Automated Reconstruction of Evolving Curvilinear Tree Structures

THÈSE N^{O} 6930 (2016)

PRÉSENTÉE LE 18 MARS 2016

À LA FACULTÉ INFORMATIQUE ET COMMUNICATIONS

LABORATOIRE DE VISION PAR ORDINATEUR

PROGRAMME DOCTORAL EN INFORMATIQUE ET COMMUNICATIONS

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

POUR L'OBTENTION DU GRADE DE DOCTEUR ÈS SCIENCES

PAR

Przemysław Rafał GŁOWACKI

acceptée sur proposition du jury:

Prof. P. Dillenbourg, président du jury
Prof. P. Fua, directeur de thèse
Dr F. Moreno-Noguer, rapporteur
Dr R. Sznitman, rapporteur
Prof. S. Süsstrunk, rapporteuse

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE
Suisse
2016

Acknowledgements

I would like to thank my supervisor, Prof. Pascal Fua for providing me with the opportunity to conduct scientific research in CVLab during these past few years. I would also like to extend my thanks to the other members of my defence committee: Prof. Pierre Dillenbourg, Prof. Sabine Süsstrunk, Dr. Francesc Moreno-Noguer and Dr. Raphael Sznitman.

My special thanks go to my girlfriend, Magda, whose continuous support allowed me to endure the most difficult times. I would also like to thank my parents and all the other friends and relatives for being there when I needed them.

Abstract

Curvilinear networks are prevalent in nature and span many different scales, ranging from micron-scale neural structures in the brain to petameter-scale dark-matter arbors binding massive galaxy clusters. Reliably reconstructing them in an automated fashion is of great value in many different scientific domains. However, it remains an open Computer Vision problem.

In this thesis we focus on automatically delineating curvilinear tree structures in images of the same object of interest taken at different time instants. Unlike virtually all of the existing methods approaching the task of tree structures delineation we process all the images at once. This is useful in the more ambiguous regions and allows to reason for the tree structure that fits best to all the acquired data. We propose two methods that utilize this principle of temporal consistency to achieve results of higher quality compared to single time instant methods.

The first, simpler method starts by building an overcomplete graph representation of the final solution in all time instants while simultaneously obtaining correspondences between image features across time. We then define an objective function with a temporal consistency prior and reconstruct the structures in all images at once by solving a mathematical optimization. The role of the prior is to encourage solutions where for two consecutive time instants corresponding candidate edges are either both retained or both rejected from the final solution.

The second multiple time instant method uses the same overcomplete graph principle but handles the temporal consistency in a more robust way. Instead of focusing on the very local consistency of single edges of the overcomplete graph we propose a method for describing topological relationships. This favors solutions whose connectivity is consistent over time. We show that by making the temporal consistency more global we achieve additional robustness to errors in the initial features matching step, which is shared by both the approaches. In the end, this yields superior performance.

Furthermore, an added benefit of both our approaches is the ability to automatically detect places where significant changes have occurred over time, which is challenging when considering large amounts of data.

We also propose a simple single time instant method for delineating tree structures. It computes a Minimum Spanning Arborescence of an initial overcomplete graph and proceeds to optimally prune spurious branches. This yields results of lower but still competitive quality compared to the mathematical optimization based methods, while keeping low computational complexity.

Our methods can applied to both 2D and 3D data. We demonstrate their performance in 3D on microscopy volumes of mouse brain and rat brain. We also test them in 2D on time-lapse images of a growing runner bean and aerial images of a road network.

Keywords: Computer vision, curvilinear networks, tubular structures, automated reconstruction, temporal consistency, integer programming.

Résumé

Les réseaux curvilignes sont dominants dans la nature et présents à toutes les échelles; leur étendue varie des structures neuronales de la taille du micron, jusqu'aux ségments de la matière noire qui lient des groupes de galaxies massifs. Les reconstruire avec précision de façon automatique est d'une énorme valeur dans de multiples domaines scientifiques. Cependant cela continue à être un problème ouvert de la Vision par ordinateur.

Dans cette thèse nous abordons le sujet de la délimitation automatique des structures arborescentes curvilignes visibles dans plusieurs images d'un même objet, mais prises à différents moments. Contrairement à la plupart des méthodes approchant la question de la délimitation des structures arborescentes, nous traitons toutes les images simultanément. Ceci est particulièrement utile dans les zones plus ambiguës et permet de créer une structure arborescente qui s'accorde le mieux avec les données. Nous proposons deux méthodes qui utilisent le principe de cohérence temporelle pour obtenir des résultats de meilleure qualité que celles qui basent sur l'instance unique.

Dans la première méthode, plus simple, nous commençons par construire un graphe saturé de la solution finale à tout instant. Simultanément, nous obtenons les correspondances entre les caractéristiques des images à travers le temps. Nous définions une fonction-objectif avec un terme de cohérence temporelle et reconstruisons ensuite les structures dans toutes les images simultanément en résolvant une optimisation mathématique. Le rôle du

terme de cohérence temporelle est de favoriser les solutions où pour deux instances consécutives, deux arrêtes potentielles sont soit toutes les deux retenues, soit toutes les deux rejetées dans la solution finale.

La deuxième méthode d'instances multiples fait appel au même principe du graphe saturé, mais gère la cohérence temporelle d'une manière plus robuste. Au lieu de se focaliser sur la cohérence très locale des arrêtes singulières du graphe saturé, nous proposons une méthode pour décrire des relations à portée plus longue. Ceci favorise les solutions dont la topologie est cohérente au cours du temps. Nous démontrons qu'en rendant la cohérence temporelle plus globale, notre méthode est plus résistante face aux erreurs des premières étapes de la reconnaissance des caractéristiques, qui est présente dans les deux approches. Cette méthode donne donc une performance supérieure.

En outre, un ultérieur aspect bénéfique de nos deux approches, est la possibilité de détecter de façon automatique tout endroit où des changements significatifs se produisent au cours du temps, ce qui facilite l'analyse d'un grand nombre de données.

Nous proposons aussi une méthode simple d'instance unique pour délimiter les structures arborescentes. Elle calcule une Arborescence Couvrante de Poids Minimal d'un graphe saturé initial et élimine les branches indésirables. Ceci donne des résultats de qualité inférieure, mais d'une valeur tout autant compétitive, comparé aux méthodes basées sur l'optimisation mathématique, tout en gardant une complexité basse.

Notre méthode peut être appliquée dans les données 2D et 3D. Nous démontrons leur efficacité en 3D sur des volumes microscopiques de cerveaux de souris et de rats. Nous les testons aussi en 2D sur des images séquentielles d'un haricot qui germe, ainsi que sur des vues aériennes de réseaux routiers.

Mots-clés: Vision par ordinateur, réseaux curvilignes, structures tubulaires, reconstruction automatique, cohérence temporelle, programmation linéaire en nombre entier

CONTENTS

List of Figures **xiii**

List of Tables **xv**

1 Introduction **1**

1.1 Applications 2

1.2 Contributions 6

1.3 Outline 7

2 Related work **9**

2.1 Manual and Semi-Automated Approaches 10

2.2 Automated Methods 11

3 Single time instant methods **17**

3.1 Tubular Graph 17

3.2 Reconstructing Tree Structures from Tubular Graph 25

3.3 Experiments and Results 33

4 Local Temporal Consistency **43**

4.1 Approach 44

4.2 Building Spatio-Temporal Graphs 48

4.3 Finding Temporally Consistent Trees 50

4.4 Experiments and Results 54

5 Topological Temporal Consistency **59**

5.1 Flow Variables and Temporal Consistency 59

5.2 Modeling Topological Consistency 61

5.3 Augmenting the Objective Function 64

5.4 Speeding Up the Computation 67

5.5 Results 68

6 Concluding Remarks **75**

6.1 Summary and Contributions 75

6.2 Limitations and Future Work 77

7 Appendices **81**

A Optimal Pruning of an Edge Pair Weighted Tree 81

B Tree Constraints in an Undirected Graph 84

References **87**

LIST OF FIGURES

1.1 Example curvilinear structures of scientific importance. 4

3.1 Tubular graph building algorithmic steps. 18
3.2 Scoring paths by classification vs Integration. 20
3.3 Feature extraction process. 22
3.4 Flowchart of the approach to extracting appearance features from tubular paths of arbitrary length. 23
3.5 An example initial graph. 27
3.6 A Minimum Spanning Arborescence computed on the example graph. . 27
3.7 Optimal pruning vs naive pruning. 28
3.8 An optimally pruned Minimum Spanning Arborescence of the example graph. 30
3.9 Flow variables in a directed graph. 34
3.10 An example Brainbow stack before and after pruning. 36
3.11 Another example Brainbow stack before and after pruning. 37
3.12 An example brightfield stack before pruning. 38
3.13 An example brightfield stack after pruning. 39
3.14 Another example brightfield stack before pruning. 40
3.15 Another example brightfield stack after pruning. 41

4.1 Reconstruction and automatic change detection using a time-lapse sequence for growing runner bean. 44
4.2 Key steps of the local consistency algorithm. 46
4.3 Fine alignment and automatic change detection. 47
4.4 Simultaneous sampling and matching. 48
4.5 An example spatio-temporal graph. 51
4.6 Example results visualisation for the local consistency method. 55
4.7 One of the 3D volumes used for training the path classifier for the brain datasets. 56

5.1 A small example graph illustrating the additional robustness of the topological consistency method. 60
5.2 Graphs illustrating situations considered to be temprorally consistent. . 62
5.3 Graphs illustrating situations considered to be temprorally inconsistent. 63
5.4 Road images used to train the path classifier for the aerial photographs dataset. 68
5.5 Example results visualisation for the topological consistency method on road data. 72
5.6 Example results visualisation for the topological consistency method on brain data. 73

6.1 Example of a reconstruction error that could be avoided by imposing a geometric prior. 78
6.2 Biconnected Components Decomposition. 79

7.1 Flow variables in an undirected graph. 86

LIST OF TABLES

2.1 Existing tools for reconstructing curvilinear structures 11

3.1 Experimental results for the pruned Minimum Spanning Arborescence method. 36

4.1 Experimental results for the local consistency method. 57

5.1 Experimental results for the local and the topological consistency method. 71

ONE

INTRODUCTION

Networks of curvilinear structures appear in many domains and reliably reconstructing them from images remains an open Computer Vision problem. So far, it has mostly been addressed in terms of modeling structures that have been captured at a specific moment in time. However, these networks, be they made of axons and dendrites seen *in vivo* in optical microscopy image stacks [32], blood vessels in retinal-scans [40], plant branches in time-lapse imagery [11], or roads in aerial images taken at long intervals [27], evolve over time. Modeling this evolution is of great scientific value to help understand underlying processes and analyzing the effects of biological [39] or geographic environmental conditions [52]. More broadly and as shown in Fig. 1.1, curvilinear networks are prevalent in nature and span many different scales, ranging from micron-scale neaural structures in the brain, through meter-scale road networks and all the way up to petameter-scale dark-matter arbors binding massive galaxy clusters. Efficient and robust methods for automatically delineating such structures could therefore accelerate scientific research in many domains.

1.1 Applications

The research presented here has been motivated by a number of applications, such as those depicted by Fig. 1.1. We briefly review here the ones we had an opportunity to directly work with. Others are presented to provide broader context.

1.1.1 Brain morphogenesis

One of the most promising domains of application for such a multi-frame modelling approach is tracking the evolution of neural networks in the brain. It might allow for better undestanding of the neuron forming and rewiring processes inside a brain during morphogenesis and their effect on the skills and behaviour of an animal.

It has been theorized that the pattern and extent by which a neuron projects its dendrites and axons maximizes its potential connectivity per given length of wire. Wen et al. [74] demonstrated the explanatory power of this model on basal dendritic arbors of pyramidal neurons. Individual cortical neurons have extensive axonal and dendritic arbors and thus harbor a large repertoire of potential connections. Only a small fraction of them, around 20%, are actual synapses [59]. As a result, a cortical neuron could, in principle, switch to other synaptic partners and thereby change its own or its partners receptive field through a small rewiring effort, e.g. simply by protruding dendritic spines, axonal boutons or through small changes in the trajectory of terminal branches. The capricious projection territories of dendrites and axons, which often fall outside of the functional cortical column from which they originate [23], make it difficult to predict the extent of structural rearrangement that is needed to support changes in circuit function.

To investigate these issues, we have been colaborating with neuroscientists from University of Geneva who are interested in the structural mechanisms and strategies that cortical neurons use to change circuit function. They provided us with in vivo 2-photon laser scanning micrographs of mouse brains that they were taking in weekly intervals in order to correlate structural and functional changes in the mouse cortex.

Images were taken through a permanently implanted cranial window, which allows for tracking identified structures over months during which the mouse learns new tasks or undergoes new experiences. An example image of that type is depicted by Fig. 1.1(a). Annotating the structures and detecting changes in the microscopic images occurring over time remains a bottleneck of studying the subject, which motivates the efforts to automatize the process.

1.1.2 Tracing road networks over time

Road network detection can be used for several applications. One example is the development of vision-based control strategies for unmanned aerial vehicles (UAVs), which may enable complex autonomous missions in environments where typical navigation systems like GPS are unavailable [28]. Another application is automated correction and updating of geographic information systems (GIS) from aerial images, which involves positionally correcting the existing roads location and generating hypotheses for new roads [27]. Automatic road delineation is also useful for integrating geo-spatial data coming from different sources. The easy accessibility of a wide range of such data allows to answer many queries that could not be answered using only a single data source. However, integrating information from all those sources is a challenging task and Computer Vision based methods for road delineation may help to automate the process [12].

We have used aerial photographs of the same region taken in long intervals of time. One such photograph of the Geneva area is shown in Fig. 1.1(g). The task we approached was to obtain more reliable results from multiple images that greatly differ in appearance. A leading cause of such appearance changes are seasonal variations and more specifically the extent of tree canopies, which create different occlusion patterns. Another important factor is the fact that the images were taken at different times of day, which produced different shading patterns. The changing conditions often affect

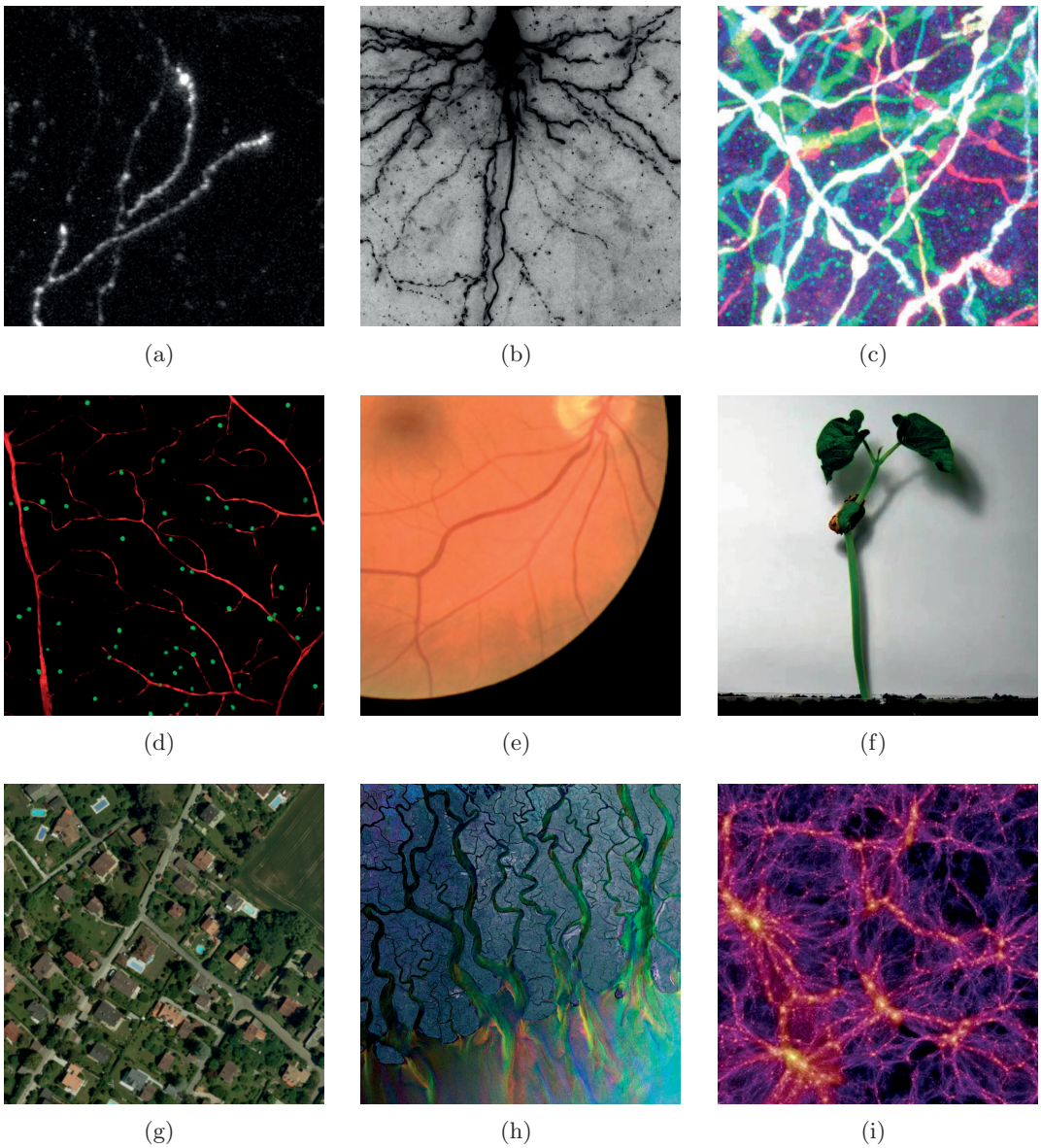


Figure 1.1: Example curvilinear structures of scientific importance. (a) A maximum intensity projection of a 2-photon micrograph of a rat brain. (b) A minimum intensity projection of a brightfield micrograph of a rat brain. (c) A maximum intensity projection of a micrograph of a rat brain. Neurons were genetically labelled with multiple, distinct colours using the Brainbow technique [42]. (d) A confocal micrograph of blood vessels and neurons. (e) An image of retinal vasculature obtained by a fundus camera. (f) A single frame from a time-lapse video of a growing runner bean. (g) An aerial image of a road network. (h) An image of the Ganges Delta taken by the Envisat Advanced Synthetic Aperture Radar [1]. (i) A visualisation of dark matter distribution in the universe at the present time produced by the Millennium Simulation [57].

different parts of the image and one might obtain more reliable results by looking at the image evidence in the multiple time instants all at once.

1.1.3 Time-lapse videos of growing plants

Time-lapse imaging has been applied in agriculture for capturing the growth of plants. It allows the creation of realistic plant development models. Modeling plant architecture is an active research area and some empirical models have already achieved the predictive value needed in practical applications [52]. However, the development of well calibrated empirical models of specific plants remains a labor-intensive task and construction of reliable mechanistic models is a current research problem. Computer Vision techniques could accelerate the research by limiting the amount of manual labor.

Obtaining reliable models has a number of practical applications. It allows research into optimal conditions, such as pruning and pinching, temperature and day length manipulation, application of chemicals, to achieve desirable plant size, shape, quality etc. In forestry it helps to identify optimal strategies for pruning and spacing trees. It may also broaden the understanding of disease dynamics through simulation of pathogen deposition and growth. Another application is the development of new or more ecofriendly ways to biologically control weeds. It might also allow for better management of pests by improving our knowledge of the action thresholds through simulation of interactions between plant architecture, pesticide deposition, insect movement and feeding, and compensatory growth of plants. In developmental biology, one might also explore hypotheses relation physiology of plant development and information in genes to integrated 3D structures.

We have obtained a time-lapse video of a growing runner bean in order to explore this area of application and to prove that the presented multi-frame methods are indeed generic in terms of the problem domain. An example frame from this video is presented in Fig. 1.1(f).

1.2 Contributions

The contributions of this thesis are as follows.

Simultaneous reconstruction at multiple time instants by enforcing local temporal consistency. We present a method for reconstructing curvilinear tree structures from multiple images of the same region of interest taken at different time instants. Instead of solving in every image independently, we take a more global approach of processing all time instants at once while enforcing temporal consistency of local corresponding areas. We start by building an overcomplete representation of the final solution in all time instants while simultaneously obtaining correspondences between image features across time. We then solve a mathematical optimization while intorducing a temporal consistency prior in the objective function. Its role is to encourage solutions where for two consecutive time instances corresponding short candidate edges are either both retained or both rejected from the final solution. We demonstrate that by doing so we can recover from some reconstruction errors by referring to neighboring images for additional evidence.

Reconstructing tree structures in multiple time instants by enforcing topological temporal consistency. We introduce a second, more robust way of reconstructing tree structures in multiple images. Instead of focusing on the very local consistency of single edges of the overcomplete graph we propose a method for describing relationships between spatially distant parts of a network in a specific time instant. We then express topological similarity of the networks in consecutive time instants in the form of a temporal consistency prior, which favors solutions whose topology is consistent over time. We demonstrate that the method is in fact a generalization of its local counterpart. We show that by making the temporal consistency more global we achieve additional robustness to errors in the initial features matching step, which is shared by both the approaches. In the end, this yields superior performance.

1.3 Outline

The remainder of the this thesis is organized as follows.

In **Chapter 2**, we present the state of the art in curvilinear structures delineation. We cover mostly single time instant methods, as we are not aware of any competing multi-frame ones that are currently published.

In **Chapter 3**, we give a detailed overview of a method directly related to the ones proposed in this work. Understanding it is required to comprehend the key ideas behind the ways of enforcing temporal consistency explained later on. It is a single time instant method, for which we proposed a simplified but much faster version. It is based on tree pruning and can produce delineations whose quality is only slightly less than that of the original method. This is valuable when dealing with large stacks, for which the original method tends to be very slow.

In **Chapter 4**, we present our local consistency mathematical optimization approach to curvilinear structures delineation in multiple time instants. We demonstrate its performance on a simple time-lapse sequence of a growing runner bean and on sequences of 3D 2-photon laser scanning microscopy volumes of mouse brain taken at different time instants.

In **Chapter 5**, we introduce our more advanced method for enforcing temporal consistency between different time instants by applying the topological temporal prior. We show that it is in fact a generalisation of its local counterpart and we exaplain in what ways it brings more robustness. Again, we demonstrate its performance on sequences of 3D microscopy volumes of mouse brain and also on a sequence of challenging road network images of the same area taken at long time intervals with varying illumination conditions and oclusion patterns.

Chapter 6 provides concluding remarks about the contributions of the presented work, its failure modes and ideas for future work.

TWO

RELATED WORK

Even though imposing temporal consistency is well-known to increase the robustness of video-based object tracking [37] or 3D body pose estimation [3, 53], we do not know of any other automated delineation algorithm besides our own that exploits it when working with time-lapse imagery.

The method of [39] explores sequences of confocal microscopy images of the brain but it attempts to achieve a different goal. It does not try to improve robustness of the tree structures reconstruction procedure by enforcing temporal consistency between consecutive time instants. Instead, dendritic spines are traced individually. Only then is spatial and structural information between two dendritic structures used to establish correspondences. What is more it searches for correspondences only between the dendric spines, which makes it very domain-specific and not easily generalizable to other domains of applicaiton.

In the absence of direct competitors, in this chapter we review briefly single time-frame delineation methods. We first review the manual and semi-automated approaches. We then proceed to describing automated methods which, depending on the search mechanism they rely on, we divide into two categories, the faster but less robust local search methods and the more robust but slower global ones.

2.1 Manual and Semi-Automated Approaches

Manual delineation tools require the cylindrical compartments to be sequentially provided by a user starting from the structure roots, such as soma for neural micrographs or optic disc for fundus scans, and ending at branch tips [6, 7, 25, 47]. Since they tend to be labor-intensive and time consuming, these tools are most often used for correcting the reconstructions obtained by more automated ones.

Semi-automated techniques, on the other hand, use a *tubularity* measure to reduce user input to a handful of carefully selected seed points to be manually specified. These points are linked with paths that tightly follow high tubularity pixels or voxels. Most methods employ an interactive and sequential procedure, where the user specifies one seed point at a time and the algorithm constructs a high-tubularity path that links the given seed to the current reconstruction [22, 43, 45, 47]. Other techniques include those that prompt the user for a new seed only when the algorithm is stuck [58] or require only the root and branch endpoints [6, 49, 50].

In Table 2.1, we list some of the existing software tools that implement these approaches. The table includes several key features such as the level of automation, the type of outputs obtained and the ability to handle 3D stacks.

Tool	**Type**	**Output**	**Platf.**	**Dim.**	**Color**	**Radii**
AxonTracker [58]	S	T: Greedy tracking	W	3D	no	no
Imaris [6]	M/S/A	T: Fast marching minimal path in image space	M/W	2D/3D	no	yes
Farsight [68, 69]	A	T: Open-curve snake	L/M/W	3D	no	yes
Geodesic-SNT [64]	S	T: Fast marching minimal path with a color prior in scale space	L/M/W	2D/3D	yes	yes
HCA-Vision [67]	S	S	W	2D	no	no
NeuriteTracer [51]	A	S: Thresholding and skeletonization	L/M/W	2D	no	no
Neurolucida [44]	S/A	T	W	2D/3D	X	yes
Neuromantic [47]	M/S	T: Dijkstra shortest path in image space	W	2D/3D	no	yes
Neuron_Morpho [7]	M	T	L/M/W	2D/3D	no	no
NeuronGrowth [22]	S	T: Greedy tracking based on Hessian eigenvector directions	L/M/W	2D+t	no	no
NeuronJ [45]	S	T: Dijkstra shortest path in image space	L/M/W	2D	no	no

Tool	Type	Output	Platf.	Dim.	Color	Radii
NeuronStudio [70]	A	T: Thresholding, skeletonization and Rayburst sampling for radius estimation	W	2D/3D	no	yes
NCTracer [14]	A	T: Voxel coding applied to binarized stacks	W	3D	no	yes
Reconstruct [25]	M/S	T: Region growing	W	2D/3D	yes	no
Simple Neurite Tracer (SNT) [43]	S	T: Bidirectional A* search	L/M/W	3D	no	yes
TrakEM2 [10]	M	T (treeline object)	L/M/W	2D/3D	no	no
TREES Toolbox [16]	M/A	T: Thresholding and skeletonization	L/M/W	2D/3D	no	yes
Vaa3D [49, 50, 75]	S, A	T: Dijkstra shortest path in image space	L/M/W	3D	no	yes

Table 2.1: Existing tools for reconstructing curvilinear structures. From left to right, the columns list tool name with a reference, algorithm type (M: manual, S: semi-automated, A: Automated), output type and short description of the algorithm (T: tracing=vector graphics, S: segmentation=binary or label image), tubularity measure, supported platforms (L: Linux, M: Apple Macintosh, W: Microsoft Windows), supported image dimensions, whether color information is taken into account in the processing, whether radius estimates are automatically produced. X denotes unknown.

2.2 Automated Methods

There has recently been a resurgence of interest in automated delineation techniques because extracting curvilinear structures automatically and robustly is of fundamental relevance to many scientific disciplines. Similar to the semi-automated approaches, most automated techniques rely on a tubularity measure to trace the curvilinear structures. However, they require only a single seed point, the root, to be specified for each connected network in the image. These points can alternatively be automatically generated by finding the local maxima of the tubularity measure or using separate detection procedures, including optic disc and soma detectors. Starting from the roots, they then grow branches that follow high tubularity paths in the image. Depending on the search mechanism employed, existing algorithms can be categorized as either local or global.

2.2.1 Local Search Methods

They make greedy decisions about which branches to retain based on local image evidence. They include methods that segment and skeletonize the tubularity image [36, 48, 51, 70, 71, 76], and active contour-based methods, which are initialized from it [8, 14, 66]. Such methods have been shown to be effective and efficient when a very good segmentation can be reliably obtained. In practice, however, they tend to be prone to errors caused by noise and imaging artifatcs.

Another important class of local approaches involves explicitly delineating the curvilinear networks. It includes greedy tracking methods that start from a set of seed points and incrementally grow branches by evaluating a tubularity measure [2, 4, 9, 13, 46, 77]. High tubularity paths are then iteratively added to the solution and their end points are treated as the new seeds from which the process can be restarted. These techniques are computationally efficient because the tubularity measure only needs to be evaluated for a small subset of the image volume near the seeds. However, these approaches typically require separate procedures to detect branching points. Furthermore, due to their greedy nature, imaging artifacts and noise can produce local tracing errors that propagate. This often results in large morphological mistakes, especially when there are extended areas where image signal to noise ratio is poor.

2.2.2 Global Search Methods

They aim at greater robustness by optimizing a global objective function over a graph of high-tubularity seed points [65] or superpixels [72]. Typically, they compute the tubularity measure everywhere. Although this is more computationally demanding, it can still be done efficiently in Fourier space or using GPUs [18, 34, 35]. Markov Chain Monte Carlo algorithms, for instance, belong to this class [21, 60]. These methods explore the search space efficiently by first sampling seed points and linking them, and then iteratively adjusting their positions and connections so as to minimize their objective function. However, while they produce smooth tree components in general,

the algorithms presented in [21, 60] do not necessarily guarantee spatial connectedness of these components.

By contrast, many other graph-based methods use the specified roots to guarantee connectivity. They sample seed points and connect them by paths that follow local maxima of the tubularity measure. This defines a graph whose vertices are the seeds and edges are the paths linking them. The edges of the graph are assumed to form an overcomplete representation of the underlying curvilinear structures and the final step is to build a solution by selecting an optimal subset of these candidate edges.

Many existing approaches weigh the edges of this graph and solve a minimum-weight tree problem. Algorithms that find a Minimum Spanning Tree (MST) [16, 26, 68, 79] or a Shortest Path Tree (SPT) [49] belong to this class. Although efficient polynomial-time algorithms exist for both SPT- and MST-based formulations, these approaches suffer from the fact that they must span all seed points, including some that might be false positives. As a result, they produce spurious branches when seed points that are not part of the tree structure are mistakenly detected, which happens all too often in noisy data. We have, however, explored the MST-based techniques and proposed an efficient algorithm that first computes a Minimum Spaning Tree and then proceeds to pruning the spurious branches. It can produce delineations whose quality is only slightly less than that of the more computationally demanding methods described below. This is valuable when dealing with large stacks, for which the more sophisticated methods tend to be slow.

The k-Minimum Spanning Tree (k-MST) formulation [65] addressed the issue of spurious branches by posing the problem as one of finding the minimum cost tree that spans only an *a priori* unknown subset of k seed points. However, it relies on a heuristic search algorithm and a dual objective function, one for searching and the other for scoring, without guaranteeing the global optimality of the final reconstruction. Furthermore, it requires an explicit optimization over the auxiliary variable k, which is not relevant to the problem.

By contrast, the Integer Programming formulation introduced in [63] involves minimizing a single global objective function that allows for linking legitimate seed points while rejecting spurious ones by finding the optimum solution to within a small user-specified tolerance. It also introduces a robust Machine Learning method for classifying tubular paths. Since no polynomial time algorithm is known for solving the resulting problem, mathematical optimization techniques need to be employed to find the final solution. The additional computational complexity is justified by the optimality guarantees. Furthermore, this mathematical optimization approach generalizes elegantly into our temporal-consistency framework. For this reason we provide a more detailed overview of the method in Chapter 3.

The very recent method of [62] can be viewed as a generalisation of [63]. It introduces a similar Integer Programming formulation and attempts to solve the delineation problem optimally. Unlike earlier approaches that assume a tree topology for the networks, it explicitly models the fact that the networks may contain loops, and can reconstruct both cyclic and acyclic ones. In case of acyclic networks the proposed formulation provides an additional level of robustness compared to [63]. The multiple time instant approaches introduced in this thesis could be adapted to benefit from this additional robustness. Since it is a very recent method, we discuss this in the Limitations and Future Work section of Chapter 6.

All the MST and SPT approaches rely on local tubularity scores to weight the graph edges. For example, global methods that rely on geodesic distances express this cost as an integral of a function of the tubularity values [38, 79]. Similarly, active contour-based methods typically define their energy terms as such integrals over the paths [35, 68]. According to [65], because they involve averaging, such measures are not particularly effective at ignoring paths that are mostly on the curvilinear structures but also partially on the background. Moreover, because the scores are computed as sums of values along the path, normalizing them so that paths of different lengths can be appropriately compared is non-trivial. By contrast, the path classification approach

introduced in [63] returns much more discriminative probabilistic costs, which can be compared for paths of arbitrary length. We therefore incorporate it into all of the algorithms presented in this thesis, as it is critical to obtaining good results on noisy data.

THREE

SINGLE TIME INSTANT METHODS

In this chapter we focus on the typical scenario of delineating curvilinear structures in single time instants. We first present the idea of a tubularity graph introduced in [63], which is an important concept in the remainder of the thesis. We then present two different methods to obtain the final reconstructions from the tubular graph. The original method of [63] involves performing a mathematical optimization to solve the delineation problem exactly. The multiple time instant methods described in the next two chapters are directly related to it. Therefore, understanding it is required to comprehend the key ideas behind the ways of enforcing temporal consistency explained later on. We also present a simplified, faster version of this method. It can produce delineations whose quality is only slightly less than that of the original method. This is valuable when dealing with large stacks in scenarios where one can sacrifice a small decrease in accuracy to obtain the delineation a lot faster.

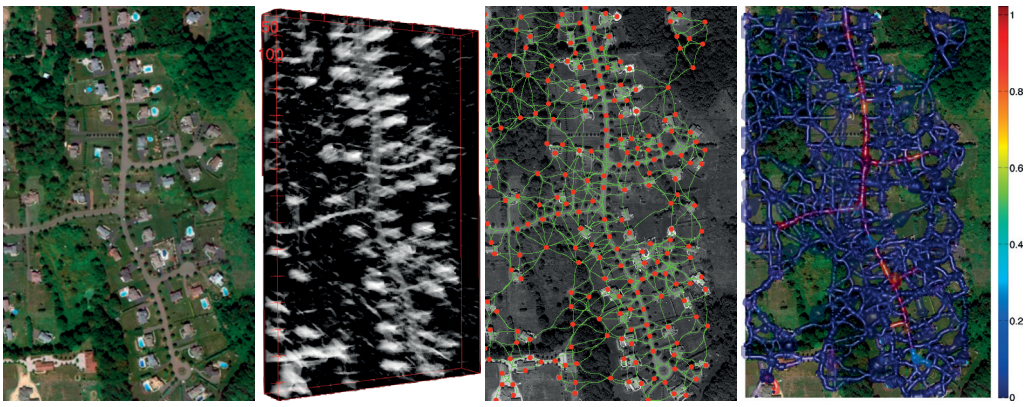
3.1 Tubular Graph

We first briefly describe the tubularity graph of [63]. It provides a mathematical model of the image data that is then processed to obtain the final results. The graph building procedure consists of the following steps depicted by Fig. 3.1:

1. We compute a *tubularity* value at each image location $\mathbf{x}_i$ and radius value r_i, which quantifies the likelihood that there exists a tubular structure of radius r_i, at location $\mathbf{x}_i$. Given an N-D image, this creates an $(N+1)$-D scale-space tubularity volume such as the one shown in Fig. 3.1(b).

2. We select regularly spaced high-tubularity points as seed points and connect pairs of them that are within a given distance from each other. This results in a directed tubular graph, such as the one of Fig. 3.1(c), which serves as an overcomplete representation for the underlying curvilinear networks.

3. Having trained a path classifier using such graphs and ground-truth trees, we assign probabilistic weights to pairs of consecutive edges of a given graph at detection time as depicted by Fig. 3.1(d).



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

Figure 3.1: Algorithmic steps. (a) A 2D aerial image of a suburban neighborhood. (b) 3-D scale-space tubularity image. (c) Graph obtained by linking the seed points. They are shown in red with the path centerlines overlaid in green. (d) The same graph with probabilities assigned to edges using our path classification approach. Blue and transparent denote low probabilities, red and opaque high ones. Note that only the paths lying on roads appear in red.

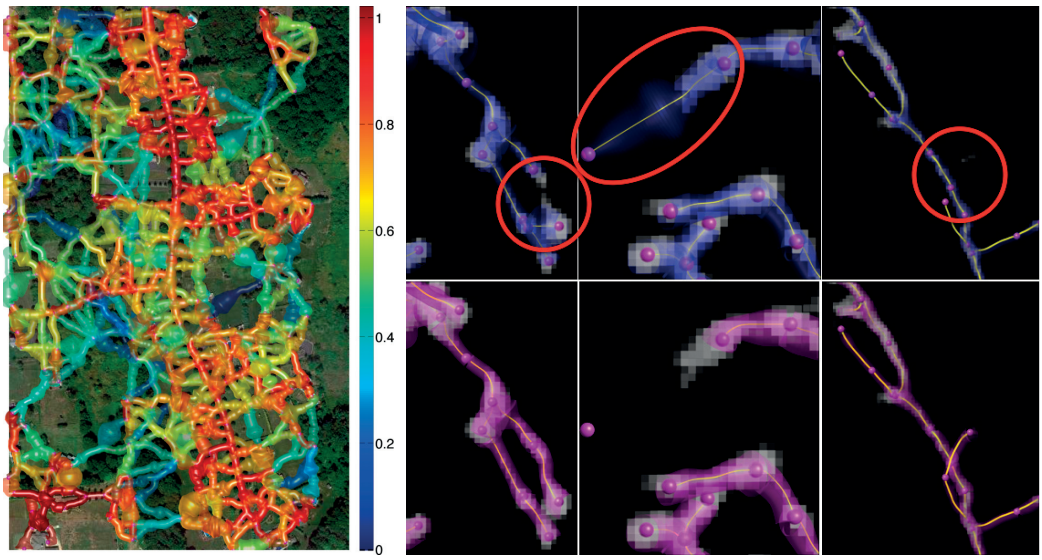
3.1.1 Graph Construction

Let us now explain the graph construction procedure in more detail. We first compute the Multi-Directional Oriented Flux tubularity measure introduced in [61]. This measure is used to assess if a voxel lies on a centerline of a filament at a given scale. Unlike the Oriented Flux approach [35] that relies on a circular model of the cross-sections, this measure allows for arbitrarily-shaped ones that are prevalent in biological imagery. This is achieved by maximizing the image gradient flux along multiple directions and radii, instead of only two with a unique radius. We then suppress non-maxima responses around filament centerlines using the NMST algorithm [61], which helps remove some artifacts from further consideration. More specifically, the NMST algorithm suppresses voxels that are not local maxima in the plane that is perpendicular to a local orientation estimate and within the circular neighborhood defined by their scale. It then computes the MST of the tubularity measure and links pairs of connected components in the non-maxima suppressed volume with the MST paths.

Next, we select maximum tubularity points in the resulting image and treat them as graph vertices, or *seeds*, to be linked. They are selected greedily, at a minimal distance of d to every point that was selected previously. Finally, we compute paths linking every pair of seed points within a certain distance $l(d)$ from each other using the minimal path method in the scale space domain [38]. We take the *geodesic tubular path* connecting vertices i and j to be

$$p_{ij} = \underset{\gamma}{\operatorname{argmin}} \int_0^1 \mathcal{P}\left(\gamma(s)\right) \mathrm{d}s, \tag{3.1}$$

where $\mathcal{P}$ is the negative exponential mapping of the tubularity measure, $s \in [0, 1]$ is the arc-length parameter and γ is a parametrized curve mapping s to a location in $\mathbb{R}^{N+1}$ [5]. The first N dimensions are spatial ones while the last one denotes the scale. The minimization relies on the Runge-Kutta gradient descent algorithm on the geodesic distance, which is computed using the Fast Marching algorithm [56].



(a) (b)

Figure 3.2: Scoring paths by classification vs Integration. (a) Tubular graph of Fig. 3.1(c) with edge weights computed by integrating tubularity values along the paths instead of using the path classification approach. We use the same color scheme as in Fig. 3.1(d) to demonstrate how much less informative these weights are. (b) In microscopy stacks scoring paths by summing tubularity values results in, from left to right, shortcuts, spurious branches, and missing branches denoted by the red circles at the top. Using the classification approach to scoring paths yields the right answer in all three cases, as shown in the bottom row.

3.1.2 Path Classification

Once the graph has been built, a key component of the algorithm is an approach to assigning weights to its edges, or more specifically pairs of consecutive edges. A standard approach to computing such weights is to integrate a function of the tubularity values along the paths, as in Eq. 3.1. However, as shown in Fig. 3.2(a), the resulting estimates are often unreliable because a few very high values along the path might offset low values and, as a result, fail to adequately penalize spurious branches and short-cuts. Furthermore, it is often difficult to find an adequate balance between allowing paths to deviate from a straight line and preventing them from meandering too much, which results in errors such as those depicted by the top row of Fig. 3.2(b) when using them to find the optimal subgraph.

In this section, we outline the path-classification approach of [63] to computing the probability estimates that proves to be more reliable. More specifically, given a tubular path computed as discussed in Section 3.1.1, we break it down into several segments and compute one feature vector for each based on gradient histograms. We then use an embedding approach [73] to compute fixed-size descriptors from the potentially arbitrary number of feature vectors we obtain. Finally, we feed these descriptors to a classifier and turn its output into a probability estimate.

As shown in the bottom row of Fig. 3.2(b), this approach penalizes paths that mostly follow the tree structure but cross the background. Thus, it discourages shortcuts and spurious branches, which the integration approach along the path fails to do.

In the remainder of this section, we describe the path features, embedding scheme, and training data collection mechanism in more detail.

3.1.2.1 Histogram of Gradient Deviation Descriptors

Gradient orientation histograms have been successfully applied to detecting objects in images and recognizing actions in videos [17, 24, 73]. In a typical setup, the image is first divided into a grid of fixed-size blocks, called cells, and then for each cell, a 1-D histogram of orientated gradients (HOG) is formed from the pixel votes within it. Histograms from neighboring cells are then combined and normalized to form features invariant to local contrast changes. Finally, these features are fed into a classifier to detect objects of interest. We adapt this strategy for tubular paths by defining *Histogram of Gradient Deviation* (HGD) descriptors as follows.

Given a tubular path $\gamma(s)$ such as the one depicted by Fig. 3.3, with s being the curvilinear abscissa, let $\mathcal{C}(s)$ be the centerline and $r(s)$ the corresponding radius mappings. We partition the path into equal-length overlapping segments $\gamma_i(s)$ and, for each, we compute histograms of angles between image gradients $\nabla I(\mathbf{x})$ and normal vectors $\mathbf{N}(\mathbf{x})$ obtained from the points $\mathbf{x} \in \gamma_i(s)$ within the tubular segment. To ensure that all the gradient information surrounding the path is captured, we enlarge the segments

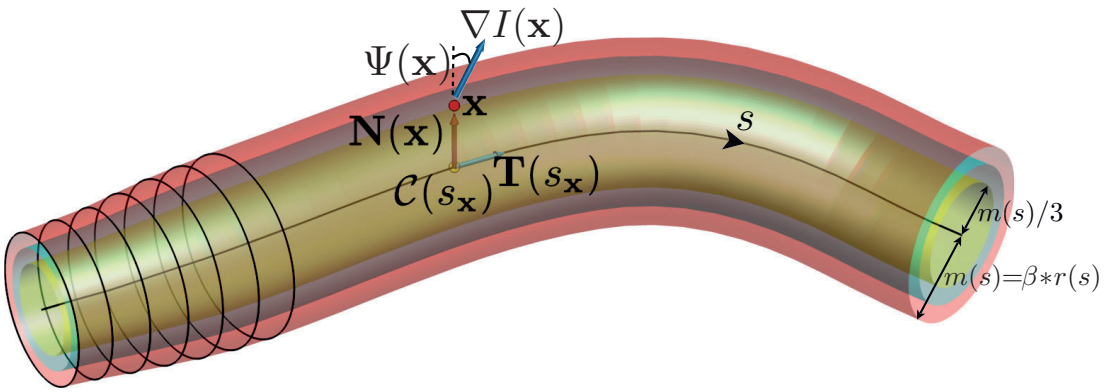


Figure 3.3: Three aspects of the feature extraction process. An extended neighborhood of points around the path centerline $\mathcal{C}(s)$ is defined as the envelope of cross-sectional circles shown in black. This neighborhood is divided into R radius intervals highlighted by the yellow, green and red tubes (here $R = 3$) and a histogram is created for each such interval. A point $\mathbf{x}$ contributes a weighted vote to an angular bin according to the angle between the normal $\mathbf{N}(\mathbf{x})$ and the image gradient $\nabla I(\mathbf{x})$ at that point.

by a margin $m(s) = \beta * r(s)$ proportional to the radius values. A fixed size margin is not preferred to avoid biased interference of background gradients for thin paths.

Furthermore, to obtain a description of paths' cross-sectional appearance profile, we split its tubular segments into R equally spaced radius intervals as shown in Fig. 3.3 and create two histograms for each such interval. While the first histogram captures the *strength* of the gradient field inside an interval, the second one captures its *symmetry* about the segment centerline.

Given a point $\mathbf{x} \in \gamma_i(s)$, let $\mathcal{C}(s_\mathbf{x})$ be the closest centerline point, and $\mathbf{N}(\mathbf{x})$ be the *normal ray vector* from $\mathcal{C}(s_\mathbf{x})$ to $\mathbf{x}$, as illustrated by Fig. 3.3. Each such point contributes a vote to a bin of the *gradient-strength* and *gradient-symmetry* histograms. The bin is determined by the following equation:

$$\Psi(\mathbf{x}) = \begin{cases} \text{angle}(\nabla I(\mathbf{x}), \mathbf{N}(\mathbf{x})) \text{, if } \|\mathbf{x} - \mathcal{C}(s_\mathbf{x})\| > \varepsilon \\ \text{angle}(\nabla I(\mathbf{x}), \mathbf{\Pi}(\mathbf{x})) \text{, otherwise,} \end{cases} \tag{3.2}$$

where $\mathbf{\Pi}(\mathbf{x})$ is the cross-sectional plane, which we use to compute the deviation angle in the special case when $\mathbf{x}$ belongs to the centerline and the normal ray vector is not defined. The votes are weighted by $\|\nabla I(\mathbf{x})\|$ and $\sqrt{< -\nabla I(\mathbf{x}), \nabla I(\mathcal{C}(s_\mathbf{x}) - \mathbf{N}(\mathbf{x})) >}$ for

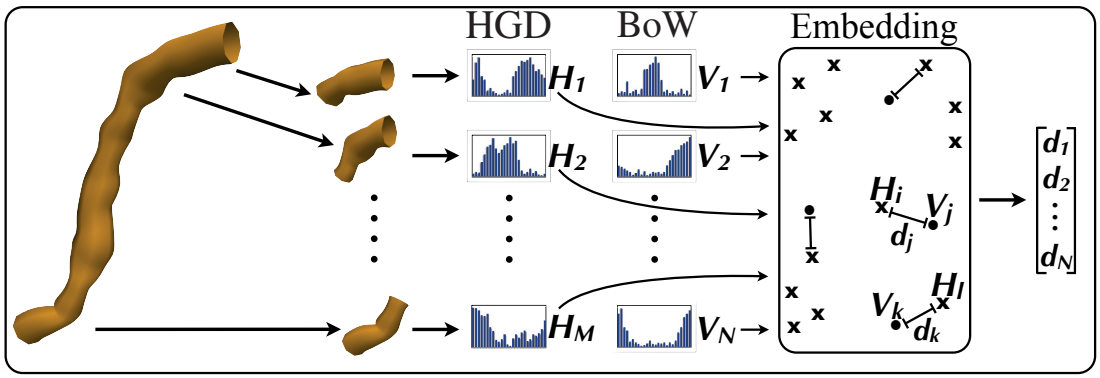


Figure 3.4: Flowchart of the approach to extracting appearance features from tubular paths of arbitrary length.

the *gradient-strength* and *gradient-symmetry* histograms respectively.

We compute the radius interval and the angular bin indices for a point $\mathbf{x}$ respectively as $\min(R-1, \lfloor R\|\mathbf{N}(\mathbf{x})\|/(r(s_\mathbf{x})+m(s_\mathbf{x}))\rfloor)$ and $\min(B-1, \lfloor B\Psi(\mathbf{x})/\pi\rfloor)$, where B is the number of histogram bins. For each segment, this produces R pairs of histograms, each one corresponding to a radius interval. Finally, we normalize each histogram by the number of points that voted for it.

This yields a set of histograms for each segment, which we combine into a single HGD descriptor.

3.1.3 Embedding

The above procedure produces an arbitrary number of HGD descriptors per path. To derive a fixed-size descriptor from them, we first use a Bag-of-Words (BoW) approach to compactly represent their feature space. The words of the BoW model are generated during training by randomly sampling a predefined number of descriptors from the training data. For a given path of arbitrary length, we then compute an embedding of the path's HGD descriptors into the *codewords* of the model. As illustrated in Fig. 3.4, computing the embedding amounts to finding the minimum Euclidean distance from the descriptors to each word in the model. The resulting feature vector of distances has the same length as the number of elements in the BoW model.

To account for geometry and characterize paths that share a common section we incorporate into these descriptors four geometry features, the maximum curvature along the centerline curve $\mathcal{C}(s)$, its tortuosity, and normalized length along the z and the scale axes defined in Section 3.1.1. For a path of length L in world coordinates and distance d between its start and end points, these features are defined respectively as argmax$\|\mathbf{T}'(s)\|$, d/L, Δ_z/L and Δ_r/L, where Δ_z and Δ_r are the path lengths along the z and the scale axes, and $\mathbf{T}(s)$ is the unit tangent vector depicted in Fig. 3.3.

3.1.3.1 Training

Given a set of training images $\{I_i\}$, the associated ground truth tracings $\{H_i\}$ annotated manually and tubular graphs $\{G_i\}$ obtained using the method of Section 3.1.1, we sample an equal number of positive and negative paths from each pair $\{H_i, G_i\}$. To train the path classifier, we obtain positive samples by simply sampling the ground truth delineations $\{H_i\}$ associated with our training images. We obtain negative samples by first randomly selecting *candidate* paths from a tubular graph G_i, and then attempting to find *matching* paths in the corresponding ground truth H_i. The *candidate* paths are considered as negatives if they satisfy certain incompatibility conditions, such as having a small overlap with their respective *matching* ones.

More specifically, given a randomly selected *candidate* path p_g of a graph G_i, we find its *matching* p_h by first finding the two centerline points in H_i that are closest to the start and end points of p_g in the world coordinates of the image I_i and then extracting the shortest directed path p_h connecting these points in H_i.

Let $l(p)$ denote the centerline length of a path p, and $l_{p_1}(p_2)$ denote the length of p_2's longest segment that is outside the volume enclosed by p_1. We consider p_g as a negative example if it satisfies at least one of the following criteria:

1. $max(l_{p_g}(p_h), l_{p_h}(p_g))$ is larger than a length threshold t_l, which is taken to be the minimum image spacing in our experiments.

2. The ratio $min(l(p_h), l(p_g))/max(l(p_h), l(p_g))$ is smaller than a threshold, which we set to 0.75. This is to detect if p_g is meandering too much with respect to p_h.

3. Volumetric intersection of p_h and p_g over their union is smaller than a threshold, taken to be 0.5 in our experiments.

We label those negatives that partially overlap with the *matching* paths as hard-to-classify examples, and those that do not overlap as easy-to-classify ones. In our experiments, hard-to-classify examples constitute 99 percent of all the negative examples.

We choose the path lengths randomly from a probability distribution built from the consecutive edge pair lengths of the graphs $\{G_i\}$ in the training dataset. This is achieved by first labeling the edge pairs as positive or negative using the above procedure and then constructing two separate distributions, one for each class, using the assigned labels.

Finally, at detection time, we run the path classifier on consecutive edge pairs and assign to them the resulting probabilities of belonging to the underlying curvilinear networks.

3.2 Reconstructing Tree Structures from Tubular Graph

In this section we first present an efficient, polynomial-time algorithm for obtaing the final results from the overcomplete graph. It first computes a Minimum Spanning Arborescence of the graph and then performs an optimal pruning. The algorithm does not have optimality guarantees but it is valuable when dealing with large stacks in scenarios where one would like to obtain results very fast. We then outline the original Quadratic Mixed Integer Programming method of [63], which solves the delineation task optimally.

3.2.1 Pruning a Minimum Spanning Arborescence

Let $\mathcal{G} = (\{\mathcal{X} = \mathbf{x}_i\}, \{\mathcal{E}_s = \mathbf{e}_{ij}\})$ be the overcomplete directed tubular graph, with vertices $\mathcal{X}$ and edges $\mathcal{E}_s$. Given the probabilities p_{ijk} produced by the path classifier approach of [63] we would like to reason for the most probable solution to our problem. Let $\{Y = y_{ij}\}$ be the set of binary variables indicating the presence or absence of specific edges in the final solution. More specifically, for every directed edge $e_{ij} \in \mathcal{E}_s$ we create a single binary variable y_{ij} whose value is equal to one if the edge is taken taken in the solution or zero if it is left out of it. As shown later in this chapter, this amounts to solving the following optimisation problem

$$y^* = \arg\max_{y \in \mathcal{Y}} P\left(\mathcal{I}, \mathcal{X}, \mathcal{E}_s | Y = y\right) \ , \tag{3.3}$$

$$= \arg\min_{y \in \mathcal{Y}} \sum_{\mathbf{e}_{ij}, \mathbf{e}_{jk} \in \mathcal{E}_s} w_{ijk} y_{ij} y_{jk}, \tag{3.4}$$

where $w_{ijk} = -\log \frac{p_{ijk}}{1-p_{ijk}}$, p_{ijk} is the probability of edge pair $(\mathbf{e}_{ij}, \mathbf{e}_{jk})$ being part of a tubular structure, and $\mathcal{Y}$ is the set of all feasible trees with root $\mathbf{x}_r$.

Unfortunately, no polynomial-time algorithm is known for solving this edge-pair weighted Minimum Arborescence problem. To obtain our solution fast we compute a Minimum Spanning Arborescence instead, that is a Minimum Arborescence that spans all vertices of the graph. As mentioned in Chapter 2, a common shortcoming of many such spanning-based techniques is that they retain many spurious branches. Therefore, the Minimum Spanning Arborescence serves as an initial solution that provides the general topology and we proceed to prune it optimally.

Once again, in case of edge-pair weighted graphs there are no polynomial-time algorithms available to compute a Minimum Spanning Arborescence. This is why we alter the overcomplete graph by putting probabilities p_{ij} on edges instead of edge-pairs. In such a setting the weight of any given tree is defined as $\sum_{\mathbf{e}_{ij} \in \mathcal{E}_s} w_{ij} y_{ij}$, where $w_{ij} = -\log\left(\frac{p_{ij}}{1-p_{ij}}\right)$ is a weight associated with edge $\mathbf{e}_{ij}$. We use the ChuLiu/Edmonds'

algorithm [15, 20] to compute our spanning solution. An example initial graph is presented in Fig. 3.5 and its Minimum Spanning Arborescence is presented in Fig. 3.6.

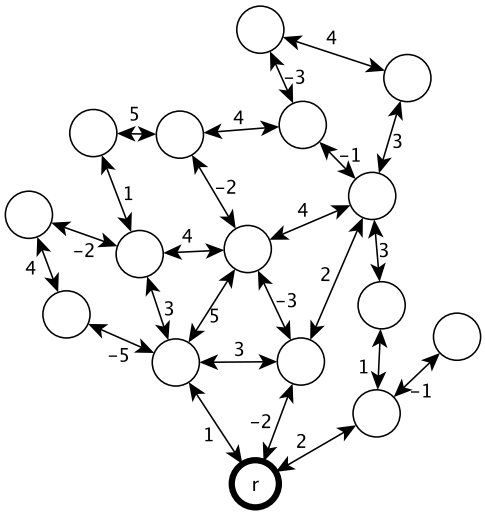


Figure 3.5: An example initial graph with the bottom vertex being the root. The numbers on the edges represent edge weights, that is, the negative log likelihood ratios.

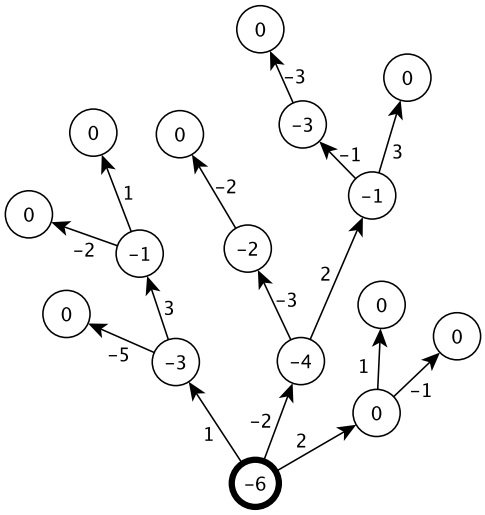


Figure 3.6: A Minimum Spanning Arborescence computed on the example graph of Fig. 3.5. Note that all of the vertices of the initial graph have been retained. The numbers on the vertices represent the weights of the subtrees rooted at them. The number on the root vertex represents the weight of the whole Minimum Spanning Arborescence, which is equal to -6.

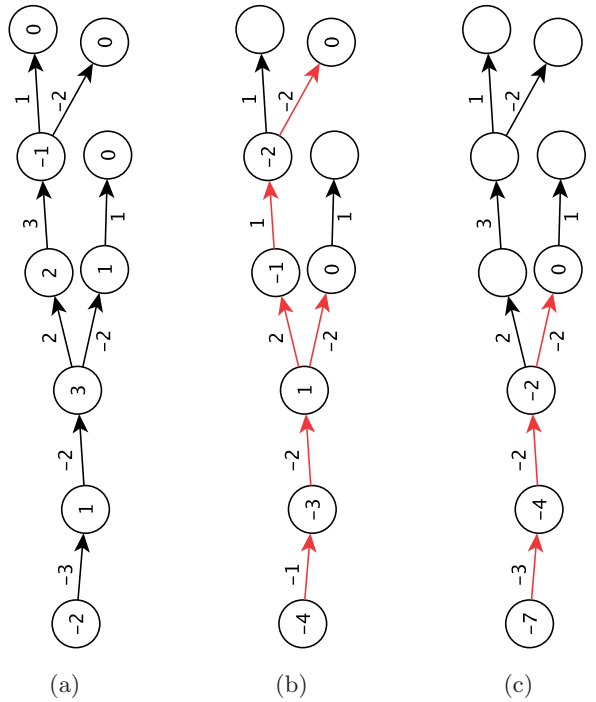


Figure 3.7: A simple illustration of why removing subtrees with all edge costs positive is suboptimal: (a) - the initial graph, (b) - naive pruning result, (c) - optimal pruning result. The numbers on vertices express costs of subtrees rooted at them. The red arrows in (b) and (c) indicate the retained edges. As can be seen, the naive approach to pruning improves the cost of the result from -2 to -4. However, this can be further impoved to -7 by preforming the pruning in an optimal way.

3.2.1.1 Optimal pruning

Given an edge-weighted Minimum Spanning Arborescence, it is possible that one could remove a number of edges from the tree and improve the overall score. A simple example of such a situation would be a branch that terminates with a subtree whose edges all have positive weights. Keeping them in the solution certainly does not make sense. An obvious approach to pruning could then be to search for all possible subtrees with that property and remove them from the solution. However, this could be suboptimal, as pictured by Figure 3.7.

In order to retrieve the optimal pruning let us express cost $c(\mathbf{x}_i)$ of the optimal

subtree rooted at vertex $\mathbf{x}_i$ as the following recursive equality.

$$c\left(\mathbf{x}_i\right) = \sum_{\mathbf{x}_j \in C(\mathbf{x}_i)} \min\left(w_{ij} + c\left(\mathbf{x}_j\right), 0\right), \tag{3.5}$$

where $C\left(\mathbf{x}_i\right)$ is the set of child vertices adjacent to $\mathbf{x}_i$. The intuition behind it is that for a subtree rooted at $\mathbf{x}_i$ we iterate through all its child nodes $\mathbf{x}_j \in C\left(\mathbf{x}_i\right)$ and for each one we decide whether it is worth taking it or not. In the former case, we add the weight of the edge $\mathbf{e}_{ij}$ and the cost of the subtree rooted at $\mathbf{x}_j$ to $c\left(\mathbf{x}_i\right)$, which is equivalent to adding to the optimal subtree rooted at $\mathbf{x}_i$ the edge $\mathbf{e}_{ij}$ and the optimal subtree rooted at $\mathbf{x}_j$. In the latter case, the edge is left out of the solution and the contribution to the cost is equal to zero.

Equation 3.5 holds for all vertices in the tree including the root node. All the $c\left(\mathbf{x}_i\right)$ values can be computed recursively in this way starting from the root while labeling the outgoing edges of each vertex as retainable or not. Once the edges are labelled, the final solution can be taken as the largest connected set of retainable edges reachable from the root. This leads to Algorithm 1. An optimal pruning of the example Minimum Spanning Arborescence of Figure 3.6 is illustrated by Figure 3.8.

The main procedure of Algorithm 1 makes two separate passes through the initial tree, one when calling GET_OPTIMAL_COST($\mathbf{x}_r$) and the other one when calling RETRIEVE_SOLUTION($\mathbf{x}_r$). Each vertex and each edge is visited at most once during both passes and so the computational complexity is $\mathcal{O}(|\mathcal{X}| + |\mathcal{E}_s|)$.

We refer to the presented method as **MSA+Prune** in the results section and we demonstrate that it yields competitive results compared to other, more complex methods.

In this section, we used edge weighted graphs for the sake of obtaining the reconstructions in the initial Minimum Spanning Arborescence step in polynomial time. However, given a specific edge-pair weighted tree it is possible to perform its optimal pruning efficiently in a similary way. We describe the algorithm in Appendix A.

Even though no polynomial-time methods are known for finding a Minimum Spanning Arborescence of an edge-pair weighted graph, the algorithm might still be useful for refining the output of another method that does not guarantee optimality. In particular, it could be used together with the original optimization method of [63] described below to obtain results of lower quality but in shorter time. One could terminate the optimization procedure prematurely and use the pruning algorithm of Appendix A to refine the reconstructions.

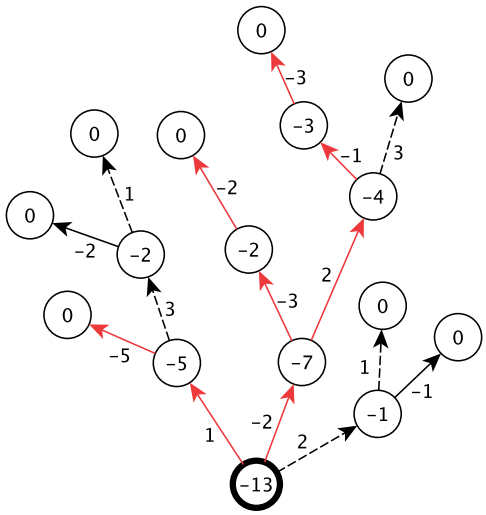


Figure 3.8: An optimally pruned edge-weighted Minimum Spanning Arborescence rooted at the bottom vertex. The numbers on the edges represent edge weights and the numbers on the vertices represent the optimal costs $c(\mathbf{x}_i)$ of their subtrees. Edges marked as retainable are represented by solid arrows. The remaining edges appear as dashed arrows. The red edges constitute the optimal subtree. Note that not all retainable edges are in fact retained in the final solution. Also, not all of the pruned edges have positive weights and not all of the retained ones have negative ones. The weight of the optimal subtree is -13.

Algorithm 1 Optimal edge-weighted tree pruning

```
for e_ij ∈ E_s do
    take_ij ← 0
    y_ij ← 0
end for
GET_OPTIMAL_COST(x_r)
RETRIEVE_SOLUTION(x_r)

function GET_OPTIMAL_COST(x_i)
    cost ← 0
    for x_j ∈ C(x_i) do
        b ← GET_OPTIMAL_COST(x_j) + w_ij
        if b < 0 then
            take_ij ← 1
            cost ← cost + b
        end if
    end for
    return cost
end function

procedure RETRIEVE_SOLUTION(x_i)
    for x_j ∈ C(x_i) do
        if take_ij = 1 then
            y_ij ← 1
            RETRIEVE_SOLUTION(x_j)
        end if
    end for
end procedure
```

3.2.2 Global Optimization

We now outline the original mathematical optimization method of [63]. The graph building procedure described in section 3.1 yields a tubular graph $\mathcal{G} = (\mathcal{X}, \mathcal{E}_s)$ with weights put on consecutive edge pairs. Finding the optimal solution tree amounts to finding the most probable subset $\mathcal{E}'_s \subseteq \mathcal{E}_s$ of edges that form a tree rooted at node $\mathbf{x}_r$. This implies that there must be exactly one directed path from $\mathbf{x}_r$ to every vertex in that solution tree.

Formally, the problem is expressed with binary variables y_{ij} indicating whether the edge $\mathbf{e}_{ij}$ is part of the solution. The resulting minimization scheme is the following.

$$y^* = \arg\max_{y \in \mathcal{Y}} P\left(\mathcal{I}, \mathcal{X}, \mathcal{E}_s | Y = y\right) \ , \tag{3.6}$$

$$= \arg\min_{y \in \mathcal{Y}} \sum_{\mathbf{e}_{ij}, \mathbf{e}_{jk} \in \mathcal{E}_s} w_{ijk} y_{ij} y_{jk}, \tag{3.7}$$

where $w_{ijk} = -\log \frac{p_{ijk}}{1-p_{ijk}}$, p_{ijk} is the probability of edge pair $(\mathbf{e}_{ij}, \mathbf{e}_{jk})$ being part of a tubular structure, and $\mathcal{Y}$ is the set of all feasible trees with root $\mathbf{x}_r$.

To ensure that the solution is indeed a tree we adapt a set of constraints from [19]. We introduce a set $F = \{f_{ij}^m\}$ of variables called *flow variables*. Each of those corresponds to one vertex-edge pair $(\mathbf{x}_m, \mathbf{e}_{ij}) \in \mathcal{X} \times \mathcal{E}_s$ in the graph. If vertex $\mathbf{x}_m$ is not part of the solution tree, all the *flow variables* f_{ij}^m are set to 0. If it is part of the solution the value of f_{ij}^m indicates whether the unique path from the root vertex $\mathbf{x}_r$ to the target vertex $\mathbf{x}_m$ traverses the edge $\mathbf{e}_{ij}$ or not. In the first case it is set to 1 and the other to 0. This way, if the solution is a tree, there is a unit flow from the root to every target vertex that is part of it. Fig. 3.9 depicts such a tree.

As shown in [19], the tree connectivity constraints can therefore be enforced by minimizing the criterion of Eq. 3.7 subject to

$$
\begin{aligned}
&\sum_{\mathbf{x}_j \in \mathcal{X} \setminus \{\mathbf{x}_r\}} f_{rj}^m \leq 1, && \forall \mathbf{x}_m \in \mathcal{X} \setminus \{\mathbf{x}_r\}, \\
&\sum_{\mathbf{x}_j \in \mathcal{X} \setminus \{\mathbf{x}_k\}} f_{jk}^m \leq 1, && \forall \mathbf{x}_m \in \mathcal{X} \setminus \{\mathbf{x}_r\}, \\
&\sum_{\mathbf{x}_j \in \mathcal{X} \setminus \{\mathbf{x}_i, \mathbf{x}_r\}} f_{ij}^m - \sum_{\mathbf{x}_j \in \mathcal{X} \setminus \{\mathbf{x}_i, \mathbf{x}_m\}} f_{ji}^m = 0, && \begin{matrix} \forall \mathbf{x}_m \in \mathcal{X} \setminus \{\mathbf{x}_r\}, \\ \forall \mathbf{x}_i \in \mathcal{X} \setminus \{\mathbf{x}_r, \mathbf{x}_m\}, \end{matrix} \\
&f_{ij}^m \leq y_{ij}, && \forall e_{ij} \in \mathcal{E},\ \mathbf{x}_m \in \mathcal{X} \setminus \{\mathbf{x}_r, \mathbf{x}_i, \mathbf{x}_j\}, \\
&f_{im}^m = y_{im}, && \forall e_{im} \in \mathcal{E}, \\
&f_{ij}^m \geq 0, && \forall e_{ij} \in \mathcal{E},\ \mathbf{x}_m \in \mathcal{X} \setminus \{\mathbf{x}_r, \mathbf{x}_i\}, \\
&y_{ij} \in \{0, 1\}, && \forall e_{ij} \in \mathcal{E}
\end{aligned}
\tag{3.8}
$$

During the optimization the edge variables are treated as integers and the flow variables are treated as real numbers. However, in the end their values need to be equal to zero or one. As shown in [19], explicitly constraining the edge variables y_{ij} to be either zero or one is enough to achieve this goal. This is why the initial Integer Program turns into a Qadratic Mixed Integer Program (QMIP). In the results section we refer to the presented method as **Arbor-IP**.

We have developed a similar set of flow variable constraints for undirected graphs in the hope that this would speed up the computation. Unfortunately, they turned out to be slower on average. An outline of the undirected constraints can be found in Appendix B.

3.3 Experiments and Results

We evaluated our pruning approach on micrographs acquired by targeting mice primary visual cortex using the brainbow technique [41] so that each neuron has a distinct color. We used one image stack for training and three for testing. We also used a set of six image stacks acquired by brightfield microscopy from biocytin-stained rat brains. We used three for training and three for testing.

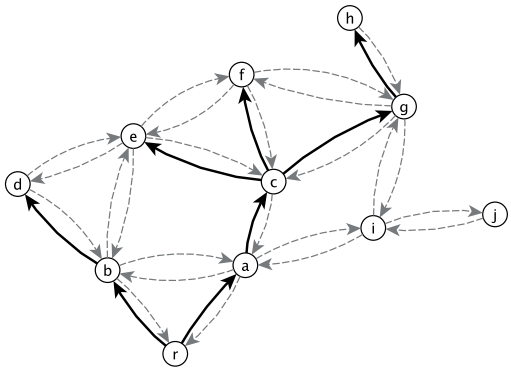


Figure 3.9: Flows in a directed graph rooted at vertex r. The arrows represent directed edges. Those that are part of the solution tree are shown as solid, the others as dashed. Note that the solution does not necessarily have to span all the vertices. In this specific case, vertices i and j do not belong to the tree. There are 11 vertices and 32 edges in this graph, which gives a total of 32 edge indicator variables y_{ij} and $11 \cdot 32 = 352$ flow variables f_{ij}^m. For the example tree, we have $f_{ra}^g = f_{ac}^g = f_{cg}^g = 1$, which creates the unit flow from r to g. All the other $f_{\cdot\cdot}^g$ variables, such as f_{ca}^g, are equal to 0. All the $f_{\cdot\cdot}^i$ and $f_{\cdot\cdot}^j$ variables are also equal to 0.

We used a semi-automated delineation tool of [64] to obtain the ground truth tracings. We built the tubular graphs using a seed selection distance of $d = 20\Delta_I$ and a seed linking distance of $l(d) = 5d$, where Δ_I denotes the minimum voxel spacing.

3.3.1 Baselines

We compare our **MSA+Prune** approach, with the original **Arbor-IP** approach, the **Loopy-IP** approach of [62], all of which compute an optimal solution according to their respective cost functions and they are all based on the same path classifier idea. We also include in our comparison the **kMST** approach of [65]. As mentioned in Chapter 2, **Loopy-IP** can be viewed as a more robust generalization of **Arbor-IP**. To demonstrate the importance of the pruning step, we also compare against a pure Minimum Spanning Arborescence approach denoted by **MSA**.

3.3.2 DIADEM Metric

For quantitative evaluation we used the DIADEM metric [29], which outputs a single number for comparing topological accuracy of a reconstructed tree against a ground

truth tree. The metric is a multi-step process that scores the connection between every pair of nodes in the ground truth reconstruction based on whether or not the test reconstruction captures that connection. Moreover, the importance of a connection may be taken into account to reflect the global topological similarity of the trees. This is done by weighting each connection by a chosen node attribute. In our case the weight is the size of the subtree to which a connection leads in terms of terminal degree.

3.3.3 Results

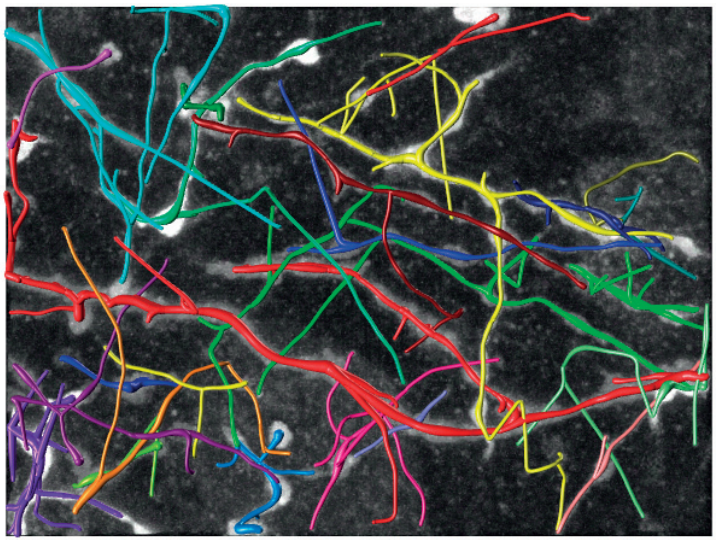
The DIADEM scores are presented in Table 3.1. Example visualisations of the Minimum Spanning Arborescence results before and after the pruning procedure are presented in Figs. 3.10 through 3.15. In all cases many of the background edges have been removed from the solutions and the final results look a lot cleaner. The diameters of the reconstructed trees in the visualisations have been decreased for clarity.

As can be seen in Table 3.1 our **MSA+Prune** approach provides competitive results compared to the other, more complicated methods. Most notably, it performs very well compared to the original **Arbor-IP** method. It is also worth noting that the pruning step plays an important role. The unpruned **MSA** method performs consistently worse. The **Loopy-IP** method clearly performs best. It is a very recent method and we include it here for the sake of completeness.

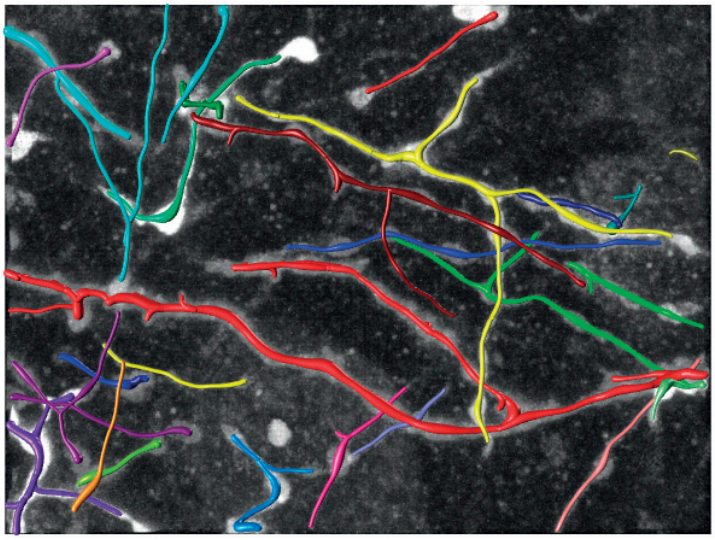
The key advantage of our **MSA+Prune** approach is the computational efficiency. Given the initial overcomplete graph, one can find the initial Minimum Spanning Arborescence in quadratic time and then perform the optimal pruning in linear time. In the end, the computation time is of course negligible compared to the other, more complex approaches. It can therefore be an alternative to the mathematical optimization based **Arbor-IP** or even **Loopy-IP** when dealing with very large image stacks in situations where a slight decrease in the quality of reconstructions is acceptable in favor of speed.

	BRBW1	BRBW2	BRBW3	BRF1	BRF2	BRF3
Loopy-IP [62]	**0.65**	**0.56**	**0.66**	**0.76**	**0.50**	**0.80**
MSA+Prune	0.46	0.40	0.49	0.44	0.37	0.55
MSA	0.33	0.29	0.25	0.25	0.22	0.26
Arbor-IP	0.44	0.32	0.45	0.59	0.32	0.71
kMST [65]	0.26	0.16	0.25	0.51	0.28	0.47

Table 3.1: DIADEM scores for the Brightfield and Brainbow datasets. Higher scores are better.

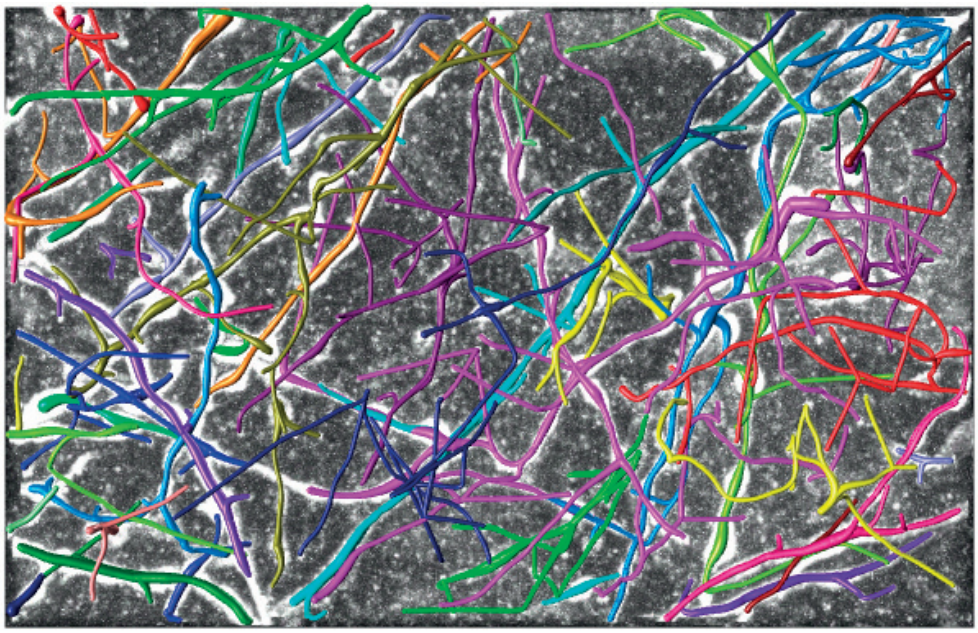


(a)

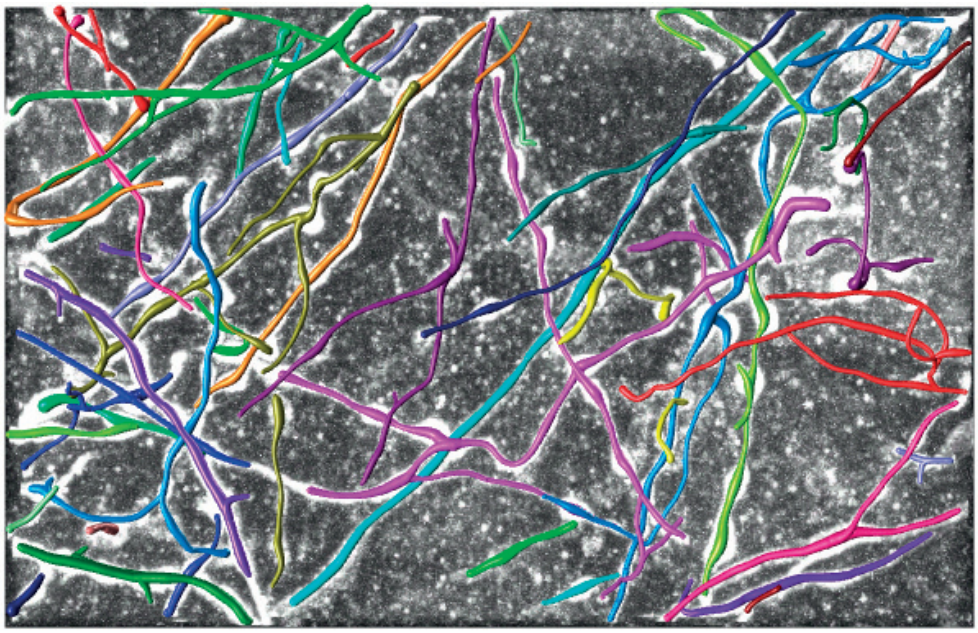


(b)

Figure 3.10: An example Brainbow stack. (a) Minimum Spanning Arborescence before pruning. (b) Minimum Spanning Arborescence after pruning.



(a)



(b)

Figure 3.11: Another example Brainbow stack. (a) Minimum Spanning Arborescence before pruning. (b) Minimum Spanning Arborescence after pruning.

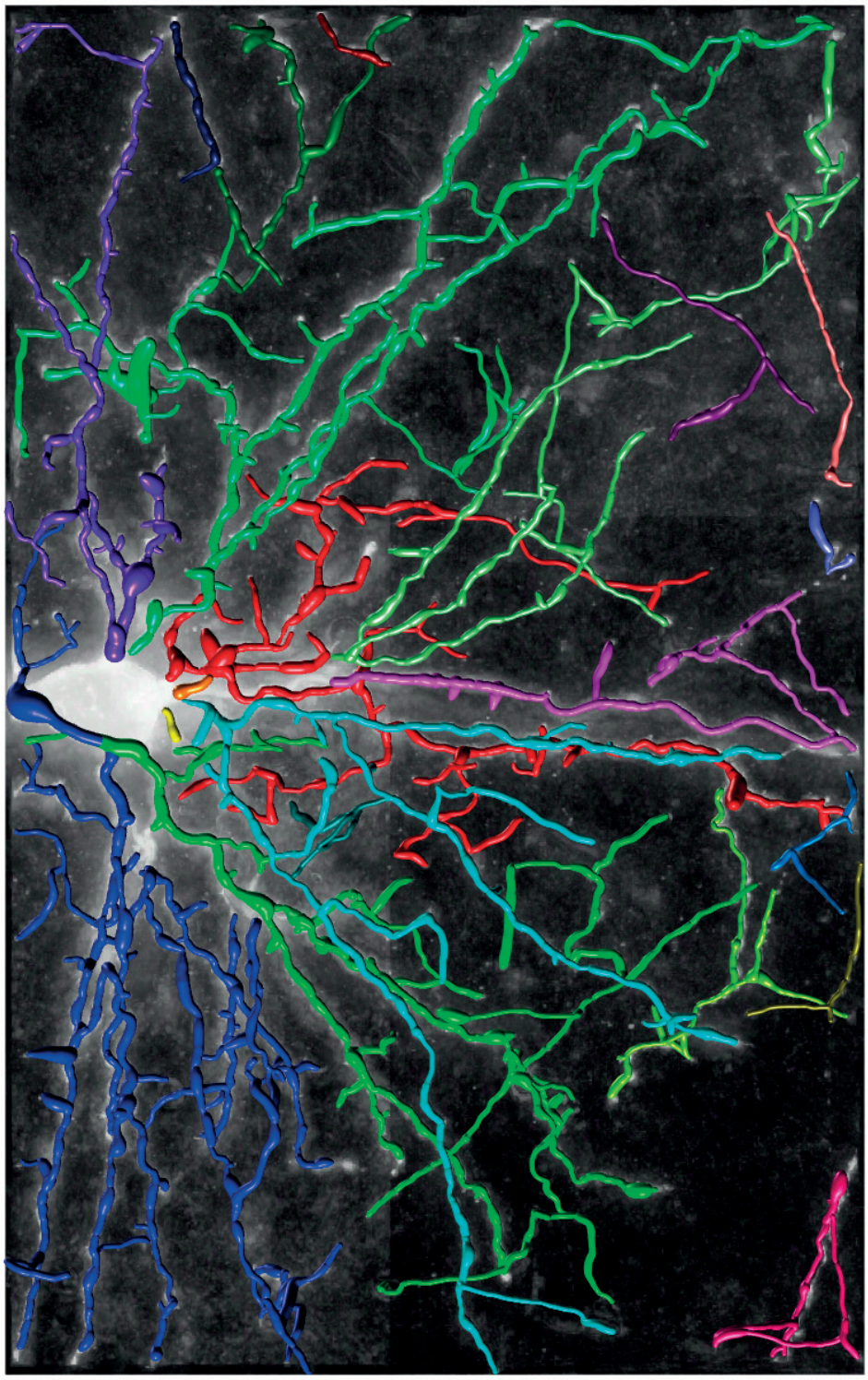


Figure 3.12: An example brightfield stack. Minimum Spanning Arborescence before pruning.

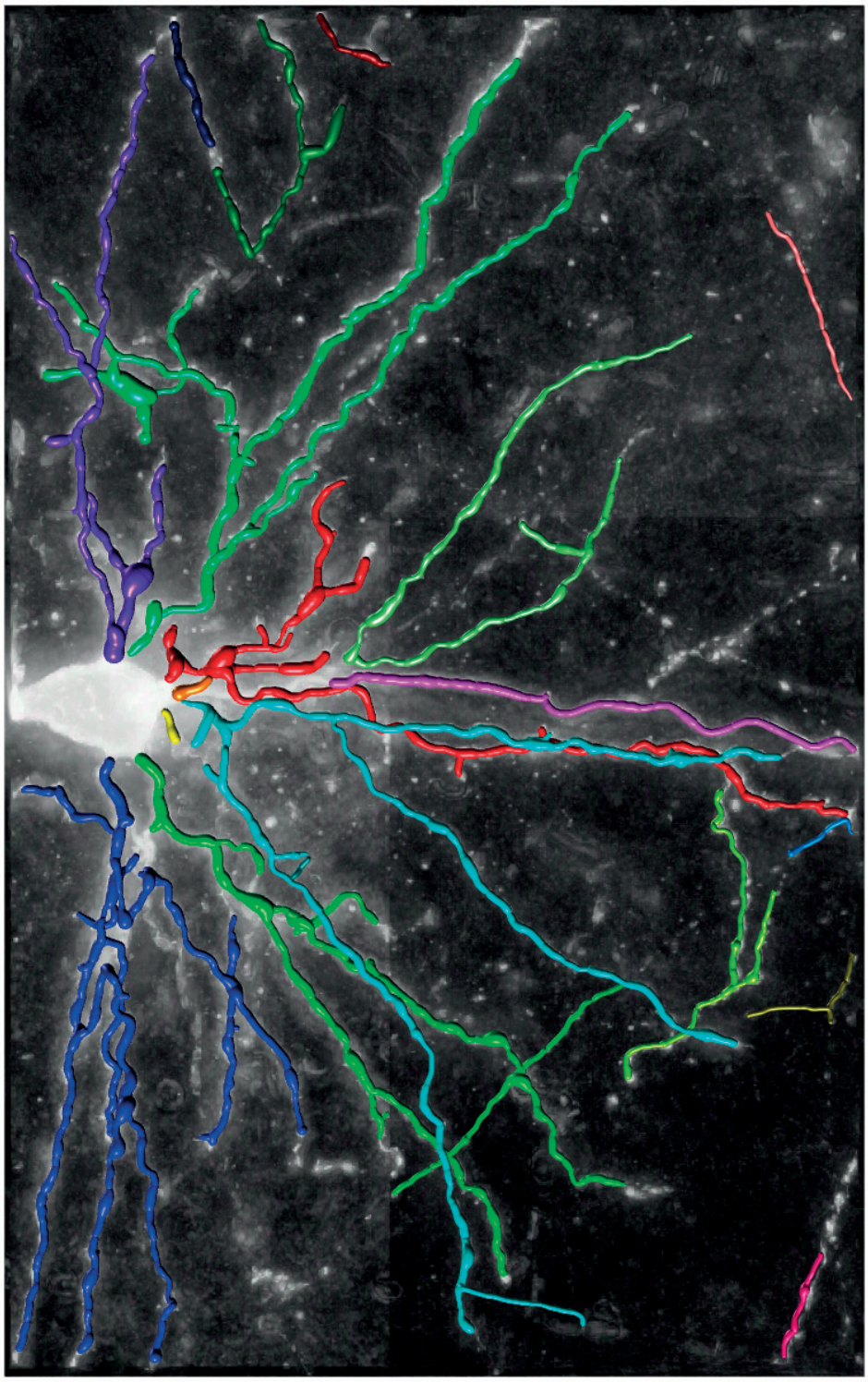


Figure 3.13: The brightfield stack of Fig. 3.12. Minimum Spanning Arborescence after pruning.

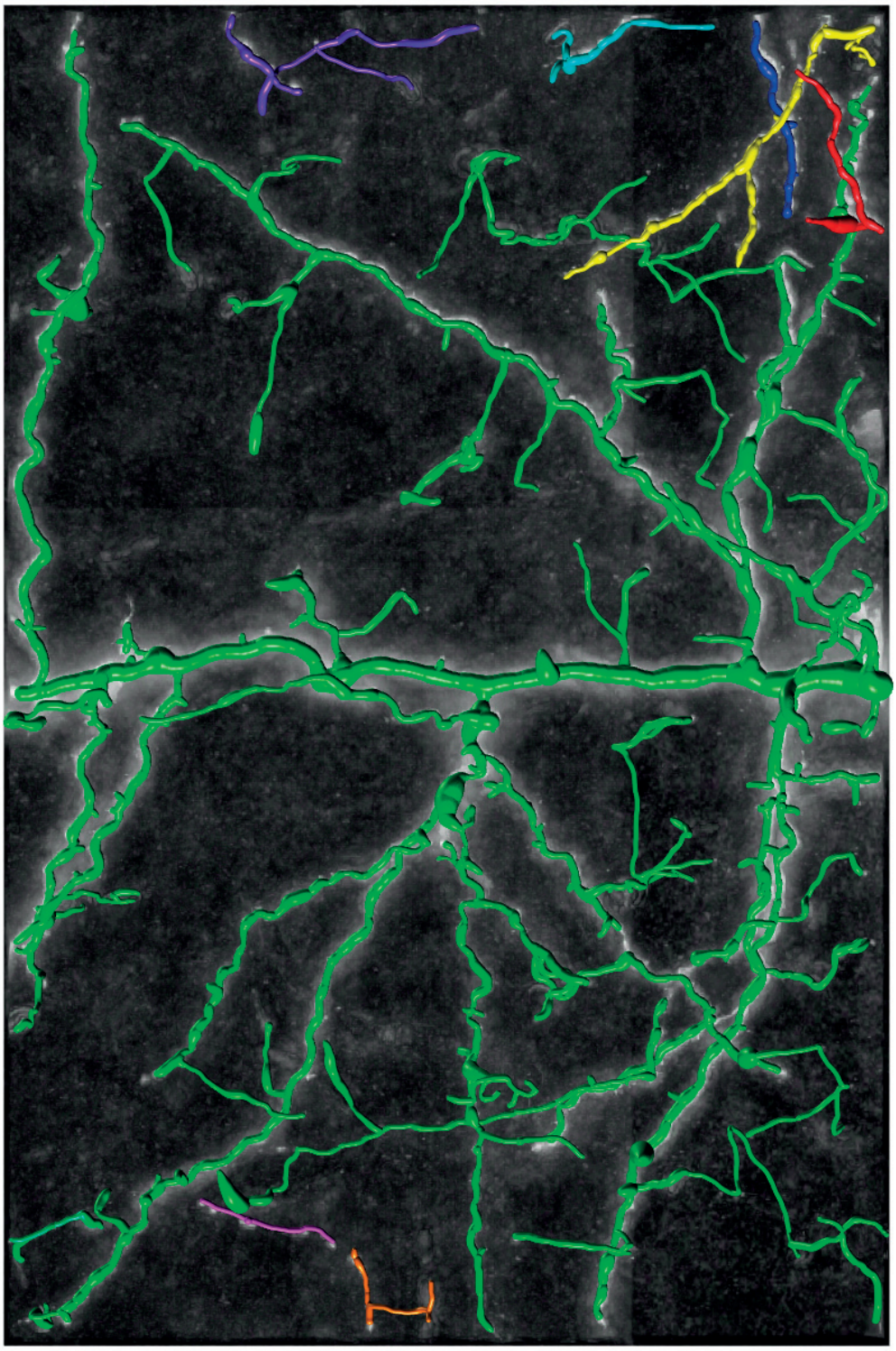


Figure 3.14: Another example brightfield stack. Minimum Spanning Arborescence before pruning.

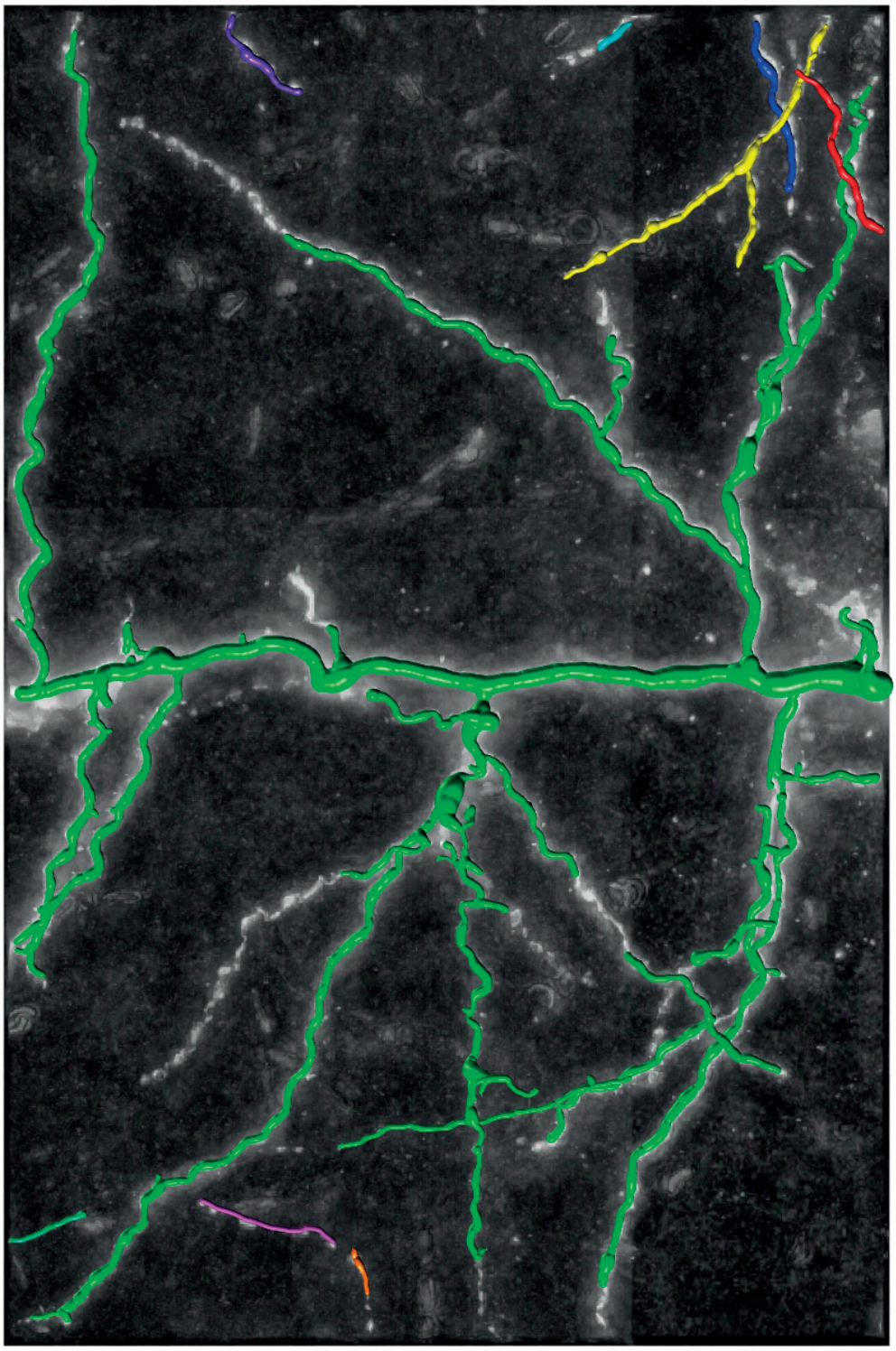


Figure 3.15: The brightfield stack of Fig. 3.14. Minimum Spanning Arborescence after pruning.

FOUR

LOCAL TEMPORAL CONSISTENCY

In this chapter we propose an approach to reconstructing evolving tree structures simultaneously in a sequence of images taken at different time instants. In this way, we can enforce temporal consistency over stable parts of the structure and reliably detect changes elsewhere. This is in contrast to recovering the relevant structures in each image individually and only then comparing them, which we will show to be less effective.

In Chapter 3 we described the method of [63] whose initial step was to find a set of uniformly distributed centerline points in a single image. Here, we find centerline points that correspond to identical features across time instances by means of a Gaussian Process Regression (GPR) model [54] and connect these temporal correspondences by *temporal edges*. Combining those with *spatial edges* yields a *spatio-temporal* graph that lets us incorporate into our objective function terms that enforce temporal consistency. Conveniently, this optimization problem remains a QMIP that can be solved efficiently. Fig. 4.1 illustrates our approach in a simple case.

Our contribution is therefore a novel approach to modeling trees over several images simultaneously while enforcing temporal consistency. Not only is this more reliable than doing so over individual images but has the added benefit of making it easy to spot the regions that have significantly changed, which is tedious and hard to do for human

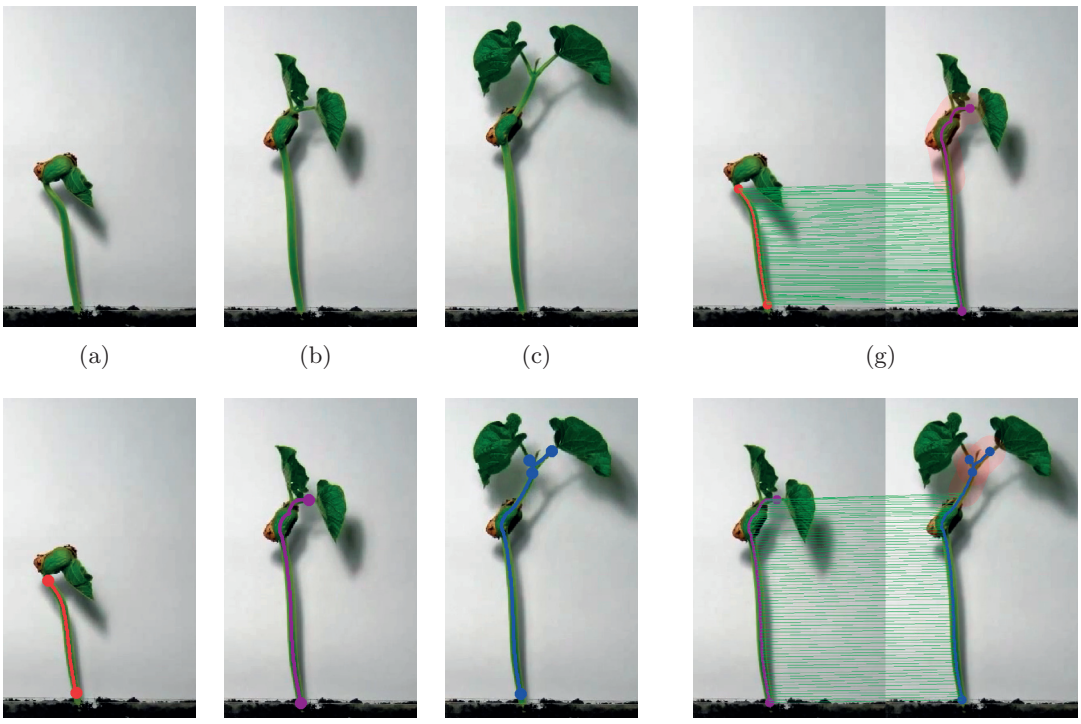


Figure 4.1: Reconstruction and automatic change detection using a time-lapse sequence for growing runner bean. (a, b, c) Original images. (d, e, f) Reconstructed trees in each one of them. (g, h) The horizontal green lines represent correspondences obtained after the fine alignment step.

operators. We demonstrate the power of our approach on a time-lapse sequence of a growing bean plant and on sequences of *in vivo* two-photon micrographs of neuronal networks.

4.1 Approach

For many tree structures that evolve over time, significant changes from one frame to the next tend to be fairly localized, while the general topology and geometry remain relatively stable up to minor local deformations. Consider, for example, a real-world tree whose branches are growing over time. In images taken at sufficiently long time intervals, there may be significant changes at the tips of existing branches while the

rest remains largely unchanged. The same principle applies in the case of the neuronal network of Fig. 4.2 captured *in vivo* at intervals of a week. Most of the structure is preserved over time, except for a few branches that have either grown to form new connections, retracted or moved to new positions. To exploit the overall consistency while allowing some degree of change, we propose the following approach.

Given N D-dimensional images $\mathcal{I} = \{I^n\}_{n=1}^N$ taken in sequence and showing an evolving tree structure, our goal is to reconstruct trees in each individual image such that they collectively form a temporally consistent sequence. By this, we mean that branches do not appear or disappear randomly. First, we find corresponding points across images and use them as nodes of a graph whose edges can either connect to nodes within the same image or to other images. As in Chapter 3, the final set of trees can then be reconstructed by solving a QMIP problem.

4.1.1 Reconstruction in all Images Simultaneously

Repeating the procedure described in Section 3.2.2 for each image I^n would yield N distinct trees that would be difficult to compare to other trees, as it is unlikely for their nodes to be at the same locations in different images. To avoid this problem and to enforce temporal consistency constraints, we modify the framework in two key ways.

First, we find temporally consistent nodes $\mathbf{x}_i^n$ in all images by looking for local maxima of tubularity in one image and then finding corresponding high-tubularity points in the others. This lets us create *temporal edges* $\mathbf{e}_{ij}^{n_1,n_2}$ between node $\mathbf{x}_i^{n_1}$ found in I^{n_1} and its matched node $\mathbf{x}_j^{n_2}$ in I^{n_2}.

Second, we build a *spatio-temporal* graph whose edges are both the spatial edges as in Chapter 3 and temporal edges that connect nodes from one individual image to another. In such a graph, minimizing an objective function that only considers the spatial edges, as described in the previous subsection, would yield the same result as before. However, we can use the temporal edges to add terms favoring edges persistent between time instances, thus enforcing time consistency. Minimizing this extended

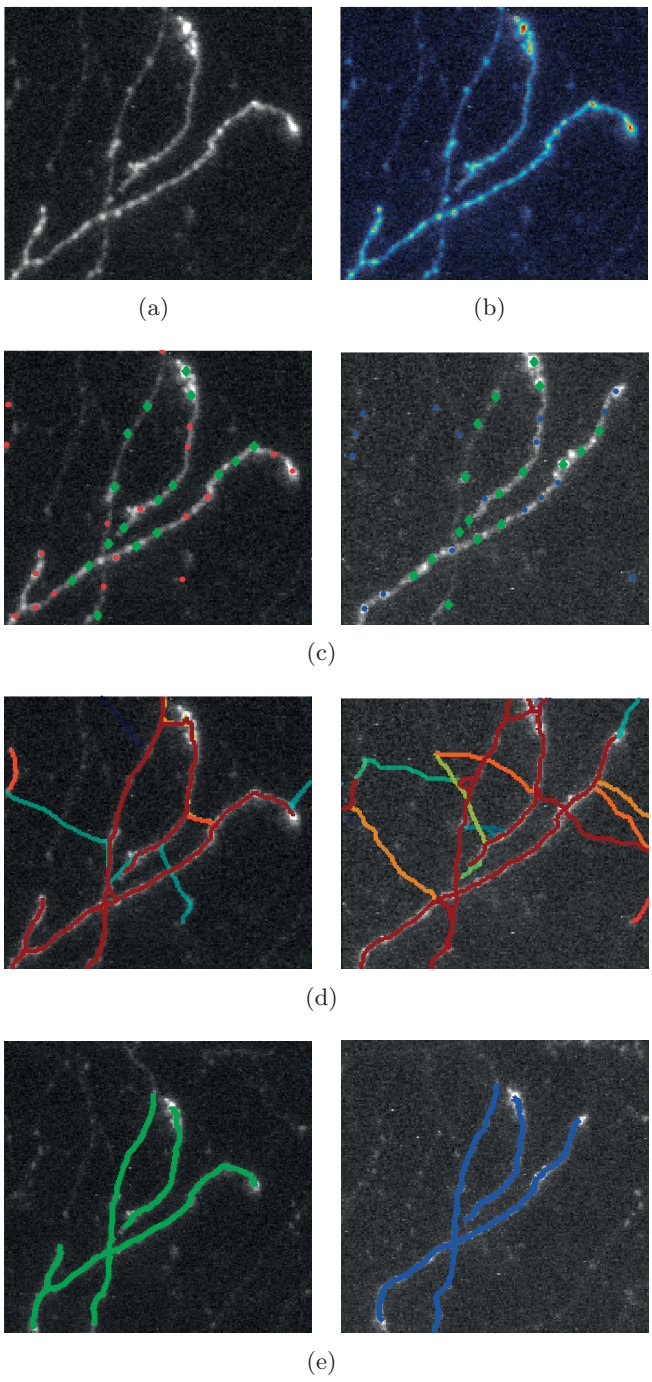


Figure 4.2: Key steps of the algorithm. (a) Maximum intensity projection of one of three *in vivo* image-stacks of a neural network taken at one week intervals. (b) Corresponding tubularity image. (c) Maxima of tubularity selected as graph nodes in two different stacks. Those shown in green have been determined to correspond to the same location in both, while those in red or blue appear in only one. (d) Connecting neighboring nodes by high-tubularity paths produces a spatial graph in each image. High-quality paths are shown as red while low quality ones appear as blue. (e) Connecting the corresponding vertices across images turns the spatial graphs into a single spatio-temporal one and solving the corresponding QMIP problem yields two temporally consistent sets of trees.

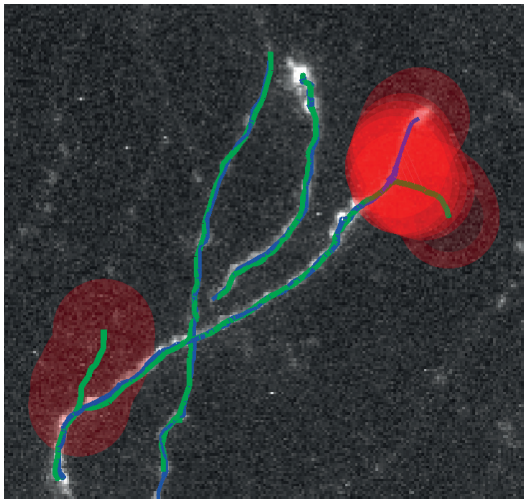


Figure 4.3: The reconstructed trees from the two time instants of Fig. 4.2(e) can be deformed and superposed on each other, making the changes highlighted in red easy to detect.

objective function can still be expressed as a Q-MIP. Our approach therefore goes through the following steps:

1. Find graph nodes in individual images as tubularity maxima and corresponding nodes, if any, in other images, as in Fig. 4.2(c).

2. Build a spatio-temporal graph such as the one depicted in Fig. 4.2(d) by linking nodes both within images when they are close enough and across images when they match. A more illustrative schematic representation of a spatio-temporal graph can be seen in Fig. 4.5.

3. Solve an extended Q-MIP problem to find a set of temporally-consistent trees, such as those of Fig. 4.2(e).

4. Align these trees spatially to identify places where substantial changes have occurred, as can be seen in Fig.4.3.

In the following two sections, we first describe the construction of our spatio-temporal graphs in more detail. We then define our Q-MIP problem and the corresponding objective function.

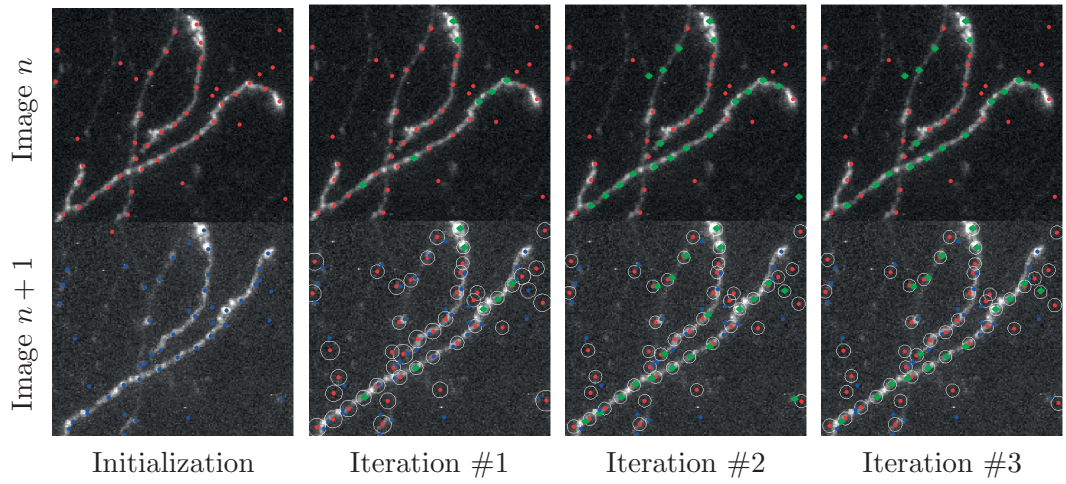


Figure 4.4: Iterating until a stable correspondence set has been found. (Initialization) A set of corresponding points with possible inconsistencies in the transformation model is found in each image using high-tubularity locations and NCC. (Iteration #1) A set of corresponding points (shown in green) with the highest tubularity likelihoods has been selected, which are then used to instantiate a GPR that maps the remaining red points in image n to the red locations in image $n+1$. The blue points in image $n+1$ that are close enough to these red locations and correlate well with the original red points in image n are taken to form new correspondences. (Iterations #2 and #3) They are added to the set of correspondences, shown in green. The process is then repeated.

4.2 Building Spatio-Temporal Graphs

The first step in building our spatio-temporal graph is to find corresponding nodes across images, such as those shown in Fig. 4.2(c). As discussed above, we assume that there may be some non-linear deformation from one image to the next but that it is smooth.

Finding an Initial Set of Correspondences We first use the Multi-Directional Oriented Flux [61] filter to compute a tubularity measure in each image independently.

Then, for $m = 1, \ldots, M$ iterations, we find the point $\mathbf{x}_m^n$ that maximizes tubularity across all images, where n refers to the image in which it was found. Then for each of the remaining images $I^{\bar{n}} \in \mathcal{I} \backslash I^n$, we compute the Normalized Cross Correlation (NCC) score of a square or cubic patch centered on a point $\mathbf{x}_m^n$ and a neighbourhood of locations around $\mathbf{x}_m^{\bar{n}}$. Within each evaluated neighbourhood, we associate the location $\mathbf{x}_m^{\bar{n}}$ with

the maximum computed NCC score provided it is above a given minimum threshold. From this set, we keep all the consecutive pairs of points $\{\mathbf{x}_m^{n'} \leftrightarrow \mathbf{x}_m^{n'+1}\}_{1\leq n'\leq N-1}$ as correspondences, as illustrated by the green points of Fig. 4.2(c). Once computed, the tubularity is set to zero in both the neighborhood of $\mathbf{x}_m^n$ and that of the found corresponding points. The procedure is then iterated until the tubularity of the selected point $\mathbf{x}_m^n$ is below a certain value.

Enforcing Geometric Consistency The procedure described above relies solely on the NCC scores computed locally and does not guarantee that the displacements of neighboring points are spatially consistent with each other. To enforce this and to remove potential mismatches, we use Gaussian Processes Regression (GPR) [54] to remove correspondences that are not consistent with a non-linear but locally smooth deformation model. This step of the sampling procedure was designed by a colleague of mine, Miguel Amável Pinheiro.

To find a geometrically consistent set of correspondences $\mathcal{S}_n$ between images I^n and I^{n+1}, we first select from our correspondences a set $\mathcal{S}_n^0 = \{\mathbf{x}_l^n \leftrightarrow \mathbf{x}_l^{n+1}\}_{1\leq l\leq L}$ of the L points with the highest average local tubularity. In the example of Fig. 4.4 (Iteration #1), the selected $\mathbf{x}_l^n$ points are shown in green. We treat $\mathcal{S}_n^0$ as being a reliable set and use the GPR to estimate the mean and covariance of the location of a point $\mathbf{x}^n$ in I^{n+1}. This can be computed as

$$\begin{aligned} m_{\mathcal{S}_n^0}(\mathbf{x}^n) &= \mathbf{k}'\mathbf{\Gamma}_{\mathcal{S}_n^0}^{-1}\mathbf{X}_{\mathcal{S}_n^0}^{n+1} \ , \\ \sigma_{\mathcal{S}_n^0}^2(\mathbf{x}^n) &= k(\mathbf{x}^n,\mathbf{x}^n) + \beta^{-1} \ - \mathbf{k}'\mathbf{\Gamma}_{\mathcal{S}_n^0}^{-1}\mathbf{k} \ , \end{aligned} \tag{4.1}$$

where k is a kernel function that implicitly defines a mapping composed of an affine and a non-linear transformation as in [55, 78], β^{-1} is a measurement noise variance, $\mathbf{\Gamma}_{\mathcal{S}_n^0}$ is the $L \times L$ symmetric matrix with elements $\Gamma_{i,j} = k(\mathbf{x}_i^n, \mathbf{x}_j^n) + \beta^{-1}\delta_{i,j}$, $\mathbf{k}$ is the vector $[k(\mathbf{x}_1^n,\mathbf{x}^n),\ldots,k(\mathbf{x}_L^n,\mathbf{x}^n)]^T$ and $\mathbf{X}_{\mathcal{S}_n^0}^{n+1}$ is the $L \times D$ matrix $[\mathbf{x}_1^{n+1},\ldots,\mathbf{x}_L^{n+1}]^T$.

We then add all correspondences that are consistent with this GPR to $\mathcal{S}_n^0$, which

is determined when the Mahalanobis distance between corresponding points $\mathbf{x}^{n+1}$ and $m_{\mathcal{S}_n^0}(\mathbf{x}^n)$ is sufficiently small. This gives us an augmented set of correspondences $\mathcal{S}_n^1$, such as the one depicted by Fig. 4.4 (Iteration #2). We then repeat the process using $\mathcal{S}_n^1$ to compute the regression of Eq. 4.1 and iterate until the set stabilizes, typically after 4 to 5 iterations, as shown in Fig. 4.4 (Iteration #3).

This is performed for each consecutive image pair, which yields sets of points in each image $\mathcal{X}^n = \{\mathbf{x}_i^n\}$ and sets of geometrically consistent correspondences $\mathcal{S}_n$ across consecutive images.

Building the Graph We treat points in all the $\mathcal{X}^n$ as nodes of our graph and create two kinds of edges. As in the single-image case of Chapter 3, the *spatial edges* $\mathcal{E}_s^n = \{\mathbf{e}_{ij}^n = (\mathbf{x}_i^n, \mathbf{x}_j^n)\}$ correspond to edges connecting points within I^n and consecutive pairs of such edges are assigned an image-based probability of being part of the final curvilinear structure. To these, we add *temporal edges* $\mathcal{E}_t^n = \{\mathbf{e}_{ij}^{n,n+1} = (\mathbf{x}_i^n, \mathbf{x}_j^{n+1}) \mid (\mathbf{x}_i^n \leftrightarrow \mathbf{x}_j^{n+1}) \in \mathcal{S}_n\}$ that connect nodes in I^n and I^{n+1} that belong to the set $\mathcal{S}_n$ of geometrically consistent correspondences.

4.3 Finding Temporally Consistent Trees

Given a spatio-temporal graph $\mathcal{G} = (\mathcal{X}, \mathcal{E})$, where $\mathcal{X} = \{\bigcup_{n=1}^N \mathcal{X}^n\}$ and $\mathcal{E} = \mathcal{E}_s \cup \mathcal{E}_t = \{\bigcup_{n=1}^N \mathcal{E}_s^n\} \cup \{\bigcup_{n=1}^{N-1} \mathcal{E}_t^n\}$ such as the one discussed in the previous section, our goal now is to find a subgraph forming a set of trees that evolve consistently over time. For every image in the sequence, the locations of the tree roots are provided by an operator and are added to the set of graph nodes. An additional imaginary root $\mathbf{x}_r$ is created and connected to all these root nodes in all time instances. This way, reconstructing the trees in all images can be achieved by finding the most likely arborescence rooted in $\mathbf{x}_r$.

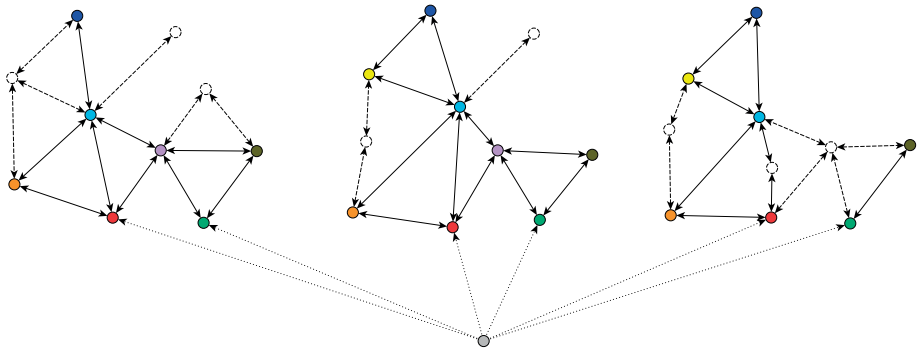


Figure 4.5: An example *spatio-temporal graph*. The imaginary root vertex $\mathbf{x}_r$ is presented in gray at the bottom. Each of the three time points contain two manually annotated physical roots marked in red and green. The vertices for which correspondences where not found in adjacent time points are represented by white circles with dashed borders. The other vertices are represented by coloured circles. The temporal edges $\mathcal{E}_t$ are not explicitely presented to avoid clutter. Instead, vertices for which correspondences were found are marked with matching colours. The dotted arrows represent the imaginary edges from $\mathbf{x}_r$ to the physical roots. The double-sided arrows between vertices in each time point represent the two oppositely directed *spatial edges* between neighbouring vertices. *Spatial edges* that are part of the corresponding edges set $\bar{\mathcal{E}}_t$ are marked with solid lines. Other *spatial edges* are marked with dashed lines.

4.3.1 Objective Function

Reconstructing the trees of interest means making a decision as to whether each edge of the graph $\mathcal{G}$ should be part of the solution or not. To this end, we take Bayesian point of view as in the QMIP method of Chapter 3. Let $Y_{ij} \in \{0, 1\}$ be a binary random variable denoting the presence or absence of the edge $\mathbf{e}_{ij}$ in the final solution and Y be the set of all Y_{ij} variables. Our goal is to infer the most likely tree Y.

To obtain the most likely Y while enforcing temporal consistency between reconstructions across time, we introduce a constant q that denotes the edge persistence probability. That is, for a given pair of edges $(\mathbf{e}_{ij}^n, \mathbf{e}_{kl}^{n+1})$, we assume that the probability of both edges being part, or not, of the final solution is equal to q. Conversely, the probability that one of the edges is part of the solution while the other is not is equal to $1 - q$. And let us therefore denote $\bar{\mathcal{E}}_t = \{(\mathbf{e}_{ij}^n, \mathbf{e}_{kl}^{n+1}) | \mathbf{e}_{ij}^n, \mathbf{e}_{kl}^{n+1} \in \mathcal{E}_s \wedge \mathbf{e}_{ik}^{n,n+1}, \mathbf{e}_{jl}^{n,n+1} \in \mathcal{E}_t\}$ be the set of all pairs of spacial edges in consecutive time frames whose endpoints are connected with temporal edges.

With this, describing the posterior distribution of Y given the spatial edges $\mathcal{E}_s$ and the temporal edges $\mathcal{E}_t$ can then be expressed as

$$P(Y=y|\mathcal{I},\mathcal{X},\mathcal{E}_s,\mathcal{E}_t) \propto P(\mathcal{I},\mathcal{X},\mathcal{E}_s|Y=y)\,P(Y=y|\mathcal{E}_t)\,, \tag{4.2}$$

assuming that the image data and the spatial edges are conditionally independent of the temporal edges given Y. We can then write

$$P\left(\mathcal{I},\mathcal{X},\mathcal{E}_s|Y=y\right) = \prod_{\mathbf{e}_{ij}^n,\mathbf{e}_{jk}^n\in\mathcal{E}_s} \left(\frac{p_{ijk}}{1-p_{ijk}}\right)^{y_{ij}y_{jk}}\,, \tag{4.3}$$

$$P\left(Y=y|\mathcal{E}_t\right) = \prod_{\left(\mathbf{e}_{ij}^n,\mathbf{e}_{kl}^{n+1}\right)\in\bar{\mathcal{E}}_t} \left(\frac{q}{1-q}\right)^{2y_{ij}y_{kl}-y_{ij}-y_{kl}}\,, \tag{4.4}$$

where p_{ijk} is the probability that the edge pair $(\mathbf{e}_{ij}^n,\mathbf{e}_{jk}^n)$ is a part of a tubular structure. The derivation of the image term of Eq. 4.3 is exactly the same as that of the QMIP method of Chapter 3. We model the prior term of Eq. 4.4 as a tree structured Bayesian network that captures temporal relationships between edges in $\bar{\mathcal{E}}_t$.

$$\begin{aligned}
&P\left(Y=y|\mathcal{E}_t\right)\\
&= \prod_{\left(\mathbf{e}_{ij}^n,\mathbf{e}_{kl}^{n+1}\right)\in\bar{\mathcal{E}}_t} P\left(Y_{kl}=y_{kl}|Y_{ij}=y_{ij}\right) \prod_{\mathbf{e}_{ij}^n\in\mathcal{E}_0} P\left(Y_{ij}=y_{ij}\right) && (4.5)\\
&\propto \prod_{\left(\mathbf{e}_{ij}^n,\mathbf{e}_{kl}^{n+1}\right)\in\bar{\mathcal{E}}_t} \left(\frac{P\left(Y_{kl}=y_{kl}|Y_{ij}=1\right)}{P\left(Y_{kl}=y_{kl}|Y_{ij}=0\right)}\right)^{y_{ij}} P\left(Y_{kl}=y_{kl}|Y_{ij}=0\right) && (4.6)\\
&\propto \prod_{\left(\mathbf{e}_{ij}^n,\mathbf{e}_{kl}^{n+1}\right)\in\bar{\mathcal{E}}_t} \left(\frac{P\left(Y_{kl}=y_{kl}|Y_{ij}=1\right)}{P\left(Y_{kl}=y_{kl}|Y_{ij}=0\right)}\right)^{y_{ij}} \left(\frac{P\left(Y_{kl}=1|Y_{ij}=0\right)}{P\left(Y_{kl}=0|Y_{ij}=0\right)}\right)^{y_{kl}}\\
&\qquad\qquad P\left(Y_{kl}=0|Y_{ij}=0\right) && (4.7)\\
&\propto \prod_{\left(\mathbf{e}_{ij}^n,\mathbf{e}_{kl}^{n+1}\right)\in\bar{\mathcal{E}}_t} \left(\frac{P\left(Y_{kl}=1|Y_{ij}=1\right)}{P\left(Y_{kl}=1|Y_{ij}=0\right)}\right)^{y_{ij}y_{kl}} \left(\frac{P\left(Y_{kl}=0|Y_{ij}=1\right)}{P\left(Y_{kl}=0|Y_{ij}=0\right)}\right)^{y_{ij}(1-y_{kl})}\\
&\qquad\qquad \left(\frac{P\left(Y_{kl}=1|Y_{ij}=0\right)}{P\left(Y_{kl}=0|Y_{ij}=0\right)}\right)^{y_{kl}} && (4.8)\\
&\propto \prod_{\left(\mathbf{e}_{ij}^n,\mathbf{e}_{kl}^{n+1}\right)\in\bar{\mathcal{E}}_t} \left(\frac{q}{1-q}\right)^{y_{ij}y_{kl}} \left(\frac{1-q}{q}\right)^{y_{ij}-y_{ij}y_{kl}} \left(\frac{1-q}{q}\right)^{y_{kl}} && (4.9)\\
&\propto \prod_{\left(\mathbf{e}_{ij}^n,\mathbf{e}_{kl}^{n+1}\right)\in\bar{\mathcal{E}}_t} \left(\frac{q}{1-q}\right)^{2y_{ij}y_{kl}-y_{ij}-y_{kl}}\,, && (4.10)
\end{aligned}$$

where $\mathcal{E}_0 \subset \mathcal{E}_S$ denotes the set of spatial edges associated to the first image at $n = 0$. We assume uniform prior for these edges and drop the $P(Y_{ij} = y_{ij})$ terms from Eq. 4.5. Eq. 4.8 is obtained by simple algebraic manipulations and dropping the constant terms that do not depend on the y_{ij} or y_{kl} variables. Finally, we substitute the persistent probabilities $P(Y_{kl} = 0|Y_{ij} = 0)$ and $P(Y_{kl} = 1|Y_{ij} = 1)$ with q, and probabilities $P(Y_{kl} = 1|Y_{ij} = 0)$ and $P(Y_{kl} = 0|Y_{ij} = 1)$ with $(1 - q)$ in Eq. 4.9, and rearrange common terms in Eq. 4.10.

Note that finding a solution that maximizes the product of probabilities of Eq. 4.3 and Eq. 4.4 is equivalent to finding one that minimizes the sum of negative logarithms of these probabilities. This leads to the *maximum a posteriori* problem of Eq. 4.11.

$$\begin{aligned} y^* &= \arg\max_{y \in \mathcal{Y}} P(\mathcal{I}, \mathcal{X}, \mathcal{E}_s | Y = y)\, P(Y = y | \mathcal{E}_t)\,, \\ &= \arg\min_{y \in \mathcal{Y}} \sum_{\mathbf{e}^n_{ij}, \mathbf{e}^n_{jk} \in \mathcal{E}_s} w_{ijk} y_{ij} y_{jk} \\ &\quad + \sum_{(\mathbf{e}^n_{ij}, \mathbf{e}^{n+1}_{kl}) \in \bar{\mathcal{E}}_t} w_p \left(2 y_{ij} y_{kl} - y_{ij} - y_{kl}\right), \end{aligned} \tag{4.11}$$

where $w_{ijk} = -\log \frac{p_{ijk}}{1-p_{ijk}}$, $w_p = -\log \frac{q}{1-q}$, and $\mathcal{Y}$ is the set of all feasible trees rooted at $\mathbf{x}_r$.

Note that the temporal constant $0.5 \leq q < 1$ allows flexibility in the amount of time consistency desired across time instances, *i.e.* higher values enforce more consistent results. In the special case where $q = 0.5$ the persistence weight w_p is equal to 0 and the problem is reduced to that of Chapter 3.

4.3.2 Finding the Optimal Tree

To find a tree that minimizes the objective function defined above, we solve the quadratic mixed integer program (Q-MIP), the same as in Chapter 3, applying a max-flow min-cut formulation of the minimum arborescence problem using the Gurobi optimization engine [31]. Note that in this formulation, the input graph must have directed

edges in order to compute the flow of a given solution. Hence, as in the QMIP method of Chapter 3, we treat each possible spatial edge pair $(\mathbf{e}_{ij}^n, \mathbf{e}_{jk}^n)$ with associated weight p_{ijk}, as a directed path and also give the opposite directed edge pair $(\mathbf{e}_{kj}^n, \mathbf{e}_{ji}^n)$ the weight p_{ijk}. As a result, the solution is a directed tree with root node $\mathbf{x}_r$, connected to a sub-tree in each image I^n, as depicted in Fig. 4.2(e).

4.3.3 Fine Alignment and Change Detection

After obtaining the final delineations in all time instants, the iterative GPR method introduced in section 4.2 can be applied once again to perform fine alignment of the solution trees and automatically detect possible changes. For every pair of consecutive time instants I^n and I^{n+1} we take the set of matching seedpoints retained in both instants to be the initial reliable set of matchings S_n^0. We also sample some additional uniformly distributed points from the paths of the solution trees and treat them as the candidate points for matching. We then iterate the GPR estimation and matching until convergence. Sequences of points without correspondences are then detected as potential differences between time instants.

4.4 Experiments and Results

We evaluated our method on 3D 2-photon images of axons in the brain of a mouse, and on 2D time-lapse images of a growing runner bean. We used the DIADEM metric [29] to quantify our results.

Throughout the experiments, we would suppress all tubularity values below 30% of the highest observed value, and set the initial number of values in $\mathcal{S}_n$ to be $L = 10$. In this section, we use $q = 0.75$ as our edge persistence probability and have observed that the results are very similar for q in the range 0.65 to 0.8.

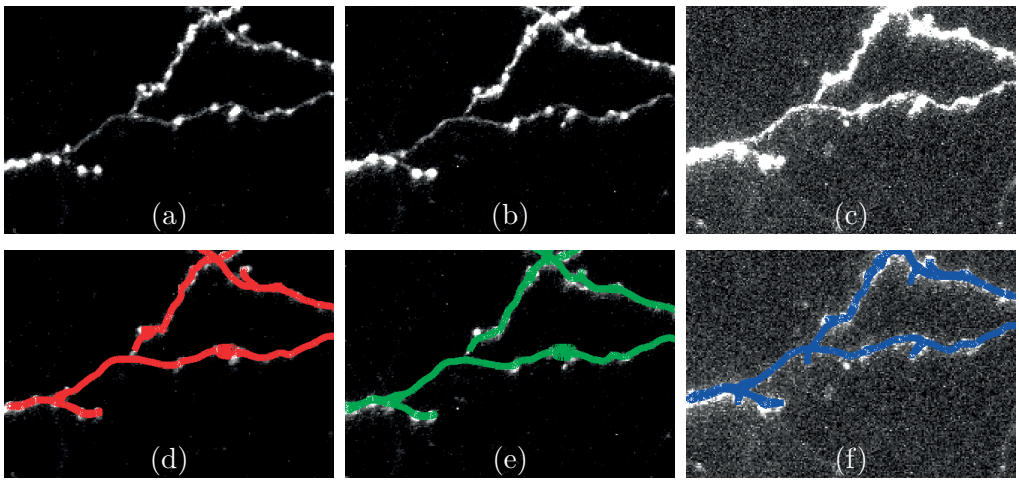


Figure 4.6: Automated reconstruction in BR2 dataset. (a,b,c) Maximum intensity projections of the images. (d,e,f) Reconstructions with DIADEM scores of 0.8471, 0.6422 and 0.5248, respectively. Note that the DIADEM score penalizes heavily even the relatively small errors in (f).

4.4.1 Change Detection in Plant Growing Time Lapse

We first tested our algorithm on a simple time lapse sequence of a growing runner bean. We trained the path classifier using 20000 positive samples and 20000 negative samples, extracted from six images from the sequence. These training images were selected at random and we manually traced the tree in each one to produce positive samples.

Fig. 4.1 depicts example results. The branch structure is correctly reconstructed and the important topological changes are automatically found. In Fig. 4.1(g) in particular, one can see that there is nonlinear deformation between the structures over time. Initially the plant is partially bent and then straightens. Nonetheless, since the GPR allows for nonlinearity, the correct correspondences between the tree structures are found and the tree reconstructions and registration are achieved accurately.

4.4.2 Automatic Change Detection in Brain Circuits

We received large-scale 2-photon laser scanning microscopy images of a sparse set of fluorescently labeled neurons in the neocortex of a rat. Images were taken through

Figure 4.7: One of the 3D volumes used for training the path classifier for the brain datasets. The manual annotation is shown in bright colors.

a permanently implanted cranial window, which allowed tracking specific structures over months during which the rat learned new tasks or underwent new experiences.

We used four large image stacks, labeled 1 to 4, of the same area of the brain at four different times. To train the path classifier, we selected a region from stacks 2 and 4, asked an expert to manually annotate them, and sampled 20000 positive and 20000 negative paths. One of the two training stacks is depicted by Fig. 4.7. Three sequences of smaller volumes were then selected from image stacks 1, 2 and 3 for testing. A single test sequence consists of three volumes representing roughly the same brain area, each one taken from a different stack. We will refer to them as BR1, BR2, and BR3.

For each volume in a dataset, we evaluated the reconstruction performance of our approach when using either zero, one, or two additional time instances. When no additional time instance is used, we simply pick regularly spaced high-tubularity points for the vertices of the graph and our approach reduces to that of Chapter 3. Figs. 4.2 and 4.6 depict our results when using all three images simultaneously on DS1 and DS2, respectively.

		Single	Pair	Triplet
BR1	Image #1	0.0944	0.9473	**0.9770**
	Image #2	0.1828	0.8720	**0.8734**
	Image #3	0.2985	0.9413	**0.9496**
BR2	Image #1	0.2312	**0.8471**	**0.8471**
	Image #2	0.1712	0.5475	**0.6422**
	Image #3	0.0165	**0.6236**	0.5248
BR3	Image #1	0.3369	0.5507	**0.7103**
	Image #2	0.3177	**0.6819**	0.6593
	Image #3	0.2423	**0.6905**	**0.6905**

Table 4.1: Tree reconstruction DIADEM score [29] on our three datasets. These scores were obtained using either single images without temporal consistency or image pairs and triplets and enforcing time consistency.

In Table 4.1, we show the resulting DIADEM scores, which can range from 0.0 to 1.0 with 1.0 being best. That is in each entry of the table, we show the reconstruction score obtained for each image when using a specific number of additional time instances to reconstruct the neural structures. Note that our approach consistently produces more reliable reconstructions than those obtained using a single instance.

FIVE

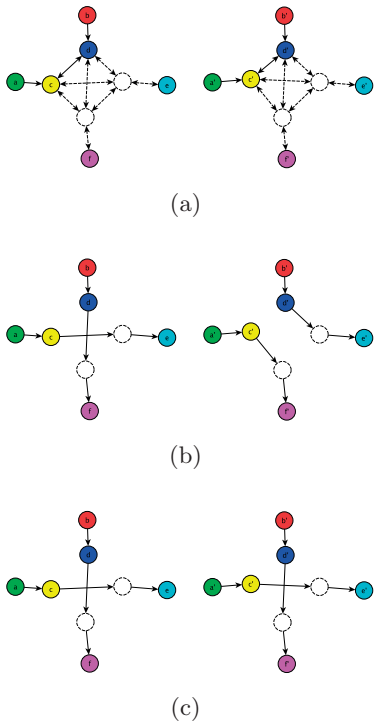
TOPOLOGICAL TEMPORAL CONSISTENCY

In the previous chapter we introduced a method for simultaneously delineating curvilinear tree structures in multiple time instants. We presented a way to enforce temporal consistency between consecutive images and showed that such an approach may bring about greater robustness. However, the method of enforcing temporal consistency applied there is somewhat local. The one we introduce here is able to enforce topological consistency between time instants and we will show that this increases overall delineation accuracy.

5.1 Flow Variables and Temporal Consistency

The starting point for our method will be the spatio-temporal graph $\mathcal{G} = (\mathcal{X}, \mathcal{E})$, where $\mathcal{X} = \{\bigcup_{n=1}^{N} \mathcal{X}^n\}$ and $\mathcal{E} = \mathcal{E}_s \cup \mathcal{E}_t = \{\bigcup_{n=1}^{N} \mathcal{E}_s^n\} \cup \{\bigcup_{n=1}^{N-1} \mathcal{E}_t^n\}$. We introduced it in chapter 4 and we will reuse it in the current formulation. We still operate under the assumption that for every image in the sequence the locations of the tree roots are provided by an operator and are added to the set of graph nodes. We also add an additional imaginary root $\mathbf{x}_r$ and connect it to all these root nodes for all time instants.

In the previous chapter, temporal consistency was enforced by penalizing situations in which binary variables associated to corresponding edges in different time frames



(a)

(b)

(c)

Figure 5.1: A small example graph with two time steps each one consisting of eight vertices. The imaginary root is omitted for clarity. The root vertices in the two time steps are labelled a, b and a', b' respectively. The solid circles represent vertices for which correspondences have been succesfully established and the corresponding vertices are represented with matching colours. The dashed circles represent vertices with no correspondences. (a) The full graph. The edges that join two vertices with correspondences are represented with solid lines. Other edges are represented with dashed lines. For clarity, every pair of edges between two specific vertices is represented by a single double-sided arrow. (b), (c) Two example solutions. While only the solution in (c) looks temporally consistent they would both be considered consistent by the local consistency term due to the missing correspondences between edges in the critical locations. The topological conistency term would penalise solution (b) as the following flow variables are inconsistent: $1 = f^{e}_{ac} \neq f^{e'}_{a'c'} = 0$, $0 = f^{e}_{bd} \neq f^{e'}_{b'd'} = 1$, $0 = f^{f}_{ac} \neq f^{f'}_{a'c'} = 1$, $1 = f^{f}_{bd} \neq f^{f'}_{b'd'} = 0$.

had different values. More specifically, we introduced a persistence probability q. It quantifies how likely it is for a pair of corresponding edges $(\mathbf{e}^{n}_{ij}, \mathbf{e}^{n+1}_{i'j'}) \in \bar{\mathcal{E}}_t$ to both be part of the solution or to be both excluded from it, as opposed to one being included and the other excluded. This yields additional terms that are added to the objective function of Eq. 3.7. Even though it does bring some level of consistency between reconstructions in consecutive images, it remains very localized. In the most ambiguous places of the

overcomplete graph where multiple connectivity patterns are possible, it relies heavily on correspondences being present at crucial locations, which cannot be guaranteed as illustrated by Fig. 5.1.

Here we show how to use the flow variables introduced in Section 3.2.2 to enforce topological consistency. To this end, we first show that this topological consistency can be expressed in terms of the flow variables introduced in Section 3.2.2. We then show that it can be enforced probabilistically by adding a term that is a function of these variables to the objective function of Eq. 3.7. In the results section we will demonstrate that this improves performance.

5.2 Modeling Topological Consistency

Let us assume that we would like to encourage solutions where any two corresponding vertices at two consecutive time instants are connected to the rest of the graph in a consistent manner. In Chapter 4 this was done in a somewhat local way by penalizing situations in which edges adjacent to the two corresponding vertices would be included or left out of the solution inconsistently. By this we mean situations where one of the edges is included in the solution and the other one left out of it, even though they both correspond to the same physical path in the underlying image. Note, however, that we do not forbid completely such situations to allow for potential changes over time.

We propose a more global, topological approach to achieving the same goal. For any two corresponding vertices in consecutive images consider not only the edges adjacent to them but all the edges in some larger neighbourhood around them. More specifically, we would like to favour solutions in which the two paths from the imaginary root vertex to each one of the two corresponding vertices both pass through corresponding edges or neither one of them does. This way, we impose the temporal consistency of the solution's topology rather than only of its local connections.

Recall that temporal edges between vertices in consecutive time frames are one of the outputs of our graph construction procedure, as described in Section 4.2. Let us assume

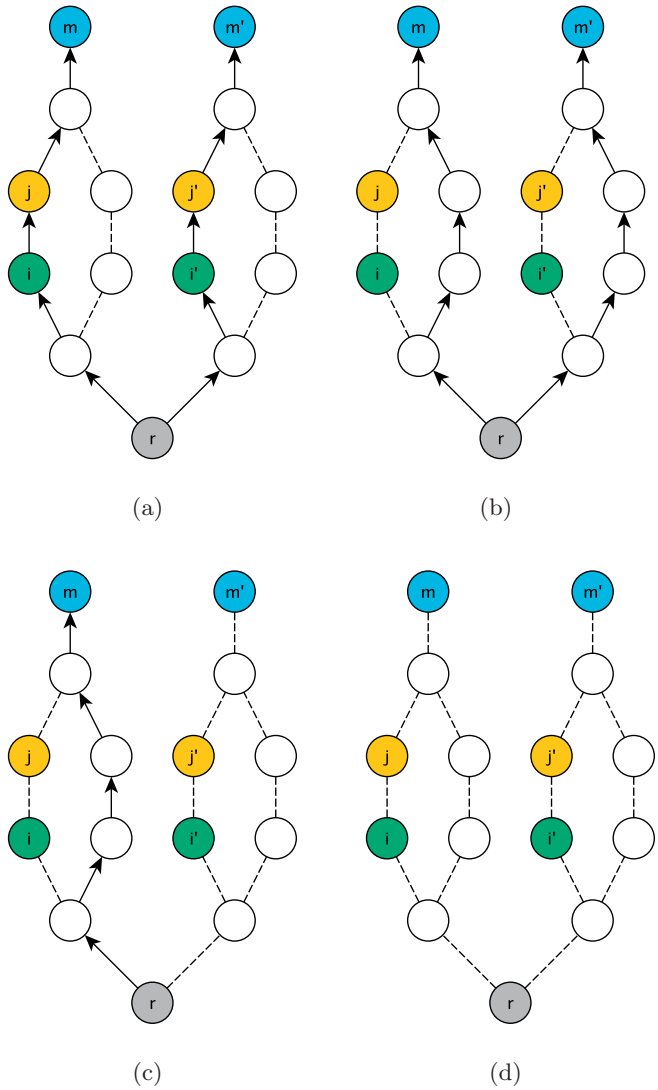
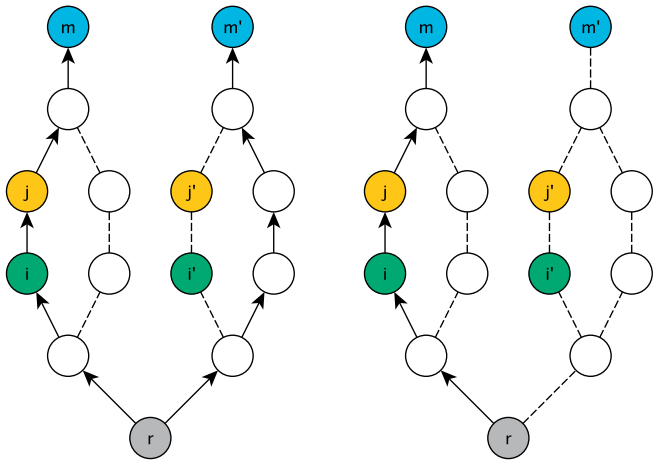


Figure 5.2: These four example graphs illustrate all possible temporal consistency situations for a pair of corresponding vertices $(\mathbf{x}_m, \mathbf{x}_{m'})$ and a pair of corresponding edges $(\mathbf{e}_{ij}, \mathbf{e}_{i'j'})$ in two consecutive time instants considered to be consistent in our approach. Note that in the asymmetric case of (c) its respective reverse case was omitted for clarity. A situation where there is no path from root vertex to $\mathbf{x}_m$ and there exists a path from the root vertex to $\mathbf{x}_{m'}$ that doesn't pass through edge $\mathbf{e}_{i'j'}$ would also be considered consistent.

that for any spatially-connected pair of vertices $\mathbf{x}_i^n$, $\mathbf{x}_j^n$ in image I^n and another pair of spatially connected vertices $\mathbf{x}_{i'}^{n+1}$, $\mathbf{x}_{j'}^{n+1}$ in image I^{n+1} the spatial edge $\mathbf{e}_{ij}^n$ corresponds to the spatial edge $\mathbf{e}_{i'j'}^{n+1}$ provided that $(\mathbf{x}_i^n \leftrightarrow \mathbf{x}_{i'}^{n+1}), (\mathbf{x}_j^n \leftrightarrow \mathbf{x}_{j'}^{n+1}) \in \mathcal{E}_t$. In other



(a) (b)

Figure 5.3: These two example graphs illustrate all possible temporal consistency situations for a pair of corresponding vertices $(\mathbf{x}_m, \mathbf{x}_{m'})$ and a pair of corresponding edges $(\mathbf{e}_{ij}, \mathbf{e}_{i'j'})$ in two consecutive time instants considered to be inconsistent in our approach. Note that their respective reverse cases were omitted for clarity. For instance, in the case of figure (a), the reverse situation where the path to $\mathbf{x}_{m'}$ passes through $\mathbf{e}_{i'j'}$ and the path to $\mathbf{x}_m$ doesn't pass through $\mathbf{e}_{ij}$ would also be considered inconsistent.

words, if for two spatial edges in two consecutive images both endpoints of the one edge correspond to the endpoints of the other edge according to $\mathcal{E}_t$ then we assume that those edges correspond to each other. This way, we define a set of corresponding edges $\bar{\mathcal{E}}_t$. In mathematical terms, we define a set $\bar{\mathcal{E}}_t = \{(\mathbf{e}_{ij}^n, \mathbf{e}_{kl}^{n+1}) | \mathbf{e}_{ij}^n, \mathbf{e}_{kl}^{n+1} \in \mathcal{E}_s \wedge \mathbf{e}_{ik}^{n,n+1}, \mathbf{e}_{jl}^{n,n+1} \in \mathcal{E}_t\}$ of edge correspondences.

Let $\mathbf{x}_m^n$ and $\mathbf{x}_{m'}^{n+1}$ be two corresponding vertices in two consecutive images I^n and I^{n+1} (*i.e.* $(\mathbf{x}_m^n \leftrightarrow \mathbf{x}_{m'}^{n+1}) \in \mathcal{E}_t$). Let also $\mathbf{e}_{ij}^n$ and $\mathbf{e}_{i'j'}^{n+1}$ be two corresponding edges in the same two images, that is $(\mathbf{e}_{ij}^n, \mathbf{e}_{kl}^{n+1}) \in \bar{\mathcal{E}}_t$. We consider $\mathbf{x}_m^n$ and $\mathbf{x}_{m'}^{n+1}$ to be connected to the imaginary root in a temporally consistent way with respect to $\mathbf{e}_{ij}^n$ and $\mathbf{e}_{i'j'}^{n+1}$ if any of the following four configurations arises:

c1) Both $\mathbf{x}_m^n$ and $\mathbf{x}_{m'}^{n+1}$ are part of the solution and the paths connecting them to $\mathbf{x}_r$ traverse the edges $\mathbf{e}_{ij}^n$ and $\mathbf{e}_{i'j'}^{n+1}$ respectively, which implies $f_{ij}^m = f_{i'j'}^{m'} = 1$. (See Fig. 5.2(a).)

c2) Both $\mathbf{x}_m^n$ and $\mathbf{x}_{m'}^{n+1}$ are part of the solution and neither of the paths connecting them to $\mathbf{x}_r$ traverses the edges $\mathbf{e}_{ij}^n$ and $\mathbf{e}_{i'j'}^{n+1}$, which implies $f_{ij}^m = f_{i'j'}^{m'} = 0$. (See Fig. 5.2(b).)

c3) Exactly one of the vertices $\mathbf{x}_m^n$ and $\mathbf{x}_{m'}^{n+1}$ is part of the solution but the path connecting it to $\mathbf{x}_r$ does not traverse the respective one of the corresponding edges $\mathbf{e}_{ij}^n$ and $\mathbf{e}_{i'j'}^{n+1}$, which implies $f_{ij}^m = f_{i'j'}^{m'} = 0$. (See Fig. 5.2(c).)

c4) Neither of the vertices $\mathbf{x}_m^n$ and $\mathbf{x}_{m'}^{n+1}$ is part of the solution, which implies $f_{ij}^m = f_{i'j'}^{m'} = 0$. (See Fig. 5.2(d).)

We introduce a temporal consistency parameter q, the probability that any one of these four situations holds. Conversely, with probability $1-q$, one of the two following inconsistent configurations may arise:

i1) Both $\mathbf{x}_m^n$ and $\mathbf{x}_{m'}^{n+1}$ are part of the solution but only one of the paths connecting them to $\mathbf{x}_r$ traverses the respective one of the corresponding edges $\mathbf{e}_{ij}^n$ and $\mathbf{e}_{i'j'}^{n+1}$, which implies either $f_{ij}^m = 1, f_{i'j'}^{m'} = 0$ or $f_{ij}^m = 0, f_{i'j'}^{m'} = 1$. (See Fig. 5.3(a).)

i2) Exactly one of the vertices $\mathbf{x}_m^n$ and $\mathbf{x}_{m'}^{n+1}$ is part of the solution and the path connecting it to $\mathbf{x}_r$ traverses the respective one of the corresponding edges $\mathbf{e}_{ij}^n$ and $\mathbf{e}_{i'j'}^{n+1}$, which implies either $f_{ij}^m = 1, f_{i'j'}^{m'} = 0$ or $f_{ij}^m = 0, f_{i'j'}^{m'} = 1$. (See Fig. 5.3(b).)

Therefore, q is a key parameter of our algorithm and we will discuss its influence in the results section. Note that the list of the four consistent and two inconsistent cases is an exhaustive one, *i.e.* it covers all possible cases; hence the q and $1-q$ probabilities.

5.3 Augmenting the Objective Function

As in Chapter 4, we assume that the image data $\mathcal{I}$ and the spatial edges $\mathcal{E}_s$ are conditionally independent of the temporal edges $\mathcal{E}_t$ given Y and express the posterior

distribution of Y as

$$P(Y=y|\mathcal{I},\mathcal{X},\mathcal{E}_s,\mathcal{E}_t) \propto P(\mathcal{I},\mathcal{X},\mathcal{E}_s|Y=y)\,P(Y=y|\mathcal{E}_t)\,. \tag{5.1}$$

We can then write

$$P\left(\mathcal{I},\mathcal{X},\mathcal{E}_s|Y=y\right) = \prod_{\mathbf{e}_{ij}^n,\mathbf{e}_{jk}^n\in\mathcal{E}_s}\left(\frac{p_{ijk}}{1-p_{ijk}}\right)^{y_{ij}y_{jk}}\,, \tag{5.2}$$

$$P\left(Y=y|\mathcal{E}_t\right) = \prod_{(\mathbf{x}_m^n,\mathbf{x}_{m'}^{n+1})\in\mathcal{E}_t}\prod_{(\mathbf{e}_{ij}^n,\mathbf{e}_{i'j'}^{n+1})\in\bar{\mathcal{E}}_t}\left(\frac{q}{1-q}\right)^{2f_{ij}^m f_{i'j'}^{m'}-f_{ij}^m-f_{i'j'}^{m'}}\,, \tag{5.3}$$

where p_{ijk} is the probability that the edge pair $(\mathbf{e}_{ij}^n,\mathbf{e}_{jk}^n)$ is a part of a tubular structure. The image term of Eq. 5.2 is exactly the same as in the method described in Chapter 4. Following similar steps as in Chapter 4, we derive the prior term of Eq. 5.3.

$$\begin{aligned}
&P\left(Y=y|\mathcal{E}_t\right) = \\
&= \prod_{(\mathbf{x}_m^n,\mathbf{x}_{m'}^{n+1})\in\mathcal{E}_t}\prod_{(\mathbf{e}_{ij}^n,\mathbf{e}_{i'j'}^{n+1})\in\bar{\mathcal{E}}_t} P\left(F_{i'j'}^{m'}=f_{i'j'}^{m'}|F_{ij}^m=f_{ij}^m\right)\prod_{\mathbf{e}_{ij}^0\in\mathcal{E}_0}P\left(F_{ij}^0=f_{ij}^m\right) && (5.4)\\
&\propto \prod_{(\mathbf{x}_m^n,\mathbf{x}_{m'}^{n+1})\in\mathcal{E}_t}\prod_{(\mathbf{e}_{ij}^n,\mathbf{e}_{i'j'}^{n+1})\in\bar{\mathcal{E}}_t}\left(\frac{P\left(F_{i'j'}^{m'}=f_{i'j'}^{m'}|F_{ij}^m=1\right)}{P\left(F_{i'j'}^{m'}=f_{i'j'}^{m'}|F_{ij}^m=0\right)}\right)^{f_{ij}^m} P\left(F_{i'j'}^{m'}=f_{i'j'}^{m'}|F_{ij}^m=0\right) && (5.5)\\
&\propto \prod_{(\mathbf{x}_m^n,\mathbf{x}_{m'}^{n+1})\in\mathcal{E}_t}\prod_{(\mathbf{e}_{ij}^n,\mathbf{e}_{i'j'}^{n+1})\in\bar{\mathcal{E}}_t}\left(\frac{P\left(F_{i'j'}^{m'}=f_{i'j'}^{m'}|F_{ij}^m=1\right)}{P\left(F_{i'j'}^{m'}=f_{i'j'}^{m'}|F_{ij}^m=0\right)}\right)^{f_{ij}^m} \\
&\qquad\qquad \left(\frac{P\left(F_{i'j'}^{m'}=1|F_{ij}^m=0\right)}{P\left(F_{i'j'}^{m'}=0|F_{ij}^m=0\right)}\right)^{f_{i'j'}^{m'}} P\left(F_{i'j'}^{m'}=0|F_{ij}^m=0\right) && (5.6)\\
&\propto \prod_{(\mathbf{x}_m^n,\mathbf{x}_{m'}^{n+1})\in\mathcal{E}_t}\prod_{(\mathbf{e}_{ij}^n,\mathbf{e}_{i'j'}^{n+1})\in\bar{\mathcal{E}}_t}\left(\frac{P\left(F_{i'j'}^{m'}=1|F_{ij}^m=1\right)}{P\left(F_{i'j'}^{m'}=1|F_{ij}^m=0\right)}\right)^{f_{ij}^m f_{i'j'}^{m'}} && (5.7)\\
&\qquad\qquad \left(\frac{P\left(F_{i'j'}^{m'}=0|F_{ij}^m=1\right)}{P\left(F_{i'j'}^{m'}=0|F_{ij}^m=0\right)}\right)^{f_{ij}^m(1-f_{i'j'}^{m'})}\left(\frac{P\left(F_{i'j'}^{m'}=1|F_{ij}^m=0\right)}{P\left(F_{i'j'}^{m'}=0|F_{ij}^m=0\right)}\right)^{f_{i'j'}^{m'}} && (5.8)\\
&\propto \prod_{(\mathbf{x}_m^n,\mathbf{x}_{m'}^{n+1})\in\mathcal{E}_t}\prod_{(\mathbf{e}_{ij}^n,\mathbf{e}_{i'j'}^{n+1})\in\bar{\mathcal{E}}_t}\left(\frac{q}{1-q}\right)^{f_{ij}^m f_{i'j'}^{m'}}\left(\frac{1-q}{q}\right)^{f_{ij}^m-f_{ij}^m f_{i'j'}^{m'}}\left(\frac{1-q}{q}\right)^{f_{i'j'}^{m'}} && (5.9)
\end{aligned}$$

$$\propto \prod_{(\mathbf{x}_m^n, \mathbf{x}_{m'}^{n+1}) \in \mathcal{E}_t} \prod_{(\mathbf{e}_{ij}^n, \mathbf{e}_{i'j'}^{n+1}) \in \bar{\mathcal{E}}_t} \left(\frac{q}{1-q}\right)^{2f_{ij}^m f_{i'j'}^{m'} - f_{ij}^m - f_{i'j'}^{m'}} , \tag{5.10}$$

where $\mathcal{E}_0 \subset \mathcal{E}_S$ denotes the set of spatial edges associated to the first image at $n = 0$. We assume uniform prior for these edges and drop the $P\left(F_{ij}^0 = f_{ij}^0\right)$ terms from Eq. 5.4. Eq. 5.8 is obtained by simple algebraic manipulations and dropping the constant terms that do not depend on the f_{ij}^m or $f_{i'j'}^{m'}$ variables. Finally, we substitute the persistent probabilities $P\left(F_{i'j'}^{m'} = 0 | F_{ij}^m = 0\right)$ and $P\left(F_{i'j'}^{m'} = 1 | F_{ij}^m = 1\right)$ with q, and probabilities $P\left(F_{i'j'}^{m'} = 1 | F_{ij}^m = 0\right)$ and $P\left(F_{i'j'}^{m'} = 0 | F_{ij}^m = 1\right)$ with $(1-q)$ in Eq. 5.9, and rearrange common terms in Eq. 5.10.

Once again, finding a solution that maximizes the product of probabilities of Eq. 5.2 and Eq. 5.3 is equivalent to finding one that minimizes the sum of negative logarithms of these probabilities. This leads to the *maximum a posteriori* problem of Eq. 5.11.

$$\begin{aligned} y^* &= \arg\max_{y \in \mathcal{Y}} P\left(\mathcal{I}, \mathcal{X}, \mathcal{E}_s | Y = y\right) P\left(Y = y | \mathcal{E}_t\right), \\ &= \arg\min_{y \in \mathcal{Y}} \sum_{\mathbf{e}_{ij}^n, \mathbf{e}_{jk}^n \in \mathcal{E}_s} w_{ijk} y_{ij}^n y_{jk}^n \\ &+ \sum_{(\mathbf{x}_m^n, \mathbf{x}_{m'}^{n+1}) \in \mathcal{E}_t} \sum_{(\mathbf{e}_{ij}^n, \mathbf{e}_{i'j'}^{n+1}) \in \bar{\mathcal{E}}_t} w_p \left(2 f_{ij}^m f_{i'j'}^{m'} - f_{ij}^m - f_{i'j'}^{m'}\right). \end{aligned} \tag{5.11}$$

where $w_{ijk} = -\log \frac{p_{ijk}}{1-p_{ijk}}$, p_{ijk} is the probability of edge pair $(\mathbf{e}_{ij}^n, \mathbf{e}_{jk}^n)$ being part of a tubular structure, $w_p = -\log \frac{q}{1-q}$, and $\mathcal{Y}$ is the set of all feasible trees with root $\mathbf{x}_r$. Note that the connectivity constraints are the same as before and can therefore be imposed by performing the minimization under the linear constraints of Eqs. 3.8, which are also expressed in terms of the the y_{ij}^n and f_{il}^m variables. The problem therefore remains a QMIP.

In addition to being more global than the method of [30], this approach has the added benefit of being able to handle cases where the sampling step returns a lot of sample points that were not assigned correspondences in adjacent time steps. This is especially important in areas crucial to the topology of the structure. In case of neurons

in 3D brain images the most prevalent such case would be two branches crossing very near to each other in the z dimension. In such a setting it is difficult to tell whether they form an actual branching or not. A simple example illustrating this extra robustness coming from our approach is shown in Fig. 5.1. The top subfigure presents a graph with two time steps each one consisting of eight vertices with the imaginary root omitted for clarity. The dashed circles represent vertices with no correspondences. The middle and the bottom subfigure represent two different solutions. While only the one in the bottom subfigure looks temporally consistent they would both be considered consistent by the local consistency term due to the missing correspondences between edges in the critical locations. The topological consistency term would penalize the solution in the middle subfigure as several flow variables would be inconsistent.

In practice, for a given pair of corresponding vertices, it is neither beneficial nor computationally efficient to include the flow consistency constraints for every pair of corresponding edges. Imposing it for edges very distant from the vertex in question might put too much weight on the persistency term as compared to the image term. It would also produce a vast amount of quadratic terms in the cost function, which would slow down the computation considerably. Instead, for a given pair of vertices $\mathbf{x}_m^n$ and $\mathbf{x}_{m'}^{n+1}$ we only take into consideration those edges whose distance from the vertex in question is smaller than some predefined threshold r.

Note that, given the fifth constraint of Eqs. 3.8, the presented method might be considered a generalization of the one presented in Chapter 4. Namely, in the extreme case of $r = 1$ it is completely equivalent to it. Setting the distance parameter r to values greater than one transforms the method into its more robust counterpart.

5.4 Speeding Up the Computation

The path from the imaginary root to any vertex $\mathbf{x}_r^n$ of the spatio-temporal graph cannot pass through edges residing at time steps different from I^n. Therefore, it is easy to predict that all of the respective flow variables will be equal to zero for any valid

Figure 5.4: Road images used to train the path classifier for the aerial photographs dataset. The manual annotation is shown in bright colors.

solution and it is unnecessary to introduce them at all. Removing them from the problem together with any constraints that might involve them results in an equivalent yet simpler optimization. This usually reduces the time and memory needed to solve it and of course does not affect the quality of the solution.

5.5 Results

We evaluated our method on 3D 2-photon images of axons in the brain of a rat, such as the ones used in the previous chapter. We also tested our approach on a road network delineation task. We used a sequence of three aerial images of the same area taken at three very different time instants. The appearance varies from one to the other due to different illumination, different level of occlusion from trees captured in different seasons etc. Six regions spanning those varying appearances were picked to train a single classifier the same way as it was done for the brain images. Two of them can be seen in Fig. 5.4. We also cropped one sequence of three images for testing.

5.5.1 Overall Performance

In Table 5.1, we compare the results of reconstructing independently in each test image of these datasets using the QMIP method of Chapter 3, of imposing local temporal constraints as in Chapter 4, and of imposing topological temporal constraints as discussed in this chapter. To this end, we measure the quality of the reconstructions in terms of the DIADEM metric [29]. It measures the similarity with the ground truth in a way that is appropriate for trees and ranges from 0.0 to 1.0, with 1.0 being best. To obtain these results, we set the persistence probability value q, introduced in section 5.1, to 0.75 in all experiments. For values close to 0.5 the consistency term proved to be completely outweighted by the image term and the temporal consistency would not be enforced at all. Values close to 1 tended to prevent any differences between the graphs in different time instants. This would sometimes cause ignoring the image term and pruning some of the branches completely, as this of course results in a highly time-consistent solution. Furthermore, high values of q tended to slow down the optimization.

We present some representative results in Figs. 5.5 and 5.6. They illustrate that both the local and the topological consistency methods favor reconstructions consistent over time, with the topological one providing a higher level of temporal consistency. Setting the range parameter to 4 instead of 1 also improves accuracy.

The range parameter for the topological consistency, also introduced in section 5.1, was set to 4 in all experiments. This proved sufficient to improve robustness without unduly increasing the computational complexity. High values of the range parameter had a tendency to slow down the optimization and to give a very high weight to the consistency prior. Similarily as in the case of high persistence probability values, this would sometimes result in pruning some of the correctly delineated branches of the overcomplete graph.

As can be seen in Table 5.1, enforcing local temporal consistency improves the overall quality of the results and imposing topological consistency improves them even

more. However, the effect of consistency depends on the quality of the path classifier weights. If the classifier provides reasonable scores more often than not, temporal consistency propagates them. If it performs poorly, combining multiple weak solutions might not help. It might even hurt by producing solutions that are consistently wrong.

We have also tested how removing the unnecessary flow variables, as described in section 5.4, influences the computation time. Due to the non-deterministic nature of the optimization procedure we performed each of the optimizations ten times and observed consistent behaviour with only minor differences between specific runs. In the case of the local consistency the computation time turned out to be consistently lower, reduced for instance from 16s to 4s on one small example and from 510s to 248s on a bigger one. In the case of the topological consistency the computation would also complete around two to three times faster. However, it is worth noting that for one example the optimization time raised from 2011s to 3172s after removing the unnecessary flow variables.

		No consistency	Local consistency	Topological consistency
Brain 1 V: 161 E: 450	Image #1	0.7722	**0.7903**	**0.7903**
	Image #2	**0.6890**	**0.6890**	0.6746
	Image #3	0.4761	0.4395	**0.9185**
	Average score	0.6458	0.6396	**0.7945**
	Computation time	9s	14s	17s
Brain 2 V: 108 E: 298	Image #1	0.6303	0.3654	**0.6345**
	Image #2	0.4653	0.3433	**0.5178**
	Image #3	0.5929	0.3374	**0.6325**
	Average score	0.5628	0.3487	**0.5949**
	Computation time	3s	4s	10s
Brain 3 V: 134 E: 328	Image #1	0.4964	**0.5444**	0.4242
	Image #2	0.2437	0.5327	**0.5884**
	Image #3	0.5200	0.5771	**0.6766**
	Average score	0.4201	0.5514	**0.5631**
	Computation time	3s	4s	5s
Brain 4 V: 200 E: 612	Image #1	0.3846	**0.6000**	0.5940
	Image #2	0.5088	0.5272	**0.5677**
	Image #3	0.5910	0.6118	**0.6340**
	Average score	0.4948	0.5797	**0.5986**
	Computation time	19s	23s	33s
Brain 5 V: 309 E: 958	Image #1	0.3713	**0.3956**	0.3952
	Image #2	0.2439	0.2915	**0.3687**
	Image #3	0.2060	0.2761	**0.3134**
	Average score	0.2737	0.3211	**0.3591**
	Computation time	83s	248s	3172s
Roads V: 375 E: 1228	Image #1	0.2650	0.4630	**0.4680**
	Image #2	0.3880	0.4000	**0.4150**
	Image #3	0.3240	0.3330	**0.4670**
	Average score	0.3257	0.3987	**0.4500**
	Computation time	30027s	86400s*	86400s*

Table 5.1: DIADEM scores [29] for the brain images datasets and the road images dataset. The scores in the first column were obtained without temporal consistency, that is, by using the QMIP method of Chapter 3. The scores in the second and the third column were obtained by imposing local temporal consistency and topological temporal consistency, respectively. The branch-and-bound algorithm used by the Gurobi optimization engine is a non-deterministic one but the computation times change very little between different runs. Therefore, the example computation times can be considered representative. In case of the roads dataset the path classifier did not put very discriminative weights on the edge-pairs of the initial graph, which resulted in very slow convergence. In case of this dataset, for both the temporal-consistency methods the computation was terminated after 24 hours and the best incumbent solution was used.

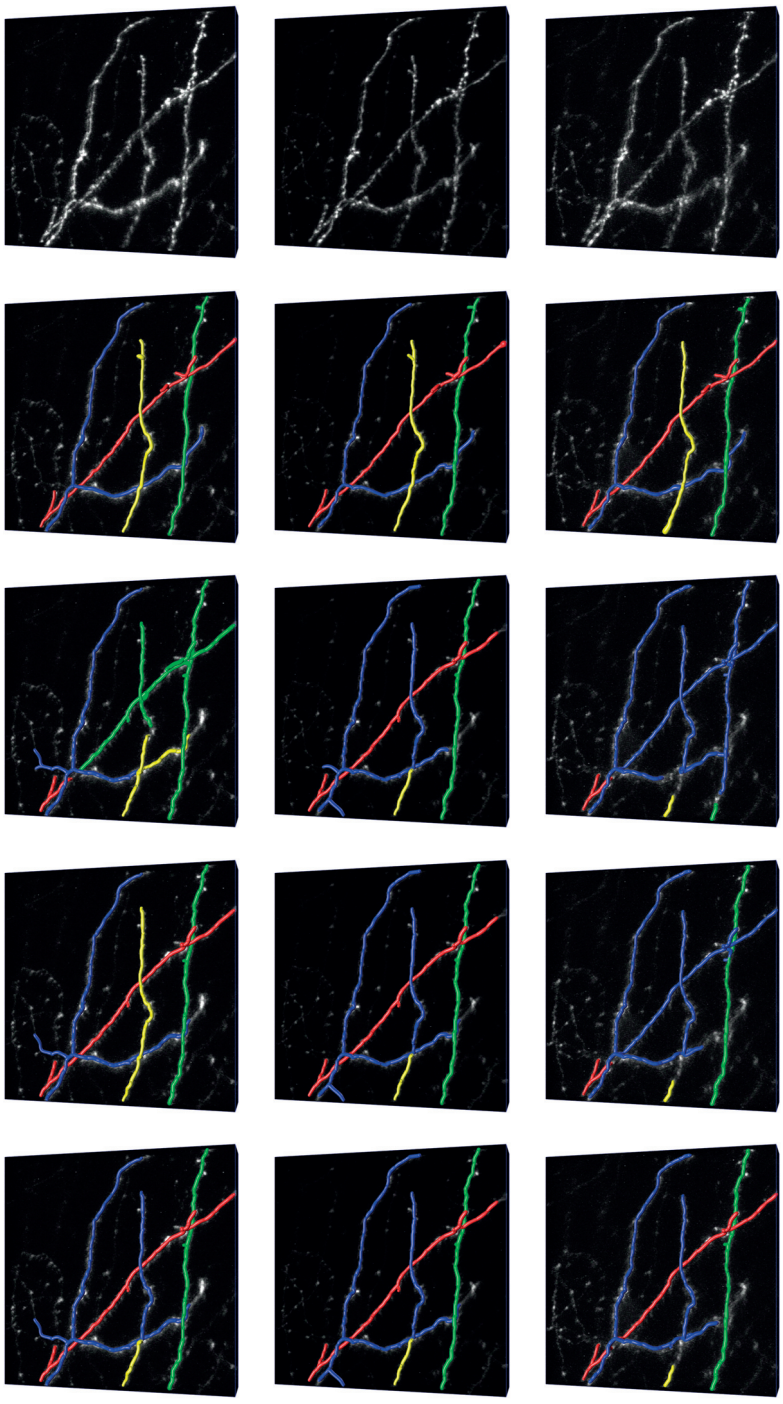


Figure 5.5: Example brain data results. The first row presents the input images. The second row presents the ground truth. The third row presents the results computed with no temporal consistency. The fourth and fifth rows present results with flow persistency set to 0.75 and range parameter set to 1 and 4, respectively.

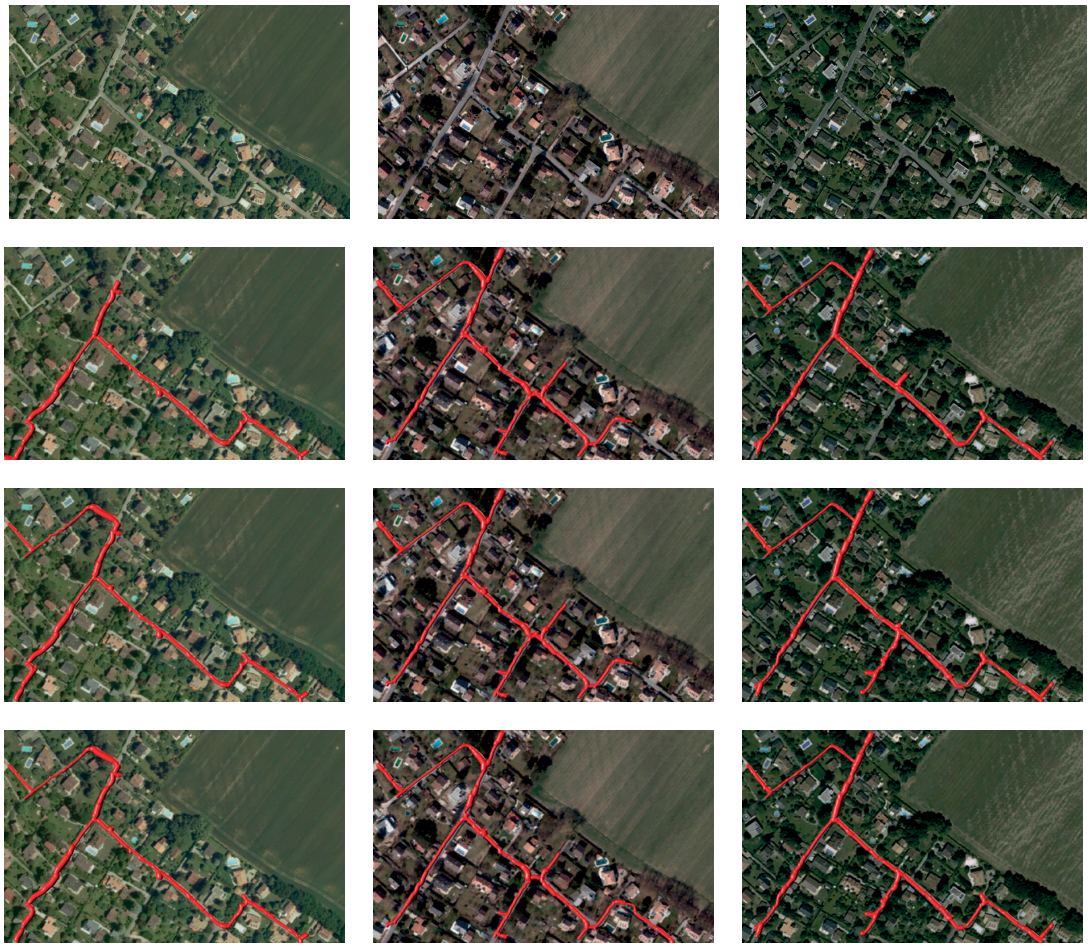


Figure 5.6: Road data results. The first row presents the input images. The second row presents the results computed with no temporal consistency. The third and fourth rows present results with flow persistency set to 0.75 and range parameter set to 1 and 4, respectively.

SIX

CONCLUDING REMARKS

In this work, we approached the problem of automatically delineating curvilinear tree structures in 2D images and 3D image stacks. As pointed out in Chapter 1, such structures are prevalent in nature and span many different scales. Reliably reconstructing them in an automated fashion is therefore of great value in many different scientific domains. We focused mainly on situations where images of the same object of interest taken at different time instants are available. Unlike virtually all of the existing methods for automatic tree structures delineation, we took into account temporal consistency constraints and looked at all the images at once to reason for the most plausible solution.

In this chapter, we first outline the contributions of this thesis. We then proceed to point out the limitations of the presented methods and discuss possible future work directions.

6.1 Summary and Contributions

Prior to this thesis, virtually all of the existing methods could be described as single time instant ones. Given a sequence of images of the same object of interest taken at different time instants they would proceed to process them independently, one by

one. In Chapter 5 we introduced a novel method that processes all images at once and finds local and stable structures that are consistent over time, and which can be used to disambiguate cases where individual time-instance reconstructions would fail. We proposed a method for finding and matching corresponding centerline points throughout the sequence. We then introduced the idea of a spatio-temporal graph that is able to encode not only spatial information in single time instants but also correspondences between edges across time. The novel mathematical optimization method, which is then applied to the graph, introduces a temporal consistency prior that encourages solutions where for two consecutive time instants corresponding candidate edges are either both retained or both rejected from the final solution. We also provided experimental results on 3D brain microscopy stacks to demostrate that the additional temporal information brings about improvement in the quality of the results.

Chapter 5 provides a more global method for reconstructing tree structures from sequences of images in a temporally consistent way. It uses the same spatio-temporal graph principle. However, we introduced a novel, improved temporal consistency prior. Instead of focusing on the very local consistency of single edges of the overcomplete graph we proposed a method for describing topological relationships. This favors solutions whose connectivity is consistent over time. We demonstrated experimentally that introducing these topological relationships results in better final reconstructions. We tested the approach both on 2D aerial road network photographs and 3D microscopy stacks.

Additionally, we proposed a simple, single time instant algorithm for delineating tree structures. It consists of computing a Minimum Spanning Arborescence as an initial, background-contaminated solution and then optimally pruning the spurious branches. Chapter 3 contains an outline of the algorithm and computational complexity analysis. We showed that computing the reconstruction can be performed in polynomial time and provided experimental validation on 3D microscopy stacks that proved that the approach provides competitive results. This is valuable when dealing with very large

image stacks in situations where a slight decrease in the quality of reconstructions is acceptable in favor of speed.

We also developed a novel set of constraints for computing a Minimum Tree from an undirected graph, a problem closely related to delineating tree structures. It is described in Appendix B. Suprisingly, and to the best of our knowledge, this problem had not been solved before.

6.2 Limitations and Future Work

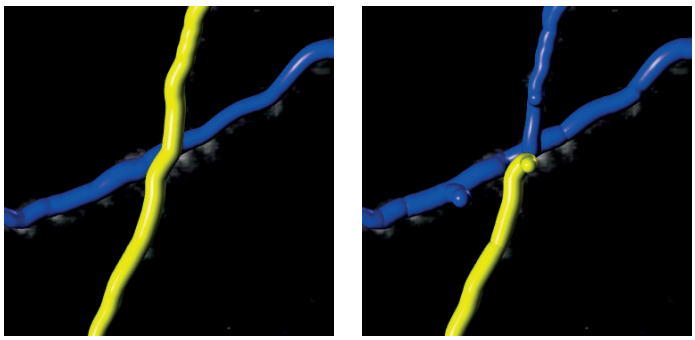
In this section we discuss the main failure modes and limitations of the presented methods and discuss possible ways to overcome them in the future.

6.2.1 Loopy Reconstructions

In all methods presented in this thesis we assumed that the curvilinear structures of interest were trees. In many cases, such as delineating neurons in the brain, which was the main focus of our work, this is not a limitation at all. The assumption stems from the natural properties of the imaged structures. However, reconstructing loopy networks in multiple time instants may also be of scientific interest. For this reason, a promising direction of future work would be to incorporate our temporal consistency ideas into the framework of [62]. This framework is a generalization of the QMIP method outlined in Chapter 3. It allows to choose between loopy and non-loopy reconstructions. This would make our methods more generic. However, the method of [62] involves a more complex optimization scheme, which would most likely pose computational efficiency challenges.

6.2.2 Geometric prior

We do not make any explicit assumptions about the geometry of the reconstructed tree structures. This sometimes results in reconstructions whose connectivity looks unrealistic at some places. An example of such a situation is presented in Fig. 6.1.



(a) (b)

Figure 6.1: Example of an error in reconstruction that could be avoided by imposing a geometric prior. (a) The ground truth reconstruction. Two separate branches pass close to each other. (b) The output of the QMIP method without temporal consistency. The error in the connectivity would be easy to spot for the human eye even without looking at the image data.

The fact that such errors are easy to spot for humans even without looking at the image data suggest that some form of geometric prior included in the reconstruction procedure could result in more accurate results. In the specific case of Fig. 6.1 it would be beneficial to discourage solutions where one branch terminates almost exactly at a branching point of a different one.

6.2.3 Speeding up the Optimization

One of the main drawbacks of the mathematical optimization methods for delineating tree structures is the computational complexity of the optimization process. Therefore, reducing this complexity is always a desirable line of research. In Chapter 3 we described the so-called flow variables, which are employed to enforce the correct structure of the solutions. To reduce the number of the flow variables and constraints related to them, one could perform a Biconnected Components Decomposition [33] of the tubular graph. This would allow to detect and eliminate unnecessary flow variables and constraints and hopefully accelerate the optimization procedure. An example initial graph and its Biconnected Components Decomposition are shown in Fig. 6.2. One needs to create a flow variable for every vertex-edge pair only within specific biconnected components.

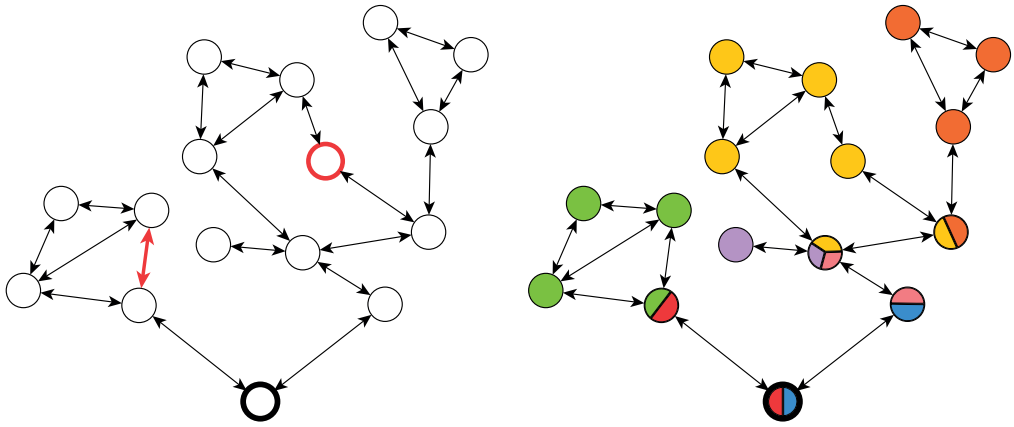


Figure 6.2: Biconnected Components Decomposition. (a) An example initial graph. The bottom vertex is the manually anotated root. Note that it is impossible to find a solution where the path from the root vertex to the red vertex passes through the red edges. We can therefore remove the respective flow variables from the program. (b) The Biconnected Components Decomposition of the initial graph. The different biconnected components are marked with different colours. It is now easy to determine which of the flow variables can be removed from the program. An exhaustive set of flow variables needs to be introduced only within a single biconnected component.

A number of flow variables and constraints also needs to be created for the so called cut vertices, *i.e.* the vertices belonging to multiple components. It is worth noting that some careful engineering would be required to express the topological consistency approach using the reduced number of flow variables.

6.2.4 Path classifier

The multiple time instant methods introduced in this thesis rely on the path classification approach outlined in Chapter 3. This results in two important considerations.

First of all, poor weights might result in the temporal consistency having an adverse effect. The methods that we proposed attempt to reason for a better solution by looking at all image data at once. However, if a given part of the structure of interest is incorrectly classified in the majority of time instants, this incorrect interpretation will be propagated to the other instants pushing them in the wrong direction.

The second important problem is that the convergence speed of the mathematical optimizations depends on how picky the classifier is. More informative probabilities both on positive and negative edge-pairs of the spatio-temporal graph result in faster convergence. Conversely, probabilities close to 0.5 tend to slow down the optimization considerably. In our experience, this effect is strongest in settings where complex background patterns can be observed, as in the case of aerial photographs of road networks.

These two issues justify the need for additional research in order to create an improved, more robust and more picky path classification method.

SEVEN

APPENDICES

A Optimal Pruning of an Edge Pair Weighted Tree

In Chapter 3 we introduced a simplified and very fast method for delineating tree structures by building an overcomplete edge weighted graph, computing its Minimum Spanning Arborescence and then optimally pruning spurious branches. The reason why we decided to use an edge weighted graph is the fact that efficient methods exist for retrieving a Minimum Spanning Arborescence in such a setting.

In this appendix we propose a similar algorithm for optimally pruning an edge-pair weighted graph. Even though no polynomial time methods are known for finding a Minimum Spanning Arborescence of such a graph, the algorithm might still be useful for refining the output of another method that does not guarantee optimality. In particular, it could be used together with the QMIP method of Chapter 3 to obtain results of lower quality but in shorter time. One could terminate the optimization procedure prematurely and use the pruning algorithm presented below to refine the reconstructions.

Let us consider the edge-pair-weighted optimisation problem of Chapter 3. Let us also consider that we are given a suboptimal solution $y \in \mathcal{Y}$ of the problem, *i.e.* a tree weighted according to the cost function of Eq. 3.7. Similarily to the edge-weighted case

we define the optimal cost of a subtree rooted at vertex $\mathbf{x}_i$ as

$$\begin{cases} c(\mathbf{x}_i) & = \sum_{\mathbf{x}_j \in N(\mathbf{x}_i)} \min\left(w_{pij} + c(\mathbf{x}_j), 0\right), \qquad \forall i \neq r \\ c(\mathbf{x}_r) & = \sum_{\mathbf{x}_j \in N(\mathbf{x}_r)} \min\left(c(\mathbf{x}_j), 0\right), \end{cases} \tag{A.1}$$

where w_{pij} is the weight of the edge-pair consisting of the edges $\mathbf{e}_{pi}$, $\mathbf{e}_{ij}$, the former of which is the edge between the unique parent of vertex $\mathbf{x}_i$ and the vertex $\mathbf{x}_i$ itself. The value $c(\mathbf{x}_i)$ expressed by the first equation represents the cost of an optimal subtree rooted at vertex $\mathbf{x}_i$ including the weights of the edge-pairs for which $\mathbf{x}_i$ is the middle vertex. The second of the two equations defines the optimal subcost of a subtree rooted at the actual root node $\mathbf{x}_r$, *i.e.* the optimal subcost of the whole tree. This leads us to the method expressed in Algorithm 2.

As in Algorithm 1 of Chapter 3, the main procedure also makes two separate passes through the initial tree, one when calling GET_OPTIMAL_COST($\mathbf{x}_j$) in a loop and the other one when calling RETRIEVE_SOLUTION($\mathbf{x}_r$). Each vertex and each edge is visited at most once during both passes and so the computational complexity of Algorithm 2 is again $\mathcal{O}(|\mathcal{X}| + |\mathcal{E}_s|)$.

Algorithm 2 Optimal edge-pair-weighted tree pruning

```
for e_ij ∈ E_s do
    take_ij ← 0
    y_ij ← 0
end for
for x_j ∈ C(x_r) do
    b ← GET_OPTIMAL_COST(x_j)
    if b < 0 then
        take_rj ← 1
    end if
end for
RETRIEVE_SOLUTION(x_r)


function GET_OPTIMAL_COST(x_i)
    cost ← 0
    for x_j ∈ C(x_i) do
        p ← P(x_i)
        b ← GET_OPTIMAL_COST(x_j) + w_pij
        if b < 0 then
            take_ij ← 1
            cost ← cost + b
        end if
    end for
    return cost
end function

procedure RETRIEVE_SOLUTION(x_i)
    for x_j ∈ C(x_i) do
        if take_ij = 1 then
            y_ij ← 1
            RETRIEVE_SOLUTION(x_j)
        end if
    end for
end procedure
```

B Tree Constraints in an Undirected Graph

In section 3.2.2 we presented the flow variable constraints adapted from [19] that are part of the Quadratic Mixed Integer Program used to compute the Minimum Arborescence from the initial overcomplete directed tubular graph. Their role is to ensure that the final solution is indeed an arborescence. Recall that for every pair of sample points $(\mathbf{x}_i, \mathbf{x}_j)$ that are sufficiently close to each other we add two oppositely directed edges. The only reason why we use directed edges is that the flow constraints of [19] are defined on directed graphs.

In this appendix we present a novel set of constraints that can constrain solutions to be trees while working with undirected graphs. Since such an undirected graph only has half the number of edges compared to an equivalent directed one, we developed the constraints in hope that they would accelerate the optimization procedure of the original QMIP method of Chapter 3. Unfortunately, they turned out to be slower on average.

B.1 The constraints

Let $\mathcal{G} = (\mathcal{X}, \mathcal{E}_{su})$ be an undirected version of the tubular graph of Chapter 3. For any pair of connected sample points $(\mathbf{x}_i, \mathbf{x}_j)$ there exists only one undirected edge $\mathbf{e}_{\{i,j\}}$. Let also $y_{\{i,j\}}$ be a binary undirected indicator variable whose value is equal to one if edge $\mathbf{e}_{\{i,j\}}$ is part of the solution and equal to zero if it is left out of the solution. Finally, let $f^l_{\{i,j\}}$ be an undirected flow variable whose value is equal to one if the flow from the root vertex to the terminal vertex $\mathbf{x}_l$ passes through the edge $e_{\{i,j\}}$, and is otherwise equal to zero. An illustrative example of such a graph is presented in Fig. 7.1. It is an undirected equivalent of the example directed graph of Fig. 3.9.

The flow constraints for an undirected graph can then be written as follows.

$$\sum_{\mathbf{x}_j \in \delta(\mathbf{x}_r)} f^l_{\{r,j\}} \leq 1, \qquad \forall \mathbf{x}_l \in \mathcal{X} \setminus \{\mathbf{x}_r\}, \tag{B.1}$$

$$\sum_{\mathbf{x}_j \in \delta(\mathbf{x}_l)} f^l_{\{l,j\}} \leq 1, \qquad \forall \mathbf{x}_l \in \mathcal{X} \setminus \{\mathbf{x}_r\}, \tag{B.2}$$

$$\sum_{\mathbf{x}_j \in \delta(\mathbf{x}_i)} f^l_{\{i,j\}} = 2c_{l,\{i,j\}}, \quad \begin{matrix} \forall \mathbf{x}_l \in \mathcal{X} \setminus \{\mathbf{x}_r\}, \\ \forall \mathbf{x}_i \in \mathcal{X} \setminus \{\mathbf{x}_r, \mathbf{x}_l\}, \end{matrix} \tag{B.3}$$

$$c_{l,\{i,j\}} \in \{0,1\}, \quad \begin{matrix} \forall \mathbf{x}_l \in \mathcal{X} \setminus \{\mathbf{x}_r\}, \\ \forall \mathbf{x}_i \in \mathcal{X} \setminus \{\mathbf{x}_r, \mathbf{x}_l\}, \end{matrix} \tag{B.4}$$

$$f^l_{\{i,j\}} \leq y_{\{i,j\}}, \qquad \forall e_{\{i,j\}} \in \mathcal{E}_{su},\ \mathbf{x}_l \in \mathcal{X} \setminus \{\mathbf{x}_r, \mathbf{x}_i, \mathbf{x}_j\}, \tag{B.5}$$

$$f^i_{\{i,j\}} + f^j_{\{i,j\}} = y_{\{i,j\}}, \quad \forall e_{\{i,j\}} \in \mathcal{E}_{su}, \tag{B.6}$$

$$y_{\{i,j\}} \in \{0,1\}, \quad \forall e_{\{i,j\}} \in \mathcal{E}_{su}, \tag{B.7}$$

Constraints B.1 and B.2 are analogous to the first two constraints of Eqs. 3.8. Constraints B.3 and B.4 collectively ensure the conservation of flow. Constraint B.5 only allows flows to pass through active edges. Constraint B.6 ensures that if edge $\mathbf{e}_{\{i,j\}}$ is active then exactly one of the unit flows terminating at $\mathbf{x}_i$ and $\mathbf{x}_j$ has to traverse $\mathbf{e}_{\{i,j\}}$. Constraint B.7 restricts the edge indicator variables to binary values.

We have implemented the undirected constraints and employed them in the QMIP method of Chapter 3. Experimenting with different examples we managed to consistently reproduce exactly the same results as the ones obtained using the original directed graphs, which confirms the correctness of the proposed constraints. As mentioned before, unfortunately they would cause the optimization procedure to take a longer time to converge.

B.2 Directedness of the edges

Surprisingly, despite the fact that we defined the constraints on an undirected graph, one can apply them to a graph where different weights are given to oppositely oriented edges or edge-pairs. Even though we only have one indicator variable for any undirected

edge $\mathbf{e}_{\{i,j\}}$, in any feasible solution it has a well defined direction. That direction can be determined by looking at the flow variables $f^i_{\{i,j\}}$ and $f^j_{\{i,j\}}$, at most one of which can be equal to one.

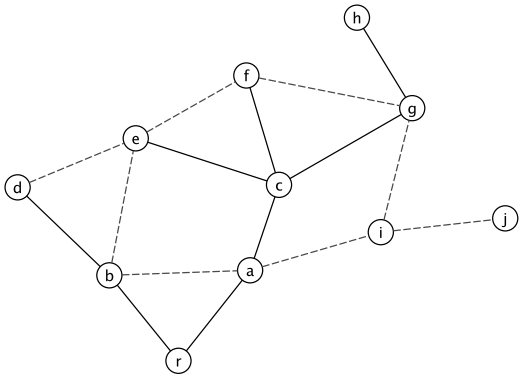


Figure 7.1: Flows in an undirected graph rooted at vertex r. The segments represent undirected edges. Those that are part of the solution tree are shown as solid, the others as dashed. Note that the solution does not necessarily have to span all the vertices. In this specific case, vertices i and j do not belong to the tree. There are 11 vertices and 16 edges in this graph, which gives a total of 16 edge indicator variables $y_{\{i,j\}}$ and $11 \cdot 16 = 176$ flow variables $f^m_{\{i,j\}}$. For the example tree, we have $f^g_{\{r,a\}} = f^g_{\{a,c\}} = f^g_{\{c,g\}} = 1$, which creates the unit flow from r to g. All the other $f^g_{\cdot\cdot}$ variables are equal to 0. All the $f^i_{\cdot\cdot}$ and $f^j_{\cdot\cdot}$ variables are also equal to 0.

REFERENCES

[1] J. Aguttes, P. Ammendola, R. Baugh, U. Benz, L. Bianchi, R. Bird, K. Blitzer, L. Borgarelli, F. Buscaglione, F. Caltagirone, et al. Space-based observation technology. 2000. 4

[2] K. Al-Kofahi, S. Lasek, D. Szarowski, C. Pace, G. Nagy, J. Turner, and B. Roysam. Rapid Automated Three-Dimensional Tracing of Neurons from Confocal Image Stacks. *TITB*, 6(2):171–187, 2002. 12

[3] M. Andriluka, S. Roth, and B. Schiele. Monocular 3D Pose Estimation and Tracking by Detection. In *CVPR*, 2010. 9

[4] E. Bas and D. Erdogmus. Principal Curves as Skeletons of Tubular Objects - Locally Characterizing the Structures of Axons. *Neur. Inf.*, 9(2-3):181–191, 2011. 12

[5] F. Benmansour and L. Cohen. Tubular Structure Segmentation Based on Minimal Path Method and Anisotropic Enhancement. *IJCV*, 92(2):192–210, 2011. 19

[6] Bitplane. Imaris: 3D and 4D Real-Time Interactive Image Visualization, 2013. http://www.bitplane.com/go/products/imaris/. 10

[7] K. M. Brown, D. E. Donohue, G. DAlessandro, and G. A. Ascoli. A cross-platform freeware tool for digital reconstruction of neuronal arborizations from image stacks. *Neuroinformatics*, 3(4):343–359, 2005. 10

[8] H. Cai, X. Xu, J. Lu, J. Lichtman, S. Yung, and S. Wong. Repulsive Force Based Snake Model to Segment and Track Neuronal Axons in 3D Microscopy Image Stacks. *NeuroImage*, 32(4):1608–1620, August 2006. 12

[9] A. Can, H. Shen, J. Turner, H. Tanenbaum, and B. Roysam. Rapid Automated Tracing and Feature Extraction from Retinal Fundus Images Using Direct Exploratory Algorithms. *TITB*, 3(2):125–138, June 1999. 12

[10] A. Cardona, S. Saalfeld, J. Schindelin, I. Arganda-Carreras, S. Preibisch, M. Longair, P. Tomancak, V. Hartenstein, and R. J. Douglas. TrakEM2 Software for Neural Circuit Reconstruction. *PLoS One*, 7(6):38011, 2012. 11

[11] L. Chaerle, K. Hulsen, C. Hermans, R. J. Strasser, R. Valcke, M. Höfte, and D. Van Der Straeten. Robotized time-lapse imaging to assess in-planta uptake of phenylurea herbicides and their microbial degradation. *Physiologia Plantarum*, 118(4):613–619, 2003. 1

[12] C.-C. Chen, S. Thakkar, C. Knoblock, and C. Shahabi. Automatically annotating and integrating spatial datasets. In *Advances in Spatial and Temporal Databases*, pages 469–488. Springer, 2003. 3

[13] A. Choromanska, S. Chang, and R. Yuste. Automatic Reconstruction of Neural Morphologies with Multi-Scale Graph-Based Tracking. *Frontiers in Neural Circuits*, 6(25), 2012. 12

[14] P. Chothani, V. Mehta, and A. Stepanyants. Automated Tracing of Neurites from Light Microscopy Stacks of Images. *Neur. Inf.*, 9:263–278, 2011. 11, 12

[15] Y. Chu and T. Liu. On Shortest Arborescence of a Directed Graph. *Scientia Sinica*, 14(10):1396, 1965. 27

[16] H. Cuntz, F. Forstner, A. Borst, and M. Häusser. One Rule to Grow Them All: A General Theory of Neuronal Branching and Its Practical Application. *PLoS Computational Biology*, 6(8):1000877, 2010. 11, 13

[17] N. Dalal and B. Triggs. Histograms of Oriented Gradients for Human Detection. In *CVPR*, 2005. 21

[18] L. Domanski, C. Sun, R. Hassan, P. Vallotton, and D. Wang. Linear Feature Detection on GPUs. In *DICTA*, 2010. 12

[19] C. Duhamel, L. Gouveia, P. Moura, and M. Souza. Models and Heuristics for a Minimum Arborescence Problem. *Networks*, 51(1):34–47, 2008. 32, 33, 84

[20] J. Edmonds. *Optimum branchings*. National Bureau of standards, 1968. 27

[21] D. Fan. *Bayesian Inference of Vascular Structure from Retinal Images*. PhD thesis, Dept. of Computer Science, U. of Warwick, Coventry, UK, 2006. 12, 13

[22] Z. Fanti, F. F. De-Miguel, and M. E. Martinez-Perez. A Method for Semiautomatic Tracing and Morphological Measuring of Neurite Outgrowth from DIC Sequences. In *Engineering in Medicine and Biology Society*, pages 1196–1199, 2008. 10

[23] D. Feldmeyer. Excitatory neuronal connectivity in the barrel cortex. *Frontiers in neuroanatomy*, 6, 2012. 2

[24] P. Felzenszwalb, R. Girshick, D. McAllester, and D. Ramanan. Object Detection with Discriminatively Trained Part Based Models. *PAMI*, 2010. 21

[25] J. C. Fiala. Reconstruct: A Free Editor for Serial Section Microscopy. *Journal of microscopy*, 218:52–61, 2005. 10, 11

[26] M. Fischler, J. Tenenbaum, and H. Wolf. Detection of Roads and Linear Structures in Low-Resolution Aerial Imagery Using a Multisource Knowledge Integration Technique. *CVGIP*, 15(3):201–223, March 1981. 13

[27] M. A. Fortier, D. Ziou, C. Armenakis, and S. Wang. Automated correction and updating of roads from aerial ortho-images. In *Proc. ISPRS*, volume 33, pages 34–41, 2000. 1, 3

[28] E. Frew, T. McGee, Z. Kim, X. Xiao, S. Jackson, M. Morimoto, S. Rathinam, J. Padial, and R. Sengupta. Vision-based road-following using a small autonomous aircraft. In *Aerospace Conference, 2004. Proceedings. 2004 IEEE*, volume 5, pages 3006–3015. IEEE, 2004. 3

[29] T. Gillette, K. Brown, and G. Ascoli. The Diadem Metric: Comparing Multiple Reconstructions of the Same Neuron. *Neur. Inf.*, 9:233–245, May 2011. 34, 54, 57, 69, 71

[30] P. Glowacki, M. Pinheiro, E. Turetken, R. Sznitman, D. Lebrecht, A. Holtmaat, J. Kybic, and P. Fua. Reconstructing Evolving Tree Structures in Time Lapse Sequences. In *CVPR*, 2014. 66

[31] Gurobi. Gurobi Optimizer, 2012. http://www.gurobi.com/. 53

[32] A. Holtmaat, J. Randall, and M. Canea. Optical Imaging of Structural and Functional Synaptic Plasticity in Vivo. *European Journal of Pharmacology*, 2013. 1

[33] J. Hopcroft and R. Tarjan. Algorithm 447: Efficient algorithms for graph manipulation. *Communications of the ACM*, 16(6):372–378, 1973. 78

[34] M. Law and A. Chung. Three Dimensional Curvilinear Structure Detection Using Optimally Oriented Flux. In *ECCV*, 2008. 12

[35] M. Law and A. Chung. An Oriented Flux Symmetry Based Active Contour Model for Three Dimensional Vessel Segmentation. In *ECCV*, 2010. 12, 14, 19

[36] T. Lee, R. Kashyap, and C. Chu. Building Skeleton Models via 3D Medial Surface Axis Thinning Algorithms. *CVGIP*, 56(6):462–478, 1994. 12

[37] V. Lepetit and P. Fua. *Monocular Model-Based 3D Tracking of Rigid Objects: A Survey*. Now Publishers, September 2005. 9

[38] H. Li and A. Yezzi. Vessels as 4-D Curves: Global Minimal 4D Paths to Extract 3-D Tubular Surfaces and Centerlines. *TMI*, 26(9):1213–1223, 2007. 14, 19

[39] Q. Li, Z. Deng, Y. Zhang, X. Zhou, U. V. Nagerl, and S. T. C. Wong. A Global Spatial Similarity Optimization Scheme to Track Large Numbers of Dendritic Spines in Time-Lapse Confocal Microscopy. *TMI*, 30(3):632–641, 2011. 1, 9

[40] Z. Li, S. Liu, R. Weinreb, J. Lindsey, M. Yu, L. Liu, C. Ye, Q. Cui, W. Yung, C. Pang, D. Lam, and C. Leung. Tracking Dendritic Shrinkage of Retinal Ganglion Cells After Acute Elevation of Intraocular Pressure. *Invest Ophthalmol Vision Science*, 52(10):7205–12, 2011. 1

[41] J. Livet, T. Weissman, H. Kang, R. Draft, J. Lu, R. Bennis, J. Sanes, and J. Lichtman. Transgenic Strategies for Combinatorial Expression of Fluorescent Proteins in the Nervous System. *Nature*, 450(7166):56–62, 2007. 33

[42] J. Livet, T. A. Weissman, H. Kang, R. W. Draft, J. Lu, R. A. Bennis, J. R. Sanes, and J. W. Lichtman. Transgenic strategies for combinatorial expression of fluorescent proteins in the nervous system. *Nature*, 450(7166):56–62, 2007. 4

[43] M. Longair, D. A. Baker, and J. D. Armstrong. Simple Neurite Tracer: Open Source Software for Reconstruction, Visualization and Analysis of Neuronal Processes. *Bioinf.*, 27(17):2453–2454, 2011. 10, 11

[44] MBF Bioscience. Neurolucida: Microscope Systems for Stereology and Neuron Morphology, 2013. `http://www.mbfbioscience.com/neurolucida/`. 10

[45] E. Meijering, M. Jacob, J.-C. F. Sarria, P. Steiner, H. Hirling, and M. Unser. Design and Validation of a Tool for Neurite Tracing and Analysis in Fluorescence Microscopy Images. *Cytometry Part A*, 58A(2):167–176, April 2004. 10

[46] A. Mukherjee and A. Stepanyants. Automated Reconstruction of Neural Trees Using Front Re-Initialization. In *SPIE*, 2012. 12

[47] D. R. Myatt, T. Hadlington, G. A. Ascoli, and S. J. Nasuto. Neuromantic - from Semi Manual to Semi Automatic Reconstruction of Neuron Morphology. *Frontiers in Neuroinformatics*, 6(4), 2012. 10

[48] K. Palagyi and A. Kuba. A 3D 6-Subiteration Thinning Algorithm for Extracting Medial Lines. *PR*, 19(7):613–627, 1998. 12

[49] H. Peng, F. Long, and G. Myers. Automatic 3D Neuron Tracing Using All-Path Pruning. *Bioinf.*, 27(13):239–247, 2011. 10, 11, 13

[50] H. Peng, Z. Ruan, D. Atasoy, and S. Sternson. Automatic Reconstruction of 3D Neuron Structures Using a Graph-Augmented Deformable Model. *Bioinf.*, 26(12):38–46, 2010. 10, 11

[51] M. Pool, J. Thiemann, A. Bar-Or, and A. E. Fournier. Neuritetracer: A Novel Imagej Plugin for Automated Quantification of Neurite Outgrowth. *Journal of Neuroscience Methods*, 168(1):134–139, 2008. 10, 12

[52] P. Prusinkiewicz. Modeling of Spatial Structure and Development of Plants: a Review. *Scientia Horticulturae*, 74(1–2):113–149, 1998. 1, 5

[53] D. Ramanan, A. Forsyth, and A. Zisserman. Strike a Pose: Tracking People by Finding Stylized Poses. In *CVPR*, 2005. 9

[54] C. E. Rasmussen and C. K. Williams. *Gaussian Process for Machine Learning*. MIT Press, 2006. 43, 49

[55] E. Serradell, P. Glowacki, J. Kybic, F. Moreno, and P. Fua. Robust Non-Rigid Registration of 2D and 3D Graphs. In *CVPR*, June 2012. 49

[56] J. A. Sethian. *Level Set Methods and Fast Marching Methods Evolving Interfaces in Computational Geometry, Fluid Mechanics, Computer Vision, and Materials Science*. Cambridge University Press, 1999. 19

[57] V. Springel. The cosmological simulation code gadget-2. *Monthly Notices of the Royal Astronomical Society*, 364(4):1105–1134, 2005. 4

[58] R. Srinivasan, Q. Li, X. Zhou, J. Lu, J. Lichtman, and S. T. C. Wong. Reconstruction of the Neuromuscular Junction Connectome. *Bioinformatics*, 26(12):64–70, 2010. 10

[59] A. Stepanyants, P. R. Hof, and D. B. Chklovskii. Geometry and structural plasticity of synaptic connectivity. *Neuron*, 34(2):275–288, 2002. 2

[60] K. Sun, N. Sang, and T. Zhang. Marked Point Process for Vascular Tree Extraction on Angiogram. In *CVPR*, pages 467–478, 2007. 12, 13

[61] E. Turetken, C. Becker, P. Glowacki, F. Benmansour, and P. Fua. Detecting Irregular Curvilinear Structures in Gray Scale and Color Imagery Using Multi-Directional Oriented Flux. In *ICCV*, December 2013. 19, 48

[62] E. Türetken, F. Benmansour, B. Andres, H. Pfister, and P. Fua. Reconstructing curvilinear networks using path classifiers and integer programming. Technical report, Institute of Electrical and Electronics Engineers, 2015. 14, 34, 36, 77

[63] E. Turetken, F. Benmansour, and P. Fua. Automated Reconstruction of Tree Structures Using Path Classifiers and Mixed Integer Programming. In *CVPR*, June 2012. 14, 15, 17, 21, 25, 26, 30, 32, 43

[64] E. Turetken, F. Benmansour, and P. Fua. Semi-Automated Reconstruction of Curvilinear Structures in Noisy 2D Images and 3D Image Stacks. Technical report, EPFL-182839, 2013. 10, 34

[65] E. Turetken, G. Gonzalez, C. Blum, and P. Fua. Automated Reconstruction of Dendritic and Axonal Trees by Global Optimization with Geometric Priors. *Neur. Inf.*, 9(2-3):279–302, 2011. 12, 13, 14, 34, 36

[66] Z. Vasilkoski and A. Stepanyants. Detection of the Optimal Neuron Traces in Confocal Microscopy Images. *Journal of Neuroscience Methods*, 178(1):197–204, 2009. 12

[67] D. Wang, R. Lagerstrom, C. Sun, L. Bishof, P. Valotton, and M. Götte. HcaVision: Automated Neurite Outgrowth Analysis. *Journal of Biomolecular Screening*, 15(9):1165–1170, 2010. 10

[68] Y. Wang, A. Narayanaswamy, and B. Roysam. Novel 4D Open-Curve Active Contour and Curve Completion Approach for Automated Tree Structure Extraction. In *CVPR*, pages 1105–1112, 2011. 10, 13, 14

[69] Y. Wang, A. Narayanaswamy, C. Tsai, and B. Roysam. A Broadly Applicable 3D Neuron Tracing Method Based on Open-Curve Snake. *Neur. Inf.*, 9(2-3):193–217, 2011. 10

[70] S. Wearne, A. Rodriguez, D. Ehlenberger, A. Rocher, S. Henderson, and P. Hof. New Techniques for Imaging, Digitization and Analysis of Three-Dimensional Neural Morphology on Multiple Scales. *Neuroscience*, 136(3):661–680, 2005. 11, 12

[71] C. Weaver, P. Hof, S. Wearne, and L. Brent. Automated Algorithms for Multiscale Morphometry of Neuronal Dendrites. *Neural Computation*, 16(7):1353–1383, 2004. 12

[72] J. Wegner, J. Montoya-Zegarra, and K. Schindler. A Higher-Order CRF Model for Road Network Extraction. In *CVPR*, 2013. 12

[73] D. Weinland, M. Ozuysal, and P. Fua. Making Action Recognition Robust to Occlusions and Viewpoint Changes. In *ECCV*, 2010. 21

[74] Q. Wen, A. Stepanyants, G. N. Elston, A. Y. Grosberg, and D. B. Chklovskii. Maximization of the connectivity repertoire as a statistical principle governing the shapes of dendritic arbors. *Proceedings of the National Academy of Sciences*, 106(30):12536–12541, 2009. 2

[75] H. Xiao and H. Peng. APP2: Automatic Tracing of 3D Neuron Morphology Based on Hierarchical Pruning of a Gray-Weighted Image Distance-Tree. *Bioinf.*, 29(11):1448–1454, 2013. 11

[76] J. Xu, J. Wu, D. Feng, and Z. Cui. Dsa Image Blood Vessel Skeleton Extraction Based on Anti-Concentration Diffusion and Level Set Method. *Computational Intelligence and Intelligent Systems*, 51:188–198, 2009. 12

[77] T. Yedidya and R. Hartley. Tracking of Blood Vessels in Retinal Images Using Kalman Filter. In *DICTA*, pages 52–58, 2008. 12

[78] X. Yu, J. Tian, and J. Liu. Transformation Model Estimation for Point Matching via Gaussian Processes. In *World Congress of Engineering*, 2007. 49

[79] T. Zhao, J. Xie, F. Amat, N. Clack, P. Ahammad, H. Peng, F. Long, and E. Myers. Automated Reconstruction of Neuronal Morphology Based on Local Geometrical and Global Structural Models. *Neur. Inf.*, 9:247–261, May 2011. 13, 14

Curriculum vitae

Przemysław Głowacki
+41 762 362 670
prz.glo@gmail.com

Strengths

- **Computer vision and mathematical optimisation skills**
- **Very good knowledge of algorithmics and advanced data structures**
- **Strong background in mathematics** (especially combinatorics and geometry)

Education

2011 to date	**PhD in Computer Science at École polytechnique fédérale de Lausanne**, Switzerland in the **Computer Vision Laboratory**
2006 – 2011	**MSc in Computer Science at AGH University of Science and Technology**, Kraków, Poland
2009 – 2011	**Mathematics and the Natural Sciences Studies - specialisation: Physics at Jagiellonian University**, Kraków, Poland
2003 – 2006	**Polish Certificate of Secondary Education, 5th Secondary School**, Kraków, **algorithmics profile**

Technical skills

Programming languages: **C/C++, Java, Python, Matlab**
Version control systems: **Git, SVN**
Operating systems: **Linux, Mac OS X, Windows**

Projects

July - August 2010	**Internship at National Physical Laboratory,** London, United Kingdom. Optimised software for controlling physics experiments. The result is software that is simpler and easier to maintain, while delivering the desired functionality.
July - September 2009	**Internship at Rutherford Appleton Laboratory,** Oxfordshire, United Kingdom. Designed an image analysis tool in Java. The tool allows for taking quick measurements in images, accelerating data analysis in laser experiments.

Languages

Polish	native language
English	fluent, Cambridge Certificate of Proficiency in English, 2006
French	conversational level
German	conversational level

Publications

E. Serradell, P, Głowacki, J, Kybic, F. Moreno-Noguer, P. Fua, **Robust Non-Rigid Registration of 2D and 3D Graphs**, Conference on Computer Vision and Pattern Recognition, Providence, RI, USA, June 2012

E. Türetken, C. Becker, P. Głowacki, F. Benmansour, P. Fua, **Detecting Irregular Curvilinear Structures in Gray Scale and Color Imagery using Multi-Directional Oriented Flux**, International Conference on Computer Vision (ICCV), Sydney, Australia, December, 2013

P. Głowacki, M. Pinheiro, E. Türetken, R, Sznitman, D. Lebrecht, J. Kybic, A. Hotlmaat, P. Fua, **Reconstructing Evolving Tree Structures in Time Lapse Sequences**, Conference on Computer Vision and Pattern Recognition (CVPR), Columbus, OH, USA, 2014

E. Türetken, F. Benmansour, B. Andres, P. Głowacki, H. Pfister, P. Fua, **Reconstructing Curvilinear Networks using Path Classifiers and Integer Programming**, IEEE Transactions on Pattern Analysis and Machine Intelligence, 2015

P. Głowacki, M. Pinheiro, E. Türetken, D. Lebrecht, R. Sznitman, A. Holtmaat, J. Kybic, P. Fua, **Reconstructing Evolving Tree Structures in Time Lapse Sequences by Enforcing Time-Consistency**, Submitted to: IEEE Transactions on Pattern Analysis and Machine Intelligence, 2015