contains 312 points, and the span of time series is from 05 September 2011 to 24 November 2014.

[1]NAD 1983 State Plane Virginia North in US feet

Dataset 2 - Route 11. The dataset is taken on a section of a primary road route 11 around the location (-8,802,382.239, 4,595,066.063) [2]. There are 396 points and the measurements are acquired from 05 September to 14 June 2014.

Dataset 3 - Winksinks. The dataset is based on large sinkholes that collapsed in West Texas. The sinkholes appear to subside linearly with time. This dataset was created using SqueeSAR on a stack of ERS 1/2 images acquired between 03 June 1992 and 21 February 1998 with 93,513 points allocated over an area of 55.37 km^2.

4.1.1.4 Results

We compare our results with three graph-based event region detection methods - Laplacian of potential (LoP) [16], difference in potential (DoP) [16] and normalized graph cut [97]. It appears that our method outperforms the competing methods in two metrics - identification of the number of active regions and reduction in the number of false positives. Furthermore, the execution time of the LaWeCo algorithm is much faster compared to the competing approaches since our method does not involve expensive operations such as spectral decomposition. The runtime for each algorithm is shown in Table 4.1 where it can be noted that, for relatively small sized datasets, LaWeCo is about 10 times faster than the competing methods. For larger datasets such as "Winksinks", the computation time is improved by a factor of 50.

The identification of active regions on route 600 over various scales is shown in Fig. 4.2. The X and Y coordinates are the latitude and longitude of the points, respectively. Fig.4.2(b) shows three active regions marked by circles using kernel $\tilde{L}$ and a scale of 2 feet. One of them subsequently disappears as the scale is gradually increased (c). However, $\tilde{L}$ is a one-hop kernel. To include the contribution over multiple hops, the infinite kernel, $\left(\mathbb{I} - \tilde{\boldsymbol{L}}\right)^{-1}$ is selected. It appears in Fig. 4.2(d)-(e) that there exist two other small-scale active regions consistent with a sequence of increments in scale. The result obtained with

[2] World Geodetic System 1984 in meters

TABLE 4.1: Run time in seconds of LaWeCo , Graph-cut, LoP, and DoP for all the datasets.

Dataset	LaWeCo	Graph-cut	LoP	DoP
Route 600	0.04	0.47	0.37	0.38
Route 11	0.07	0.76	0.43	0.42
Winksinks	6.43	257.3	264.1	281.2

infinity kernel was validated by Virginia DOT field engineers who provided the ground truth.

The graph cut method in Fig. 4.2(g) shows several false positive locations where there are no sign of activities or events as confirmed by the field engineers. LoP and DoP, intrinsically band pass in nature, also erroneously identify points which are not a part of activities.

The results for dataset 2 and 3 are given in Fig. 4.3 and 4.4. In both datasets, LoP, DoP and the graph cut method detect an unacceptable number of false positives. In contrast, the active regions obtained from our method do not grow substantially as the the scale increases.

LaWeCo shows promising results both in the identification of active regions and their extents. The contribution lies in its simple framework and the adaptability to large scale scenarios in contrast to graph signal processing methods that employ spectral decomposition. In addition, our method proves to be superior in reducing the number of false positives. This framework is also amenable to parallel processing for fast computation due to its grid structure. LaWeCo is efficient for spatial localization. However, in order to localize an event temporally, we need to adopt a temporal window based method and check LaWeCo on each window, incurring a huge computational load. The problem becomes worse when the underlying event is propagating in a decentralized fashion, such as sinkhole formation and oil-spills in sea-water.

4.1.2 Decentralized event detection and tracking (DDT)

The majority of existing methodologies in the literature assume a static event in space, in which the centroid of locations of the event remains temporally stationary. However, the problem of localization is exacerbated when the event is propagating in an arbitrary direction, which is known as decentralized behavior. The generation and propagation of activation potentials over neuronal topology of connectivity [100] inside the brain is an ideal example of a decentralized event. The complexity of the spatiotemporal localization of such decentralized events becomes more daunting when the underlying topology is also time-varying. For example, the Deepwater Horizon oil spill [16] was a real-life hazard in which the underlying topology residing on the sea surface was anisotropically drifting at a slow pace. In our work, we focus on the detection followed by the estimation of locations and extents of multiple decentralized events on an irregularly sampled topology which is time varying in terms of a predefined functional relationship.

Also known as distributed network boundary detection [17, 95], the methodologies on event detection can be broadly categorized as spatial domain techniques and spectral domain techniques. The extraction of statistics from the time series of all the sampled nodes on a topology followed by inferences about the characterization of events is one of the earliest approaches for event detection. The set of statistics includes global trend, seasonal variational trend, coefficient of variation and covariance with an assumption of regular sampling of the topology and time, and a reasonable spatiotemporal granularity.

Location based approaches [94, 95], where the location of each sampled spatial node is provided by an external service, often face computational burden and storage overload. By leveraging a graph-theoretic framework, [101] introduced a graph Laplacian weighted covariance based measure to detect multiple events. The spatial extent of each event is determined by a scale-based recursive grouping method. However, the selection of proper thresholds and the imperfect grouping routine introduce imprecision and error in

detecting and localizing events. In addition, the method fails to work on the detection of decentralized events. The work in [18] modeled an event, such as sinkhole formation with a time-varying Gaussian surface. Despite the fact that the algorithm is fast and scalable and can be applied to decentralized event detection, the constraint of Gaussian shape restricts its application. On the other hand, the working principles of location-free methods [96] assume that the signal time series over the topology is either static or stationary in space.

Spectral methods suffer from the drawback that the spatial topology is unknown beforehand. It is a fact that different spatial connections provide different spectra and spectral responses. This issue is marginally addressed in [16], where authors demonstrated their approach on a regular grid and randomly connected testbeds. It is therefore essential to estimate the topological structure *a priori*. Research works related to the estimation of topological structure resort to various assumptions. The work in [102–104] assumed that graph signals are stationary and smooth. The method in [105] modeled an event as a causal autoregressive graph process and estimated the underlying graph structure in terms of the adjacency matrix. Authors in [102] constructed a dictionary to learn the graph topology and expressed the graph signal sparsely and linearly with the dictionary atoms. The temporal evolution of the underlying topology was captured within a reasonable approximation by [21], where an arbitrary change in the topological connection is discouraged.

In our work on decentralized event detection and tracking (DDT), we recursively update the probability of each sampled location (node/vertex) to be associated with events by using an ensemble of vertex reinforced walks [106]. Iterative computation of such probability of all the vertices at each instance automatically splits multiple events with their boundaries and tracks each of the events temporally. Simultaneously, to estimate the topology, the relationships between all the vertices at each instance are updated based

on the walk frequency at that instance. We show the effectiveness of DDT regarding spatiotemporal localization of multiple decentralized events in the result section.

4.1.2.1 Bipartite graph construction

We model a timeseries dataset using a sequence of bipartite graphs. Let the dataset have N regularly or irregularly sampled locations, also called vertices, nodes or entities, and T time points. The set of vertices at time t_i is labeled by $\mathcal{V}_i = [v_{1,i}, v_{2,i}, \cdots, v_{N,i}];\ i \in \{1, 2, ..., T\}$ with the corresponding vector of measurements, also called as the signal, as $S_i = \{s_{1,i}, s_{2,i}, \cdots, s_{N,i}\}$. As an example, in the case of interferometric synthetic aperture radar (InSAR) data, the locations with their geographical coordinates serve as $\mathcal{V}$, with the surface displacements in millimeters at time instance t_i as S_i [101]. We are interested in the absolute forward deviation of the signal over time on each location. Let $\tilde{S}_i$ be the absolute forward deviation, which is given by $\tilde{S}_i = |S_i - S_{i-1}| \in \mathbb{R}^N;\ i \in \{2, ..., T\}$. As there is no prior information, we set S_1 as a zero vector. We construct a fully-connected bipartite graph between times t_i and t_{i+1} as $\mathcal{G}_{i,i+1}, i \in \{1, 2, .., T-1\}$. Therefore, the sequence of bigraphs is denoted by $\mathcal{G}_b = [\mathcal{G}_{1,2}, \mathcal{G}_{2,3}, ..., \mathcal{G}_{T-1,T}]$.

As each vertex in $\mathcal{V}$ is associated with a location, we can compute the Euclidean distance between any pair of vertices. Notice that the graph topology, which is constructed by Euclidean distance only, is invariant over time as the geographical coordinates are fixed. However, the relationship between a pair of vertices is modeled as a function of both the locations and the signals, which indicates that the underlying *functional* topology can change with the changes in signal over time. To incorporate the effect of locations and signal in the bigraphs, we define the weight, w_{ij} of an edge between vertices v_i and v_j as $w_{ij} = |s_{i,p}||s_{j,p}| exp\big(-(\frac{||loc(v_i) - loc(v_j)||_2}{\sigma})^2\big)$. Here, $loc(v_i)$ is the coordinates of location of the vertex v_i, and p is an arbitrary time instance. For notational simplicity, we write $s_{i,p}$ as s_i. The expression of w_{ij} implies that $w_{ij} = 0$ whenever $\tilde{s}_i$ or $\tilde{s}_j$ or both are zero, (no

event). With the weighted bigraphs defined in this manner, we extract an ensemble of walks.

4.1.2.2 Ensemble of walks

In order to extract an ensemble of length-2 walks, let us consider two weighted bigraphs $\mathcal{G}_{i-1,i}$ and $\mathcal{G}_{i,i+1}$. Between a pair of vertices $v_{k,i-1} \in \mathcal{V}_{i-1}$ and $v_{l,i+1} \in \mathcal{V}_{i+1}$, there are $|\mathcal{V}_i| = N$ possible length-2 paths through $\mathcal{V}_i$. Let us take a vertex $v_{m,i} \in \mathcal{V}_i$ which is a vertex on such a length-2 walk $\theta^{i-1,i+1}_{k,m,l} = (v_{k,i-1}, w_{(k,i-1)(m,i)}, v_{m,i}, w_{(m,i)(l,i+1)}, v_{l,i+1})$. The weight $\lambda^{i-1,i+1}_{k,m,l}$ of the walk $\theta^{i-1,i+1}_{k,m,l}$ can be given by $\lambda^{i-1,i+1}_{k,m,l} = w_{(k,i-1)(m,i)} + w_{(m,i)(l,i+1)}$. For a pair of vertices $(v_{k,i-1}, v_{l,i+1})$, the weights of all such N length-2 walks form a weight vector, $\lambda^{i-1,i+1}_{k,l}$. We assign a user-defined parameter, *walk-strength* τ, which acts as a threshold on the sorted $\lambda^{i-1,i+1}_{k,l}$ (decreasing order) to find the subset of vertices $\mathcal{V}_{i,k,l} \subset \mathcal{V}_i$ such that $\sum_{p \in \mathcal{V}_{i,k,l}} \lambda^{i-1,i+1}_{k,p,l} \leq \tau$. $\mathcal{V}_{i,k,l}$ is the subset of vertices $\subset \mathcal{V}_i$, which are intermediate vertices of all length-2 walks from $v_{k,i-1}$ to $v_{l,i+1}$ with a condition of threshold, τ. An example of a timeseries with 4-locations-3-timestamps is given in Fig. 4.5.

After obtaining all such $\mathcal{V}_{i,k,l}$ for all pairs of vertices $v_{k,i-1} \in \mathcal{V}_{i-1}$ and $v_{k,i-1} \in \mathcal{V}_{i-1}$, we define the ensemble of walks as a set of vertices $\mathcal{V}^{en}_i = \cup_{v_k \in \mathcal{V}_{i-1}, v_l \in \mathcal{V}_{i+1}} \mathcal{V}_{i,k,l}$. The rationale of $\mathcal{V}^{en}_i$ is as follows: let us assume that $v_{m,i} \in \mathcal{V}_i$ registers an activity with the magnitude of $\tilde{s}_{m,i}$ that crosses a threshold. To maximize $\lambda^{i-1,i+1}_{k,l}$, all the vertices from $\mathcal{V}_{i-1}$ will have the tendency to pass through $v_{m,i}$ to reach the vertices in $\mathcal{V}_{i+1}$. As a result, $v_{m,i}$ dominates in $\mathcal{V}^{en}_i$.

4.1.2.3 Prior probability and vertex reinforcement

Once $\mathcal{V}^{en}_i$ is computed, we calculate the frequency of each vertex from $\mathcal{V}^{en}_i$, $\chi_i \in \mathbb{R}^N$. $\chi_i \in \mathbb{R}^N$ is then subjected to a vertex reinforced function via pointwise multiplication: $\chi_{i,f} = f \odot \chi_i$. $\chi_{i,f}$, which is normalized and provides a probability estimate $\Gamma_i \in \mathbb{R}^N$ of event membership of all vertices. When f is a constant function, $\chi_{i,f}$ is the normalized

frequency which can be directly computed from χ_i. For the detection and localization of decentralized events, it is preferable to set the reinforcement function as the local trend of the time series. For example, the value of the reinforcement function at node $v_{k,i}$ can be given by $f(v_{k,i}) = |\frac{\tilde{s}_{k,i-1}+\tilde{s}_{k,i}+\tilde{s}_{k,i+1}}{3}|$.

We use the probability Γ_i as a prior to detect the events between $\mathcal{G}_{i,i+1}$ and $\mathcal{G}_{i+1,i+2}$. Accordingly, the probability Γ_i will be updated to Γ_{i+1} and so on. It is to note that the subset of vertices ($\subset \mathcal{V}_i$) which are not associated with any event, will not appear in $\mathcal{V}_i^{en}$. This is due to the fact for any vertex $v_{p,i}$ belonging to that subset, the weights $w_{(\bullet,i-1)(p,i)}$ and $w_{(p,i)(\bullet,i+1)}$ become zero as explained at the end of section 4.1.2.1. Therefore, this subset of vertices is ruled out by the threshold τ while searching for the ensemble of walks.

The selection of $\tau \in (0, 1]$ is crucial as it helps delineate the neighborhood of a vertex which participates in an event. It is evident that when $\tau = 1$, all the vertices in $\mathcal{V}_i$ will appear with identical frequency, giving a uniform prior probability with $P(\mathcal{V}_{m,i}) = \frac{1}{N} \forall m$. This means that all vertices participate in the event, which results worst-case event localization. It is empirically observed that $\tau \in [0.7, 0.85]$ provides a reasonable localization with negligible error.

In order to formulate mathematically, let A_c^i be a matrix$\in \mathbb{R}^{|\mathcal{V}_{i-1}|\times|\mathcal{V}_i|}$, which is called as the cross adjacency matrix. Let Γ_{i-1} be the prior estimate of events for $\mathcal{V}_{i-1}$. The weight vector of 2-length walks, $\lambda_{k,l}^{i-1,i+1}$ can be given by

$$\lambda_{k,l}^{i-1,i+1} = [A_c^i(k,:)]^T \odot \Gamma_{i-1} + A_c^{i+1}(:,l), \tag{4.7}$$

where $\odot$ is the Hadamard product. The inclusion of Γ in eq. 4.7 guides the algorithm to track the probable locations of events.

4.1.2.4 Extraction of functional topology

The functional topology at any time indicates how the vertices are related to each other in terms of the weight function $w_{ij} = |\tilde{s}_i||\tilde{s}_j|exp\big(-\frac{||loc(v_i)-loc(v_j)||_2}{\sigma}\big)^2$. Once the $\mathcal{V}_i^{en}$ is obtained, let $\mathcal{E}_i^{en}$ be the set of edges $\cup_{v_k\in\mathcal{V}_{i-1},v_l\in\mathcal{V}_{i+1}}\{(v_{k,i-1},v_{m,i})|m\in\mathcal{V}_{i,k,l}\}\cup\{(v_{p,i},v_{l,i+1})|p\in\mathcal{V}_{i,k,l}\}$ with the set of corresponding weights $\mathcal{W}_i^{en}$. It is obvious that an edge can appear multiple times in $\mathcal{E}_i^{en}$. In addition, as the bigraphs are simple and undirected, $(v_a,v_b)=(v_b,v_a)\forall a,b$. We define the weight of an edge$(v_a,v_b)\in\mathcal{E}_i^{en}$ with $v_a\neq v_b$ as

$$\phi((v_a,v_b)) = \sum\big(w_{(v_a,v_b)}+w_{(v_b,v_a)}\big); w_{(v_a,v_b)}\in\mathcal{W}_i^{en}, \tag{4.8}$$

The graph between N vertices as a representation of functional topology at a time instance is expected to be sparse and sometimes, disconnected. A sample example is explained in Fig. 4.5(b).

4.1.2.5 Results

We show the effectiveness of DDT in terms of detection, spatial localization and temporal tracking on two datasets, which are a synthetic dataset and an InSAR dataset. We compare the results on InSAR dataset with the state-of-the-art techniques - LaWeCo [101], difference of potential (DoP) [16], and laplacian of potential (LoP) [16]. DDT involves two hyperparameters - σ and τ. If the coordinates of locations are available, we set $\sigma = \frac{\text{maximum distance over all pairs of locations}}{2N}$ and $\tau = 0.7$ respectively. The synthetic dataset consists of measurements of surface displacements (in mm) over six time points on nine different locations with 2D coordinates. Overall, there is one centralized event and one decentralized event as shown in Fig. 4.6(a) and (c). The connectivity of the underlying functional topology is shown in Fig. 4.6(b) and (d) respectively. Fig. 4.6 provides a visualization of the probabilities $P(\mathcal{V}_{m,i});\ i\in\{1-5\};\ m\in\{1-9\}$ of the locations to be

associated with the events. At the beginning, it is a uniform distribution, and later the two events with their locations and boundaries are tracked.

The InSAR dataset as depicted in Fig. 4.7(a) is captured near the Calfpasture river on Route 600 at approximate location $(11213, 601.181, 6, 727, 296.005)$. The dataset contains 312 points and the surface displacements (in millimeter) are measured from 05-September-2011 to 24-November-2014. The dataset contains two in-place centralized events and one decentralized event of impending rock slope failure. The results on spatial localizations of the events in case of LoP, DoP, and LaWeCo are shown in Fig. 4.7(b),(c), and (d) respectively. Both LoP and DoP produce significant number of false positives as they detect places where there is no evidence of any event. LaWeCo performs better compared to LoP and DoP in terms of the number of false positives. However, LaWeCo fails to identify a region with a centralized activity, which is due to the improper selection of threshold. By using the event probability estimate of the locations, in Fig. 4.7(e), we captured the slow decentralized drift of the rock slope as is evident by the green curve $(10 - 23 - 2014)$ that moves past the red $(01 - 19 - 2012)$ curve. This detection and finding out the evidence of the actual movement is useful for monitoring such progressive geological hazards. The event probability map over all the locations and all the time instance are shown in Fig. 4.7(f).

Our algorithm, DDT excels in identifying, localizing and tracking multiple decentralized events. DDT provides a functional relationship between locations or entities at each time instance. In addition, each time the algorithm computes a probability estimate of event membership of each location. The algorithm is fast, scalable, computationally tractable, and inexpensive in terms of memory consumption.

Now, designing an automated methodology for event detection and recognition from videos is a critical task in security and surveillance. This problem demands localization of an event both in space and time from video data.

4.1.3 Graph based dictionary for event detection (GraDED)

In order to obtain a potentially acceptable solution to the problem, it is imperative to separate the foreground, which is the region of our interest, from the background. Conventional methods assume a static or quasi-static background [107] where the foreground is designed as a sparse matrix by considering the event as a local phenomenon. Other methods such as the saliency-driven event detection developed in [108] also assume the static nature of background. As an alternative, frame-wise or block-wise principle component analysis (PCA) [109] captures the direction and magnitude of maximum variability in a sequence of video frames. Robust PCA [110] extends this framework by retrieving the background as a low-rank matrix to incorporate minor spatio-temporal non-stationarity due to jitter, clutter and variation in illumination. In other work, the background is modeled as a mixture of Gaussians [111] to capture variations along with scene changes.

Real-time videos captured by car-mounted or hand-held cameras cannot be effectively analyzed with the aforementioned models. The high-degree of non-stationarity in the background, sporadic changes in camera angle (especially while driving on uneven road surface or persistent swaying motion of human gait while holding the camera for example), and illumination changes affect the performance of existing algorithms. In this work, we aim at detecting events exhibiting limited duration and sufficient local extent in the presence of dynamic background. In the majority of these scenarios (one such example is shown in Fig. 4.8) there may be no actual object present to track. So, analyzing the motion of objects to detect anomalies in videos using a tracking algorithm [112, 113] or trajectory estimation [114, 115] is not an appropriate solution for such data.

In recently reported work, it has been shown that subspace-based methods are keen to detect subtle changes of dynamic background [116–118]. The key idea is to represent variability of a feature in terms of the coefficients of a subspace. It is expected that the high-degree nonlinearity caused by the variation in an agent such as jitter can be

projected onto a subspace allocated for that agent. We seek to find the assembly of these subspaces, which we call a dictionary [119], to detect events in a video. There have been few emerging methods that used dictionary learning to build subspaces and exploit the reinforced sparsity of the input data to formulate suitable measures for event detection (SSPARED) [120]. The SSPARED algorithm exploits sparse codes obtained by cross-dictionary representation in conjunction with K-L divergence to detect substantial changes in consecutive frames. The algorithm is computationally expensive due to the dictionary construction per frame and cross-dictionary representation for each pair of consecutive frames and only provides the temporal extent of an event.

We develop a block-based graph-assisted dictionary learning algorithm (*GraDED*) to identify both spatial and temporal extents of an event in a video with a dynamic background as shown in fig 4.8. In this work, the graph Laplacian weights are employed to detect the temporal extent of the event. The learned dictionary is analyzed to determine the spatial localization of the event.

Before delving into the formulation of our algorithm, we give an overview of the graph tools that are used in the derivation.

4.1.3.1 Spatio-temporal graph representation of video

Let V be a video containing K frames, where each frame is partitioned into P non-overlapping blocks (see fig. 4.8). Collectively the set of i^{th} block of all frames constitutes the i^{th} sub-volume of the video. $B^i_j \in \mathcal{R}^{1\times S}; j \in \{1, 2, \cdots, K\},\ i \in \{1, 2, \cdots, P\}$ is a feature extracted from i^{th} block. The set of all features in i^{th} sub-volume, denoted as $B^i \in \mathcal{R}^{K\times S}$, is used to construct the dictionary of i^{th} block.

Firstly, we perform PCA on B^i to obtain maximum coherence within the basis of each block. We take a subset of leading eigenvectors in terms of the magnitude of eigenvalues. The span of the eigenvectors forms the subspace of principle variation in i^{th} block over

all the frames. The coherence [121] between two basis matrices can be defined as

$$\mu\big(X, Y\big) = \max_{k,j} | < x_k, y_j > |, \tag{4.9}$$

where x_k and y_j are the k^{th} and j^{th} columns of X and Y respectively. The coherence within i^{th} block is called *intra-block coherence*; whereas between two different blocks, it is *inter-block coherence*. As eigenvectors are orthonormal, μ_i is unity for i^{th} block. However, no inference can be made regarding the coherence between two different blocks.

Since, B^i_j the principal components of B^i can be exploited to analyze the temporal variations in a video. Let M_i be the number of leading eigenvectors, which is denoted as $\chi_i \in \mathcal{R}^{K \times M_i}$. The choice of the number of eigenvectors is critical, since a large number would increase the computational time of the algorithm, whereas a small number may fail to capture the background and foreground variations.

Now, motivated by [122], the idea is to subject χ_i by a distance-preserving linear transformation such that a subset of P blocks will have significant mutual incoherence with respect to the rest without affecting the *intra-block mutual coherence*. Then, that subset of blocks can be identified as the spatial localization of the event in a video. Let U be such a desired transformation. Precisely, we want $\mu_i(U^T\chi_i, U^T\chi_i) = \mu_i(\chi_i, U * U^T\chi_i) = \mu_i(\chi_i, \chi_i)$. It is evident that U has to be unitary. We want such a transformation to be data-driven, which motivates us to apply graph theory.

The graph provides a useful framework to iteratively retrieve meaningful relationships among data samples. We consider each block of a frame as a vertex of a graph. In the i^{th} sub-volume, there are K number of vertices which are connected as a linear graph having $(K-1)$ weights as defined in section 2.1. The eigenmatrix of the graph Laplacian, L_i, is derived by using the set of weights as our desirable transformation for the i^{th} sub-volume. The weights are essentially free parameters, which are updated according to a

cost function. The changes in weights lead to a change in L_i, which consequently induces a change in U_i, which is the eigenmatrix of L_i.

To build a compact mathematical framework, χ_i is left-multiplied by a unitary matrix U_i, which is considered to be the i^{th} sub-dictionary, D_i. The overall dictionary structure is given by,

$$D = [D_1 \ D_2 \ \dots \ D_P] = [U_1^T\chi_1 \ U_2^T\chi_2 \ \dots \ U_P^T\chi_P]. \tag{4.10}$$

Here $M = \sum_{i=1}^{P} M_i$ and $D \in \mathcal{R}^{K\times M}$. In eq. (4.10), there are in total $P(K-1)$ parameters of the dictionary which needs to be updated iteratively for P number of linear graphs. We select each graph to be linear as it contains the least number of edges for a given set of vertices in a connected graph. Therefore, this configuration contains the least number of parameters, which, in effect, reduces the possibility of over-fitting during optimization. It should be noted here, from graph-theoretic perspective, that there are multiple isomorphic candidates which share the property of least number of edges. For example, a star-graph contains the same number of edges as of a linear graph with a fixed cardinality of vertex sets. However, only linear graph maintains the temporal order of the event occurring in a video.

4.1.3.2 Graph based parametric dictionary learning

In this section, we attempt to find the set of transformations, U_is by using graph Laplacians. Let us consider, the sequential stack of sub-volume features as $Y = [B^1 \ B^2 \ ... \ B^P]$. The corresponding high-dimensional sparse code is $X = [X^1 \ X^2 \ ... \ X^P]$. The optimization problem by using D, Y, and X can be given by,

$$\left(D^*, X^*\right) = \min_{D,X} \|Y - DX\|_F^2 \ \text{ s.t. } \|X\|_0 \leq T. \tag{4.11}$$

The objective function in the above equation is non-convex. Conventionally, we resort to an alternating minimization technique in which each quantity is minimized by keeping the other one fixed. The above non-convex cost function and the constraints in eq. (4.11) can be stitched together with the help of Lagrangian coefficient λ as

$$\phi\big(D, X, \lambda\big) = \|Y - DX\|_F^2 + \lambda \|X\|_0 . \tag{4.12}$$

It is to be noted here that the dictionary D is a function of a set of Laplacians L_i via U_i as evident from eq. (4.10). In addition, each L_i is a function of diagonal weight matrix W_i. Therefore, the cost function in eq. (4.22) can be rewritten as

$$\begin{aligned} \phi\big(\{W_1, ..., W_P\}, X, \lambda\big) &= \|Y - DX\|_F^2 + \lambda \|X\|_0 \\ L_i = U_i \Gamma_i U_i^T, L_i &= \Upsilon W_i \Upsilon^T; i \in \{1, 2, ..., P\}. \end{aligned} \tag{4.13}$$

In alternating minimization method, the sparse feature matrix X is kept fixed during the update step of dictionary D. By using eq. (4.10), the feature Y in eq. (4.13) can be expanded as $Y = \sum_{i=1}^{P} D_i X_i$.

Note that X_i is not the sparse code for B^i, which we distinguish by subscript and superscript in notation. Rather, it can be interpreted by the row-wise partition of X i.e. the i^{th} row-block of X is multiplied by D_i. By using $Y = \sum_{i=1}^{P} D_i X_i$, the first term in eq. (4.13), which is the reconstruction error, can be restructured as a function of D_i as

$$\phi\big(D_i\big) = ||E_i - D_i X_i||_F^2; \;\; E_i = Y - \sum_{j \neq i} D_j X_j. \tag{4.14}$$

To update the block dictionary D_i iteratively, the parameters W_i needs to be estimated by gradient descent. With the help of eq. (4.13) and (4.14), the update equation for W_i

can be given by,

$$\begin{aligned} w_{ij}^{(t+1)} &= w_{ij}^{t} - \eta Tr\Big(\big[\frac{\partial \phi(D_i)}{\partial U_i}\big]^T \frac{\partial U_i}{\partial w_{ij}}\Big); \\ \frac{\partial U_i}{\partial w_{ij}} &= \Big[Tr\Big((\frac{\partial L}{\partial u_{kl}})^{-T} \frac{\partial L}{\partial w_{ij}}\Big)\Big]_{kl}. \end{aligned} \tag{4.15}$$

Here, w_{ij}^t denotes the j^{th} weight of W_i in eq. (4.13) at t^{th} iteration. η is the learning step parameter, a scalar value selected from $(0, 1)$. Tr is the matrix trace operator which sums up the diagonal values of a matrix. $[\bullet]_{kl}$ is the kl^{th} element of the matrix expressed as $[\bullet]$. It is evident from eq. (4.15) that in order to update weight w_{ij}, three partial derivatives are to evaluated at each step - $\frac{\partial \phi(D_i)}{\partial U_i}$, $\frac{\partial L}{\partial u_{kl}}$, and $\frac{\partial L}{\partial w_{ij}}$. Precisely, in order to obtain $\frac{\partial U_i}{\partial w_{ij}}$, for each (k, l) a total of K^2 computation of $\frac{\partial L}{\partial u_{kl}}$ is needed, which is computationally expensive. However, it appears that the loss of computational simplicity is compensated by considerably fast convergence of the algorithm.

Now, $\frac{\partial \phi(D_i)}{\partial U_i}$ in eq. (4.15) can be evaluated by using eq. (4.14).

$$\begin{aligned} \frac{\partial \phi(D_i)}{\partial U_i} &= \frac{\partial \left\| E_i - D_i X_i \right\|_F^2}{\partial U_i} \\ &= \frac{\partial}{\partial U_i}\Big(E_i - D_i X_i\Big)^T \Big(E_i - D_i X_i\Big) \\ &= \frac{\partial}{\partial U_i}\Big(E_i^T E_i - 2E_i^T D_i X_i + X_i^T D_i^T D_i X_i\Big). \end{aligned} \tag{4.16}$$

By construction $D_i = U_i \chi_i$ from eq. (4.10), which asserts that $D_i^T D_i = \chi_i^T U_i U_i^T \chi_i = \chi_i^T \chi_i = I$ by using $U_i U_i^T = I$. It immediately appears that $X_i^T D_i^T D_i X_i = X_i^T X_i$. After inserting this expression in eq. (4.16), $\frac{\partial \phi(D_i)}{\partial U_i}$ becomes $-2\chi_i X_i E_i^T$. The second partial derivative, $\frac{\partial L}{\partial u_{kl}}$ can be evaluated by using $L_i = U_i \Gamma_i U_i^T$, where Γ is a diagonal matrix containing the eigenvalues of L_i. By standard rule of matrix derivative and using $J^{mn} =$

$\delta_{mk}\delta_{nl}$,

$$\frac{\partial L_i}{\partial u_{kl}} = \frac{\partial}{\partial u_{kl}} U_i \Gamma_i U_i^T = U_i \Gamma_i J^{mn} + J^{nm} \Gamma_i U_i^T. \tag{4.17}$$

4.1.3.3 Event detection using learned graph weights

The event detection is performed by identifying its spatial and temporal extent independently. Both problems can be categorized as binary classification problems - associated with event or no-event.

To figure out the temporal changes, each set of weights between consecutive frames is taken as a vector to represent temporal variability. The weight vectors are then clustered by k-means with Euclidean distance. The spatial localization is obtained via clustering dictionary atoms. As dictionary atoms are inherently orthonormal, a refined approach would be to consider clustering over a Grassmann manifold. However, such clustering involves a computationally intensive intermediate step of Karcher mean calculation. Instead, we perform agglomerative clustering to sort out the subset blocks carrying event signatures. We use an unweighted average distance to measure similarity between clusters and inter-block coherence as the distance between two features in the hierarchical clustering.

4.1.3.4 Implementation, results & discussion

We evaluate the performance of our proposed algorithm on three video datasets [120], as shown in fig. 4.9. Video I contains 75 frames depicting the disappearance and reappearance of a pilot boat. In this video, the mounted camera position is kept fixed but the background variation is primarily caused by intermittent sea-waves. Video II with

69 frames shows an explosion in a gas station. The camera mounted on a running vehicle captures significant background motion. Video III contains 77 frames showing a car accident followed by a fire during daytime from a car-mounted camera.

For temporal localization of an event, we set the ground-truth by tagging the frames manually in which the event persists including the start and end frames. We compare our results with three recent methods - SSPARED [120], ADM [118], and DRMF [117]. In addition, we also provide spatial localization by identifying the subset of blocks containing the event. We use *sensitivity* and *specificity* as metrics to evaluate our results for spatial and temporal localizations separately.

In all of our experiments, we extract block-wise histogram of oriented gradient (HOG) [41] features with a cell size of 16×16, followed by dimension-wise normalization of the features as a prerequisite for orthogonal matching pursuit to generate sparse code. With the user-specified parameters, the alternative minimization procedure is executed for dictionary update and sparse code generation. During this procedure, as an intermediate step, the weights of all the graphs are normalized before obtaining the Laplacian matrices at every iteration.

We demonstrate the effects of different parameters - number of blocks (P), downsampling of frames, and number of leading eigenvectors (M) on sensitivity and specificity. We do not set M directly for our experiments. Instead, we introduce a parameter, *eigenfactor* such that $M_i = \lceil \frac{K}{eigenFactor} \rceil \ \forall i$. The results shown in fig. 4.10(a) suggest that the desirable choice of the number of blocks, P would be 16, implying the preference for coarse spatial partition. It is because, for significantly large number of blocks per frame, the signature of an event will be distributed among the blocks making it harder to distinguish from the background. Similarly, from fig. 4.10, the preferred downsample rate is 3. For higher rates, the variation between consecutive frames would be significant leading to an erratic prediction of event frames. It is to note that downsample reduces the

number of graph weights $(K-1)$, which provides two advantages - computational speed and reduced possibility of over-fitting. Likewise, according to fig. 4.10(c), the preferred eigenfactor would be 4 or 5. Higher eigenfactors make the sub-dictionaries thinner, which precludes to capture major variations. Dictionaries with a lower eigenfactor encounter almost every possible unwanted variation and loss of computational speed. Fig. 4.10(d) presents the comparison with SSPARED, ADM, and DRMF methods, and it shows improvements in specificity and sensitivity by 0.08 and 0.6 respectively for time localization. The results of spatial localization for the three video datasets by our method are given in Table 4.2.

TABLE 4.2: Sensitivity & specificity scores of GraDED for spatial localization.

Dataset	Sensitivity	Specificity
Video I	0.75	0.59
Video II	1	0.54
Video III	0.8	0.64

We present a parametric dictionary learning approach by leveraging the graph framework to detect short-time, spatially-local rare events. In this work, the variations of moving backgrounds, jitter, clutter and varied illumination are overcome by utilizing block-wise subspaces in the dictionary. The graph plays a crucial role in spatial localization of events by iteratively updating the edge-weights. We show the effectiveness of our algorithm by identifying the temporal extent of the events compared to three different state-of-the-art methods. In addition, we perform simultaneously spatial localization by locating the set of blocks in frames in which the events happened. One potential improvement involves the derivation of the graph Laplacian from the sparse code, and the control of the regularization of the Laplacian in order to obtain spatio-temporal event localization.

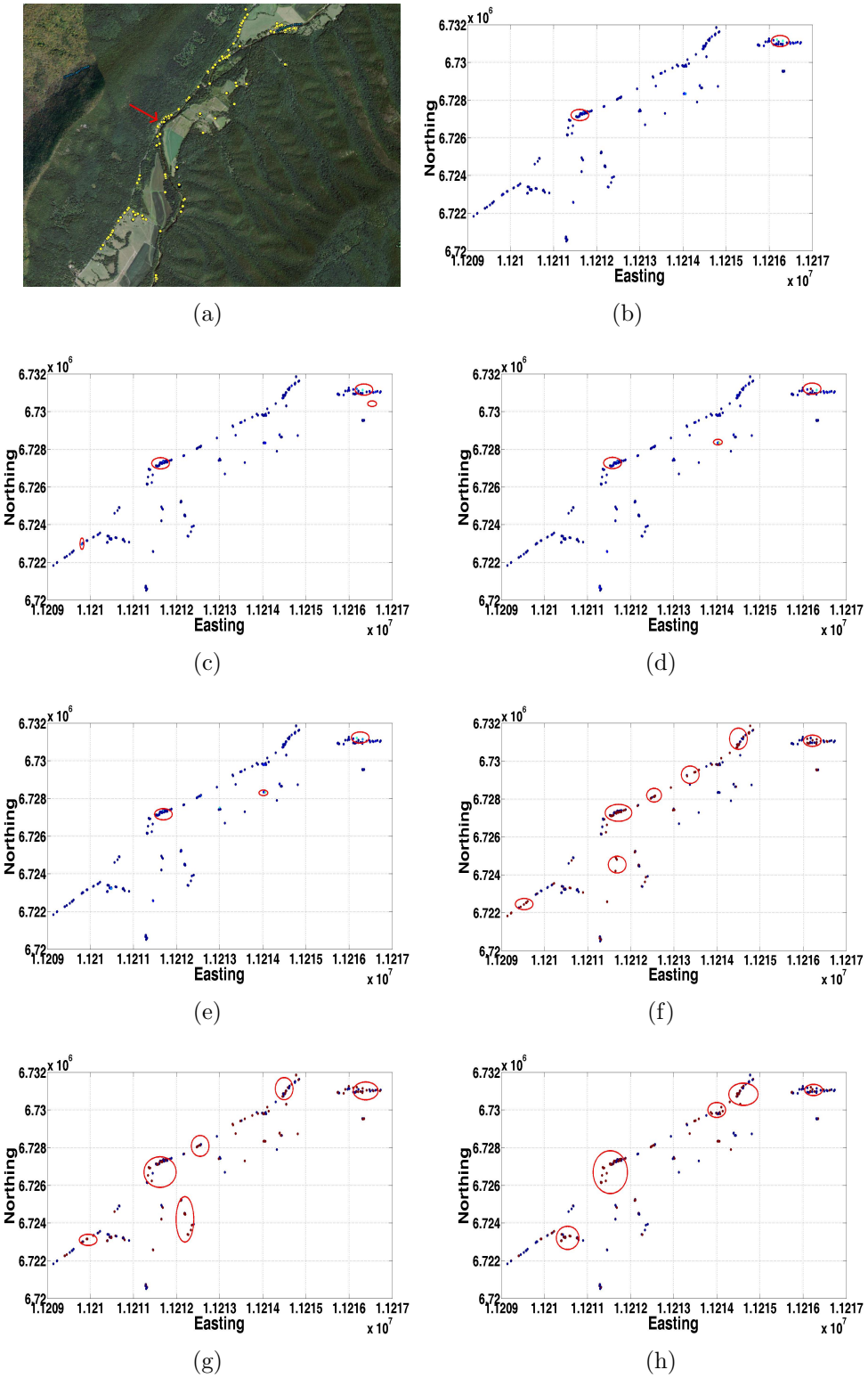


FIGURE 4.2: Dataset-1: Active regions identified using different kernels and scales are marked by red circles. (a) The original dataset, (b) LaWeCo: kernel $\tilde{L}$, scale 2 feet, (c) LaWeCo: kernel $\tilde{L}$, scale 10 feet, (d) LaWeCo: kernel $(I - \tilde{L})^{-1}$, scale 10 feet, (e) LaWeCo: kernel $(I - \tilde{L})^{-1}$, scale 30 feet (f) graph cut, (g) LoP, (h) DoP.

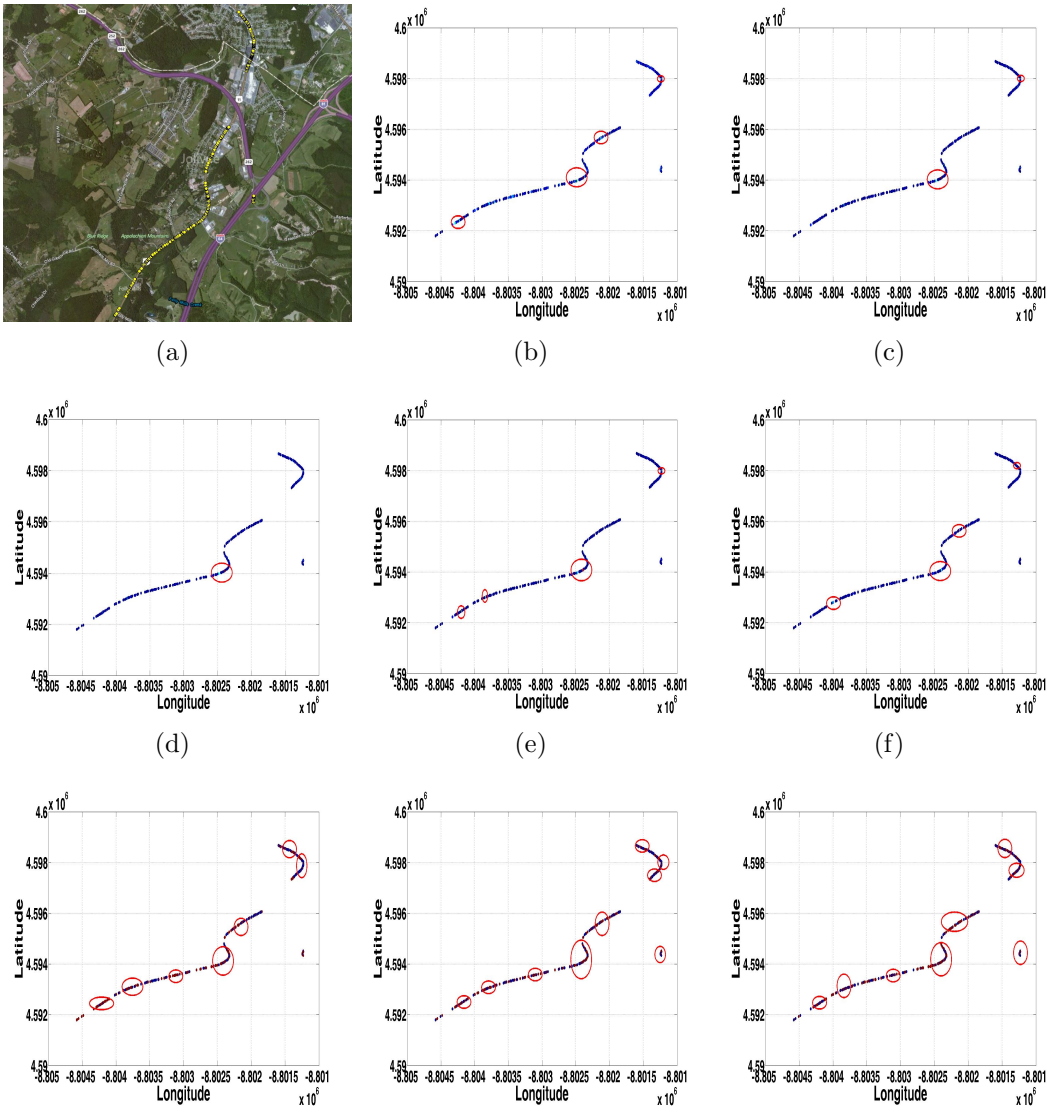


FIGURE 4.3: Dataset-2: (a) The original dataset. Active regions using different kernels and scales are marked by red circles. (b) LaWeCo: kernel $\tilde{L}$, scale 1 meter, (c) LaWeCo: kernel $\tilde{L}$, scale 50 meter, (d) kernel $(I - \tilde{L})^{-1}$, scale 100 meter, (e) LaWeCo: kernel $\sum_{i=1}^{2} \tilde{L}^i$, scale 10 meter (f) LaWeCo: kernel $(I - \tilde{L})^{-1}$, scale 50 meter (g) graph cut, (h) LoP, (i) DoP.

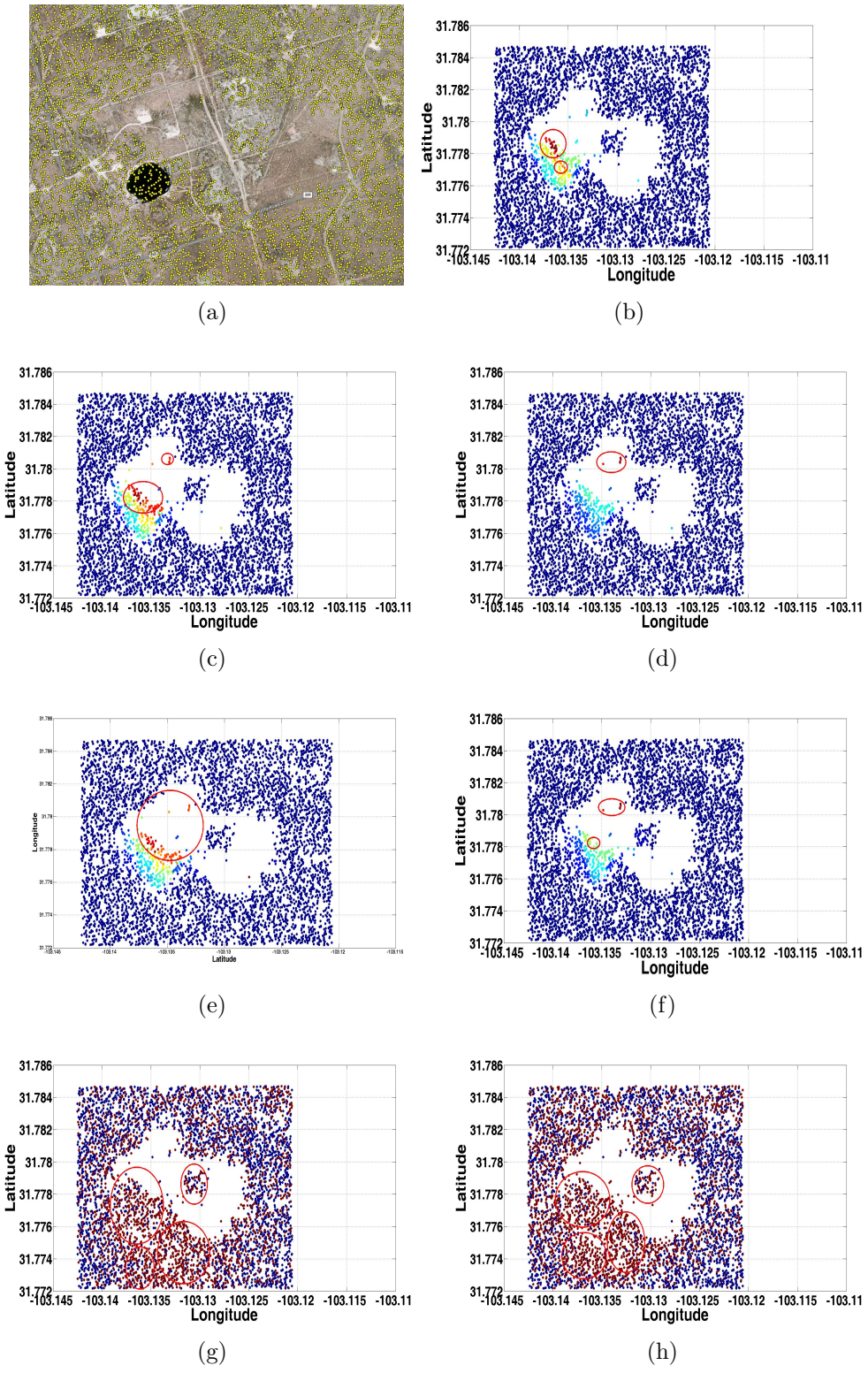
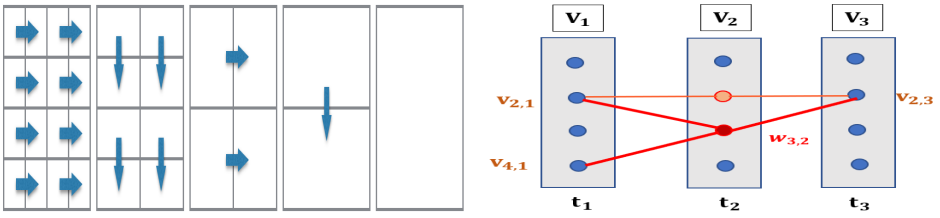
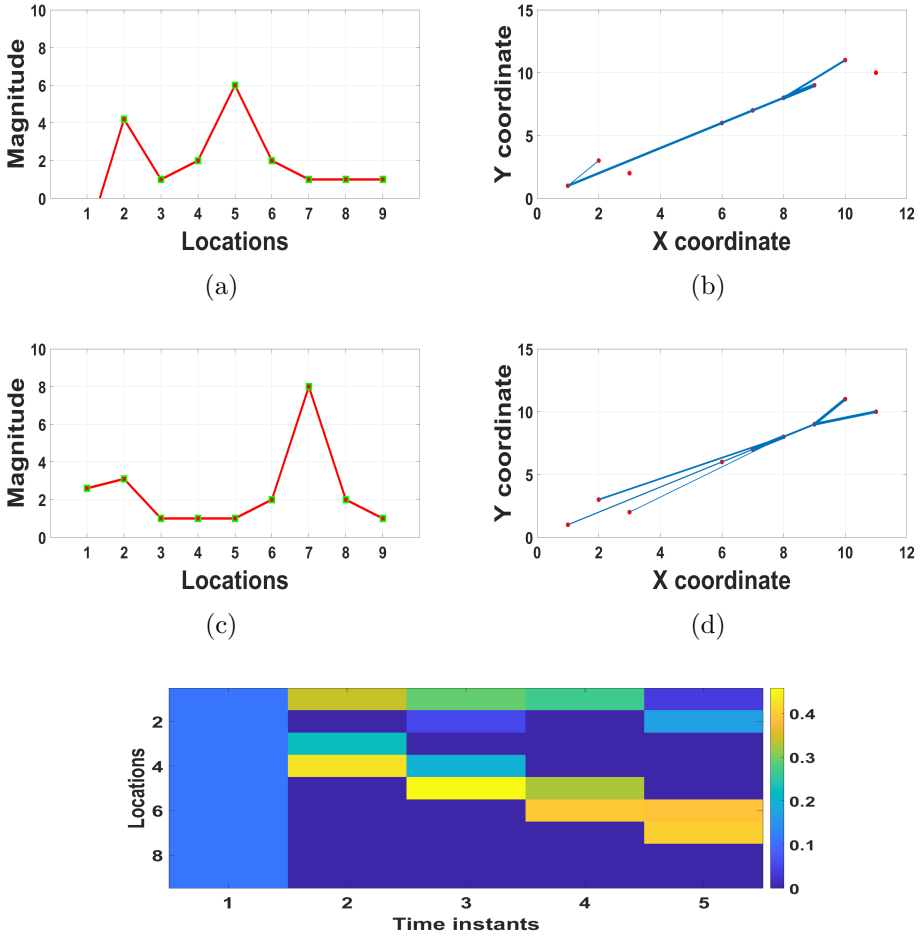


FIGURE 4.4: Dataset-3: (a) cropped original dataset. Active regions using different kernels and scales are marked by red circles. (b) LaWeCo: kernel $\tilde{L}$, scale 0.002 degree, (c) LaWeCo: kernel $\tilde{L}$, scale 0.004 degree, (d) LaWeCo: kernel $\tilde{L}$, scale 0.008 degree, (e) LaWeCo: kernel $\sum_{i=1}^{4} \tilde{L}^i$, (f) LaWeCo: kernel $(I - \tilde{L})^{-1}$, (g) LoP, (h) DoP.



(a) (b)

FIGURE 4.5: The above figure is a representation of a process with four locations and three times, with the $v_{3,2}$ and $v_{2,2}$ registering an event. (a) It shows the set of 2-length walks $\theta^{1,3}_{2,m,3}; m \in \{1,2,3,4\}$ from $v_{2,1}$ to $v_{3,3}$. Out of four possible walks, the combined normalized weights of $\theta^{1,3}_{2,3,3}$ and $\theta^{1,3}_{2,2,3}$ exceed the threshold τ. $\mathcal{V}_{2,2,3} = \{v_{2,2}, v_{3,2}\}$. (b) It shows a subset of walks from $v_{2,1}$ and $v_{4,1}$ to $v_{2,3}$. Notice that the edge from $v_{3,2}$ to $v_{2,3}$ is counted twice. Therefore, the edge weight of (v_2, v_3) in the topology at time t_2(see eq. 4.8) is $2w_{3,2}$.



(a) (b) (c) (d) (e)

FIGURE 4.6: Results on the synthetic dataset describing two events - one centralized and one decentralized. (a) and (c) are the displacement at time instances $t = 3$ and 5. The corresponding functional topologies are shown in (b) and (d). The detection, localization ad tracking of the two events are shown in (e) with the probabilities of event membership of the 9 locations.

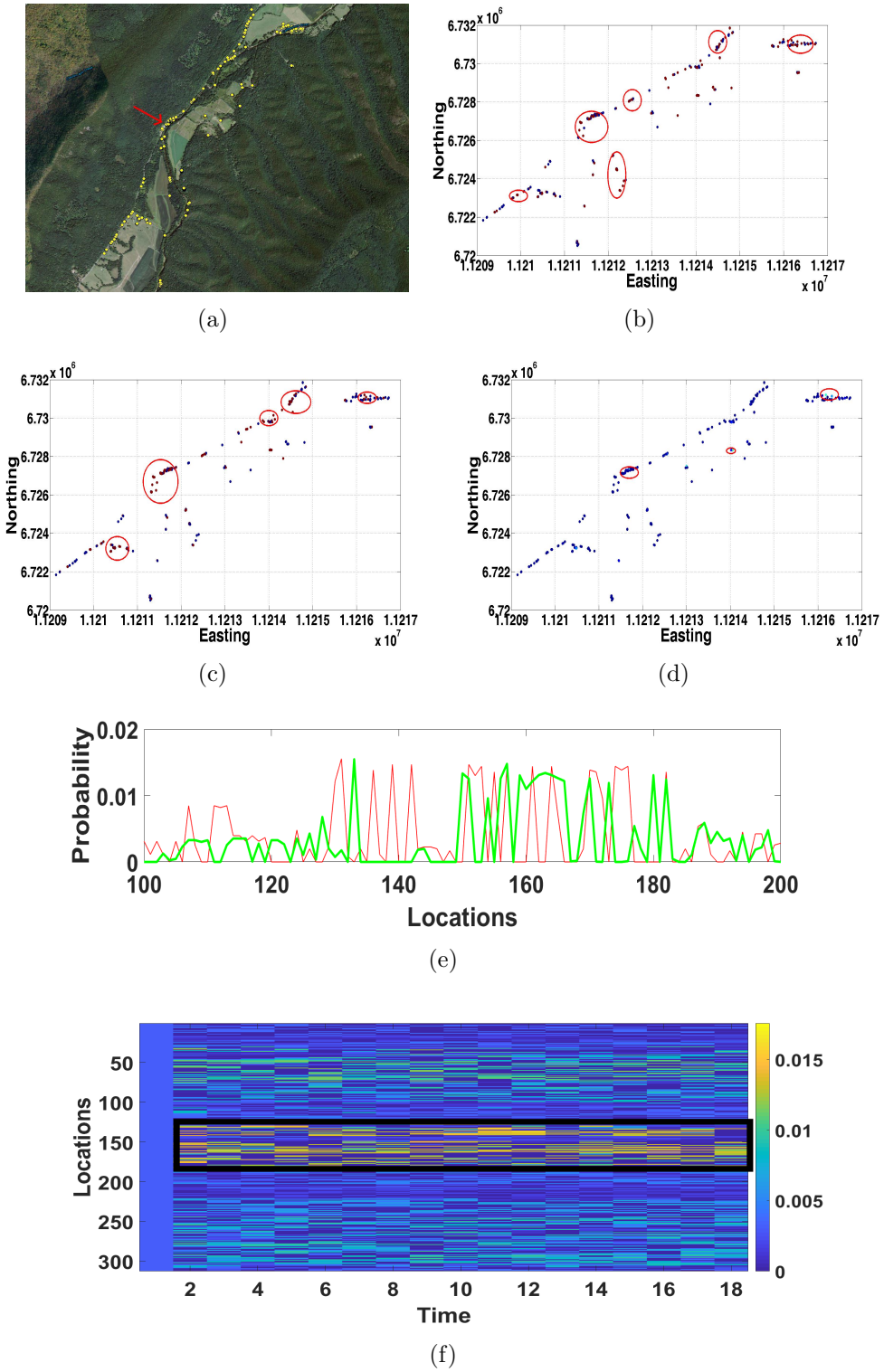


FIGURE 4.7: (a)The map of Route 600 which depicts two centralized surface deformations and a decentralized rock slope fault (region of interest). The performance of three state-of-the-art methods in terms of the detection and localization only are shown in (b),(c), and (d). (e) shows the slow drift of the impending rock slope fault from $01-19-2012$(red) to $10-23-2014$(green). (f) The event probability of all the locations over all the time periods. The black rectangle encapsulates the behavior of rock slope fault over time.

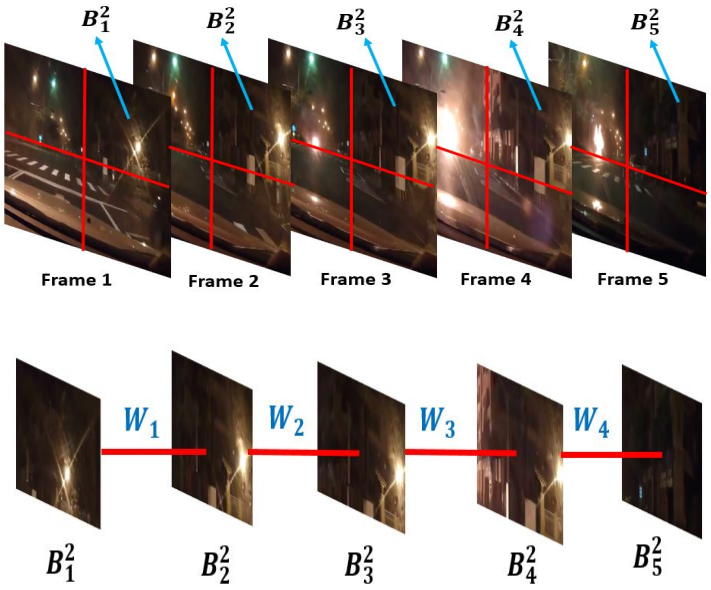
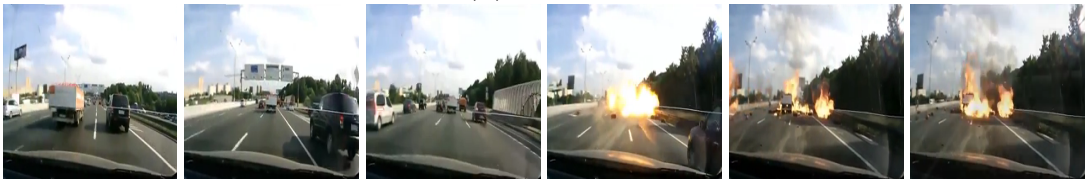


FIGURE 4.8: Top row: An event video with four segments per frame. Bottom row: The linear graph with five nodes and four weight parameters for the 2^{nd} sub-volume.

(a) Video 1

(b) Video 2



(c) Video 3

FIGURE 4.9: The video in (a) shows disappearance and reappearance of pilot boat; (b) shows a spontaneous fire on a street. (c) shows an accident on a highway.

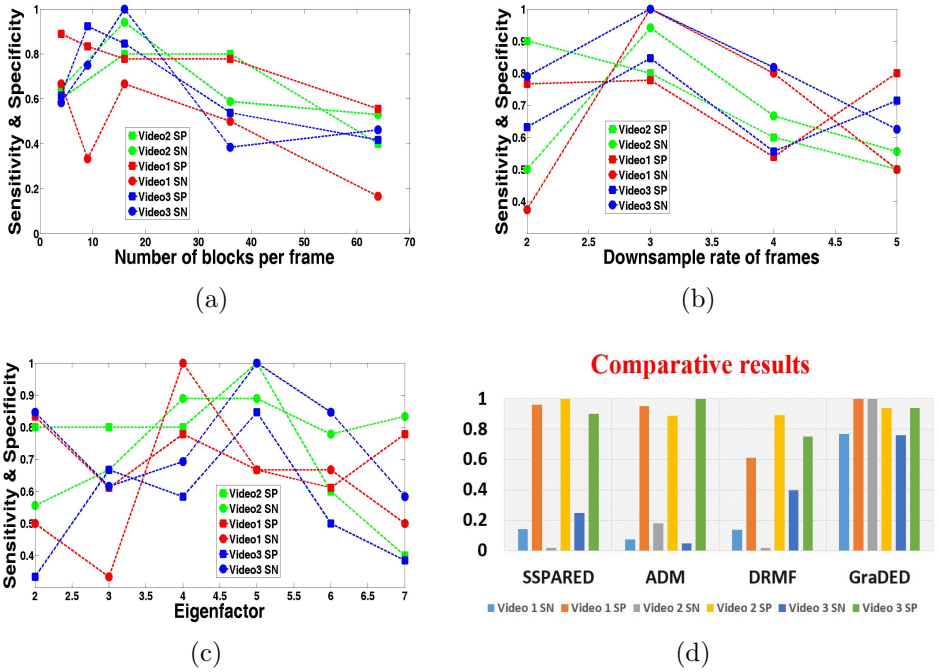
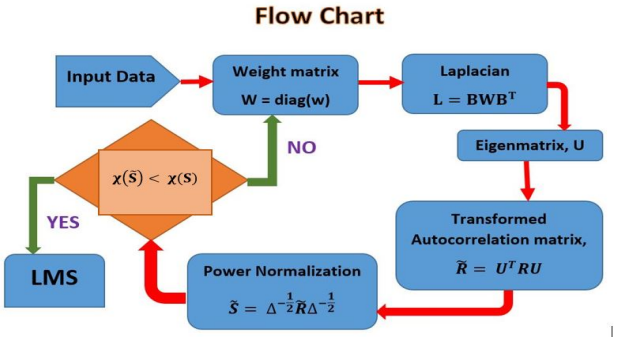


FIGURE 4.10: For three video datasets: sensitivity and specificity vs. (a) number of blocks per frame P, (b) downsample rate of frames and (c) eigenfactor; (d) comparison with state-of-the-art methods. SP = Specificity, SN = Sensitivity.

4.2 System preconditioning

In 1960, Bernard Widrow and Ted Hoff [123] proposed a class of *least mean squares* (LMS) algorithms to recursively compute the coefficients of an N-tap finite impulse response (FIR) filter that minimizes the output error signal. This computation is achieved by a stochastic gradient descent approach where the filter coefficients are evaluated as a function of the *current error* at the output. Two of the major issues with this approach are the convergence speed and stability. The filter coefficients (or weights) converge in mean while showing small fluctuation in magnitude around the optimal value. The convergence speed depends on the condition number of the autocorrelation matrix of the input, where a condition number close to unity connotes a fast and stable convergence. Later, adaptive algorithms, such as LSL (least square lattice) and GAL (gradient adaptive lattice) [124, 125] filters were designed to achieve faster convergence, immunity to poor condition number of input autocorrelation matrix, and better finite precision implementation compared to the LMS filter. However, these stochastic gradient filters may sometimes produce significant numerical errors, and the convergence is poor compared to recursive least squares (RLS) filters [126].

In order to obtain well-conditioned autocorrelation matrix of any real world input data, we transform the input *a priori*, which is popularly known as transform-domain LMS (TDLMS). The discrete Fourier transform (DFT), discrete cosine transform (DCT) [127–130] and others act as suitable off-the-shelf transformations of the input data for such problems. The aforementioned step is immediately followed by a power normalization stage [131, 132] and then used as input to the LMS filter. As a geometrical interpretation, the unitary transformation rotates the mean square error (MSE) hyperellipsoid without changing its shape on the axes of LMS filter weights [132]. The rotation



(a)

FIGURE 4.11: A schematic of our algorithm.

tries to align the axes of the hyperellipsoid to the axes of weights. The power normalization is crucial in enhancing the speed of convergence of LMS filter. The normalization forces the hyperellipsoid to cross all the axes at equal distance from the center of the hyperellipsoid. For a perfect alignment after the transformation, the normalization step turns the MSE hyperellipsoid into a hypersphere [132].

The TDLMS filter is flexible as it does not attempt to change the working principles and the architecture of LMS filter. Therefore, the transform-domain module can precede other algorithms, such as RLS, GAL and LSL. Notice that the conventional unitary transformations are independent of the underlying data, hence not optimal in regularizing condition numbers of the autocorrelation matrices of arbitrary real-time datasets. As an example, the DCT has been shown to be near-optimal for Toeplitz matrices. However, the DCT loses its near-optimality in conditioning sparse linear systems.

From a different perspective, the transformation of a matrix, such as autocorrelation matrix to improve the condition number is regarded as a subproblem of preconditioning of matrices [133–136]. Jacobi [137, 138], Gauss-Seidel [139], approximate inverse [140, 141], incomplete LU factorization [142] preconditioners are examples of such data-dependent transformations that utilize a decomposition of the input autocorrelation matrix. These algorithms are well-suited for solving linear system of equations.

Notice that there is a difference between TDLMS and linear systems in terms of the usage of a preconditioner. The preconditioning action is *implicit* in TDLMS filters. The

autocorrelation matrix is not explicitly used in the LMS architecture. Instead, it is the input data or the transformed input data (in case, we transform the data) that flows through the LMS lattice and the update of weights is based on error-correcting learning. The convergence of the algorithm depends on the input autocorrelation matrix. Whereas, in case of solving linear system of equations, $(Ax = b)$ type, the use of a preconditioner is *explicit*, which is $M^{-1}Ax = M^{-1}b$, with M as a preconditioner matrix, such as the Gauss-Seidel type.

Let A be a matrix to be preconditioned by another matrix ζ. ζ is said to be a left, right, and split preconditioner if $\zeta^{-1}A$, $A\zeta^{-1}$, and $U_1^{-1}AU_2^{-T}$ with $\zeta = U_1U_2^T$ respectively provide improved condition numbers compared to that of A. Gauss-Seidel, incomplete LU, and approximate inverse are examples of a left preconditioner. The transformations in TDLMS algorithm are the unitary split preconditioner type. By unitary, we have $\left(U_1U_2^T = I\right)$ and $U_1 = U_2 = U$.

4.2.1 Preconditioning using graph (PrecoG)

In PrecoG, we aim to learn such unitary split preconditioner from input data. It is because unlike a left or right preconditioner, the operation of a split preconditioner entails the possibility of a *transformation of input data* in which the input data vector is left-multiplied by U^T. It is the split preconditioner which helps apply PrecoG to TDLMS as well as solving a linear system of equations, and possibly any applications where preconditioner or transformation is needed. There are additional benefits of this approach. As the transformation is unitary, the energy of input data remains unchanged after transformation. In addition, the procedure of preconditioning that transforms the input vector as an intermediate step can utilize a number of input signal properties (e.g., the sparsity of a signal).

The derivation of our split conditioner is motivated by the topology of the structured input data. The topology determines neighborhood relationship between data points, which can be represented using graph-theoretic tools [103, 143, 144]. In recent years, manifold processing and regularization have shown promise in different areas of research [19,20,145,146]. Based on such evidence, we hypothesize that the intrinsic topology of the input data affects the construction of a suitable preconditioner matrix. We estimate a data manifold that provides an alternate set of basis acting as a split preconditioning matrix. The data when projected onto the basis are expected to be decorrelated.

The *main contribution* of this work is to provide an *optimization* framework that finds the desired unitary transformation for the preconditioning matrix. We iteratively estimate the underlying topology leveraging graph theory, followed by the computation of desired unitary transformation by using the graph Laplacian. Most importantly, we show that our approach is equally applicable in preconditiong arbitrary linear systems apart from ameliorating the convergence of LMS filters. In addition, PrecoG can be extended to exploit additional input data constraints, such as sparsity of the transformed input. Another advantage of PrecoG is that it can be applied without having a prior knowledge about the process that generates the input data.

4.2.1.1 Problem statement

Let $x_k = [x(k)\ x(k-1)\ \ldots\ x(k-N+1)]$ be a N-length real valued tap-delayed input signal vector at k^{th} instant. The vector representation is convenient for estimating the input autocorrelation as an ergodic process. Let the autocorrelation matrix, denoted by R_N be defined as $R_N = E\big(x_N x_N^T\big)$. We assume that Δ_Y is the main diagonal of a square matrix Y ($\Delta_Y = diag(Y)$). Following this notation, after power normalization the autocorrelation matrix becomes $S_N = \Delta_{R_N}^{-\frac{1}{2}} R_N \Delta_{R_N}^{-\frac{1}{2}}$. In general, the condition number of S_N, χ_{S_N}, happens to be significantly large in practical datasets. For example, the condition number of the autocorrelation matrix of a Markov process with signal correlation factor

as 0.95 has a χ of $\mathcal{O}(10^3)$. Notice that we seek U_N to minimize the condition number of S_N. Let a unitary transformation be U_N ($U_N U_N^T = I$) such that the transformed autocorrelation matrix becomes $\tilde{R}_N = E\big[U_N^T x_k x_k^T U_N\big]$. Next, $\tilde{R}_N$ is subjected to a power normalization stage that produces $\tilde{S}_N = \Delta_{\tilde{R}_N}^{-\frac{1}{2}} \tilde{R}_N \Delta_{\tilde{R}_N}^{-\frac{1}{2}}$. Precisely, we want the eigenvalues of $\lim_{N\to\infty} \tilde{S}_N \in \big[1-\epsilon_2,\ 1+\epsilon_1\big]$, where ϵ_1 and ϵ_2 are arbitrary constants such that $\chi_{max} \simeq \frac{1+\epsilon_1}{1-\epsilon_2}$. A schematic of our algorithm is given in Fig. 4.11.

Let us take an example of a 1^{st} order Markov input with the signal correlation factor ρ and autocorrelation matrix, R_N as

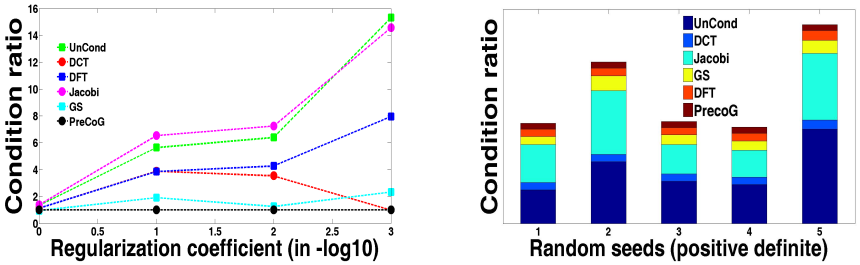
$$R_N = E[\boldsymbol{x}_k \boldsymbol{x}_k^H] = \begin{pmatrix} 1 & \rho & \cdots & \rho^{n-1} \\ \rho & 1 & \cdots & \rho^{n-2} \\ \vdots & \vdots & \ddots & \vdots \\ \rho^{n-1} & \rho^{n-2} & \cdots & 1 \end{pmatrix} \tag{4.18}$$

It is shown in [132] that $\chi_{S_N} \simeq (\frac{1+\rho}{1-\rho})^2$, which suggests that $\epsilon_1 = \rho^2 + 2\rho$, and $\epsilon_2 = 2\rho - \rho^2$. After applying the DFT, the condition number becomes $\lim_{N\to\infty} \chi_{\tilde{S}_N} = (\frac{1+\rho}{1-\rho})$, which indicates that $\epsilon_1 = \epsilon_2 = \rho$. On applying the DCT, $\lim_{N\to\infty} \chi_{\tilde{S}_N} = 1+\rho$ with $\epsilon_1 = \rho$ and $\epsilon_2 = 0$.

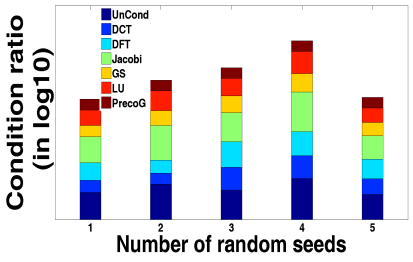
4.2.1.2 Methodology

The search for U_N is carried out through an iterative optimization of an associated cost function. It can be argued from section 4.2.1.1 that the optimal convergence properties are obtained when $\tilde{S}_N$ converges to the identity matrix in the *rank zero perturbation* sense [132]: A and B with $\eta = A - B$ have the same asymptotic eigenvalue distribution if

$$\lim_{N\longrightarrow\infty} rank(\eta) = 0. \tag{4.19}$$



(a) (b)



(c)

FIGURE 4.12: Condition ratios obtained by applying the algorithms on (a) regularized Hilbert matrices with varied regularization parameters, (b) a set of random matrices containing entries $\sim Gaussian(0, 1)$, and (d) random matrices of varied sparsities (for sparse linear systems).

In our case, with λ as an eigenvalue, this translates to

$$\lim_{N\to\infty} \det(\tilde{S}_N - \lambda \mathbb{I}_N) = 0, \tag{4.20}$$

which can be expanded as,

$$\lim_{N\to\infty} \det\left(\Delta_{\tilde{R}_N}^{-1/2} \tilde{R}_N \Delta_{\tilde{R}_N}^{-1/2} - \lambda \mathbb{I}_N\right) = 0. \tag{4.21}$$

Eq. (4.21) can be rearranged as,

$$\lim_{N\to\infty} \det\left(\tilde{R}_N - \lambda \Delta_{\tilde{R}_N}\right) = 0, \tag{4.22}$$

which is a quadratic polynomial of U_N as $\tilde{R}_N = U_N^T R_N U_N$. Given the orthonormality of the eigenvectors U_n, we can rewrite (4.22) as

$$U_N = \underset{U_N \in O(N)}{\arg\min} \left[det\left(\tilde{R}_N - \lambda \Delta_{\tilde{R}_N}\right)\right]. \tag{4.23}$$

Here, $O(N)$ is the set of unitary matrices. However, in presence of the determinant in (4.23), obtaining a closed-form expression for U_N is difficult to obtain. To overcome this obstacle, we apply the Frobenius norm in (4.24).

$$U_N = \underset{U_N \in O(N)}{\arg\min} \|\tilde{R}_N - \lambda\Delta_{\tilde{R}_N}\|_F^2. \tag{4.24}$$

The Frobenius norm imposes a stronger constraint compared to (4.20). In fact, while (4.22) can be solved if at least one column of $\tilde{R}_N - \lambda\Delta_{\tilde{R}_N}$ can be expressed as a linear combination of rest of the columns, (4.24) becomes zero only when $\tilde{R}_N - \lambda\Delta_{\tilde{R}_N}$ is a zero matrix. In effect, it reduces the search space of U_N. It is due to the fact that the set of U_N that solves eq. (4.24) is a subset of the U_N that are the solutions of eq. (4.23). Here, we address two aspects of the problem. First, (4.24) attempts to minimize the difference between $\tilde{R}_N$, which is $U_N^T R_N U_N$, and the scaled diagonal matrix of $\tilde{R}_N$. This is necessary because it accounts for the spectral leakage [132] as mentioned later in this section. The second aspect is that (4.24) seems to diagonalize $\tilde{R}_N$ apart from attempting to make the eigenvalues unity only. Here, a solution may be hard to obtain in practice. So, we relax the unity constraint by forcing the eigenvalues to lie within a range $[1-\epsilon_2,\ 1+\epsilon_1]$. By enforcing the constraint, (4.24) with parameters $p = (\boldsymbol{w}, \epsilon_1, \epsilon_2)$ becomes

$$U_N = \underset{U_N \in O(N)}{\arg\min} || \underbrace{\tilde{R}_N - s_+\Delta_{\tilde{R}_N}\|_F^2 + ||\tilde{R}_N - s_-\Delta_{\tilde{R}_N}\|_F^2}_{E(p)}. \tag{4.25}$$

where $s_+ = 1+\epsilon_1$ and $s_- = 1-\epsilon_2$ are the upper and lower bounds respectively for the eigenvalues of S_N. Using the Hadamard product notation, we can express $\Delta_{\tilde{R}_N}$ as $U_N^T R_N U_N \circ \mathbb{I}_N$. The first two constraints in (4.25) provide a valley in the space spanned by the eigenvectors in U_N if R_N is positive definite. The valley exists between two surfaces $(1-\epsilon_2)U_N^T R_N U_N \circ I$ and $(1+\epsilon_1)U_N^T R_N U_N \circ I$. At this point, there might be infinitely-many possible solutions. We add a regularizer on $\boldsymbol{w}$ to obtain an acceptable set of solutions.

The undesired result of the imposed restriction given above is that the convergence time may be significant, and the set of solutions of (4.25) in terms of U_N is significantly smaller than that of (4.24). Upon approaching a minimum of (4.24), the speed of gradient descent algorithm drops significantly. Although it is theoretically expected that $\lim_{N\to\infty} \tilde{S}_N \in [1-\epsilon_2,\ 1+\epsilon_1]$, in practice, it is difficult to guarantee after a prescribed number of iterative steps.

4.2.1.3 Laplacian parametrization

The problem of finding a sub-optimal transform by optimizing eq. (4.25) is solved by leveraging the graph framework. In this framework, the input data is modeled with a finite, single-connected, simple, and undirected graph endowed with a set of vertices, edges and edge weights. For example, for an LMS filter with N taps, an input signal vector x_k has length N, which can be represented with N vertices. Basically, each vertex corresponds to one tap of the LMS filter. Using the graph, the unknowns of the optimization in (4.25) are the number of edges and the associated edge weights. A fully-connected graph with N vertices contains $\frac{N(N-1)}{2}$ edges. A deleted edge can be represented with zero edge weight. We denote the the set of unknown parameters as $\boldsymbol{w}$, which is the set of nonzero weights of the graph.

To find the desired transformation, the algorithm initializes the weights $\boldsymbol{w}$ with random numbers sampled from a Gaussian distribution with zero mean and unit variance. Let W is the diagonal matrix containing $\boldsymbol{w}$. Then by definition, the graph Laplacian, which is symmetric and positive semidefinite by construction, is given by $L = BWB^T$. B is the incidence matrix as mentioned in section 2.1. The spectral decomposition of L provides the matrix of eigenvectors U. Finally, $U^T x_k$ is the transformation that is expected to decorrelate the dataset, which may not be possible due to random initialization. Then the cost function (4.25) helps update the weights and the search for the

desired transformation continues in an iterative fashion until the objective conditions are met.

The framework provides a couple of advantages. Firstly, in order to obtain U directly from eq. (4.25), iterative estimation of N^2 parameters is needed with the unitary constraint, which might result in overfitting. Moreover it does not seem reasonable to make U sparse. Sparsification is justified in the graph domain, where the sparsifying action leads to $\mathcal{O}(N)$ number of edges and edge weights. A smaller set of weights helps prevent the risk of overfitting during optimization.

The cost function in eq. (4.25) is nonconvex. Therefore, the solution is not guaranteed to be a global optimum. In our work, the required solution is obtained through gradient descent with μ as the step size parameter. We also obtain that $L = BWB^T$. Let $\Theta_i = \frac{\partial L}{\partial w_i}$, which can be evaluated as $\frac{\partial L}{\partial w_i} = B\frac{\partial W}{\partial w_i}B^T$. Using μ and Θ_i, the update equation is given by,

$$\begin{aligned} w_i^{t+1} &= w_i^t - \mu Tr\Big(\big[\frac{\partial E(p)}{\partial U_N}\big]^T \frac{\partial U_N}{\partial w_i}\Big); 0 < \mu < 1 \\ \frac{\partial u_{k,l}}{\partial w_i} &= Tr\Big(\frac{\partial u_{k,l}}{\partial L}\Theta_i\Big) = Tr\Big(\big[\frac{\partial L}{\partial u_{k,l}}\big]^{-T}\Theta_i\Big), \end{aligned} \tag{4.26}$$

where, by using $J^{mn} = \delta_{mk}\delta_{nl}$, $\frac{\partial L}{\partial u_{kl}}$ can be given by,

$$L = U_N \Gamma U_N^T \Longrightarrow \frac{\partial L}{\partial u_{kl}} = U_N \Gamma J^{mn} + J^{nm}\Gamma U_N^T. \tag{4.27}$$

In eq. (4.26), t is the iteration index, and the computation of $\frac{\partial E(p)}{\partial U_N}$ is given in the Appendix. To prevent each w_i from erratic values during iteration, we impose $2-$norm on the weight vector $\boldsymbol{w}$. On adding the regularization term to eq. (4.25), the new cost function becomes

$$E_N(\boldsymbol{w}, \epsilon_1, \epsilon_2, \beta) = E(p) + \beta(\boldsymbol{w}^T\boldsymbol{w} - 1). \tag{4.28}$$

On differentiating E_N with respect to $\boldsymbol{w}$, we get

$$\frac{\partial E_N}{\partial \boldsymbol{w}} = \frac{\partial E(p)}{\partial \boldsymbol{w}} + 2\beta \boldsymbol{w} \tag{4.29}$$

Following eq. (4.29), the iterative update of each weight can be given by,

$$w_i^{t+1} = w_i^t\big(1 - 2\beta\big) - \mu Tr\Big(\big[\frac{\partial E(p)}{\partial U_N}\big]^T \frac{\partial U_N}{\partial w_i}\Big). \tag{4.30}$$

4.2.1.4 Complexity analysis

The update of weights w_i requires the computation of three partial derivatives (see Appendix 4.2.1.6.1). Notice that $\frac{\partial L}{\partial u_{kl}}$ and $\frac{\partial L}{\partial w_i}$ are significantly sparse. This can be seen from (4.27), where J^{mn} is a matrix containing only one nonzero entry of magnitude unity. Using the fact that Γ is a diagonal matrix containing eigenvalues of L, $\frac{\partial L}{\partial u_{kl}}$ in (4.27) contains at most $(2N-1)$ nonzero entries out of N^2 entries, where N^2 is the dimension of $\frac{\partial L}{\partial u_{kl}}$. This sparse matrix is singular by construction, and the Moore-Penrose pseudoinverse ($\mathcal{O}(N^3)$) is computed in order to obtain $\big(\frac{\partial L}{\partial u_{kl}}\big)^{-1}$. It can also be observed that $\frac{\partial L}{\partial w_i}$ is sparse. This sparsity is due to the fact that $\frac{\partial L}{\partial w_i} = B\frac{\partial W}{\partial w_i}B^T$, and each weight w_i appears in exactly four entries of L. It asserts that $\frac{\partial L}{\partial w_i}$ has four nonzero entries out of N^2 entries in $\frac{\partial L}{\partial w_i}$. Let $L(k,l) = w_i;\ 1 \leq k,l \leq N$. Then, $\Theta_i(k,k) = \Theta_i(l,l) = 1$ and $\Theta_i(k,l) = \Theta_i(l,k) = -1$, which implies that only k^{th} and l^{th} columns of Θ_i contain nonzero entries. Therefore, $Tr\Big(\big[\frac{\partial L}{\partial u_{k,l}}\big]^{-T}\Theta_i\Big)$ can be given by the sum of (k,k) and (l,l) entries in $\big[\frac{\partial L}{\partial u_{k,l}}\big]^{-T}\Theta_i$. The lone computationally expensive operation is computation of the pseudoinverse $\big(\frac{\partial L}{\partial u_{kl}}\big)^{-1};\ 1 \leq k,l \leq N$, which incurs in total a complexity of $\mathcal{O}(N^5)$ for each weight. It is to note that the constructions of preconditioners for solving linear systems by comparative methods such as, Jacobi ($\mathcal{O}(N^2)$), successive over-relaxation (SOR) ($\mathcal{O}(N^3)$), symmetric SOR ($\mathcal{O}(N^3)$), Gauss-Seidel ($\mathcal{O}(N^3)$) have faster associated

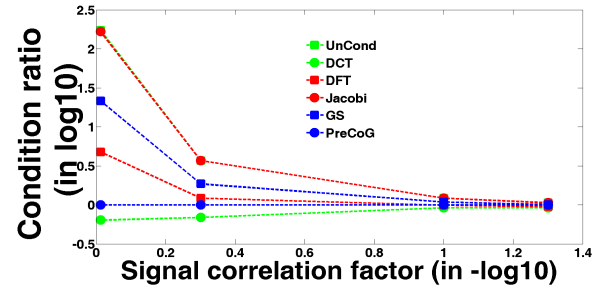
run times compared to PrecoG. Here, complexity accounts for the inversion of each preconditioner matrix. However, the acceleration in the convergence of the LMS filter using PrecoG is also expected to partially compensate for the computational overload of PrecoG.

4.2.1.5 Sparse signal and sparse topology estimation

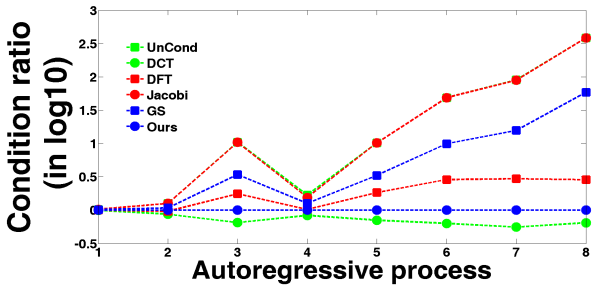
For an input signal vector of length N, the graph can be fully connected with $\frac{N(N-1)}{2}$ edge weights ($\mathcal{E}$) , which implies a dense topology. However, dense topology may lead to overfitting during the optimization using gradient descent in (4.26). We employ a fixed regular topology by setting the connectivity among vertices such that $e_{ij} \in \mathcal{E}$ exists if $0 < |i-j| \leq 2$. This fixed topology maintains the temporal order in which the data point arrives and keep the topology sparse. The estimation of the sparse topology [104] along with the constraint of sparsity of the transformed signal can also be achieved by adding a regularization term to (4.28) as

$$E_N(p, \beta, \alpha_1, \alpha_2) = E_N - \alpha_1 \mathbf{1}^T log(A\mathbf{1}) + \alpha_2 ||U_N^T x_k||_0. \tag{4.31}$$

Here, A is the adjacency matrix consisting of $\boldsymbol{w}$, and $A\mathbf{1}$ is the degree vector containing weighted degree of all the data points. By penalizing high degree vertices (data points) the *log* penalty term promotes sparsity in A. On the other hand, it strongly discourages any vertex to have degree zero, maintaining the graph to be single connected. In (4.31), $||U_N^T x_k||_0$ regulates the sparsity of the transformed signal. Such a sparse signal involves reduced multiplication with filter coefficients in the LMS filter than a non-sparse signal, which saves significant computation time. This feature can not be obtained by applying conventional transforms, such as DCT.



(a)



(b)

FIGURE 4.13: Condition ratios obtained by applying the algorithms on (a) 1^{st} order Markov process as a function of signal correlation factor ρ and (b) 2^{nd} order autoregressive process with parameters (ρ_1, ρ_2).

4.2.1.6 Results

We show the effectiveness of our approach in preconditioning different matrices against the preconditioners - DCT, DFT, Jacobi (tridiagonal matrix type) and GS ($Gauss-Seidel$). In addition, for sparse linear systems, we include a comparison with the incomplete LU factorized preconditioning procedure. It is not reasonable to compare our results with LSL, GAL, and RLS because they are the improvements of the traditional LMS algorithm. To represent the strength of an individual algorithm, we incorporate the condition number of each unconditional matrix with the aforementioned methods. In order to scale the condition numbers obtained from several methods with respect to ours, we define a metric, *condition ratio* $= \frac{\text{condition number obtained from a method}}{\text{condition number obtained from our method}}$. In some of the results, we compute log_{10}(condition ratio) to mitigate the enormous variance present in the condition ratio scores. First, we apply our method to precondition a Hilbert matrix [147] which is severely ill-conditioned. Hilbert matrix, H is defined as $H(i,j) = \frac{1}{i+j-1}$. In the experiment, we add a regularizer using $\left(\alpha\mathcal{I}\right)$ with $0 < \alpha \leq 1$ as the regularization

coefficient. The condition ratios of the existing algorithms including PrecoG on preconditioning the Hilbert matrices, which are regularized by changing the α, are shown in Fig. 4.12(a). Notice that the X-axis is given in $-log_{10}$ scale. Therefore, smaller values at X coordinate indicates higher regularization of the Hilbert matrix. On increasing the value of α, the Hilbert matrix becomes diagonally-dominant, and it suggests that the condition number of the Hilbert matrix approaches unity. This fact is clearly visible from Fig. 4.12(a), where all the state-of-the-art techniques including PrecoG performs significantly well when the value of $-log10(\alpha)$ falls in the vicinity of zero. However, on decreasing the value of α, the Hilbert matrix becomes severely ill-conditioned, and the performance of the competitive algorithms except $Gauss - Seidel$ exhibit inconsistent behavior. The DCT performs better near $\alpha = 1$ ($-log10(\alpha) = 0$). This result is expected because diagonally dominant matrix behaves similar to a 1^{st} order Markov process with ρ significantly small. However, the DCT shows inconsistency at $\alpha = 0.001$, where it again attains noticeable preconditioning of Hilbert matrix as opposed to other competitive methods.

We also evaluate our algorithm on five different random positive definite matrices with the values taken from a zero-mean and unit-variance Gaussian process. We regularize the matrices to ensure positive-definiteness. The results in Fig. 4.12(b) are bar-plotted. For example, the first bar corresponds to the results on the first random Gaussian positive definite matrix. Each bar is shown using six different colors corresponding to six methods. The width of each color is proportional to the condition ratio obtained by applying the corresponding preconditioning method. The condition numbers of the five positive definite matrices are 23.6, 31.8, 28.1, 26.5, and 269.1 respectively. The condition numbers that we obtain by applying PrecoG are 4.3, 3.1, 4.1, 4.2, and 17.5 respectively. It is evident from Fig. 4.12(b), the condition ratios obtained by applying PrecoG are at least 1.3 times better compared to DCT, and effectively well compared to Gauss-Seidel, DFT,

and Jacobi.

The condition ratios (in $log10$ scale) by applying PrecoG on sparse systems of equations are shown in Fig. 4.12(c). The five sparse matrices are random by construction with sparsity $\left(\frac{\text{number of nonzero elements}}{\text{total number of elements}}\right)$ levels as $[\frac{5}{6}, \frac{2}{3}, \frac{1}{3}, \frac{1}{2}, \frac{1}{5}]$ respectively. For example, $\frac{5}{6}$th of all the entries in the first matrix which is symmetric and positive definite by construction, are nonzero. We present the experiments in decreasing order of sparsity with the most sparse one, in our experiment, containing only $\frac{1}{5}$th nonzero elements. The condition numbers of these matrices are 138.68, 1620, 22.88, 312.3, and 51.84. On applying PrecoG to these matrices, we obtain the corresponding condition numbers as 54.6, 491.94, 8.29, 81.8, and 22.3. From Fig. 4.12(c), it can be seen that PrecoG improves the condition numbers at least twice compared to DCT, DFT, Jacobi, incomplete LU, and at least 1.4 times compared to Gauss-Seidel method.

Comparison on 1^{st} order Markov process and 2^{nd} order autoregressive process:

The DCT is known as the near-optimal preconditioner for 1^{st} and 2^{nd} order Markov processes. The purpose of this section is to compare the preconditioning strength of our algorithm with that of DCT. We also include the strength of other methods as shown in the following figures.

First, we consider the autocorrelation matrix of a first order Markov process of signal correlation factor ρ $(0 \leq \rho \leq 1)$ [132]. The autocorrelation matrix, $R_N(\rho)$ of such process is given in (4.18), which has a Toeplitz structure. Higher values of ρ indicate stronger correlation among data samples, which implies significantly higher eigenvalue spread of the autocorrelation matrix. Smaller ρ values imply weaker correlation among data instances, which is reflected in the diagonally-dominant structure of the input autocorrelation matrix R_N. It has been proved that the DCT is a near-optimal unitary transformation for this process. We compare our method against the DCT as presented in Fig. 4.13(a). Unlike other methods, our result exhibits a consistent behavior with respect to the DCT

over different values of ρ. The fact that the condition ratios of other methods are non-linearly increasing as the signal correlation factor approaches unity confirms the weaker performance of those methods compared to PrecoG.

In Fig. 4.13(b), we present the condition ratios computed by applying the algorithms on the autocorrelation matrices of eight 2^{nd} order autoregressive process with parameters (ρ_1, ρ_2) [127]. The input autocorrelation matrix R_N of such process is given by $R_N = c_1 R_N(\rho_1) + c_2 R_N(\rho_2)$. $R_N(\rho_1)$ and $R_N(\rho_2)$ are two Toeplitz matrices, similar to R_N of 1^{st} order Markov process. c_1 and c_2 are constants and are given by $c_1 = \frac{\rho_1(1-\rho_2^2)}{(\rho_1-\rho_2)(1+\rho_1\rho_2)}$, $c_2 = \frac{-\rho_2(1-\rho_1^2)}{(\rho_1-\rho_2)(1+\rho_1\rho_2)}$. The values of the parameters (ρ_1, ρ_2) considered in the experiments are $(0.015, 0.01)$, $(0.15, 0.1)$, $(0.75, 0.7)$, $(0.25, 0.01)$, $(0.75, 0.1)$, $(0.9, 0.01)$, $(0.95, 0.1)$, and $(0.99, 0.7)$. Although the DCT provides a near-optimal option when the autocorrelation matrix is Toeplitz in nature, our approach is consistent and almost approximates the DCT, while the others significantly deviate from the DCT in terms of the condition ratios.

Comments: With the condition ratio scores by the existing preconditioning techniques on different datasets at hand, it can be observed that the preconditioning capability of each method is essentially dataset-specific. For example, the DCT works significantly well for the Markov process, autoregressive process, whereas Gauss-Seidel exhibits better performances on preconditioning the *Hilbert* matrix and sparse linear systems. In all the aforementioned datasets except Markov process and autoregressive process, PrecoG shows its efficacy over other methods. For 1^{st} order Markov process, 2^{nd} order autoregressive process, PrecoG approximates the DCT in terms of the condition ratios. Therefore, it can be argued that if the input process is not known *a priori*, PrecoG can reduce the condition number and should be the preferred choice.

4.2.1.6.1 Performance by changing parameters Next, we look at the performance of our algorithm in terms of condition ratio by varying a set of internal parameters - initialization of weights, number of iterations and length of the input vector. Fig. 4.14(a) shows the behavior of condition numbers for aforementioned datasets with five different initial weights, $\boldsymbol{w}$. It is evident that PrecoG yields better performance in the cases of Markov and autoregressive inputs compared to the remainder of the cases considered here. During experimentation, we obtained a few instances where PrecoG gives a substantial improvement not possible with existing techniques. For example, PrecoG achieves a condition number 10.64 in case of the Hilbert matrix as shown in Fig. 4.14(a).

Fig. 4.14(b) exhibits the result of the preconditioning capacity of PrecoG over iteration. For each dataset, we randomly initialize the weights and keep the input vector length as 10. There is a gradual decline in the trend of condition ratio for each dataset, which implies that PrecoG improves preconditioning over iteration.

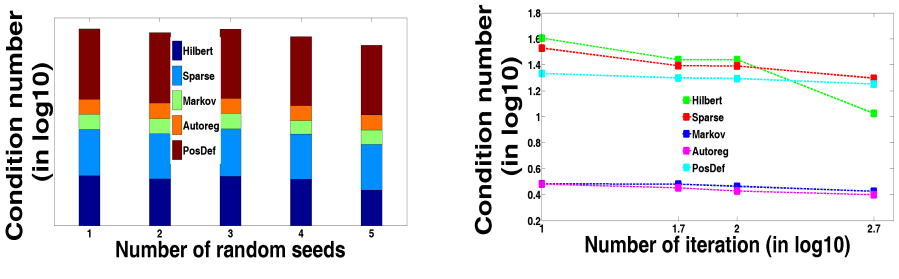
Fig. 4.14(c) illustrates the consistent performance of PrecoG on the length of input vectors. There is a positive slope of the condition ratio with increments in the length of input vector. However, the results are shown using 15 iterations only. It is observed that PrecoG with longer input vector (larger graph) needs linearly more iterations to output better condition ratio.

Appendices

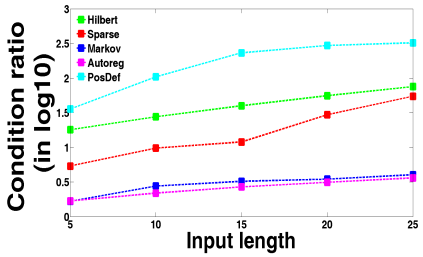
Evaluation of $\left[\frac{\partial E(p)}{\partial U_N}\right]$

Let, $M(\epsilon) = ||U^T RU - (1+\epsilon)U^T RU \circ \mathcal{I}||_F^2$. Then,

$$\begin{aligned} M(\epsilon) &= Tr\Big(\{U^T RU - (1+\epsilon)U^T RU \circ \mathcal{I}\} \\ &\quad \{U^T RU - (1+\epsilon)U^T RU \circ \mathcal{I}\}^T\Big). \end{aligned} \tag{4.32}$$



(a) (b)



(c)

FIGURE 4.14: The performance of our algorithm on (a) different initialization of weights, $\boldsymbol{w}$, (b) the number of iterations for gradient descent, and (c) the length, N of input signal vector.

Eq. (4.32) on expansion gives

$$\begin{aligned} M(\epsilon) &= Tr\Big(U^T R^2 U - 2(1+\epsilon)(U^T RU \circ \mathcal{I})(\\ &\quad U^T RU) + (1+\epsilon)^2 (U^T RU \circ \mathcal{I})^2\Big). \end{aligned} \tag{4.33}$$

Eq. (4.33) is obtained using the fact that $UU^T = \mathcal{I}$. By performing the partial derivative,

$$\begin{aligned} \frac{\partial M(\epsilon)}{\partial U} &= \frac{\partial Tr(U^T R^2 U)}{\partial U} - 2(1+\epsilon)\frac{\partial}{\partial U} Tr\{(U^T RU \circ \mathcal{I}) \\ &\quad (U^T RU)\} + (1+\epsilon)^2 \frac{\partial}{\partial U} Tr(U^T RU \circ \mathcal{I})^2. \\ &= 2R^2 U - 4(1+\epsilon)RU + (1+\epsilon)^2 (U^T RU \circ \mathcal{I})RU \end{aligned}$$

Next, $\frac{\partial E(p)}{\partial U}$ is computed using $\frac{\partial M}{\partial U}$ as $\frac{\partial E(p)}{\partial U} = \frac{\partial M(+\epsilon_1)}{\partial U} + \frac{\partial M(-\epsilon_2)}{\partial U}$.

$$\begin{aligned} \frac{\partial E(p)}{\partial U_n} &= 2\Big[2R_n - 2(2-\epsilon_2-\epsilon_2)\mathcal{I} - \{(1+\epsilon_1)^2 + \\ &\quad (1-\epsilon_2)^2\} U_n^T R_n U_n \circ \mathcal{I}\Big] R_n U_n. \end{aligned} \tag{4.34}$$

If $\epsilon_1 = \epsilon_2$, the above expression can be simplified as

$$\begin{aligned} \frac{\partial E(p)}{\partial U_n} &= 4\Big(R_n - (1-\epsilon)\mathcal{I} - (1+\epsilon^2) \\ &\quad U_n^T R_n U_n \circ \mathcal{I}\Big) R_n U_n \end{aligned} \tag{4.35}$$

Discussion:

In this letter, we present a method to obtain a unitary split preconditioner by utilizing nonconvex optimization and graph theory. We demonstrate the efficacy of our approach over prevalent state-of-the-art techniques. In addition, we show that our algorithm is amenable to precondition linear systems constrained with sparsity. As a future endeavor, we will attempt to exploit the signal structure and embed this structure into the optimization framework by including a set of constraints. In continuation, we will try to extend our approach to solve a sparse underdetermined linear system of equations in order to implement dictionary learning.

Chapter 5

Conclusion, current & future work

In 1736, Leonhard Euler proposed a solution of the seven bridges of Königsberg, which laid the foundations of graph theory. Graph theory is such a powerful tool that it serves as the bedrock for numerous scientific problems. The popularity of graphs even transcended to the domain of arts. Georges Seurat, Paul Signac developed a revolutionary style of neo-impressionistic art, called pointillism, in which repeated dots of color and cohesive placements of the dots as opposed to having continuous brush strokes create a portrait. These painters were creating a signal by place graph vertices instead of continuous tones.

Over the last 200 years, there has been extensive research and theoretical developments in combinatorial graph theory and graph topology. Using such tools, in the last two decades, we also witness the widespread use of graphical models for probabilistic inference, and of complex network models in several real life problems, such as modeling the spread of disease, evacuation, and biosphere networks. However, suitable modeling of a data with graph is a difficult problem, and the quality of downstream analysis entirely depends on how the parameters of a graph are selected. There is no universal algorithm that can select the graph parameters of an arbitrary dataset, demanding the need for case by case analysis. In this thesis, we select problems with varying degrees of the structural complexity of graph models, and propose several algorithms to meet certain objectives.

Although, it is impossible to consider all the problems and investigate the construction and application of graph models, the modeling and analysis of our selected problems can be seamlessly transferred with minor modifications to numerous problems in other domains. Before discussing about the strength, weakness, and potential application areas of our proposed algorithms, we first intend to provide the inferences that can be drawn from the thesis. The inferences are listed below.

(**I1**) Graph model of a problem can provide various combinatorial features that can be exploited in several applications. In many instances, the graph models provide compact and abstract representations, and computationally-efficient, low-storage, scalable algorithms can be designed on the graphs. The model offers significant degrees of freedom in the selection of the associated parameters, and the downstream analysis depends on the initialization of such models. For example, in order to perform geomorphological event detection, by selecting a nearest graph model and the covariance of features LaWeCo achieves only spatial localization. To capture subtle temporal changes in displacements for event monitoring purpose, we resort to a different graph model in DDT.

(**I2**) One of the principal factors behind the complexity of graph models is the nature of data acquisition. If the data already has a graph structure with fixed connectivity and fixed number of vertices, the graph structure is comparatively simple. We witness this fact in case of Kinect data.

If one of the parameters, let us assume the number of vertices, varies with data, the graph model exhibits enormous complexity. We observe this variation in case of 3D reconstructed neurons. There are graph–theoretic problems, such as subgraph matching, subgraph isomorphism, and vertex cover that comes with such graph models. In case of neurons, a relevant problem would be: if a neuron is traced by two different tracers with different number of samples, how to know that these

two traced neurons are actually the same one? The problem is termed as the *graph sparsification* problem and the solution in case of the unweighted version of graphs, as proposed by Nikhil Srivastava and Daniel Spielman, has a deep-rooted connection with the famous Kadison-Singer problem. Unfortunately, we use the weighted tree model, in which graph sparsification is computationally difficult. Our proposed algorithms, NeuroSoL and ElasticPath2Path are significantly affected by the discrepancy in the number of samples. However, in neuroPath2Path, we use only the bifurcation locations for feature construction, which majorly eliminates the above problem.

When both parameters, vertices and edges, are either not fixed or not defined properly, the complexity of the graph model rises rapidly. This is the case with video based datasets, where a pixel, a superpixel, a patch, or a frame can be regarded as a vertex. There are two different avenues to tackle such problem. In the first one, the vertices are fixed manually and the connections are defined with a criteria, such as full connectivity, K-nearest neighborhood and others. In the other avenue, the vertices are manually chosen and the connections among the vertices are found by optimizing a cost function that encodes certain properties, such as smoothness of the data.

(**I3**) The selection of auxiliary structures that are derived from the graph models plays important roles in diverse tasks that includes feature discrimination, encoding dynamics, and incorporating biological features. The combination of graph and its complementary graph has been shown to be effective for encoding the structural and geometrical features of a neuron (NeuroSoL, NeuroPath2Path). Complementary and bipartite graphs are useful for capturing event dynamics (DDT, UGrAD). Linear graph is used where the information of sequence needs to be preserved (GraDED, UGrAD).

(**I4**) Suitable selection of a model can integrate different tasks that can be performed using features from the same model. For example, in neuromorphology, the task of classifying different neuronal cell types can be accomplished using several methods, such as topological morphological descriptor [69], caulescence [92], NeuroSoL [79], and BlastNeuron [78]. However, the augmentation of these models to other problems, such as modeling of structural degeneration of neuronal arbors, neurogenesis, glia surveillance pattern is difficult. Our proposed method NeuroPath2Path offers such flexibility and it has the potential to be used to build cell-specific informatics.

In the following section, the advantages, disadvantages and scope of our proposed algorithms are discussed.

5.1 Current and future work

Graph based data modeling stitches an unthinkably broad spectrum of data into the problem of constructing an abstract representation based on the application demand. Although we categorize the applications based on whether the data inherit graph structures or not. There might exist other ways to catalogue them. Each problem that we discuss has its own uniqueness that is reflected in its graph parameters. It would be wise to unravel the possibilities of each one of the applications.

5.1.1 Neuromorphology

While the shape based categorization is an active area of research now a days, especially after the introduction of Neuromorpho.org, Linking region-based anatomical variation to functional heterogeneity is still not actively pursued. As we mention in Section 3.2, the pyramidal cells shows significant structural variation with the associated regions, the

question becomes how to link appropriate functions to the neurons in a specific region. Does there exist a 'functional gradient' of the neurons as a function of regions? or equivalently, if one takes two samples of pyramidal cells, one from the prefrontal cortex and the other from the primary visual cortex of mice, can we identify a relationship between the functional difference with the structural difference so that when we take two arbitrary samples of neurons, we can quickly associate their functional differences just by computing the structural variation in terms of morphometrics?

The study of the Connectome does not necessarily answer this question. It is because of the fact that in the Connectome, researchers are interested in looking for patterns among the giant connections of nearly 100 billion neurons. In the Connectome, each neuron is considered as a vertex, overriding the possibility to focus on the shape and synaptic connections and plasticity of each neuron. NeuronPath2Path is definitely well-suited for incorporating features for the modeling of synaptic connections, and for modeling plasticity by path rearrangements. However, to address the region-based functional mapping, we need more path-based features to pinpoint such functional variability.

The availability of single-cell transcriptomics data in understanding gene expression and regulation revolutionizes the field of computational genomics. Once the functional variability as a function of structural heterogeneity is estimated, the next goal would be to find the correspondence between the gene expression (transcriptomics) and the functional variation of neurons. The integration of transcriptomics data with the 3D reconstructed neuron data would be a challenging but exciting avenue in future as, if is possible, it will bridge two entities with significant difference in scale (Cellular morphology in μm and transcriptomics in nm).

Neuroscientists are also interested in the neurogenesis, where a neuron progenitor cell differentiates into successive cell-states and finally, becomes a mature neuron with

synaptic connections. There are lot of unanswered questions involved: what genes precisely drive the decision of a branch sprouting from a vestige of a neuron? how structural patterns in terms of morphometrics are affected by the transcriptomics? what is the cost of growing a dendritic or axonal branch? In NeuroPath2Path, the costs of splitting, growing, and terminating a branch are taken as unity, which is not true in biological settings. A neuron generates and rearranges its branches in a cost effective way, and an ad hoc mathematical formula or algorithm tweaks without interpreting cellular factors, such as cellular metabolism would be inappropriate in this context.

As discussed in Section 3.2, NeuroPath2Path can be tuned to model structural degeneration of neuronal branches. However, we need to customize new features, such as spine density, branch length, branch volume, number of clumps, clumps density per branch segment and related others. Once the feature customization is completed, the continuous morphing is expected to reveal the progress of degeneration. Structural degeneration of neuronal branches in neurodegenerative diseases also demands suitable model that can identify the genetic principle behind such phenomenon, a problem that lures the molecular biologists.

On a different note, ElasticPath2Path, NeuroSoL, NeuroBFD, NeuroPath2Path can also be applied to cell types with tree-type ramified branches with minor modifications of the associated features. Examples of cell types include microglia, astrocytes.

Microglia, also called as the 'Brain Police' [148], generally exhibit two distinct shapes, which are ramified (homeostasis) during surveillance, and amoeboid (during pathogen invasion and phagocytosis). There are unanswered questions regarding these glia. How does a microglia cell perform surveillance? how does it change shape by coiling up from the ramified to amoeboid? What are the structural differences of microgial morphology based on the regions, such as striatum and cerebellum ? It is found that the microglia at the striatum have strong homeostatic surveillance profile, whereas the microglia cells

at the cerebellum have high clearance or phagocytosis profile [149–152]. Does there exist structural variation to answer such functional differences? NeuroPath2Path can be certainly tuned to extract relevant features for glia cells, and used for identifying morphological cues. For example, to quantify the surveillance profile of a microglia cell, one can extract features, such as the average relative distance among the terminals of glia processes (equivalently paths in a neuron), the retraction and elongation rates of each path, and the average bifurcation angle. Tracking glia processes during surveillance would be identical to the continuous morphing between the processes (paths).

NeuronPath2Path have potential applications beyond the field of biology. Full-graph matching algorithm by nature, NeuroPath2Path can be deployed to measure the similarity between any two tree-structured networks. If a network does not inherit tree-structure, one can extract spanning trees and then, compare two spanning trees from two different networks. Owing to its path (walk) based construction, NeuroPath2Path can be augmented and employed as a graph-kernel in several problems.

Lastly, there is a trend that needs to be addressed in the context of feature extraction in machine learning. The practice of feature customization is gradually taken over by the deep learning and transfer learning based features. However, a feedforward neural net is still basically a mystical black box. Despite various tricks and techniques, a principled approach to design a neural net is yet to be developed. Graph signal processing based tools can provide some direction regarding the design methodology. In short, we can couple certain biological traits, such as potentiation, path rearrangement with graph theoretic features in a quest for progressive neural network design.

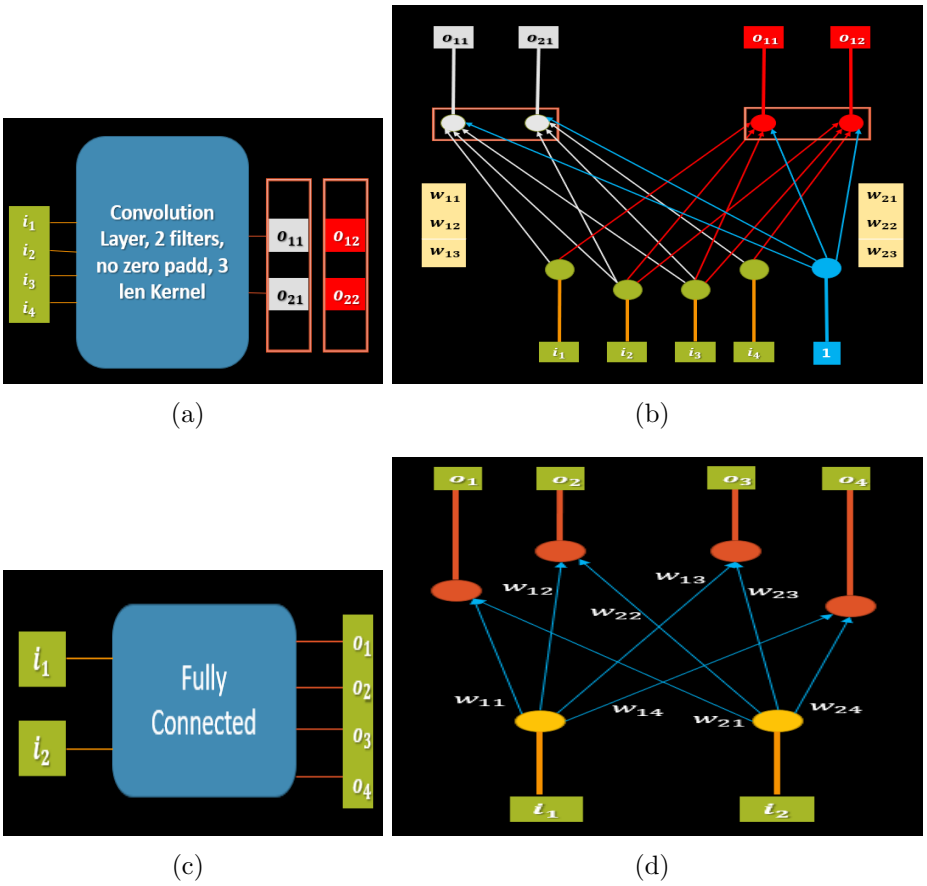
Before jumping to an example of such graph based learning procedure, we can summarize that there is a huge potential of our work on neuromorphology and the work offers a simple framework that can be applied into diverse fields.

5.1.1.1 An example: Graph based learning

The problem that we are seeking answers to is to find a way to learn with fewer labeled data (Learning with Less Labels – LwLL). The majority of datasets are notoriously class-imbalanced, which imposes several restrictions on the underlying models. In effect, the training schedules and various parameters of an architecture in such a case are restricted. To resolve this problem, popular techniques such as data augmentation have been utilized extensively. However, it has been argued that data augmentation is unrealistic and biologically implausible in many scenarios, especially in the context of biological/biomedical datasets.

In contrast to the conventional approaches to solve the problem of LwLL, such as few-shot learning, one-shot learning, and semi-supervised learning, in which either each exemplar is provided with a series of manual labels, or the classifier (or regressor) architecture is accomplished with *ad hoc* techniques, or features are borrowed from pre-trained machines, we investigate the problem of learning with fewer labels through the lens of developmental psychology. The inclusion of techniques inspired by how our brain works to designing a classifier generally improves the performance of the classifier. Instead of transferring the learned features, employing functions similar to the fundamental working principles of human brain to machines may reap significant benefits. For example, in 2018, authors [153] imposed Hebbian plasticity, mimicking synaptic plasticity in human brains, to train a recurrent neural network with millions of parameters, yielding significant improvements in meta-learning tasks.

From developmental psychology, we know that roughly 100 billion neurons progressively innervate and establish synapses based on the external stimuli, the majority of which are vision-based. According to the Swiss psychologist Jean Piaget [154], within six months, a child has an early numerical cognition on adding and subtracting numbers, and within two years (the *sensorimotor* stage), a child is able to manipulate 3D objects and



(a) (b)

(c) (d)

FIGURE 5.1: (a) A Convolution layer and its (b) graph signal representation. (c) A dense layer with 2D input and 4D output. (d) Graph representation.

lay out long-term planning with causal reasoning. Between 2 to 7 years of age, a child develops memory and imagination such that the child can distinguish an unknown object from the learned ones with a simple glimpse. The progressive development of sensory ecology dictates the ethologically relevant behaviors of a child. This outstanding feat seems unattainable even for the current deep neural networks.

In almost all of the approaches, whether it is transfer learning or meta learning or embedding, the requirement of large datasets remains the same. For example, meta-learners need sufficient data in order to be well-trained, and they are assumed to work as infallible and indefatigable oracles. Transfer learning requires large training set *a priori*. To find proper embedding, one seeks a large dataset in embedding based methods. Therefore, in conventional approaches, a classifier, which is supposed to be trained with k-shot exemplars, takes advantage of a large volume of data implicitly. This is inharmonious with

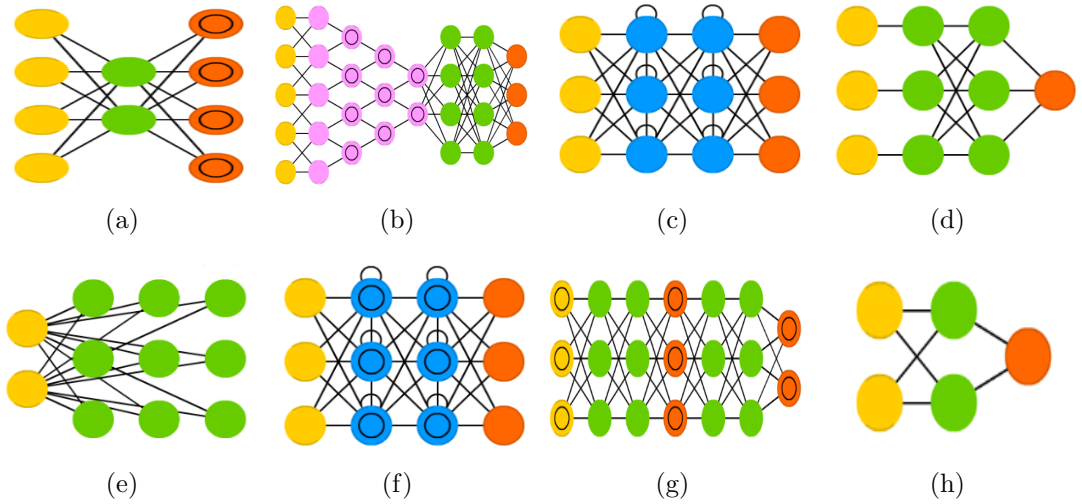


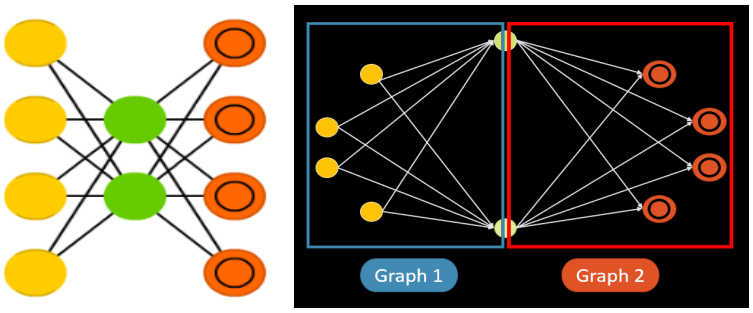
FIGURE 5.2: Neural network representation of (a) autoencoder (b) convolutional neural network (CNN), (c) recurrent neural network (RNN), (d) support vector machine (SVM), (e) Kohonen map, (f) long short term memory (LSTM), (g) generative adversarial networks (GAN), and (h) radial basis network (RBN).

the definition of LwLL. Instead of using the benefits of huge datasets (transfer learning, meta-learning, embedding), *we conceive the problem of LwLL consisting of the following stages - training (depending on the mode of learning), adaptation, validation, and testing,* which are anchored in the psychological development of the human brain. Particularly *we aim to uncover how to foresee the unseen variation of population using fewer labeled data (adaptation); to correlate and quantify any measurable changes in a classifier when trained with fewer data of wide variation (training and adaptation); to distinguish between classifier networks when trained with fewer data containing small variation and enormous variation (training); to augment the classifier parameters in order to cope up with unseen variation of data (adaptation).* We resort to graph theory to find answers to these queries.

5.1.1.2 Preliminary results

Schematic diagrams of two fundamental computational blocks used in a classifier and their equivalent graph signal representations are shown in Figs. 5.1. The network models of several conventional classifiers are shown in Fig. 5.2. From Fig. 5.3. it is evident that a neural network can be modeled as a sequence of layer-to-layer graphs. The metrics of such graphs, such as centrality, max flow, shortest path, smoothness and others, change every

iteration in accordance with the learning method. Therefore, a classifier with a network under a learning model can be described as a sequence of snapshots of its internal states taken over all iterations. We have experimented using graph smoothness and graph centrality so far, and performed tests using publicly available datasets. We hypothesize that the use of other graph metrics, such as max-flow, eccentricity *etc.* might be useful for the problem of LwLL.

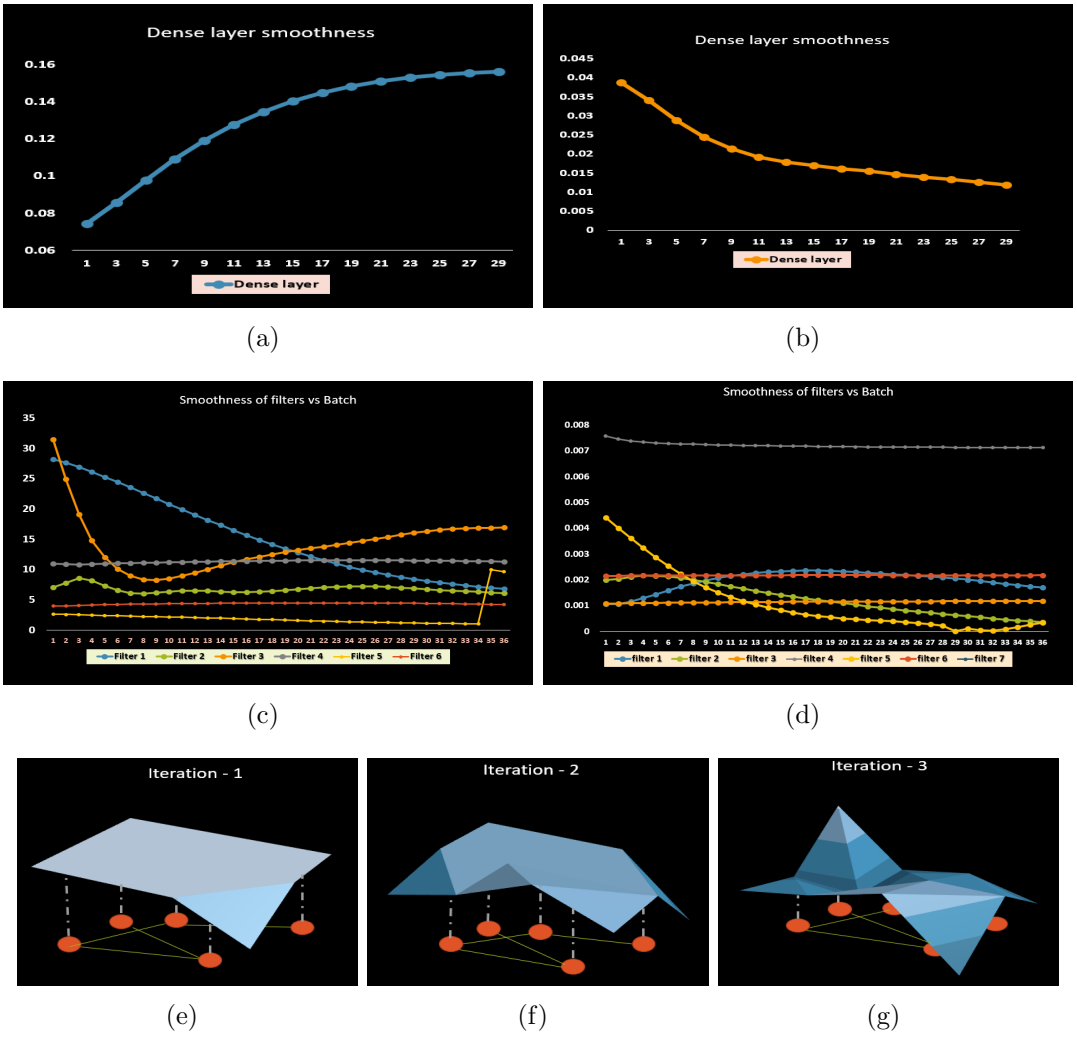


(a) (b)

FIGURE 5.3: (a) Autoencoder network and its (b) sequence of layer-to-layer graph

Let us consider the problem of overfitting, a chronic problem due to scarcity of data. We show explicit visualization to assess overfitting by inspecting the status of internal states of a classifier architecture using graph smoothness (see Section 2.1). Different classifiers are chosen to monitor the behavior of graph smoothness as shown in Fig 5.4.

The top two figures are the ideal responses of dense layers in terms of graph smoothness over batch iterations. This also holds true for convolution layers (not shown here). The smooth monotonic profile are found to be directly correlated with the capacity of the layer under observation. As both the profiles attain saturation, it implies that given a dataset, there is a maximum capacity of each layer-to-layer graph. To get more accuracy via rigorous training, we need either more neurons in a layer or more layers. Degree of regularization of weights affect layer-to-layer smoothness profile. In Fig. 5.4(c)-(d), the effects of regularization on 6 convolutional filters are shown. Higher the regularization, flatter the smoothness profiles.



(a) (b) (c) (d) (e) (f) (g)

FIGURE 5.4: Perfectly fit (a) input-hidden and (b) hidden-output layers while training an MLP with MNIST dataset. Smoothness profiles of six convolution filters over batch iterations. (c) Extremely regularized, and (d) moderately regularized. (e)-(g) A schematic representation of signal surfaces in the decreasing order of smoothness. Each surface is sampled at five locations which constitute a graph.

Currently, we are conducting rigorous experimentation using graph metrics to identify when to add more layers or to include more neurons in a feedforward neural network.

5.1.2 Activity recognition and event detection

UGraSP and UGrAD are the two algorithms that we propose for activity recognition and person identification. The videos that we apply our algorithms on contain single subject per frame. An interesting avenue of research would be to test the resilience of the algorithms when multiple subjects are present in a video.

Another fact to note is that we exclude the bending, and pick up and throw activities, because the joint correspondences in those videos are erratic. This demands refinement and correction of faulty skeletons as a required preprocessing step for activity recognition. As we can safely assume that the the movement of a skeleton is continuous and smooth, we can leverage the path2path deformation to estimate the correct postures of a subject at intermediate frames where the skeletons are erroneous.

For geomorphological event detection, we develop LaWeCo and DDT algorithms. LaWeCo is robust and simple for identifying such events spatially, whereas DDT can achieve spatiotemporal tracking even for decentralized events. DDT can be applied to other events, such as identification and tracking of oil spill in sea (Deep Horizon oil spill, a decentralized event), internet traffic patterns *etc.*. GraDED has been adopted for event detection from videos. Some of the bottlenecks of GraDED are the number of parameters in its dictionary, the eigendecomposition of the Laplacian matrices and the non-convex optimization function.

Proposed improvement of GrDED:

The framework given in Section 4.1.3 does not utilize the sparse code X which is generated iteratively corresponding to the input features. We claim that the sparse codes also contain the signatures of a problem for which the input features are extracted beforehand. In addition, the framework has limited scope of identifying the spatio-temporal events from videos. We try to augment this framework to a classification problem by leveraging the graph framework so that it can be used in segmentation and tracking.

In the work given in Section4.1.3, the dictionary and sparse codes are updated in an alternate fashion. During the learning process, sparse codes, X, corresponding to the input features, Y are generated as an intermediate step. The sparse codes are not re-utilized in section 4.1.3, which incurs computational loss during the runtime of our

algorithm. With an algorithm that extracts graph Laplacian from the sparse codes, we aim to extend the previous approach to apply for supervised multi-class classification problems.

For a mathematical description, let X^i is the sparse code of Y^i with $i \in \{1, 2, ..., c\}$, where c is the number of classes. It can be unarguably assumed that X^i contains the signature of the input Y^i. Unlike GraDED, in this work, we can first derive graph Laplacians from X^i. Let the Laplacian derived from the sparse code X_i be L^i. L^i contains essential information about the relationship between features of i^{th} class. However, L^i does not contain any free parameter to tweak by an algorithm. To construct a cost function by including the Laplacians in order to provide discriminatory ability to the dictionary, the Laplacians are regularized and the regularization parameter is to be optimized.

Let $||\bullet||_0$ as the L_0 norm of a matrix. The optimization function can be given by

$$\begin{aligned} \phi\big(\{D_1, D_2\}, \alpha, \beta X\big) &= ||Y^1 - D_1X^1||_F^2 + ||Y_2 - D_2X_2||_F^2 - \beta \sum_j ||D_1X^1 - D_2X^j||_F^2 \\ & s.t.\ ||X_1||_0 \leq T_1,\ ||X_2||_0 \leq T_2. \end{aligned} \tag{5.1}$$

In eq. (5.1), Y^1 is the features of class 1, and Y_2 consists of the features from rest of the classes, $Y_2 = [Y^2\ Y^3\ ...\ Y^c]$. Similar explanation holds for X_2 and D_2. Each dictionary D^i is constructed using a single step $D^i = (L^i + \alpha\Gamma)^{-1}$, where Γ is a diagonal matrix. α is a parameter to be optimized. From computational perspective, it is wise to solve the above optimization problem by using the regularized graph Laplacian as opposed to the use of graph spectrum in Section 4.1.3.

Publications resulting from this thesis

Journal Publications

1. Vaccari, A., **Batabyal, Tamal**; Tabassum, N.; Hoppe, E. J.; Bruckno, B. S.; Acton, S. T. (2018). "Integrating Remote Sensing Data in Decision Support Systems for Transportation Asset Management." Transportation Research Record, 0361198118786645.

2. **Batabyal, Tamal**.; Weller, D.S.; Acton S.T., "PrecoG: an efficient unitary split preconditioner for the transform-domain LMS filter via graph Laplacian regularization." (Under review)

3. Sadeghzadehyazdi, N.; **Batabyal, Tamal**; Acton, S.T."Toward person identification by soft biometric feature correction of flash Lidar data." (In preparation)

4. **Batabyal, Tamal**; Condron, B.; Acton, S.T. "NeuroPath2Path: Classification and elastic morphing between neuronal arbors using path-wise similarity." (under review)

5. Dutta, A.; **Batabyal, Tamal**; Acton, S.T. "Neural network based prediction of coronary and congestive heart disease using clinical data." (In preparation)

6. **Batabyal, Tamal**; Vaccari, A.; Condron, B.; Acton, S.T. "GraSPEL: Graph signal processing based efficient learning." (In preparation)

Conference Publications

1. **Batabyal, Tamal** and Acton, Scott T.. "DDT: Decentralized event Detection and Tracking using an ensemble of vertex-reinforced walks on a graph." 2018 IEEE Southwest Symposium on Image Analysis and Interpretation (SSIAI). IEEE, 2018. [Oral][Conference]

2. **Batabyal, Tamal**; Andrea Vaccari; and Scott T. Acton. "NeuroBFD: Size-independent automated classification of neurons using conditional distributions of morphological

features." Biomedical Imaging (ISBI 2018), 2018 IEEE 15th International Symposium on. IEEE, 2018. [Poster][Conference]

3. **Batabyal, Tamal** and Scott T. Acton. "ElasticPath2Path: Automated morphological classification of neurons by elastic path matching." arXiv preprint arXiv:1802.06913 (2018). [Oral][ICIP 2018]

4. **Batabyal, Tamal**; Rituparna Sarkar and Scott T. Acton. "GraDED: A graph-based parametric dictionary learning algorithm for event detection." Image Processing (ICIP), 2017 IEEE International Conference on. IEEE, 2017. [Oral][Conference]

5. **Batabyal, Tamal** and Scott T. Acton. "NeuroSoL: Automated classification of neurons using the sorted Laplacian of a graph." Biomedical Imaging (ISBI 2017), 2017 IEEE 14th International Symposium on. IEEE, 2017. [Oral][Conference]

6. Sadeghzadehyazdi, N., **Batabyal, Tamal**, Barnes, L. E.; Acton, S. T. (2016, November). "Graph-based classification of healthcare provider activity." In Signals, Systems and Computers, 2016 50th Asilomar Conference on (pp. 1268-1272). IEEE. [Oral][Conference]

7. **Batabyal, Tamal**; Scott T. Acton, and Andrea Vaccari. "Ugrad: A graph-theoretic framework for classification of activity with complementary graph boundary detection." Image Processing (ICIP), 2016 IEEE International Conference on. IEEE, 2016. [Oral][Conference]

8. **Batabyal, Tamal**; Andrea Vaccari, and Scott T. Acton. "LaWeCo: Active region detection in non-uniformly sampled data using Laplacian-weighted covariance." Image Analysis and Interpretation (SSIAI), 2016 IEEE Southwest Symposium on. IEEE, 2016. [Oral][Conference]

9. **Batabyal, Tamal**; Andrea Vaccari, and Scott T. Acton. "Ugrasp: A unified framework for activity recognition and person identification using graph signal processing." Image Processing (ICIP), 2015 IEEE International Conference on. IEEE, 2015. [Poster][Conference]

10. Wang Jie; **Batabyal, Tamal**; Zhang, M; Zhang, J; Gahlmann, Andreas; Acton, S.T., "lCuts: Linear clustering of bacteria using recursive graph cuts." (submitted)

11. Sadeghzadehyazdi, Nasrin; **Batabyal, Tamal**; Galndon, A.; Dhar, Nibir K.; Familoni, B.O.; Ifthekharuddin, K. M.; Acton, S.T., "$\mathcal{G}$lidar3DJ: A View-Invariant gait identification via flash lidar data correction." (submitted)

12. Ly Tiffany T.; **Batabyal, Tamal**; Thompson, Jeremy; Harris, Tajie; Weller, Daniel S.; Acton, Scott T.,"Hieroglyph: Hierarchical Glia Graph Skeletonization and Matching" (submitted)

Bibliography

[1] L. Zicheng, "MSR Action3d dataset," http://research.microsoft.com/en-us/um/people/zliu/ActionRecoRsrc/default.htm, accessed: December, 2014.

[2] L. Xia, C.-C. Chen, and J. Aggarwal, "View invariant human action recognition using histograms of 3d joints," in *CVPRW*. IEEE, 2012, pp. 20–27.

[3] S. Masood, C. Ellis, A. Nagaraja, M. Tappen, J. LaViola, and R. Sukthankar, "Measuring and reducing observational latency when recognizing actions," in *ICCV Workshops*, November 2011, pp. 422–429.

[4] D. Myatt, T. Hadlington, G. Ascoli, and S. Nasuto, "Neuromantic–from semi-manual to semi-automatic reconstruction of neuron morphology," *Frontiers in neuroinformatics*, vol. 6, p. 4, 2012.

[5] S. Murphy, K. Rokicki, C. Bruns, Y. Yu, L. Foster, E. Trautman, D. Olbris, T. Wolff, A. Nern, Y. Aso *et al.*, "The janelia workstation for neuroscience," *Keystone Big Data in Biology. San Francisco, CA*, 2014.

[6] S. Basu, B. Condron, and S. T. Acton, "Path2path: Hierarchical path-based analysis for neuron matching," in *2011 IEEE International Symposium on Biomedical Imaging: From Nano to Macro*. IEEE, 2011, pp. 996–999.

[7] T. Batabyal, A. Vaccari, and S. T. Acton, "Neurobfd: Size-independent automated classification of neurons using conditional distributions of morphological features," in *Biomedical Imaging (ISBI 2018), 2018 IEEE 15th International Symposium on*. IEEE, 2018, pp. 912–915.

[8] E. Stockley, H. Cole, A. Brown, and H. Wheal, "A system for quantitative morphological measurement and electrotonic modelling of neurons: three-dimensional reconstruction," *Journal of neuroscience methods*, vol. 47, no. 1-2, pp. 39–51, 1993.

[9] A. Srivastava, E. Klassen, S. H. Joshi, and I. H. Jermyn, "Shape analysis of elastic curves in euclidean spaces," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 7, pp. 1415–1428, 2011.

[10] M. Swan, "The quantified self: Fundamental disruption in big data science and biological discovery," *Big data*, vol. 1, no. 2, pp. 85–99, 2013.

[11] M. A. Waller and S. E. Fawcett, "Data science, predictive analytics, and big data: a revolution that will transform supply chain design and management," *Journal of Business Logistics*, vol. 34, no. 2, pp. 77–84, 2013.

[12] D. Koller, N. Friedman, and F. Bach, *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

[13] M. Rubinov and O. Sporns, "Complex network measures of brain connectivity: uses and interpretations," *Neuroimage*, vol. 52, no. 3, pp. 1059–1069, 2010.

[14] H. Wang, A. Klaser, C. Schmid, and C.-L. Liu, "Action recognition by dense trajectories," in *CVPR*. IEEE, 2011, pp. 3169–3176.

[15] W. Li, Z. Zhang, and Z. Liu, "Action recognition based on a bag of 3d points," in *CVPRW*. IEEE, 2010, pp. 9–14.

[16] A. Loukas, M. Zuniga, I. Protonotarios, and J. Gao, "How to identify global trends from local decisions? event region detection on mobile networks," in *INFOCOM, 2014 Proceedings IEEE*, 2014, pp. 1177–1185.

[17] A. Loukas, M. Cattani, M. Zuniga, and J. Gao, "Graph scale-space theory for distributed peak and pit identification," in *Proc. of the 14th Intl. Conf. on Information Processing in Sensor Networks*, 2015, pp. 118–129.

[18] A. Vaccari and S. T. Acton, "Spatiotemporal gaussian feature detection in sparsely sampled data with application to insar," in *SPIE Defense, Security, and Sensing*, 2013, pp. 87 460U–87 460U.

[19] J. Abernethy, O. Chapelle, and C. Castillo, "Graph regularization methods for web spam detection," *Machine Learning*, vol. 81, no. 2, pp. 207–225, 2010.

[20] H. Bunke and K. Riesen, "Recent advances in graph-based pattern recognition with applications in document analysis," *Pattern Recognition*, vol. 44, no. 5, pp. 1057–1067, 2011.

[21] D. Hallac, Y. Park, S. Boyd, and J. Leskovec, "Network inference via the time-varying graphical lasso," *arXiv preprint arXiv:1703.01958*, 2017.

[22] U. Von Luxburg, "A tutorial on spectral clustering," *Statistics and computing*, vol. 17, no. 4, pp. 395–416, 2007.

[23] F. Harary *et al.*, "Graph theory," 1969.

[24] D. Shuman, S. Narang, P. Frossard, A. Ortega, and P. Vandergheynst, "The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains," *Signal Processing Magazine, IEEE*, vol. 30, no. 3, pp. 83–98, 2013.

[25] J. Lafferty, A. McCallum, and F. C. Pereira, "Conditional random fields: Probabilistic models for segmenting and labeling sequence data," 2001.

[26] C. H. Papadimitriou and K. Steiglitz, *Combinatorial optimization: algorithms and complexity*. Courier Corporation, 1998.

[27] S. Polavaram, T. A. Gillette, R. Parekh, and G. A. Ascoli, "Statistical analysis and data mining of digital reconstructions of dendritic morphologies," *Frontiers in neuroanatomy*, vol. 8, 2014.

[28] K. W. Church, "A stochastic parts program and noun phrase parser for unrestricted text," in *International Conference on Acoustics, Speech, and Signal Processing,*. IEEE, 1989, pp. 695–698.

[29] J. A. Gallian, "A dynamic survey of graph labeling," *The electronic journal of combinatorics*, vol. 16, no. 6, pp. 1–219, 2009.

[30] H. Zhu, H. Zang, K. Zhu, and B. Mukherjee, "A novel generic graph model for traffic grooming in heterogeneous wdm mesh networks," *IEEE/ACM Transactions on Networking (TON)*, vol. 11, no. 2, pp. 285–299, 2003.

[31] M. E. Newman, D. J. Watts, and S. H. Strogatz, "Random graph models of social networks," *Proceedings of the National Academy of Sciences*, vol. 99, no. suppl 1, pp. 2566–2572, 2002.

[32] R. Albert, "Scale-free networks in cell biology," *Journal of cell science*, vol. 118, no. 21, pp. 4947–4957, 2005.

[33] M. D RADMACHER, R. Simon, R. Desper, R. Taetle, A. A. Schäffer, and M. A. Nelson, "Graph models of oncogenesis with an application to melanoma," *Journal of theoretical biology*, vol. 212, no. 4, pp. 535–548, 2001.

[34] O. Sporns, G. Tononi, and R. Kötter, "The human connectome: a structural description of the human brain," *PLoS computational biology*, vol. 1, no. 4, p. e42, 2005.

[35] J. Shotton, T. Sharp, A. Kipman, A. Fitzgibbon, M. Finocchio, A. Blake, M. Cook, and R. Moore, "Real-time human pose recognition in parts from single depth images," *Communications of the ACM*, vol. 56, no. 1, pp. 116–124, 2013.

[36] D. Cunado, M. Nixon, and J. Carter, "Automatic extraction and description of human gait models for recognition purposes," *CVIU*, vol. 90, no. 1, pp. 1–41, 2003.

[37] A. Sanin, C. Sanderson, M. Harandi, and B. Lovell, "Spatio-temporal covariance descriptors for action and gesture recognition," in *WACV*. IEEE, 2013, pp. 103–110.

[38] A. Kovashka and K. Grauman, "Learning a hierarchy of discriminative space-time neighborhood features for human action recognition," in *CVPR*. IEEE, 2010, pp. 2046–2053.

[39] J. Aggarwal and M. S. Ryoo, "Human activity analysis: A review," *ACM Computing Surveys (CSUR)*, vol. 43, no. 3, p. 16, 2011.

[40] I. Laptev, "On space-time interest points," *IJCV*, vol. 64, no. 2-3, pp. 107–123, 2005.

[41] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *CVPR*, vol. 1. IEEE, 2005, pp. 886–893.

[42] I. Laptev, M. Marszalek, C. Schmid, and B. Rozenfeld, "Learning realistic human actions from movies," in *CVPR*. IEEE, 2008, pp. 1–8.

[43] M. Müller and T. Röder, "Motion templates for automatic classification and retrieval of motion capture data," in *Proceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on Computer animation*. Eurographics Association, 2006, pp. 137–146.

[44] L. Han, X. Wu, W. Liang, G. Hou, and Y. Jia, "Discriminative human action recognition in the learned hierarchical manifold space," *Image and Vision Computing*, vol. 28, no. 5, pp. 836–849, 2010.

[45] N. Troje, C. Westhoff, and M. Lavrov, "Person identification from biological motion: Effects of structural and kinematic cues," *Perception & Psychophysics*, vol. 67, no. 4, pp. 667–675, 2005.

[46] R. Brunelli and D. Falavigna, "Person identification using multiple cues," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 17, no. 10, pp. 955–966, 1995.

[47] L. Xia, C.-C. Chen, and J. Aggarwal, "Human detection using depth information by kinect," in *CVPRW*. IEEE, 2011, pp. 15–22.

[48] A. Sinha, K. Chakravarty, and B. Bhowmick, "Person identification using skeleton information from kinect," in *ACHI 2013*, 2013, pp. 101–108.

[49] B. Munsell, A. Temlyakov, C. Qu, and S. Wang, "Person identification using full-body motion and anthropometric biometrics from kinect videos," in *ECCV 2012. Workshops and Demonstrations*. Springer, 2012, pp. 91–100.

[50] M. Hussein, M. Torki, M. Gowayyed, and M. El-Saban, "Human action recognition using a temporal hierarchy of covariance descriptors on 3d joint locations," in *IJCAI*. AAAI Press, 2013, pp. 2466–2472.

[51] J. Martens and I. Sutskever, "Learning recurrent neural networks with hessian-free optimization," in *ICML*, 2011, pp. 1033–1040.

[52] J. Wang, Z. Liu, J. Chorowski, Z. Chen, and Y. Wu, "Robust 3d action recognition with random occupancy patterns," in *Computer Vision–ECCV 2012*. Springer, 2012, pp. 872–885.

[53] J. Wang, Z. Liu, Y. Wu, and J. Yuan, "Mining actionlet ensemble for action recognition with depth cameras," in *CVPR*. IEEE, 2012, pp. 1290–1297.

[54] T. Batabyal, T. Chattopadhyay, and D. P. Mukherjee, "Action recognition using joint coordinates of 3d skeleton data," in *Image Processing (ICIP), 2015 IEEE International Conference on*. IEEE, 2015, pp. 4107–4111.

[55] T. Batabyal, A. Vaccari, and S. Acton, "Ugrasp: A unified framework for activity recognition and person identification using graph signal processing," in *Image Processing (ICIP), 2015 IEEE International Conference on*. IEEE, 2015, pp. 3270–3274.

[56] F. Ofli, R. Chaudhry, G. Kurillo, R. Vidal, and R. Bajcsy, "Sequence of the most informative joints (smij): A new representation for human skeletal action recognition," *Journal of Visual Communication and Image Representation*, vol. 25, no. 1, pp. 24–38, 2014.

[57] V. Mnih, H. Larochelle, and G. E. Hinton, "Conditional restricted boltzmann machines for structured output prediction," *arXiv preprint arXiv:1202.3748*, 2012.

[58] G. Taylor, G. E. Hinton, and S. T. Roweis, "Modeling human motion using binary latent variables," in *Advances in neural information processing systems*, 2006, pp. 1345–1352.

[59] R. Vemulapalli, F. Arrate, and R. Chellappa, "Human action recognition by representing 3d skeletons as points in a lie group," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 588–595.

[60] K. M. Brown, T. A. Gillette, and G. A. Ascoli, "Quantifying neuronal size: summing up trees and splitting the branch difference," in *Seminars in cell & developmental biology*, vol. 19, no. 6. Elsevier, 2008, pp. 485–493.

[61] G. A. Ascoli, L. Alonso-Nanclares, S. A. Anderson, G. Barrionuevo, R. Benavides-Piccione, A. Burkhalter, G. Buzsáki, B. Cauli, J. DeFelipe, A. Fairén *et al.*, "Petilla terminology: nomenclature of features of gabaergic interneurons of the cerebral cortex," *Nature Reviews Neuroscience*, vol. 9, no. 7, p. 557, 2008.

[62] S. R. y Cajal, *Histologie du système nerveux de l'homme et des vertébrés: Ed. française revue et mise a jour par l'auteur. Trad. de l'espagnol par L. Azoulay.* Inst. Ramon y Cajal, 1972.

[63] C. Bielza, R. Benavides-Piccione, P. López-Cruz, P. Larranaga, and J. DeFelipe, "Branching angles of pyramidal cell dendrites follow common geometrical design principles in different cortical areas," *Scientific reports*, vol. 4, p. 5909, 2014.

[64] S. Romand, Y. Wang, M. Toledo-Rodriguez, and H. Markram, "Morphological development of thick-tufted layer v pyramidal cells in the rat somatosensory cortex," *Frontiers in neuroanatomy*, vol. 5, p. 5, 2011.

[65] J. R. Glaser and E. M. Glaser, "Neuron imaging with neurolucida—a pc-based system for image combining microscopy," *Computerized Medical Imaging and Graphics*, vol. 14, no. 5, pp. 307–317, 1990.

[66] M. London and M. Häusser, "Dendritic computation," *Annu. Rev. Neurosci.*, vol. 28, pp. 503–532, 2005.

[67] Y.-N. Jan and L. Y. Jan, "Branching out: mechanisms of dendritic arborization," *Nature Reviews Neuroscience*, vol. 11, no. 5, p. 316, 2010.

[68] Q. Wen and D. B. Chklovskii, "A cost–benefit analysis of neuronal morphology," *Journal of neurophysiology*, vol. 99, no. 5, pp. 2320–2328, 2008.

[69] L. Kanari, P. Dłotko, M. Scolamiero, R. Levi, J. Shillcock, K. Hess, and H. Markram, "A topological representation of branching neuronal morphologies," *Neuroinformatics*, vol. 16, no. 1, pp. 3–13, 2018.

[70] E. P. Cervantes, C. H. Comin, R. M. C. Junior, and L. da Fontoura Costa, "Morphological neuron classification based on dendritic tree hierarchy," *Neuroinformatics*, pp. 1–15, 2018.

[71] P. L. López-Cruz, C. Bielza, P. Larrañaga, R. Benavides-Piccione, and J. DeFelipe, "Models and simulation of 3d neuronal dendritic trees using bayesian networks," *Neuroinformatics*, vol. 9, no. 4, pp. 347–369, 2011.

[72] M. Migliore and G. M. Shepherd, "Emerging rules for the distributions of active dendritic conductances," *Nature Reviews Neuroscience*, vol. 3, no. 5, p. 362, 2002.

[73] S. Gasparini, M. Migliore, and J. C. Magee, "On the initiation and propagation of dendritic spikes in ca1 pyramidal neurons," *Journal of Neuroscience*, vol. 24, no. 49, pp. 11 046–11 056, 2004.

[74] S. R. Williams, "Spatial compartmentalization and functional impact of conductance in pyramidal neurons," *Nature neuroscience*, vol. 7, no. 9, p. 961, 2004.

[75] G. A. Ascoli, D. E. Donohue, and M. Halavi, "Neuromorpho. org: a central resource for neuronal morphologies," *The Journal of Neuroscience*, vol. 27, no. 35, pp. 9247–9251, 2007.

[76] E. Meijering, "Neuron tracing in perspective," *Cytometry Part A*, vol. 77, no. 7, pp. 693–704, 2010.

[77] R. Scorcioni, S. Polavaram, and G. A. Ascoli, "L-measure: a web-accessible tool for the analysis, comparison and search of digital reconstructions of neuronal morphologies," *Nature protocols*, vol. 3, no. 5, p. 866, 2008.

[78] Y. Wan, F. Long, L. Qu, H. Xiao, M. Hawrylycz, E. W. Myers, and H. Peng, "Blastneuron for automated comparison, retrieval and clustering of 3d neuron morphologies," *Neuroinformatics*, vol. 13, no. 4, pp. 487–499, 2015.

[79] T. Batabyal and S. T. Acton, "Neurosol: Automated classification of neurons using the sorted laplacian of a graph," in *Biomedical Imaging (ISBI 2017), 2017 IEEE 14th International Symposium on*. IEEE, 2017, pp. 397–400.

[80] T. Batabyal, S. T. Acton, and A. Vaccari, "Ugrad: A graph-theoretic framework for classification of activity with complementary graph boundary detection," in *2016 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2016, pp. 1339–1343.

[81] R. Sarkar, S. Mukherjee, and S. T. Acton, "Shape descriptors based on compressed sensing with application to neuron matching," in *2013 Asilomar Conference on Signals, Systems and Computers*. IEEE, 2013, pp. 970–974.

[82] T. Gillette and G. Ascoli, "Topological characterization of neuronal arbor morphology via sequence representation. i," *Motif analysis*, 2015.

[83] T. A. Gillette, P. Hosseini, and G. A. Ascoli, "Topological characterization of neuronal arbor morphology via sequence representation: Ii-global alignment," *BMC bioinformatics*, vol. 16, no. 1, p. 209, 2015.

[84] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[85] S. V. N. Vishwanathan, N. N. Schraudolph, R. Kondor, and K. M. Borgwardt, "Graph kernels," *Journal of Machine Learning Research*, vol. 11, no. Apr, pp. 1201–1242, 2010.

[86] T. Batabyal and S. T. Acton, "Elasticpath2path: Automated morphological classification of neurons by elastic path matching," *arXiv preprint arXiv:1802.06913*, 2018.

[87] J. Munkres, "Algorithms for the assignment and transportation problems," *Journal of the society for industrial and applied mathematics*, vol. 5, no. 1, pp. 32–38, 1957.

[88] W. Kabsch, "A discussion of the solution for the best rotation to relate two sets of vectors," *Acta Crystallographica Section A: Crystal Physics, Diffraction, Theoretical and General Crystallography*, vol. 34, no. 5, pp. 827–828, 1978.

[89] J. Miina and T. Pukkala, "Application of ecological field theory in distance-dependent growth modelling," *Forest Ecology and Management*, vol. 161, no. 1-3, pp. 101–107, 2002.

[90] A. Genet, P. Grabarnik, O. Sekretenko, and D. Pothier, "Incorporating the mechanisms underlying inter-tree competition into a random point process model to improve spatial tree pattern analysis in forestry," *Ecological modelling*, vol. 288, pp. 143–154, 2014.

[91] N. Puškaš, I. Zaletel, B. D. Stefanović, and D. Ristanović, "Fractal dimension of apical dendritic arborization differs in the superficial and the deep pyramidal neurons of the rat cerebral neocortex," *Neuroscience letters*, vol. 589, pp. 88–91, 2015.

[92] A. V. Samsonovich and G. A. Ascoli, "Morphological homeostasis in cortical dendrites," *Proceedings of the National Academy of Sciences*, vol. 103, no. 5, pp. 1569–1574, 2006.

[93] Y. Lin and S.-T. Yau, "Ricci curvature and eigenvalue estimate on locally finite graphs," *Mathematical research letters*, vol. 17, no. 2, pp. 343–356, 2010.

[94] K.-P. Shih, S.-S. Wang, H.-C. Chen, and P.-H. Yang, "Collect: Collaborative event detection and tracking in wireless heterogeneous sensor networks," *Comp. Comm.*, vol. 31, no. 14, pp. 3124–3136, 2008.

[95] F. Li, J. Luo, C. Zhang, S. Xin, and Y. He, "Unfold: uniform fast on-line boundary detection for dynamic 3d wireless sensor networks," in *Proceedings of the Twelfth ACM International Symposium on Mobile Ad Hoc Networking and Computing*, 2011, p. 14.

[96] G. Jin and S. Nittel, "Ned: An efficient noise-tolerant event and event boundary detection algorithm in wireless sensor networks," in *Mobile Data Management. 7th International Conference on*, 2006, pp. 153–153.

[97] J. Shi and J. Malik, "Normalized cuts and image segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 8, pp. 888–905, 2000.

[98] A. Vaccari, M. Stuecheli, B. Bruckno, E. Hoppe, and S. T. Acton, "Detection of geophysical features in InSAR point cloud data set using spatiotemporal models," *International Journal of Remote Sensing*, vol. 34, no. 22, pp. 8215–8234, 2013.

[99] A. Ferretti, A. Fumagalli, F. Novali, C. Prati, F. Rocca, and A. Rucci, "A new algorithm for processing interferometric data-stacks: Squeesar," *IEEE Trans. Geosci. Remote Sens.*, vol. 49, no. 9, pp. 3460–3470, 2011.

[100] A. Karbasi, A. H. Salavati, and M. Vetterli, "Learning network structures from firing patterns," in *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on.* IEEE, 2016, pp. 699–703.

[101] T. Batabyal, A. Vaccari, and S. T. Acton, "LaWeCo: Active region detection in non-uniformly sampled data using laplacian-weighted covariance," in *Image Analysis*

and Interpretation (SSIAI), 2016 IEEE Southwest Symposium on. IEEE, 2016, pp. 129–132.

[102] H. Petric Maretic, D. Thanou, and P. Frossard, "Graph learning under sparsity priors," in *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, no. EPFL-CONF-224359, 2017.

[103] M. G. Rabbat, "Inferring sparse graphs from smooth signals with theoretical guarantees," in *Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on.* IEEE, 2017, pp. 6533–6537.

[104] V. Kalofolias, "How to learn a graph from smooth signals," in *Artificial Intelligence and Statistics*, 2016, pp. 920–929.

[105] J. Mei and J. M. Moura, "Signal processing on graphs: Estimating the structure of a graph," in *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on.* IEEE, 2015, pp. 5495–5499.

[106] C. Cotar, D. Thacker *et al.*, "Edge-and vertex-reinforced random walks with superlinear reinforcement on infinite graphs," *The Annals of Probability*, vol. 45, no. 4, pp. 2655–2706, 2017.

[107] M. Piccardi, "Background subtraction techniques: a review," in *Systems, man and cybernetics, 2004 IEEE international conference on*, vol. 4. IEEE, 2004, pp. 3099–3104.

[108] F. F. E. Guraya, F. A. Cheikh, A. Tremeau, Y. Tong, and H. Konik, "Predictive saliency maps for surveillance videos," in *Distributed Computing and Applications to Business Engineering and Science (DCABES), 2010 Ninth International Symposium on.* IEEE, 2010, pp. 508–513.

[109] P. W. Power and J. A. Schoonees, "Understanding background mixture models for foreground segmentation," in *Proceedings image and vision computing New Zealand*, vol. 2002, 2002, pp. 10–11.

[110] E. J. Candès, X. Li, Y. Ma, and J. Wright, "Robust principal component analysis?" *Journal of the ACM (JACM)*, vol. 58, no. 3, p. 11, 2011.

[111] C. Stauffer and W. E. L. Grimson, "Adaptive background mixture models for real-time tracking," in *Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on.*, vol. 2. IEEE, 1999, pp. 246–252.

[112] R. Sarkar, S. Das, and N. Vaswani, "Tracking sparse signal sequences from nonlinear/non-gaussian measurements and applications in illumination-motion tracking," in *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on.* IEEE, 2013, pp. 6615–6619.

[113] S. Mukherjee, R. Sarkar, J. Vandenbrink, S. T. Acton, and B. Blackman, "Tracking sunflower circumnutation using affine parametric active contours," in *Image Analysis and Interpretation (SSIAI), 2014 IEEE Southwest Symposium on.* IEEE, 2014, pp. 93–96.

[114] W. Hu, X. Xiao, Z. Fu, D. Xie, T. Tan, and S. Maybank, "A system for learning statistical motion patterns," *IEEE transactions on pattern analysis and machine intelligence*, vol. 28, no. 9, pp. 1450–1464, 2006.

[115] A. Basharat, A. Gritai, and M. Shah, "Learning object motion patterns for anomaly detection and improved object detection," in *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on.* IEEE, 2008, pp. 1–8.

[116] J. Hamm and D. D. Lee, "Grassmann discriminant analysis: a unifying view on subspace-based learning," in *Proceedings of the 25th international conference on Machine learning.* ACM, 2008, pp. 376–383.

[117] L. Xiong, X. Chen, and J. Schneider, "Direct robust matrix factorizatoin for anomaly detection," in *Data Mining (ICDM), 2011 IEEE 11th International Conference on.* IEEE, 2011, pp. 844–853.

[118] A. Adler, M. Elad, Y. Hel-Or, and E. Rivlin, "Sparse coding with anomaly detection," *Journal of Signal Processing Systems*, vol. 79, no. 2, pp. 179–188, 2015.

[119] M. Aharon, M. Elad, and A. Bruckstein, "*rmk*-svd: An algorithm for designing overcomplete dictionaries for sparse representation," *IEEE Transactions on signal processing*, vol. 54, no. 11, pp. 4311–4322, 2006.

[120] R. Sarkar, A. Vaccari, and S. T. Acton, "Sspared: Saliency and sparse code analysis for rare event detection in video," in *Image, Video, and Multidimensional Signal Processing Workshop (IVMSP), 2016 IEEE 12th.* IEEE, 2016, pp. 1–5.

[121] E. Candes and J. Romberg, "Sparsity and incoherence in compressive sampling," *Inverse problems*, vol. 23, no. 3, p. 969, 2007.

[122] X. Zhang, X. Dong, and P. Frossard, "Learning of structured graph dictionaries," in *Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on.* IEEE, 2012, pp. 3373–3376.

[123] B. Widrow and M. E. J. Hoff, "Adaptive switching circuits." in *1960 IRE WESCON Convention record: at the Western Electronic Show and Convention, Los Angeles, Calif., August 23-26, 1960.* Institute of Radio Engineers, 1960.

[124] C. Paleologu, S. Ciochina, A. A. Enescu, and C. Vladeanu, "Gradient adaptive lattice algorithm suitable for fixed point implementation," in *Digital Telecommunications, 2008. ICDT'08. The Third International Conference on.* IEEE, 2008, pp. 41–46.

[125] H. Fan and X. Q. Liu, "Gal and lsl revisited: new convergence results," *IEEE transaction on signal processing*, vol. 41, no. 1, pp. 55–66, 1993.

[126] S. Haykin, *Adaptive Filter Theory (3rd Ed.)*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1996.

[127] S. Zhao, Z. Man, S. Khoo, and H. R. Wu, "Stability and convergence analysis of transform-domain lms adaptive filters with second-order autoregressive process," *Signal Processing, IEEE Transactions on*, vol. 57, no. 1, pp. 119–130, 2009.

[128] S. Hosur and A. H. Tewfik, "Wavelet transform domain adaptive fir filtering," *Signal Processing, IEEE Transactions on*, vol. 45, no. 3, pp. 617–630, 1997.

[129] M. H. Costa, J. C. M. Bermudez, and N. J. Bershad, "Stochastic analysis of the lms algorithm with a saturation nonlinearity following the adaptive filter output," *Signal Processing, IEEE Transactions on*, vol. 49, no. 7, pp. 1370–1387, 2001.

[130] D. I. Kim and P. De Wilde, "Performance analysis of the dct-lms adaptive filtering algorithm," *Signal Processing*, vol. 80, no. 8, pp. 1629–1654, 2000.

[131] B. Farhang-Boroujeny, "Transform domain adaptive filters," *Adaptive Filters: Theory and Applications*, pp. 207–250, 1998.

[132] F. Beaufays, "Transform-domain adaptive filters: an analytical approach," *Signal Processing, IEEE Transactions on*, vol. 43, no. 2, pp. 422–431, 1995.

[133] K. Chen, *Matrix preconditioning techniques and applications*. Cambridge University Press, 2005, vol. 19.

[134] M. Benzi, "Preconditioning techniques for large linear systems: a survey," *Journal of computational Physics*, vol. 182, no. 2, pp. 418–477, 2002.

[135] Y. Cao, M.-Q. Jiang, and Y.-L. Zheng, "A splitting preconditioner for saddle point problems," *Numerical Linear Algebra with Applications*, vol. 18, no. 5, pp. 875–895, 2011.

[136] Q. Zheng and L. Lu, "Extended shift-splitting preconditioners for saddle point problems," *Journal of Computational and Applied Mathematics*, vol. 313, pp. 70–81, 2017.

[137] Y. Dauphin, H. de Vries, and Y. Bengio, "Equilibrated adaptive learning rates for non-convex optimization," in *Advances in Neural Information Processing Systems*, 2015, pp. 1504–1512.

[138] O. Chapelle and D. Erhan, "Improved preconditioner for hessian free optimization," in *NIPS Workshop on Deep Learning and Unsupervised Feature Learning*, vol. 201, no. 1, 2011.

[139] A. Dziekonski, A. Lamecki, and M. Mrozowski, "Jacobi and gauss-seidel preconditioned complex conjugate gradient method with gpu acceleration for finite element method," in *Microwave Conference (EuMC), 2010 European.* IEEE, 2010, pp. 1305–1308.

[140] W.-P. Tang, "Toward an effective sparse approximate inverse preconditioner," *SIAM journal on matrix analysis and applications*, vol. 20, no. 4, pp. 970–986, 1999.

[141] M. Benzi, J. K. Cullum, and M. Tuma, "Robust approximate inverse preconditioning for the conjugate gradient method," *SIAM Journal on Scientific Computing*, vol. 22, no. 4, pp. 1318–1332, 2000.

[142] E. Chow and A. Patel, "Fine-grained parallel incomplete lu factorization," *SIAM Journal on Scientific Computing*, vol. 37, no. 2, pp. C169–C193, 2015.

[143] H. P. Maretic, D. Thanou, and P. Frossard, "Graph learning under sparsity priors," in *Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on.* Ieee, 2017, pp. 6523–6527.

[144] E. Pavez, H. E. Egilmez, and A. Ortega, "Learning graphs with monotone topology properties and multiple connected components," *IEEE Transactions on Signal Processing*, 2018.

[145] O. Lezoray, V. T. Ta, and A. Elmoataz, "Nonlocal graph regularization for image colorization," in *Pattern Recognition, 2008. ICPR 2008. 19th International Conference on.* IEEE, 2008, pp. 1–4.

[146] D. Hallac, J. Leskovec, and S. Boyd, "Network lasso: Clustering and optimization in large graphs," in *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining.* ACM, 2015, pp. 387–396.

[147] M.-D. Choi, "Tricks or treats with the hilbert matrix," *The American Mathematical Monthly*, vol. 90, no. 5, pp. 301–312, 1983.

[148] S. Lewis, "Neuroimmunology: Brain police," *Nature Reviews Neuroscience*, vol. 19, no. 2, p. 60, 2018.

[149] C. Madore, C. Baufeld, and O. Butovsky, "Microglial confetti party," *Nature neuroscience*, vol. 20, no. 6, p. 762, 2017.

[150] F. Rossi and C. Lewis, "Microglia's heretical self-renewal," *Nature neuroscience*, vol. 21, no. 4, p. 455, 2018.

[151] S. D. Bilbo, "The diverse culinary habits of microglia," *Nature neuroscience*, p. 1, 2018.

[152] P. Ayata, A. Badimon, H. J. Strasburger, M. K. Duff, S. E. Montgomery, Y.-H. E. Loh, A. Ebert, A. A. Pimenova, B. R. Ramirez, A. T. Chan *et al.*, "Epigenetic regulation of brain region-specific microglia clearance activity," *Nature neuroscience*, vol. 21, no. 8, p. 1049, 2018.

[153] T. Miconi, J. Clune, and K. O. Stanley, "Differentiable plasticity: training plastic neural networks with backpropagation," *arXiv preprint arXiv:1804.02464*, 2018.

[154] J. Piaget, "Piaget's theory," 1970.