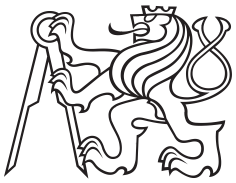
Doctoral Thesis



Czech Technical University in Prague

F3

Faculty of Electrical Engineering
Department of Cybernetics

Graph and Point Cloud Matching for Image Registration

Miguel Amável Pinheiro

Supervisor: prof. Dr. Ing. Jan Kybic

Ph.D. Programme: Electrical Engineering and Information Technology

Field of study: Artificial Intelligence and Biocybernetics

Prague, January 2017

Acknowledgements

I would like to express my gratitude to my colleagues at the Center for Machine Perception, who helped me throughout my time as a doctoral student. A special thanks to prof. Jan Kybic for the great supervision of my work, together with his patience and enthusiasm, and for providing me with great opportunities in my research and career.

I would also like leave my thanks to the research group at CvLab of EPFL, who received me for half a year altogether and kept collaborating throughout my studies. In particular prof. Pascal Fua and prof. Raphael Sznitman for their guidance.

I would like to thank my family and friends for their support that made it possible for me to finish this thesis. A special thanks to my friends in Prague (_\|m), my friends back from Porto and my parents for providing me with motivation and inspiration. Finally I would like to leave my at most appreciation for my significant other.

I gratefully acknowledge the support by the Portuguese Fundação para a Ciência e Tecnologia.

Abstract

This thesis focuses on the topic of image or volume registration of data containing tree and graph shaped structures, with a special focus on medical imaging. The geometrical information is first extracted from the volumes or images and then used for registration. We propose a method for the segmentation of trees in images acquired at different time instances, by enforcing time consistency. This results in an overall improvement of the extraction accuracy. The method was tested on medical, biological and road images.

The focus of this thesis is finding the alignment between segmented graphs and trees. We first propose a method called Active Testing Search (ATS) that explores partial correspondences of branching points of the structures. The method estimates the probability of partial match correctness based on training data and incrementally grows these partial matches. The ATS approach was able to align real data from several different medical imaging modalities, and is robust to initial position, rotation, deformation, missing data and noise.

The second proposed method is called Graph Matching using Monte Carlo tree search (GMMC). The approach uses a stochastic state-space search algorithm inspired by the Monte Carlo tree search method to build a large set of compatible curves. Further acceleration is achieved by pruning using novel curve descriptors. The method can handle partial matches, topological differences, geometrical distortion, does not use appearance information and foes not require an initial alignment. Moreover, our method is very efficient – it can match graphs with thousands of nodes, which is an order of magnitude better than the best competing method.

Supervisor: prof. Dr. Ing. Jan Kybic

Abstrakt

Tato disertační práce se zabývá tématem registrace 2D a 3D obrazů obsahujících stromové a grafové geometrické struktury, zejména v oblasti lékařství a biologie.

Součástí práce je metoda pro extrakci stromových struktur z časové sekvence obrazů, kde využití časové koherence zlepšuje přesnost.

Hlavním tématem práce je nicméně hledání korespondencí mezi dvěma geometrickými grafy či stromy. První navrhovanou metodou je aktivní testování (Active Testing Search (ATS)), založené na postupném prohledávání stavového prostoru částečných korespondencí. Metoda funguji i v případech neznámé počáteční pozice, nelineární deformace, chybějících dat, topologických rozdílů, chyb měření atp.

Druhá navrhovaná metoda je založená na stochastickém prohledávání stavového prostoru (Graph Matching using Monte Carlo tree search (GMMC)), která je oproti ATS výrazně efektivnější. Dalšího zrychlení je dosaženo rychlým prořezáním nevhodných dvojic hran pomocí nově vyvinutých křivkových deskriptorů. Tímto způsobem jsme schopni registrovat grafy s tisíci uzlů, což o řád lepší, než u jiných existujících metod.

Contents

1 Introduction **1**

1.1 Graph matching approach to image registration 3

1.1.1 Problem definition 4

1.2 Contributions of this thesis 6

1.2.1 Tree segmentation with time-wise consistency constraints (Chapter 3) 6

1.2.2 Coarse geometrical graph matching 6

1.2.3 Publications related to this thesis 8

2 State of the art **11**

2.1 Segmentation of graph and tree structures 11

2.1.1 Single frame methods 11

2.1.2 Multiple frames 14

2.2 Matching geometrical features 14

2.2.1 Point cloud matching 14

2.2.2 Geometrical graph matching 15

2.2.3 Curve matching 17

3 Tree Segmentation with Time Consistency **19**

3.1 Approach 20

3.1.1 Reconstruction without Time Consistency 21

3.1.2 Reconstruction in all Images Simultaneously 22

3.2 Building Spatio-Temporal Graphs 24

3.3 Enforcing Temporal Consistency 26

3.3.1 Solving without Time Consistency 26

3.3.2 Flow Variables and Spatial Connectivity 28

3.3.3 Flow Variables and Temporal Consistency 29

3.3.4 Fine alignment 34

3.3.5 Speeding Up the Computation 34

3.4 Results 34

3.4.1 Image Datasets 34

3.4.2 Overall Performance 37

3.4.3 Change Detection 39

4 Active Testing Search **43**

4.1 Approach 44

4.2 Gaussian Process Regression 45

4.3 Coarse Alignment 47

4.3.1 Greedy Search 47

4.3.2 Active Test Search 49

4.4 Fine Alignment 53

4.5 Experiments 54

4.5.1 Synthetic Experiments 58

4.5.2 Real Data Experiments 64

5 Graph Matching using Monte Carlo Tree Search **67**

5.1 Problem definition 68

5.1.1 Proposed approach 70

5.2 Transformation model 71

5.3 Monte Carlo tree search 73

5.3.1 Implementation details 74

5.3.2 Adding a child node 76

5.4 Path descriptors 77

5.5 Experiments 78

5.5.1 Tested methods 78

5.5.2 Synthetic datasets 81

5.5.3 Real datasets 85

6 Conclusion **97**

A Active Testing Search derivation **99**

Notation and Assumptions 99

Objective 100

Observation Selection 102

B Graph Matching using Monte Carlo tree search implementation **105**

B.1 Input 105

B.2 Graph representation 105

B.3 Algorithm 107

B.3.1 Tree contents and parameters 107

C Graph Matching using Monte Carlo tree search code **115**

C.1 Compilation 115

C.2 Usage 116

C.3 Implementation 118

Bibliography **119**

Chapter 1

Introduction

The focus of our work is the registration of images with tree or graph-like structures. This type of structures is pervasive in biomedical 2D and 3D images. Examples are blood vessels, pulmonary bronchi, or nerve fibers in medical imaging, and also road networks, plant roots or real trees in other fields of imaging. They can be acquired at different times and scales, or using different modalities, which may result in vastly diverse image appearances. For example, neuronal structures acquired using a light microscope (LM), such as those in the upper row of Figure 1.1, look radically different when imaged using an electron microscope (EM) that, as shown in the bottom row of Figure 1.1. Registering these images is desirable in order to identify the same region in both images and to combine the specific information each modality provides, in this case large-scale connectivity from the low-resolution data and fine details such as dendritic spines from the high-resolution data. Another example are brain blood vessels obtained using two-photon microscopy and bright-field optical microscopy as shown in Figure 5.17. Similarly as the previous application, the modalities provide different information on the observed vessels. An example of using graph shaped structures to register information outside of the medical imaging field is aerial image localization by matching detected roads to map networks, as showed in Figure 5.1.

We wish to propose a general method which can register very large images containing geometrical graph structures obtained using the same or different types of acquisition techniques. The size of the graphs in the images can be in the range of tens of thousands of vertices, and therefore the method should be able to efficiently match very large structures. The kind of drastic appearance changes as listed above make it impractical to use registration techniques that rely on maximizing image similarity [79, 110] – in particular, when the images

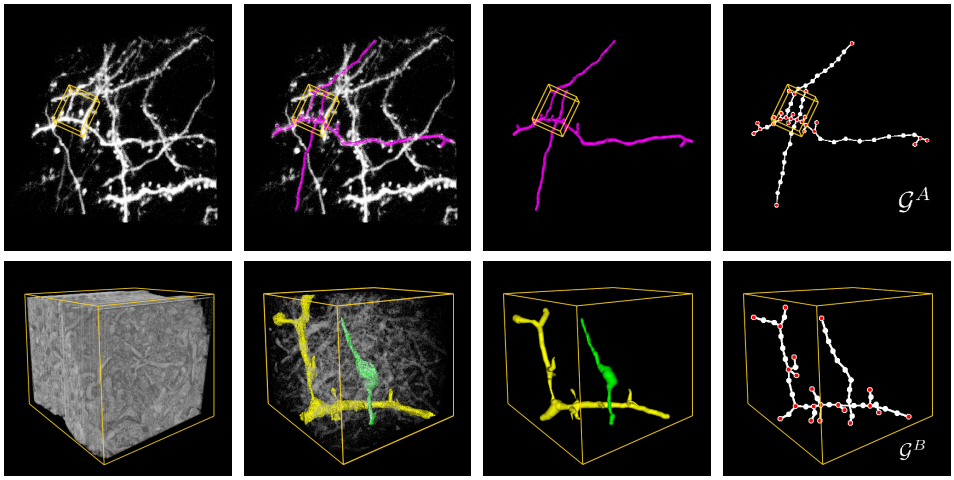


Figure 1.1: Brain tissue at different resolutions. **(a)** Image stack acquired using a two-photon light microscope from live brain tissue at a 1 micrometer resolution and a smaller area of the same tissue imaged using an electron microscope, at a 20 nanometer resolution. The orange box in the top image denotes the area from which the EM sample has been extracted. **(b)** Semi-automated delineation of some dendrites overlaid in magenta and manual segmentation of an axon overlaid in green and a dendrite in yellow. **(c)** The segmented structures on a black background. Since the resolution is much higher in the EM data, dendritic spines and synapses are clearly visible. **(d)** Graph representation of the neuronal structures. The red dots represent vertices and the white dots are sampled points along each edge.

are very different and when dealing with thin structures, such as blood vessels or neuronal fibers. The lack of distinguishing features of individual branching points or edges makes the use of feature-based correspondence techniques equally impractical [110]. We therefore propose to first detect the graph-like structures and then register the resulting graphs. Since the geometrical and topological structure may be the only property shared across modalities, graph or point cloud matching becomes the only effective registration means. This also includes subgraph matching when the images have been acquired at different resolutions with a different field of view.

Most existing techniques that attempt to match geometrical graphs rely on matching Euclidean or geodesic distances between graph junction points [29, 45, 92]. The methods are usable when the transformation is locally close to rigid, such as for pulmonary vessels, which undergo a smooth deformation, or retinal fundus images that show only slight non-linearities produced when the curved surface of the retina is viewed from different viewpoints. Also when dealing with images acquired using distinct modalities and at different resolutions, the acquired structures exhibit significant topology changes, for example, due to the failure of one of the methods to display parts of the

structure. Similarly, large non-linear deformations may occur because we work with a living specimen and the acquisitions are separated in time or because the deformation is introduced by the sample preparation or handling process. We know of no other graph matching method apart from our own method that can simultaneously handle all issues related to this kind of data, i.e. with the presence of

- Non-linear deformation – the difference in time and possible acquisition technique generally causes the transformation to be non-rigid.
- Unknown initial position – especially when the graphs are extracted using different approaches, the structures may not be roughly aligned in space and therefore the method should be rotation independent.
- Lack of distinguishing local features.

Graph and point cloud matching has been used in many applications for image registration. In retinal fundus photography, it is possible to match the branching points [17] but also the complete vessel graphs [108, 29]. Tree matching has been used for matching 3D lung structures [78, 49] and also blood vascular systems [18]. Graph matching has been applied in the registration of volumes of neuronal networks and brain blood vessels [87]. Nonetheless, there is not a universal approach which would be able to efficiently and accurately register all types of images independently of the application.

We propose a method which registers these images by accurately matching the segmented graph-like structures. Our approach is scalable for graphs with a large number of elements [51] – in the case of large volumes of brain networks we are able to match graphs with up to 10,000 vertices.

1.1 Graph matching approach to image registration

We propose to register images or volumes by first extracting geometrical from both images the graph structures we assume to be present in the images (the **Segmentation** step in Section 1.1.1), which we represent by graphs. We then find matches between the graphs to obtain a coarse alignment, i.e. a rough estimate of the alignment or transformation (the **Matching and coarse alignment** step), and then finely tune the alignment to obtain the

complete alignment of the whole graph, including individual edge points (the **Fine alignment** step). If necessary, the deformation is then extrapolated to all image points. An example of these steps is shown in Figure 1.2. We now define each task separately.

1.1.1 Problem definition

Segmentation

Given two D-dimensional images or volumes, the task is to segment the graph-like structures present in the images. The resulting segmented graphs are represented by $\mathcal{G}^A$ and $\mathcal{G}^B$ where each of the graphs is $\mathcal{G} = (\mathbf{V}, \mathbf{E})$ where the vertices $\mathbf{V}$ are points in $\mathbb{R}^D$, and the edges $\mathbf{E} \subseteq \mathbf{V} \times \mathbf{V}$ are associated with curves connecting the two incident vertices. This is a generalization of a *geometric graph* [35]. For more information, see Section 3.1.

Graph matching and coarse alignment

The graphs $\mathcal{G}^A$ and $\mathcal{G}^B$ obtained from segmentation are assumed to be related by a geometrical transformation $\mathcal{T}\colon \mathbb{R}^D \to \mathbb{R}^D$. Additionally, the removal or addition of subgraphs can occur. The task of graph matching is to find $\mathcal{T}$ and a matching $\mathbf{M}^V \subseteq \mathbf{V}^A \times \mathbf{V}^B$, $\mathbf{M}^E \subseteq \mathbf{E}^A \times \mathbf{E}^B$ between vertices and edges of the graphs. In fact, it is possible to estimate $\mathcal{T}$ from the matching and therefore throughout this thesis we focus on finding $\mathbf{M}^\mathbf{V}$ and $\mathbf{M}^\mathbf{E}$, while an estimate of $\mathcal{T}$ will be available as a by-product.

Fine alignment

Given the graphs $\mathcal{G}^A$ and $\mathcal{G}^B$ and their matching $\mathbf{M}^\mathbf{V}$, $\mathbf{M}^\mathbf{E}$, we can infer $\mathcal{T}$ and consequently obtain an approximate alignment $\mathcal{T}(\mathcal{G}^A)$ of $\mathcal{G}^A$ onto $\mathcal{G}^B$, which aligns all matched vertices. The task of fine alignment is to locally align $\mathcal{T}(\mathcal{G}^A)$ to $\mathcal{G}^B$, so that also the edge points are individually aligned.

See Figure 1.2 for a depiction of the steps.

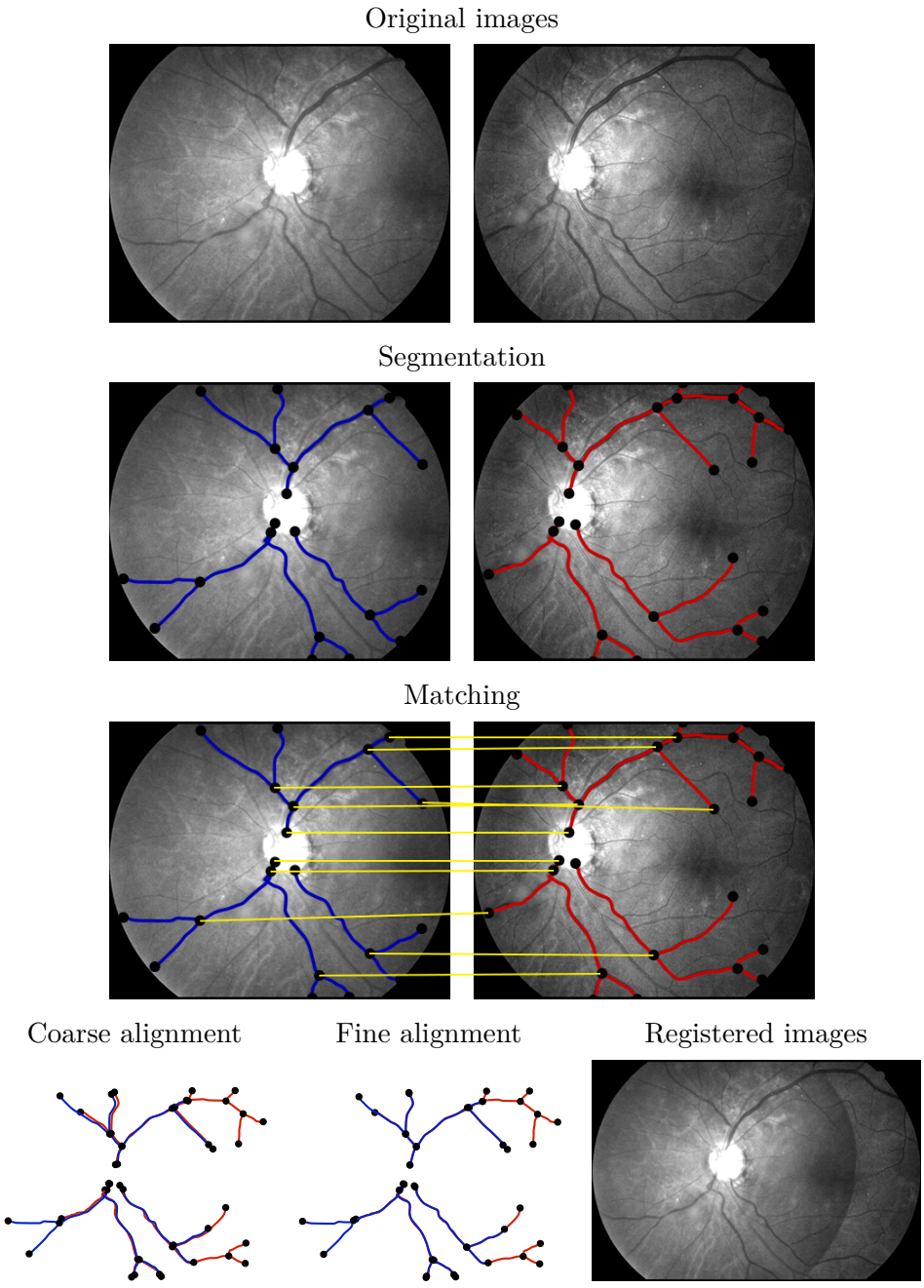


Figure 1.2: Simplified example of the complete steps for image registration using graph matching. The retinal photography images are acquired with different orientations of the eye [29].

1.2 Contributions of this thesis

This thesis contributes to the first two steps described above. Some of the work was done in collaboration with others, see Section 1.2.3 for details.

1.2.1 Tree segmentation with time-wise consistency constraints (Chapter 3)

We propose a new method for the segmentation of tree structures evolving in time [44, 43]. We describe the approach in detail in Chapter 3, which is based on the text of our submitted article [43].

For this particular work, we assume that we are given N images for the segmentation step, and we assume our structures to be trees. Besides the segmentation, the task is to find differences between the trees, which are assumed to be minor.

The approach builds upon an earlier work, based on [97]. The method proposes to enforce time consistency in order to improve the overall segmentation accuracy. This is achieved by considering the various trees at different time instances together, where connections are made between corresponding points consistently detected in the volumes. These connections are then considered in the optimization process to enforce these time-consistent detections.

We show experimentally that our approach successfully takes advantage of temporal information to produce more reliable and accurate reconstructions of tree structures. In addition, we show that the approach has the added benefit of automatically detecting regions where significant topology change occur in tree structures.

1.2.2 Coarse geometrical graph matching

We have developed two methods for the graph matching step – ATS and GMMC.

Active Testing Search (Chapter 4)

Active Testing Search (ATS) is described in detail in Chapter 4. The text of Chapter 4 is strongly based on our article [89], where the biggest difference is the unification of notation with the remainder of this thesis. As a baseline, we use a method based on greedy search, described in [87]. The Active Testing Search coarse alignment algorithm was also published as a conference paper [77].

ATS [77, 89] prioritizes between partial matches during the search using several match quality descriptors. It uses training data to learn distributions of scores from true and false partial matchings. At runtime, it ranks different matchings by calculating the likelihood of the matching being a true solution. The method automatically gives priority to more likely matchings, expanding them with more correspondences to find a larger set of correspondences. The matchings with a score indicating a probably incorrect solution are kept, but with a low priority, allowing the possibility of backtracking.

Similarly an earlier method [87], the approach for predicting a transformation $\mathcal{T}$ given a set of partial correspondences is based on Gaussian Processes, and an algorithm for fine alignment is also presented. In [77] we proposed a new search algorithm for coarse matching which speeds up significantly the performance and in [89] we present further improvements on the approach and also significantly extend the experiments done with the method, including various synthetic and medical datasets.

We show that our algorithm can match graphs with neither appearance information nor initial pose estimate being available, while allowing for partial matches and non-linear deformations.

Graph Matching using Monte Carlo Tree Search (Chapter 5)

The second graph matching algorithm we have developed is called Graph Matching using Monte Carlo tree search (GMMC) which is described in Chapter 5. The text is strongly based in the text of our submission [76] – only some notation was unified and more visual results are included. There are two separate contributions – a new curve descriptor and a tree search algorithm based on the Monte Carlo tree search.

The curve descriptors are used to evaluate compatibility between edges and sets of consecutive edges of two different graphs in order to quickly prune the search. Our path descriptor is based on an implicit Lipschitz transformation model. We show that our descriptor performs better than competitive curve descriptors.

We propose a tree search algorithm inspired by the Monte Carlo tree search to efficiently explore the search space, using the set of compatible paths [75]. The search algorithm is restricted on growing the matching using only adjacent edges, taking advantage of the exploration versus exploitation balance of the Monte Carlo method to quickly obtain a solution, even when matching large graphs with thousands of vertices. A preliminary version of the algorithm was described in a short paper [75].

GMMC is able to match all our tests, while the competing methods fail. It is also faster than competing methods. Particularly in the road datasets, where the search space is considerably larger, ATS (as well as other competing methods) is not able to obtain a solution, while GMMC quickly obtains a large match.

1.2.3 Publications related to this thesis

Tree segmentation with time-wise consistency constraints (Chapter 3).

- Przemysław Głowacki, Miguel Amável Pinheiro, Engin Türetken, Raphael Sznitman, Daniel Lebrecht, Jan Kybic, Anthony Holtmaat, and Pascal Fua. Reconstructing evolving tree structures in time lapse sequences. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3035–3042, 2014. [44]
  Authorship 35-35-6-6-6-6-3-3
- Przemysław Głowacki, Miguel Amável Pinheiro, Agata Mosinska, Engin Türetken, Daniel Lebrecht, Raphael Sznitman, Anthony Holtmaat, Jan Kybic, and Pascal Fua. Reconstructing evolving tree structures in time lapse sequences by enforcing time-consistency. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 2015 (submitted). [43]
  Authorship 35-20-15-6-6-6-6-3-3
  Impact factor 6.077

The author of this thesis developed the contents described in Section 3.2 where a Gaussian Processes Regression is used to predict which vertices of

the graph have correspondences in other images. The author also developed the fine alignment algorithm to detect differences in the segmented trees, described in 3.3.4.

Active Testing Search (Chapter 4).

- Miguel Amável Pinheiro, Raphael Sznitman, Eduard Serradell, Jan Kybic, Francesc Moreno-Noguer, and Pascal Fua. Active testing search for point cloud matching. *Information Processing in Medical Imaging*, pages 572–583, 2013. [77]
  Authorship 50-15-15-15-3-2
- Eduard Serradell, Miguel Amável Pinheiro, Raphael Sznitman, Jan Kybic, Francesc Moreno-Noguer, and Pascal Fua. Non-rigid graph registration using active testing search. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 37(3):625–638, 2015. [89]
  Authorship 40-40-8-8-2-2
  Impact factor 6.077

The author of this thesis has a direct contribution in all the work described in the conference paper introducing ATS [77]. The journal paper is the result of the merge of the earlier paper [87] with our paper [77], and also extended experiments, where accuracy results were produced using ATS and time complexity was analyzed to show that ATS outperforms the previous [87] and other competing methods. The author of this thesis contributed therefore equally as a first author to [89].

Graph Matching using Monte Carlo Tree Search (Chapter 5).

- Miguel Amável Pinheiro, and Jan Kybic. Path descriptors for geometric graph matching and registration. *International Conference on Image Analysis and Recognition*, pages 3–11, 2014. [74]
  Authorship 50-50
- Miguel Amável Pinheiro, and Jan Kybic. Geometrical Graph Matching using Monte Carlo Tree Search. *IEEE International Conference in Image Processing (ICIP)*, pages 3145–3149, 2015. [75]
  Authorship 50-50
- Miguel Amável Pinheiro, Jan Kybic, and Pascal Fua. Geometric graph matching using Monte Carlo tree search. *IEEE Trans. on Pattern*

Analysis and Machine Intelligence, 2016 (early access). [76]
Authorship 45-45-10
Impact factor 6.077

The author of this thesis has a direct contribution in all the sections described in these papers.

Chapter 2

State of the art

Let us review the existing work related to the topic of this thesis, i.e. graph structure segmentation, registration, graph matching and curve descriptors.

2.1 Segmentation of graph and tree structures

The task is to segment geometrical tree structures present in given images. This is the first step to register images using geometrical tree or graph-shaped structures.

2.1.1 Single frame methods

For most automatic reconstruction techniques of tree-like structures, the process begins by estimating a local measure of *tubularity*, i.e. the likelihood that a point sits along the centerline of a tubular structure [32, 54, 67]. Matched filters [2, 109], Hessian and Oriented Flux functionals [39, 56, 57, 83], and classification scores derived from steerable filter responses [48, 52] have all been used for this purpose.

These tubular measures are then used within a search or optimization framework to reconstruct the tree structures image by image. In this context,

the search techniques come in one of two forms, either local or global. Local methods reconstruct the tree structure piece by piece in a greedy fashion, making them extremely efficient [2, 5, 101] but less robust to image noise and prone to errors when there are large gaps separating filaments. Conversely, global methods optimize the entire tree in one shot making them more computationally demanding but also more robust. This typically involves connecting high tubularity points to form a weighted graph and then finding a tree within that graph by optimizing an objective function. This last step can be done using Minimum Spanning Trees (MST) [37, 101, 109], Shortest Path Trees (SPT) [73], k-Minimum Spanning Trees (k-MST) [98], or Quadratic Mixed Integer Programming (QMIP) [97].

Local search methods

This type of methods make greedy decisions about which segments of the tree to retain based on local image evidence. They include methods that segment and skeletonize the tubularity image [103, 58, 80], and active contour-based methods, which are initialized from it [15, 100, 24]. 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 artifacts.

Another important class of local approaches involve methods that start from a set of seed points and incrementally grow branches by evaluating a tubularity measure [16, 2, 5]. 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.

Global Search Methods

Global methods aim at greater robustness by optimizing a global objective function over a graph of high-tubularity seed points [98] or superpixels [104].

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 [56, 57, 31]. An example of a global method is a Markov Chain Monte Carlo algorithm class [36, 93]. 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, the algorithms presented in [36, 93] while producing smooth tree components in general, do not necessarily guarantee spatial connectivity of these components.

Some graph-based methods use user-specified roots. The methods connect neighboring seed points 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 graph structures. The final step is to build a solution by selecting an optimal subset of the candidate edges.

Many existing approaches weigh the edges of this graph and solve some variant of a minimum-weight tree problem. Algorithms that find a Minimum Spanning Tree (MST) [38, 109, 101] or a Shortest Path Tree (SPT) [73] 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 often in noisy data.

The structures can also be obtained locally using tracking by particle filtering [30]. The segmentation is then obtained by global supervised seed clustering.

The k-Minimum Spanning Tree (k-MST) formulation [98] addressed this issue by posing the problem as one of finding the minimum cost tree that spans only an *a priori* unknown subset of k seed points. It relies on a heuristic search algorithm and two objective functions, 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. By contrast, the Integer Programming formulation introduced in [97] involves minimizing a single global objective function that allows us to link legitimate seed points while rejecting spurious ones by finding the optimum solution to within a small user-specified tolerance.

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 [62, 109]. Similarly, active contour-based methods typically define their energy terms as such integrals over the paths [101, 57]. Since integrals amount to averaging, such measures are not particularly effective at ignoring paths that take *shortcuts* through the background [98]. 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 [97] returns much more discriminative probabilistic costs, which can be compared for paths of arbitrary length.

2.1.2 Multiple frames

Existing strategies generally reconstruct structures one image at a time. Using temporal information to enforce time-consistency has not yet to be exploited for the purpose of tree structure reconstruction, as we propose in our approach described in Chapter 3. Nonetheless, such a technique are commonly used in other applications such as in tracking multiple people [9], tracking dendritic spines in time-lapse microscopy [63], or segmentation and tracking in echocardiographic sequences [69].

2.2 Matching geometrical features

Once the geometrical features such as points, curves or graphs are detected in the images, the task is now to match them between the images to be registered.

2.2.1 Point cloud matching

Point clouds are sets of points in the Euclidean space. We define point cloud matching as the task of finding a set of correspondences $\mathbf{M}^{\mathbf{X}}$ and a transformation $\mathcal{T}$ between two sets of points $\mathbf{X}^A$ and $\mathbf{X}^B$, where $\mathbf{X}^A = \{\mathbf{x}_1^A, \cdots, \mathbf{x}_{|\mathbf{X}^A|}^A\}$ is a set of points in $\mathbb{R}^D$ and similarly for $\mathbf{X}^B$, and $\mathcal{T}(\mathbf{x}_i^A) \approx \mathbf{x}_j^B$, if $(\mathbf{x}_i^A, \mathbf{x}_j^B) \in \mathbf{M}^{\mathbf{X}}$.

If the number of degrees of freedom of the geometrical transformation $\mathcal{T}$ being sought is small, its parameters can be recovered by RANSAC [37], based on randomly sampling partial correspondences $\mathbf{M}_C^{\mathbf{X}} = \{(\mathbf{x}_l^A, \mathbf{x}_l^B)\}_{1 \leq l \leq C}$, where C is a small number (depending on the transformation model used) and fitting the geometrical model to them. Many improvements to RANSAC have been proposed [23], such as Guided-MLESAC [95] or PROSAC [26]. When appearance information is not available to reduce the number of possible matches, more sophisticated search strategies have to be used, such as accelerated hypothesis sampling with information derived from the residual sorting [20]. The advantage of RANSAC-like approaches is that it does not require initialization but its computational complexity grows sharply with more general deformation models.

Another class of point matching algorithms is represented by the Iterative Closest Point (ICP) [10], which alternatively identifies closest points and updates the deformation parameters until convergence. There are variants of ICP that increase robustness [72, 19] or employ non-linear transformations [3, 61] such as thin-plate splines [25]. The softassign algorithm [46] constraints the types of transformations and uses a soft assigning on correspondences to obtain a matching. Shape context [7] analyzes the local distribution of points to obtain a matching between points. The Coherent Point Drift (CPD) method [71] uses probabilistic assignment based on Gaussian mixture models between point pairs. It introduces a fast algorithm based on the expectation maximization algorithm which results on an optimization close to linear complexity. As we see further on, in our experiments (Section 5.5) it is the method which is the fastest and with the best time complexity of the previously presented methods, which are tested against our approaches. These methods are fast and can handle large number of points but nonetheless require a good initial estimate of the transformation.

2.2.2 Geometrical graph matching

Geometrical graph extends a point cloud with edges connecting the points. By representing the segmentation with geometrical graphs, we can use information such as connectivity or curvature. Geometrical graphs are graphs $\mathcal{G} = (\mathbf{V}, \mathbf{E})$ where to each vertex $\mathbf{v}_i \in \mathbf{V}$ we associate a point in $\mathbb{R}^D$ and where each edge $\mathbf{e}_j \in \mathbf{E}$ is associated with a continuous curve $\zeta_{\mathbf{e}_j} : I \to \mathbb{R}^D$ connecting two vertices, where $I = [0, 1]$. Throughout this thesis, we will occasionally refer to geometrical graphs simply as graphs, even though we do not mean to refer to *classical graphs* which do not have any geometrical interpretation. The task of geometrical graph matching is defined in Section 1.1.

Graph matching can be formulated by considering compatibilities between edges or vertices, for example by comparing their Euclidean or geodesic distances [92, 29] or neighborhood similarity [47]. Matching can be then approached as finding an approximate minimum vertex cover of pairwise consistent matches [34], or maximum weighted independent set [88], where the cost reflects the dissimilarity between vertices or edges. The algorithms are fast as long as the transformation can be estimated using a low number of vertex or edge pairs, which is in practice only possible for rigid transformations or using very discriminative appearance descriptors. The vertex or edge compatibility criteria can be also used to prune the list of possible matches.

Binary compatibilities between vertex pairs can be relaxed to real-valued affinities, which leads to an integer quadratic program (IQP). The problem is formulated as follows

$$\mathbf{x}^* = \arg\max_{\mathbf{x}} \mathbf{x}^\intercal \mathbf{W} \mathbf{x}, \quad \mathbf{x} \in [0,1]^{|\mathbf{V}^A||\mathbf{V}^B|}, \qquad (2.1)$$
$$\forall j \sum_{i=1}^{|\mathbf{V}^B|} \mathbf{x}_{ij} \leq 1, \quad \forall i \sum_{j=1}^{|\mathbf{V}^A|} \mathbf{x}_{ij} \leq 1,$$

where $\mathbf{x}$ are the correspondences between the graphs and $\mathbf{W}$ is an affinity matrix where each element represents an affinity value between two pairs of points from the different graphs. The affinity value usually try to preserve Euclidean or geodesic distances [29], but can also include other descriptors such as SIFT descriptors [92] when local appearance is consistent. This type of formulation can be solved by spectral techniques [85, 59], spectral matching with affine constraints [27], iterative projections [60], random walks [22], dual decomposition [96] or path following algorithms [108]. These methods are mathematically elegant but even the most advanced ones cannot handle more than a few tens of vertices.

Topological differences are common in graphs created by segmentation of real data. Most algorithms can only handle missing nodes or edges; more general cases can be handled by graph edit distance minimization [68] at the expense of increased computational complexity.

Search methods which hypothesize different partial matching between graphs, similarly to RANSAC, were also proposed for graph matching. A greedy search can be used by considering adjacent vertices pairs to partially grow the matching [87]. This greedy expanding approach uses also consistency in Euclidean and geodesic distances between the elements of the matchings to prune the search. The matches are also checked using a Gaussian Processes Regression [82] which parametrizes the nonlinear transformation and evaluates the match.

2.2.3 Curve matching

By representing the segmented structures as curves in Euclidean space, we can then find similarity features such as curvature or length. We use these curve descriptors to establish whether two edges are similar or compatible in Euclidean space, which is very useful for pairing the search space, becoming therefore a subtask of our geometrical graph matching.

A simple but computationally demanding method consists of first aligning the curves using a chosen family of transformations and then calculating the residual Euclidean distance [72, 89] or the Frechet distance [14].

Descriptors of curve segments invariant to similarity transformations can be established [64], they can be based on angle [1] or curvature [105, 28]. In [41], the authors present an affine-invariant descriptor for a curve, based on curvature and high order derivatives. Other path descriptors use image information such as gradient to build a descriptor [102].

Curves can be assumed to match completely or partially. In the latter case, the match can be found by iterative closest point algorithm [72], dynamic programming [86] or the Hungarian method for the assignment problem [89].

Chapter 3

Tree Segmentation with Time Consistency

We propose a novel approach to reconstructing curvilinear tree structures imaged at different points in time such as 2D aerial images of road networks or growing plants, or 3D brain microscopy stacks. In order to utilize temporal consistency constraints we take a global approach of processing a sequence of images all at once, as opposed to the traditional way of reconstructing tree structures in each image independently.

The problem can be formulated as a Quadratic Mixed Integer Program and solved using off the shelf optimization engines. We demonstrate the additional robustness that comes from utilizing all the available visual clues at once as compared to single-frame oriented methods. The presented method is also more robust and handles time-consistency in a more global way than another multi time-frame method, to the best of our knowledge the only one published so far. In case of datasets where the imaged structure undergoes some local changes over time, the presented method has the added benefit of detecting those changes in an automated way.

This chapter is strongly based on the text of the paper [43]. Several images were enlarged and some notation was changed to unify with the rest of the thesis.

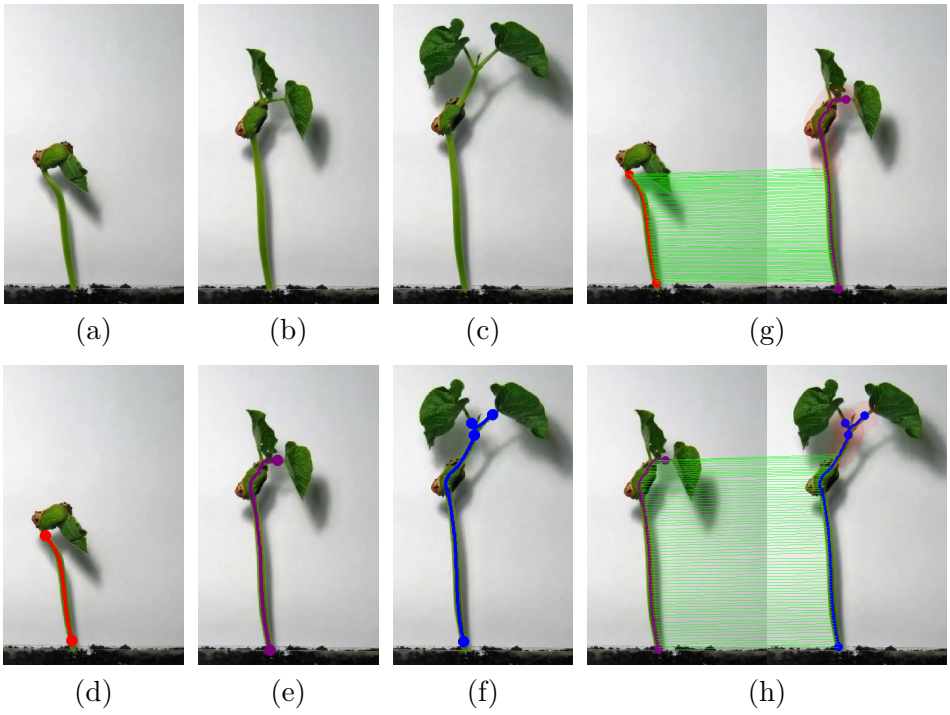


Figure 3.1: Reconstruction and automatic change detection using a time-lapse sequence for growing runner bean. (a, b, c) Original images. (d, e, f) Reconstructed trees in each one of them. (g, h) The horizontal green lines represent temporal edges between their vertices. This figure, as well as most of the subsequent ones, is best viewed in color.

3.1 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, the plant of Fig. 3.1, 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. 3.2 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 evolving tree structures, our goal is to reconstruct a number of 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 and the topology is preserved from one time frame to the other. As a starting point, 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 [97], the final set of trees can then be reconstructed by solving a QMIP problem.

In the remainder of this section, we first briefly describe the method of [97]. We then discuss how to extend it to take into account both local and global temporal-consistency constraints.

3.1.1 Reconstruction without Time Consistency

The method of [97] was designed for reconstructing tree-like structures in single images. For a given image I, a local scale-space tubularity measure is computed for every pixel, in the case of 2D data, or voxel, in the case of 3D data, using the oriented flux cross-section trace measure [56]. It expresses how likely it is that a given spacial position lies on a centerline of a tubular structure of a specific radius. A set of evenly distributed sample points $\mathcal{X} = \{\mathbf{x}_i\}$ is selected by first thresholding the tubularity image and then iteratively choosing the highest tubularity point and suppressing its neighborhood until no non-zero tubularity points are left. A number of tree roots are also manually annotated by a human operator. A *spatial graph*, $\mathcal{G} = (\mathcal{X}, \mathcal{E}_s)$, is then built taking the manually annotated roots and automatically selected sample points as vertices. Every two vertices that are close to each other are connected by two oppositely directed edges. For every pair of consecutive edges $\mathbf{e}_{ij}, \mathbf{e}_{jk} \in \mathcal{E}_s$ in the graph a probability score p_{ijk} is computed to assess how likely it is that the underlying tubular path is indeed part of the solution.

The final reconstruction is then obtained by choosing a subset of edges from the graph $\mathcal{G}$ that forms the most likely set of trees. Formally, the problem is formulated as a quadratic mixed-integer program (QMIP) with binary variables y_{ij} indicating whether the edge $\mathbf{e}_{ij}$ is part of the solution. A set of constraints adapted from [33] ensures that the result truly is a set of trees emanating from the manually annotated root vertices.

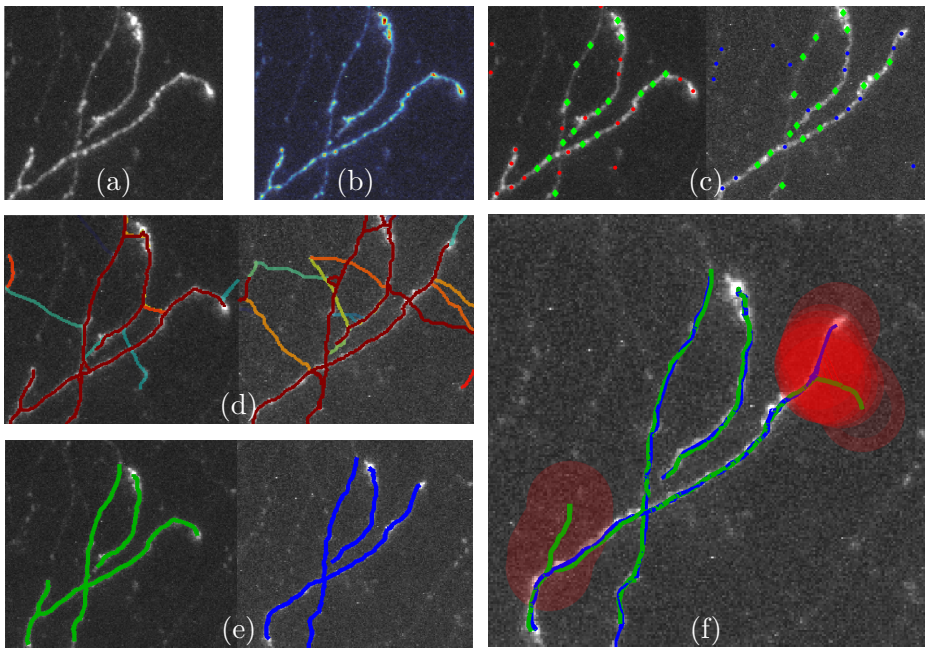


Figure 3.2: Key steps of the algorithm, best viewed in color. (a) Maximum intensity projection of one of three *in vivo* image-stacks of a neural network taken at one week intervals. (b) Corresponding tubularity image. (c) Maxima of tubularity selected as graph nodes in two different stacks. Those shown 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 trees. (f) The red tree from the first image can be deformed and superposed on the blue tree in the second one, making the changes highlighted in red easy to detect.

3.1.2 Reconstruction in all Images Simultaneously

When dealing with sequences of images depicting the same region of interest at different time points one can process them step by step using the approach described above. This, however, ignores constraints arising from temporal consistency. To account for them, we perform the initial sampling of tubularity images at all time instances simultaneously while trying to match corresponding sample points between consecutive images. We then create a *spatio-temporal* graph comprising all the sample points in all time points, such as the one depicted in Fig. 3.3. For convenience we connect all of them to an imaginary root vertex shown in gray at the bottom of the figure. This choice is explained in more detail in the next section. In addition to creating *spatial edges* connecting neighboring vertices in specific images we also create

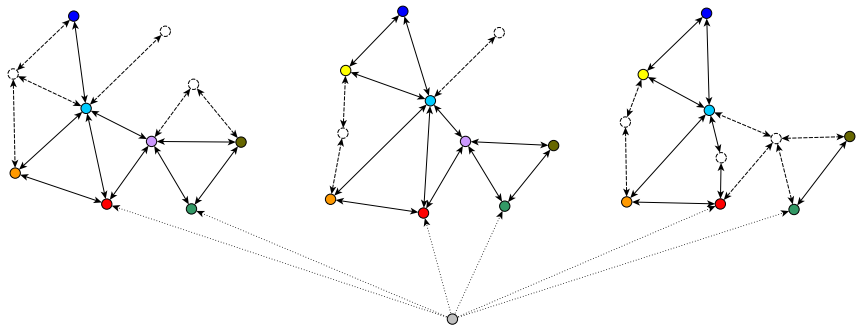


Figure 3.3: An example *spatio-temporal graph*. The imaginary root vertex $\mathbf{x}_r$ is presented in gray at the bottom. Each of the three time points contain two manually annotated physical roots marked in red and green. The vertices for which correspondences where not found in adjacent time points are represented by white circles with dashed borders. The other vertices are represented by colored circles. The temporal edges $\mathcal{E}_t$ are not explicitly presented to avoid clutter. Instead, vertices for which correspondences were found are marked with matching colors. 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 neighboring 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.

temporal edges between matching vertices in consecutive images. Instead of searching for the most probable set of trees in every image independently, we reconstruct them simultaneously and look for trees whose topology is consistent across time.

In our earlier approach [44], we enforced consistency by encouraging solutions in which corresponding edges in two consecutive time frames were either both present of both absent from the final solution. This accounts for the fact that it is unlikely for spatial edges to appear and disappear randomly throughout a time series. However, this remains a very local constraint. Here, we enforce a longer-range form of consistency by traversing broader vertex neighborhoods and trying to encouraging the persistence of their topology over time. As we will see, this can also be expressed as a QMIP and solved using off-the-shelf optimization software. In the end, a fine alignment may be performed to detect actual changes in the tree.

To achieve this, our approach goes through the following steps:

1. Find graph nodes in individual images as tubularity maxima and corresponding nodes in other images, as in Fig. 3.2(b-c).
2. Build a spatio-temporal graph such as the one depicted in Figs. 3.2(d)

and 3.3 by linking nodes both within images when they are close enough and across images when they match.

3. Solve an extended QMIP problem to find a set of trees whose local topology is temporally consistent, such as those of Fig. 3.2(e).

4. Align these trees spatially to identify places where substantial changes have occurred, as can be seen in Fig. 3.2(f).

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

3.2 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. 3.2(c). 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 Scale-Space Distance Transform method of [91] 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 neighborhood of locations around $\mathbf{x}_m^{\bar{n}}$. Within each evaluated neighborhood, 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. 3.2(c). Once computed, the tubularity is set to zero in both the neighborhood of $\mathbf{x}_m^n$ and that of the 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

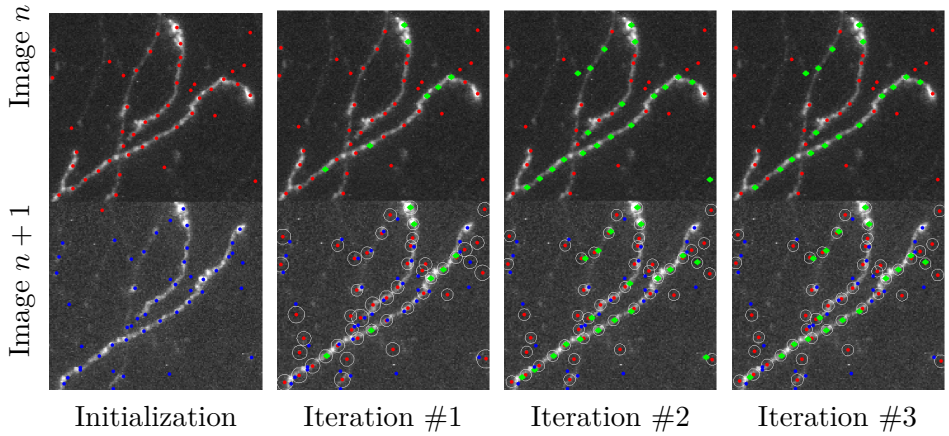


Figure 3.4: Iterating until a stable correspondence set has been found. (Initialization) A set of corresponding points with possible inconsistencies in the transformation model is found in each image using high-tubularity locations and NCC. (Iteration #1) A set of corresponding points (shown in green) 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.

the displacements of neighboring points are spatially consistent with each other. To enforce this and remove potential mismatches, we use a Gaussian Processes Regression (GPR) [82] to remove correspondences that are not consistent with a non-linear but locally smooth deformation model.

Hence, 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. 3.4 (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

$$\begin{aligned} m_{\mathcal{S}_n^0}(\mathbf{x}^n) &= \mathbf{k}'\mathbf{\Gamma}_{\mathcal{S}_n^0}^{-1}\mathbf{X}_{\mathcal{S}_n^0}^{n+1} \ , \\ \sigma_{\mathcal{S}_n^0}^2(\mathbf{x}^n) &= k(\mathbf{x}^n,\mathbf{x}^n) + \beta^{-1} - \mathbf{k}'\mathbf{\Gamma}_{\mathcal{S}_n^0}^{-1}\mathbf{k} \ , \end{aligned} \tag{3.1}$$

where k is a kernel function that implicitly defines a mapping composed of an affine and a non-linear transformation as in [87, 106], 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 $\mathcal{S}_n^0$. A correspondence is considered to be consistent if 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 correspondence $\mathcal{S}_n^1$, such as the one depicted by Fig. 3.4 (Iteration #2). We then repeat the process using $\mathcal{S}_n^1$ to compute the regression of Eq. 3.1 and iterate until the set stabilizes, typically after 4 to 5 iterations, as shown in Fig. 3.4 (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 Section 3.1.1, 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.

3.3 Enforcing Temporal Consistency

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 for all time instants. This way, reconstructing the trees in all images can be achieved by finding the most likely arborescence rooted in $\mathbf{x}_r$.

3.3.1 Solving without Time Consistency

Reconstructing the trees of interest means making a decision which edges of the graph $\mathcal{G}$ should be part of the final solution. To this end, we take

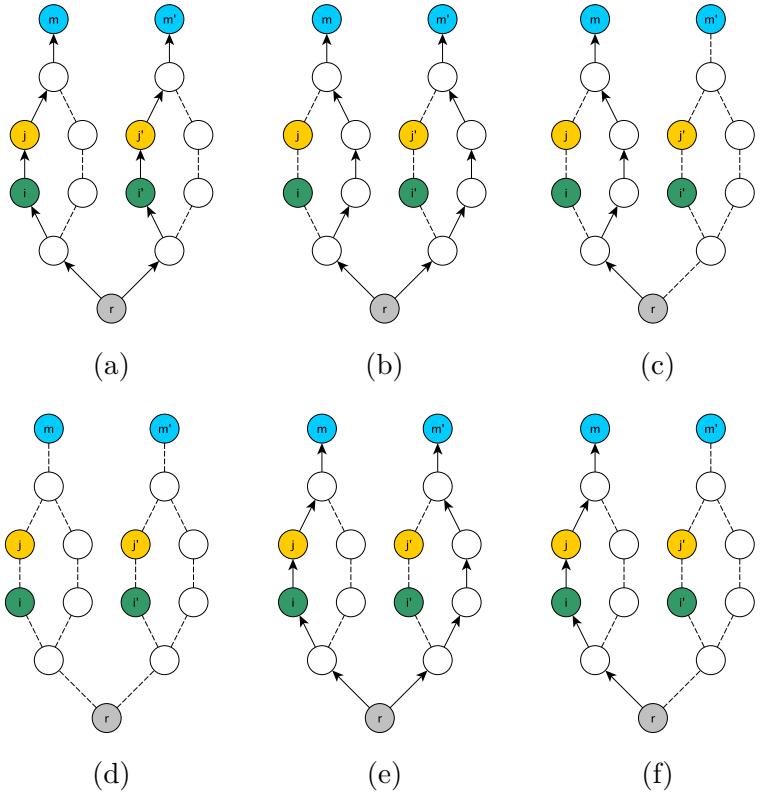


Figure 3.5: These six example graphs illustrate all possible temporal consistency situations for a pair of corresponding vertices $(\mathbf{x}_m, \mathbf{x}_{m'})$ and a pair of corresponding edges $(\mathbf{e}_{ij}, \mathbf{e}_{i'j'})$ in two consecutive time instants. Situations considered consistent are shown in figures (a) through (d). Inconsisntent cases are presented in (e) and (f). Note that in the asymmetric cases of (c), (e) and (f) their respective reverse cases were omitted for clarity. For instance, for the inconsistent case of figure (e), 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.

a Bayesian point of view as in [97]. Let $Y_{ij}^n \in \{0, 1\}$ be a binary random variable denoting the presence or absence of the edge $\mathbf{e}_{ij}^n$ in the final solution and Y be the set of all Y_{ij}^n variables. Our goal is to infer the most likely tree Y.

We could ignore the existence of the *temporal edges* and directly use the algorithm of [97]. In this case, computing the optimal tree would simply amount to solving the *maximum a posteriori* problem

$$y^* = \arg\max_{y \in \mathcal{Y}} P\left(\mathcal{I}, \mathcal{X}, \mathcal{E}_s | Y = y\right) \ , \tag{3.2}$$

$$= \arg\min_{y \in \mathcal{Y}} \sum_{\mathbf{e}_{ij}^n, \mathbf{e}_{jk}^n \in \mathcal{E}_s} w_{ijk} y_{ij}^n y_{jk}^n, \tag{3.3}$$

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 introduced in Section 3.1.1, and $\mathcal{Y}$ is the set of all feasible trees with root $\mathbf{x}_r$.

As time consistency is not enforced, this is exactly equivalent to independent reconstructions in all images, as described in Section 3.1.1.

However, what makes this relevant to our problem is that minimizing the criterion of Eq. 3.3 under the constraints that the result be a tree can be done by introducing auxiliary floating point *flow variables* and imposing *flow constraints* that enforce spatial connectivity [33, 97]. This turns the minimization problem into a Q-MIP, as explained below. In the following sections, we first describe these variables and then show how can they be used to also enforce temporal consistency.

3.3.2 Flow Variables and Spatial Connectivity

Given the spatio-temporal graph $\mathcal{G} = (\mathcal{X}, \mathcal{E})$ introduced above, solving the minimization problem of Eq. 3.3 amounts to finding a subset $\mathcal{E}'_s \subseteq \mathcal{E}_s$ of edges that form a tree rooted at node $\mathbf{x}_r$ that minimizes the objective function. This implies that there must be exactly one directed path from $\mathbf{x}_r$ to every vertex in that solution tree.

To ensure that this is the case, 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, otherwise it is set 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. 3.6 depicts such a tree.

As shown in [33], the tree connectivity constraints can therefore be enforced

by minimizing the criterion of Eq. 3.3 subject to

$$\begin{aligned}
&\sum_{\mathbf{x}_j \in \mathcal{X}\setminus\{\mathbf{x}_r\}} f_{rj}^m \leq 1, && \forall \mathbf{x}_m \in \mathcal{X}\setminus\{\mathbf{x}_r\},\\
&\sum_{\mathbf{x}_j \in \mathcal{X}\setminus\{\mathbf{x}_k\}} f_{jk}^m \leq 1, && \forall \mathbf{x}_m \in \mathcal{X}\setminus\{\mathbf{x}_r\},\\
&\sum_{\mathbf{x}_j \in \mathcal{X}\setminus\{\mathbf{x}_i,\mathbf{x}_r\}} f_{ij}^m - \sum_{\mathbf{x}_j \in \mathcal{X}\setminus\{\mathbf{x}_i,\mathbf{x}_m\}} f_{ji}^m = 0, && \begin{array}{l}\forall \mathbf{x}_m \in \mathcal{X}\setminus\{\mathbf{x}_r\},\\ \forall \mathbf{x}_i \in \mathcal{X}\setminus\{\mathbf{x}_r,\mathbf{x}_m\},\end{array}\\
&f_{ij}^m \leq y_{ij}^n, && \forall e_{ij} \in \mathcal{E},\ \mathbf{x}_m \in \mathcal{X}\setminus\{\mathbf{x}_r,\mathbf{x}_i,\mathbf{x}_j\},\\
&f_{im}^m = y_{im}^n, && \forall e_{im} \in \mathcal{E},\\
&f_{ij}^m \geq 0, && \forall e_{ij} \in \mathcal{E},\ \mathbf{x}_m \in \mathcal{X}\setminus\{\mathbf{x}_r,\mathbf{x}_i\},\\
&y_{ij}^n \in \{0,1\}, && \forall e_{ij} \in \mathcal{E}
\end{aligned} \tag{3.4}$$

where $\mathbf{x}_r$ is the imaginary root vertex.

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 [33], explicitly constraining the edge variables y_{ij}^n to be either zero or one is enough to achieve this goal. This is why our initial integer program turns into a qadratic mixed integer program (Q-MIP).

3.3.3 Flow Variables and Temporal Consistency

We now turn to enforcing temporal consistency. In our earlier work [44], this was done by penalizing situations in which binary variables associated to corresponding edges in different time frames had different values. More specifically, we introduced a persistence probability q. It quantifies how likely it is for a pair of corresponding edges $(\mathbf{e}_{ij}^n, \mathbf{e}_{i'j'}^{n+1}) \in \bar{\mathcal{E}}_t$ 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. 3.3. 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. 3.7.

Here we show how to use the flow variables introduced in Section 3.3.2 to enforce more long-range consistency. To this end, we first show that

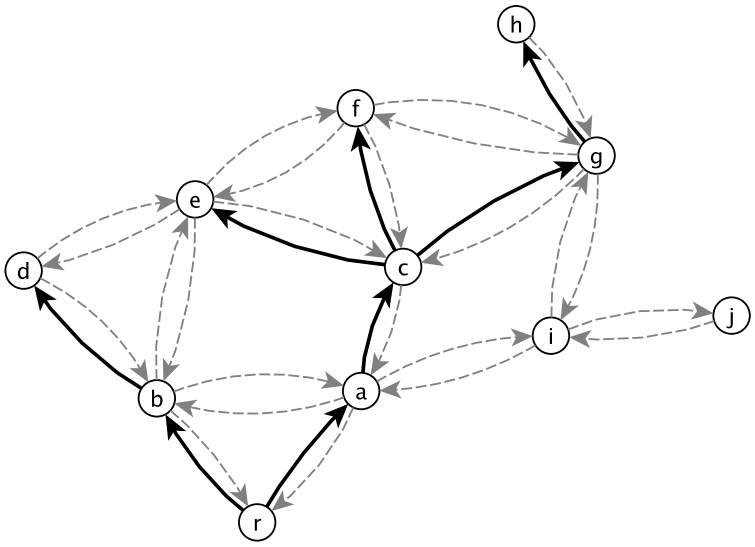


Figure 3.6: Flows in a graph rooted at vertex r. The arrows represent directed edges. Those that are part of the solution tree are shown as solid, the others as dashed. Note that the solution does not necessarily have to span all the vertices. In this specific case, vertices i and j do not belong to the tree. There are 11 vertices and 32 edges in this graph, which gives a total of 32 edge indicator variables y_{ij}^n and $11 \cdot 32 = 352$ flow variables f_{ij}^m. For the 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.

this long-range consistency can be expressed in terms of the flow variables introduced above. 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. 3.3. We will demonstrate that this significantly improves performance in the results section.

Modeling Long-Range 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 the approach of [44] 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, long-range 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 neighborhood 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 3.2. Let us assume 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 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. 3.5(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. 3.5(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. 3.5(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. 3.5(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. 3.5(e).)

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. 3.5(f).)

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.

Augmenting the Objective Function

The posterior distribution of Y given the spatial edges $\mathcal{E}_s$ and the temporal edges $\mathcal{E}_t$ can be expressed as

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

assuming that the image data and the spatial edges are conditionally independent of the temporal edges given Y. As in Section 3.3.1, we can look for the optimal tree as the one that maximizes this probability. That amounts to finding

$$\begin{aligned} y^* &= \arg\max_{y\in\mathcal{Y}} P\left(\mathcal{I},\mathcal{X},\mathcal{E}_s|Y=y\right) P\left(Y=y|\mathcal{E}_t\right), \qquad (3.5)\\ &= \arg\min_{y\in\mathcal{Y}} \sum_{\mathbf{e}_{ij}^n,\mathbf{e}_{jk}^n\in\mathcal{E}_s} w_{ijk} y_{ij}^n y_{jk}^n \\ &+ \sum_{(\mathbf{x}_m^n,\mathbf{x}_{m'}^{n+1})\in\mathcal{E}_t} \sum_{(\mathbf{e}_{ij}^n,\mathbf{e}_{i'j'}^{n+1})\in\bar{\mathcal{E}}_t} w_p \left(2 f_{ij}^m f_{i'j'}^{m'} - f_{ij}^m - f_{i'j'}^{m'}\right). \end{aligned}$$

where $w_p = -\log\frac{q}{1-q}$. 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. 3.4, which are also expressed in terms of the the y_{ij}^n and f_{il}^m variables. The problem therefore remains a Q-MIP.

In addition to being more global than the method of [44], 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. 3.7. 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 middle subfigure looks temporally consistent they would both be considered consistent by the short range consistency term due to the missing correspondences between edges in the critical locations. The long range consistency term would penalize the solution in the bottom 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. 3.4, the presented method might be considered a generalization of the one presented in [44]. 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.

3.3.4 Fine alignment

After obtaining the final delineations in all time instants, the iterative GPR method introduced in section 3.2 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 seed points 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.

3.3.5 Speeding Up the Computation

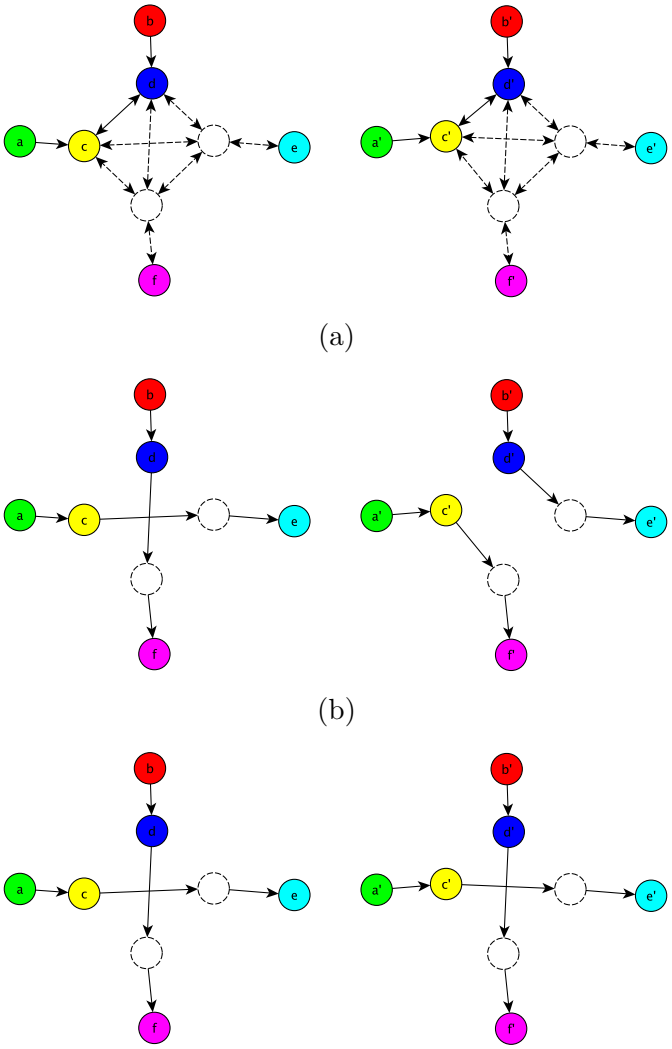
The path from the imaginary root to any vertex $\mathbf{x}_r^n$ of the spatio-temporal graph cannot pass through edges residing at time steps different from I^n. Therefore, it is easy to predict that all of the respective flow variables will be equal to zero for any valid solution and it is unnecessary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

3.4 Results

We first present the three very different image datasets we used to test our approach. We then show that enforcing time-consistency allows us to improve overall performance.

3.4.1 Image Datasets

We evaluated our method on 3D 2-photon images of axons in the brain of a mouse and on aerial images of the same area take in different years and seasons.



(a)

(b)

(c)

Figure 3.7: 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 successfully established and the corresponding vertices are represented with matching colors. 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 short range consistency term due to the missing correspondences between edges in the critical locations. The long range consistency term would penalize solution (b) as the following flow variables are inconsistent: $1 = f^{e}_{ac} \neq f^{e'}_{a'c'} = 0$, $0 = f^{e}_{bd} \neq f^{e'}_{b'd'} = 1$, $0 = f^{f}_{ac} \neq f^{f'}_{a'c'} = 1$, $1 = f^{f}_{bd} \neq f^{f'}_{b'd'} = 0$.

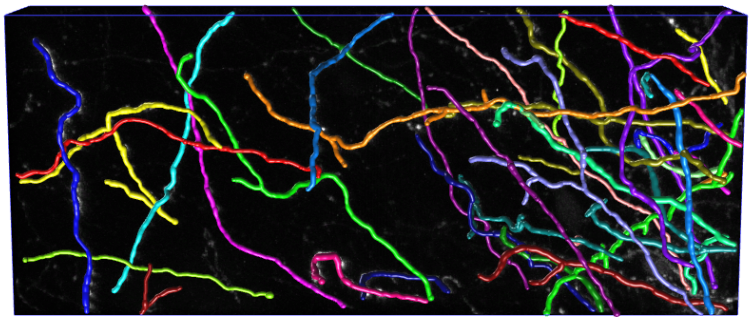


Figure 3.8: A 3D volume used for training the path classifier for the brain datasets. The manual annotation is shown in bright colors.

Brain Structures. Long-term memory is thought to be stored in the configuration of the synaptic wiring diagram of brain circuits. The synaptic connections between neurons are found on tree-like dendrites and axons through which they receive input and provide output respectively. The complex nature of dendrites and axons allows neurons to gather and distribute information from and to a plethora of other neurons that reside in spatially segregated areas. The rewiring of synaptic circuits could be accomplished by structural changes in the branches of those input and output trees, which would thereby reprogram the circuit's function. This may be important for learning and memory formation.

We collaborate with neuroscientists who aim at mapping structural circuit changes in the mouse brain during the learning processes. To this end they acquire large-scale 2-photon laser scanning microscopy images of a sparse set of fluorescent labeled neurons in the neocortex. Images are taken through a permanently implanted cranial window, which lets them track specific structures over months during which the mouse learns new tasks or undergoes new experiences.

We used three large image stacks, labeled 1 to 3, of the same area of the brain at three different time instants. To train the path classifier, we selected a region from one of the stacks, asked an expert to manually annotate it, and sampled 20000 positive and 20000 negative paths. The training stack is depicted by Fig. 3.8. Five sequences of smaller volumes were then selected from the three image stacks 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, BR3, BR4, and BR5.

Urban Roads. 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

Figure 3.9: Road images used to train the path classifier. The manual annotation is shown in bright colors.

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. 3.9. We also cropped one sequence of three images for testing.

Runner Bean. We used a nine-frame time-lapse sequence of a growing runner bean. This is relevant because monitoring the growth of a plant has many uses. They include testing different environmental conditions, getting to understand the influence of specific pesticides or other agricultural products, and evaluating models of plant development and growth [81]. Again, we picked six of the images for training the path classifier and the other three images constituted the testing sequence.

3.4.2 Overall Performance

In Table 3.1, we compare the results of reconstructing independently in each test image of these datasets using the method of [97], of imposing short-range temporal constraints as in [44], and of imposing long-range constraints

temporal constraints as discussed in this paper. To this end, we measure the quality of the reconstructions in terms of the DIADEM metric [4]. 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 of persistence probability value q, introduced in section 3.3.3, 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. 3.11 and 3.10. They illustrate that both the short range and the long range consistency methods favor reconstructions consistent over time, with the long range 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 long-range consistency, also introduced in section 3.3.3, 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. Similarly 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 3.1, enforcing local temporal consistency improves the overall quality of the results and imposing long-range 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 3.3.5, influences the computation time. Due to the nondeterministic nature of the optimization procedure we performed each of the optimizations ten times and observed consistent behavior with only minor differences between specific runs. In the case of the short-range consistency the computation time turned out to be consistently lower, reduced for instance from 16.3s to 3.9s on one small example and from 470s to 247s on a bigger one. In the case of the long-range consistency the computation would also complete around two to three times faster. However, it is worth noting that

		Single [97]	Short-range [44]	Long-range
BR1	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 BR1	0.5628	0.3487	**0.5949**
BR2	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 BR2	0.4201	0.5514	**0.5631**
BR3	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 BR3	0.6458	0.6396	**0.7945**
BR4	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 BR4	0.4948	0.5797	**0.5986**
BR5	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 BR5	0.2737	0.3211	**0.3591**
RD	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 RD	0.3257	0.3987	**0.4500**

Table 3.1: DIADEM scores [4] for the brain images datasets (denoted BRi) and road images dataset (denoted RD). The scores in the first column were obtained without time-consistency, that is, by using the method of [97]. The scores in the second and third columns were obtained by imposing short-range time consistency as in our earlier method [44] and long-range time consistency as described in this paper.

for one example the average optimization time raised from 1958s to 3079s after removing the unnecessary flow variables.

3.4.3 Change Detection

Fig. 3.1 depicts our results on the runner-bean sequence. The branch structure is correctly reconstructed and the important topological changes are automatically found. In Fig. 3.1(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. Since the GPR allows for nonlinearity, the correct correspondences between the tree structures are nevertheless found and the tree reconstructions and registration are achieved accurately.

Figure 3.10: 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.

A similar phenomenon can be observed in Fig. 3.2(f). In the lower left corner, a branch seems to have retracted while, in the upper right corner, another seems to have changed orientation. Both of these areas are denoted by a red overlay.

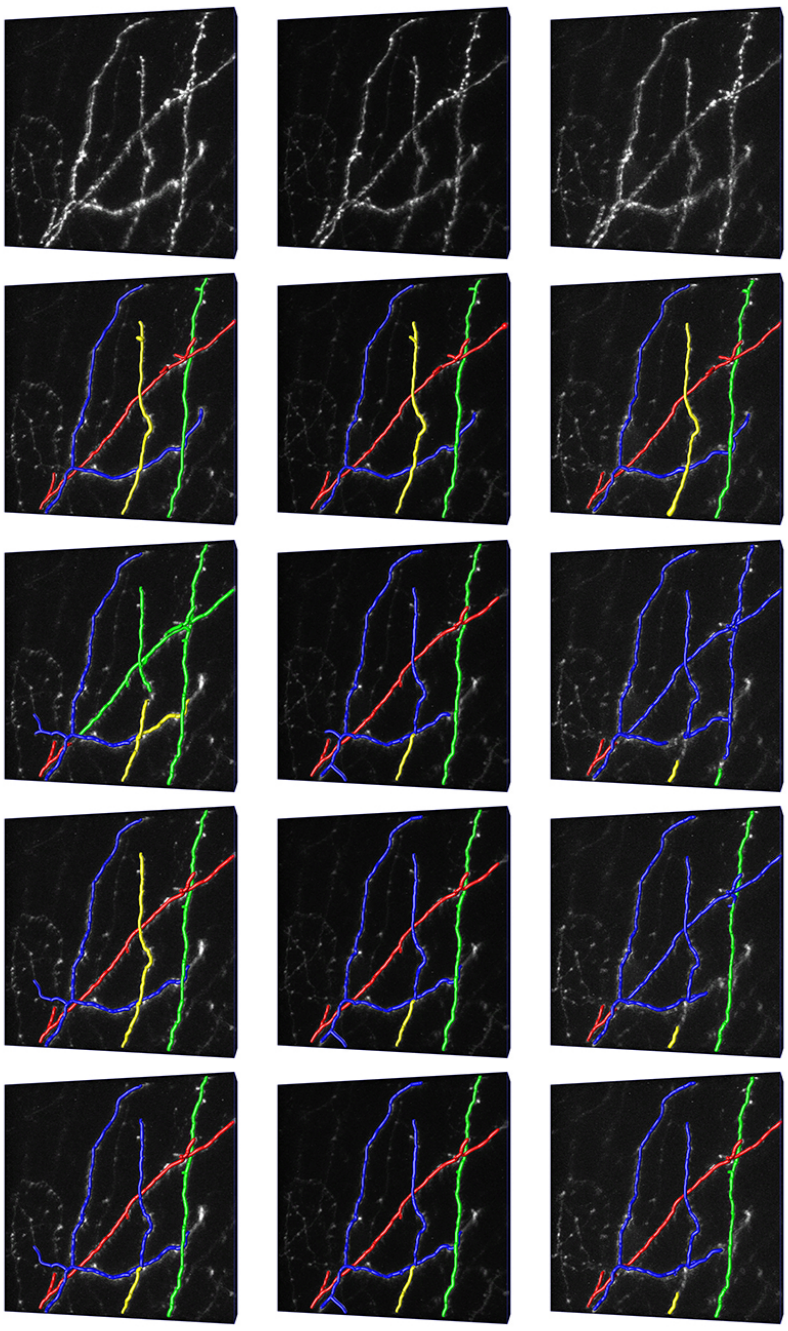


Figure 3.11: Example brain data results. The first row presents the input images. The second row presents the ground truth. The third row presents the results computed with no temporal consistency. The fourth and fifth rows present results with flow persistency set to 0.75 and range parameter set to 1 and 4, respectively.

Chapter 4

Active Testing Search

We first present a new approach for matching sets of branching curvilinear structures that form graphs embedded in $\mathbb{R}^2$ or $\mathbb{R}^3$ and may be subject to deformations. Unlike earlier methods, ours does not rely on local appearance similarity nor does require a good initial alignment. Furthermore, it can cope with non-linear deformations, topological differences, and partial graphs. To handle arbitrary non-linear deformations, we use Gaussian process regressions to represent the geometrical mapping relating the two graphs. In the absence of appearance information, we iteratively establish correspondences between points, update the mapping accordingly, and use it to estimate where to find the most likely correspondences that will be used in the next step. To make the computation tractable for large graphs, the set of new potential matches considered at each iteration is not selected at random as with many RANSAC-based algorithms. Instead, we introduce a so-called Active Testing Search strategy that performs a priority search to favor the most likely matches and speed-up the process. We demonstrate the effectiveness of our approach first on synthetic cases and then on angiography data, retinal fundus images, and microscopy image stacks acquired at very different resolutions.

The text in this chapter is strongly based on the text of the paper [89], an extended version of the paper [77].

4.1 Approach

Let us assume we are given two graphs $\mathcal{G}^A = (\mathbf{V}^A, \mathbf{E}^A)$ and $\mathcal{G}^B = (\mathbf{V}^B, \mathbf{E}^B)$, such as the one of Fig. 1.1-right, extracted from two images or image-stacks A and B. The $\mathbf{E}$s denote the graphs' edges and the $\mathbf{V}$s their nodes—shown as red dots in the figure—that are points in $\mathbb{R}^D$, where we assume $D \in \{2, 3\}$. The edges, in turn, are represented by dense sets of points—shown as white dots in the figure—that form $\mathbb{R}^2$ or $\mathbb{R}^3$ paths connecting the nodes.

Our goal is to use these two graphs to find a geometrical mapping m from A to B such that $m(\mathbf{v}_i^A)$ is as close as possible to $\mathbf{v}_j^B$ in the least-squares sense assuming that $\mathbf{v}_i^A$ and $\mathbf{v}_j^B$ are corresponding pixels or voxels.

If correspondences between points belonging to the two graphs were given, we could directly use the Gaussian Process Regression (GPR) [82] to estimate a non-linear mapping that would yield a prediction of m and its associated variance [11]. In our case, however, the correspondences are initially unavailable and cannot be established on the basis of local image information because the A and B are too different in appearance. In short, this means that we must rely only on geometrical properties to simultaneously establish the correspondences and estimate the underlying non-linear transform. Since attempting to do this directly for all edge points would be computationally intractable, our algorithm goes through the following two steps:

1. **Coarse alignment:** We begin by only matching graph nodes so that the resulting mapping is a combination of an affine deformation and a smooth non-linear deformation. We initialize the search by randomly picking D correspondences, which roughly fixes relative scale and orientation, and using them to instantiate a Gaussian Process (GP). We then recursively refine it as follows: Given some matches between $\mathcal{G}^A$ and $\mathcal{G}^B$ nodes, the GP serves to predict where other $\mathcal{G}^A$ nodes should map and restricts the set of potential correspondences. Among these possibilities, we select the most promising one based on geometric or information gain criteria we will define in Section 4.3, and use it to refine the GP. Repeating this procedure recursively until enough mutually consistent correspondences have been established and backtracking when necessary lets us quickly explore the set of potential correspondences and recover an approximate geometric mapping.
2. **Fine alignment:** Having been learned only from potentially distant graph nodes, the above-mapping is coarse. To refine it, we also establish correspondences between points that form the edges connecting the nodes

in such a way that distances along these edges, which we will refer to as *geodesic* distances, are changed as little as possible between the two graphs. Because there are many more such points than nodes, this would be extremely expensive to do from scratch. Therefore, we constrain the correspondence candidates to edges between already matched nodes and rely on a Hungarian algorithm [70] to perform the optimal assignment quickly.

In the remainder of this chapter, we first outline the GPR model that we use. We then introduce our procedures for coarse and fine alignments.

4.2 Gaussian Process Regression

Without loss of generality, let us assume that the elements of $\mathbf{V}^A$ and $\mathbf{V}^B$ have been reordered so that the set $\mathbf{M} = \left\{\mathbf{v}_l^A \leftrightarrow \mathbf{v}_l^B\right\}_{1\leq l\leq n_c}$ denotes correspondences between D-dimensional points from A and B respectively. Using the GP approach to non-linear regression and assuming Gaussian i.i.d. noise of precision β in all coordinate values, these correspondences can be used to predict that a point $\mathbf{v}^B$ in B corresponding to $\mathbf{v}^A$ in A can be expected to be found at a location with the following mean $m_{\mathbf{M}}(\cdot)$ and isotropic variance $\sigma^2_{\mathbf{M}}(\cdot)$:

$$m_{\mathbf{M}}(\mathbf{v}^A) = \mathbf{k}^T\mathbf{C}_{\mathbf{M}}^{-1}\mathbf{V}_{\mathbf{M}}^B \ , \tag{4.1}$$

$$\sigma^2_{\mathbf{M}}(\mathbf{v}^A) = k(\mathbf{v}^A,\mathbf{v}^A) + \beta^{-1} \ - \mathbf{k}^T\mathbf{C}_{\mathbf{M}}^{-1}\mathbf{k} \ , \tag{4.2}$$

where k is a kernel function, β^{-1} is the measurement noise variance, $\mathbf{C}_{\mathbf{M}}$ is the $n_c \times n_c$ symmetric matrix with elements $c_{i,j} = k(\mathbf{v}_i^A,\mathbf{v}_j^A) + \beta^{-1}\delta_{i,j}$, $\mathbf{k}$ is the vector $[k(\mathbf{v}_1^A,\mathbf{v}^A),\ldots,k(\mathbf{v}_{n_c}^A,\mathbf{v}^A)]^T$, and $\mathbf{V}_{\mathbf{M}}^B$ is the $n_c \times D$ matrix $[\mathbf{v}_1^B,\ldots,\mathbf{v}_{n_c}^B]^T$.

Among the different types of existing kernel functions [82], we chose the widely used summation of a squared-exponential, a constant term, and a linear term

$$k(\mathbf{v}_i,\mathbf{v}_j) = \theta_0 + \theta_1\mathbf{v}_i^T\mathbf{v}_j + \theta_2\exp\left\{-\frac{\theta_3}{2}||\mathbf{v}_i-\mathbf{v}_j||^2\right\}. \tag{4.3}$$

Thus, the kernel of (4.3) models transformations as being rigid with non-linear deformations. Such transformation covers a wide range of situations in medical imaging [29, 79, 90, 92].

Given this expression for k, the geometric mapping from Eq. (4.1) can now be rewritten as

$$\begin{aligned} m_{\mathbf{M}}(\mathbf{v}^A) &= \sum_{i=1}^{n_c} \mathbf{a}_i k(\mathbf{v}_i^A, \mathbf{v}^A) \\ &= \sum_{i=1}^{n_c} \mathbf{a}_i (\theta_0 + \theta_1 (\mathbf{v}_i^A)^T \mathbf{v}^A) + \\ &\quad \sum_{i=1}^{n_c} \mathbf{a}_i \, \theta_2 \exp\left\{ -\frac{\theta_3}{2} ||\mathbf{v}_i^A - \mathbf{v}^A||^2 \right\} , \end{aligned} \tag{4.4}$$

where $\mathbf{a}_i$ is the i^{th} row of the matrix $\mathbf{C}_{\mathbf{M}}^{-1}\mathbf{V}_{\mathbf{M}}^B$. The first term of Eq. (4.4), which contains the θ_0 and θ_1 hyperparameters, is a linear function of the input coordinates while the second one, which involves the θ_2 and θ_3, allows for additional non-linear deformations.

Apart from the mapping $m_{\mathbf{M}}(\cdot)$, we also need to evaluate the mapping quality for any particular set of correspondences $\mathbf{M}$. Let us define a quality score as $S_{\mathbf{M}} \in \mathbb{R}$, which is a deterministic function. We use the following two methods to evaluate the quality of a correspondence set:

- **Assigned distance:** We compute the minimum possible total distance between $m_{\mathbf{M}}(\mathbf{v}_i^A)$ and points $\mathbf{v}_j^B \in \mathbf{V}^B$ for all $1 \leq i \leq n_A$

$$S_{\mathbf{M}} = \sum_{i=1}^{n_A} \sum_{j=1}^{n_B} H_{i,j} \cdot \mathrm{dist}(m_{\mathbf{M}}(\mathbf{v}_i^A), \mathbf{v}_j^B), \tag{4.5}$$

  where 'dist' is a Euclidean distance and H is the assignment matrix computed by the Hungarian algorithm [70] which, given the distances between all points $m_{\mathbf{M}}(\mathbf{v}_i^A)$ and $\mathbf{v}_j^B$, iteratively calculates the optimal assignment of correspondences between all points, by minimizing $S_{\mathbf{M}}$.

- **Number of inliers:** We compute the proportion of edge and branching points in $\mathbf{V}^A$ that are mapped near a point in $\mathbf{V}^B$ as

$$S_{\mathbf{M}} = \frac{|I|}{|\mathbf{V}^A|}, \tag{4.6}$$

  $$I = \left\{ \mathbf{v}_j^B \mid \exists m_{\mathbf{M}}(\mathbf{v}_i^A),\ \mathrm{dist}(m_{\mathbf{M}}(\mathbf{v}_i^A), \mathbf{v}_j^B) < \beta^{-\frac{1}{2}} \right\}.$$

Our experiments show that these two measures suffice to recognize good sets of correspondences.

4.3 Coarse Alignment

Let $\mathbf{V}^A = \left\{\mathbf{v}_1^A, \ldots, \mathbf{v}_{n_A}^A\right\}$ and $\mathbf{V}^B = \left\{\mathbf{v}_1^B, \ldots, \mathbf{v}_{n_B}^B\right\}$ be the nodes of our two graphs. Our first goal is to simultaneously retrieve as many correspondences $\mathbf{M} = \{\mathbf{v}^A \leftrightarrow \mathbf{v}^B\}$ as possible and to determine the underlying non-linear mapping $\mathbf{v}^B = m_{\mathbf{M}}(\mathbf{v}^A)$ that best aligns them.

In this section, we present two different approaches for doing this. The first one—Section 4.3.1—relies on first assigning correspondences to nodes for which there are few to choose from. The second—Section 4.3.2—uses a more sophisticated strategy that ranks partial solutions and attempts to extend the most promising ones first. We introduced the first strategy in [87] and tested it successfully on relatively small graphs. However, as will be shown in Section 4.5, its computational requirements grow quickly with the number of graph nodes. The second strategy, while slightly more complex, scales better.

4.3.1 Greedy Search

Let $m_{\mathbf{M}}(\cdot)$ be a GP written using the formulation of Section 4.2, which we instantiate by first selecting a set of D random correspondences (line 1 in Algorithm 1) – we set the initial size of potential candidates to be the data points dimensionality. This gives us an initial correspondence set $\mathbf{M}_0$. We then iteratively construct sets of correspondences in T steps as follows.

1. At iteration t, we have a set of correspondences $\mathbf{M}_t$ from which we compute (line 3) the mapping $m_{\mathbf{M}_t}(\cdot)$ and covariance $\sigma^2_{\mathbf{M}_t}(.)$ using Eqs. (4.1) and (4.2).

2. For each unmatched node $\mathbf{v}_i^A \in \mathbf{V}^A$, we search for potential correspondences $\mathbf{v}_j^B \in \mathbf{V}^B$ in the bounded region $\mathcal{B}_i$ determined by the predicted covariance $\sigma^2_{\mathbf{M}_t}(\mathbf{v}_i^A)$ (lines 6–7). We use the Mahalanobis distance to define the boundary:

$$\mathcal{M}^2 = \left(m_{\mathbf{M}_t}(\mathbf{v}_i^A) - \mathbf{v}_j^B\right)^T \left(\sigma^2_{\mathbf{M}_t}(\mathbf{v}_i^A)\right)^{-1} \left(m_{\mathbf{M}_t}(\mathbf{v}_i^A) - \mathbf{v}_j^B\right)$$
$$\mathcal{B}_i = \left\{\forall \mathbf{v}_j^B \in \mathbf{V}^B | \mathcal{M}^2(m_{\mathbf{M}_t}(\mathbf{v}_i^A), \mathbf{v}_j^B) < 2\right\}$$

3. We choose the node $\mathbf{v}_{i^*}^A$ with the smallest number of potential candidates (line 9), and randomly pick one of them to define the match $\mathbf{v}_{i^*}^A \leftrightarrow \mathbf{v}_{j^*}^B$, which we add to the correspondence set $\mathbf{M}_t$ (lines 11–12). If there is no

Algorithm 1 Greedy Alignment $(\mathcal{G}^A, \mathcal{G}^B; \mathbf{\Theta}, \beta; T)$

1: Initialize Correspondence Set:
2: $\mathbf{M}_0 \leftarrow \{\mathbf{v}^A_{i_1} \leftrightarrow \mathbf{v}^B_{j_1}, \ldots, \mathbf{v}^A_{i_D} \leftrightarrow \mathbf{v}^B_{j_D}\}$
3: **for** $t = 0, \ldots, T$ **do**
4: $\quad\{m_{\mathbf{M}_t}, \sigma^2_{\mathbf{M}_t}\} = \text{ComputeMapping}(\mathbf{V}^A, \mathbf{V}^B, \mathbf{\Theta}, \mathbf{M}_t)$
5: $\quad S_{\mathbf{M}_t} = \text{QualityScore}(m_{\mathbf{M}_t}, \beta)$
6: $\quad$**for** $i = 1 \ldots n_A$ **do**
7: $\quad\quad\mathcal{B}_i = \text{ComputeBoundary}(m_{\mathbf{M}_t}(\mathbf{v}^A_i), \sigma^2_{\mathbf{M}_t}(\mathbf{v}^A_i), \mathbf{V}^B)$
8: $\quad\quad\text{PotCand}_i = \{\mathbf{v}^B_j; \mathbf{v}^B_j \in \mathbf{V}^B \wedge \mathbf{v}^B_j \in \mathcal{B}_i\}$
9: $\quad$**end for**
10: $\quad i^* = \arg\min_i\{|\text{PotCand}_i|\}$ for $|\text{PotCand}_i| \neq 0$
11: $\quad$**if** $i^* \neq \emptyset$ **then**
12: $\quad\quad\mathbf{v}^B_{j^*} = \text{PickRandom}(\text{PotCand}_{i^*})$
13: $\quad\quad\mathbf{M}_{t+1} \leftarrow \mathbf{M}_t \cup \{\mathbf{v}^A_{i^*} \leftrightarrow \mathbf{v}^B_{j^*}\}$
14: $\quad$**else**
15: $\quad\quad\mathbf{M}_{t+1} \leftarrow \mathbf{M}_{t-1}$
16: $\quad$**end if**
17: **end forreturn** $\mathbf{M}^* = \arg\max_{\{1,\ldots,T\}} S_{\mathbf{M}_t}$

point from $\mathbf{V}^B$ which satisfies the conditions to be selected, we remove the last added correspondence from $\mathbf{M}_t$ and continue searching (line 14).

4. We take the quality score $S_{\mathbf{M}_t}$ to be the number of inliers as defined in Eq. (4.6).

As described in Algorithm 1, this process uses a depth-first search and is repeated T times, backtracking and selecting sets of correspondences of different sizes, unlike RANSAC which uses a fixed set size at each iteration. We then return the correspondence set $\mathbf{M}^*$ with the highest score, and its corresponding $m_{\mathbf{M}^*}$. We also terminate if the inlier consenus $S_{\mathbf{M}_t}$ becomes large enough. An example of this process is shown in Fig. 4.1.

The process is controlled by the noise parameter β of Eq. (4.2) and the vector $\mathbf{\Theta} = \{\theta_0, \theta_1, \theta_2, \theta_3\}$ containing the kernel hyperparameters of Eq. (4.3). To avoid having to tune these parameters for each new dataset, we center and scale the $\mathbf{V}^A$ and $\mathbf{V}^B$ coordinates so that their average distance to the origin is one and perform the computation on the scaled datasets. As a result, we were able to use the same $\mathbf{\Theta}$ for all experiments described in Section 4.5.

To speed up the computation, we reject matches that would produce large changes in geodesic distances, which we define as the length of a path connecting the edges between two graph nodes $\mathbf{v}_i$ and $\mathbf{v}_j$. Given already established correspondences $\mathbf{M}$ between graphs, then for each new potential

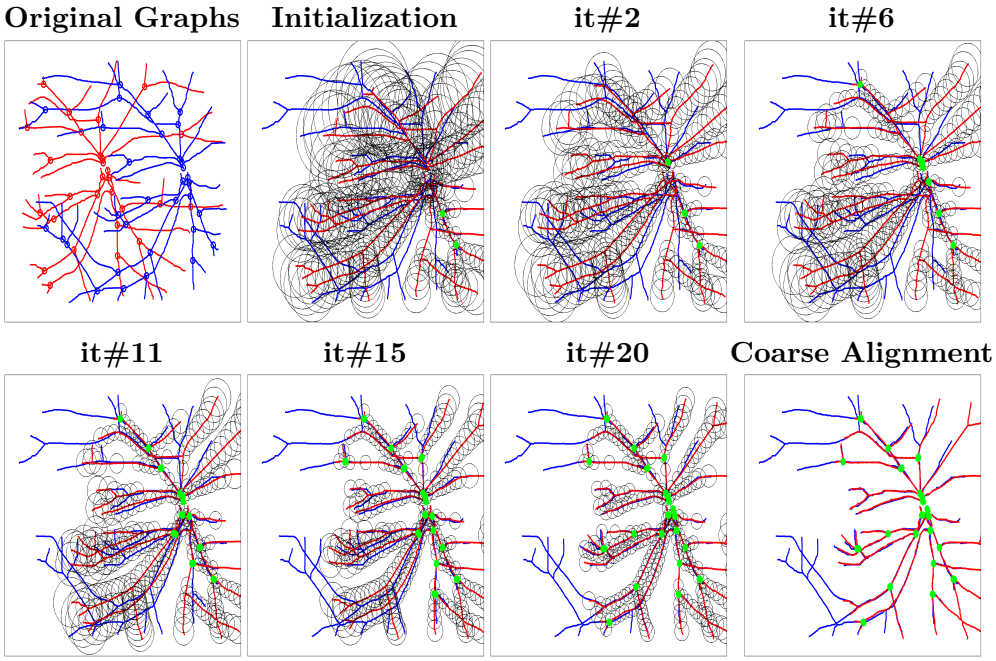
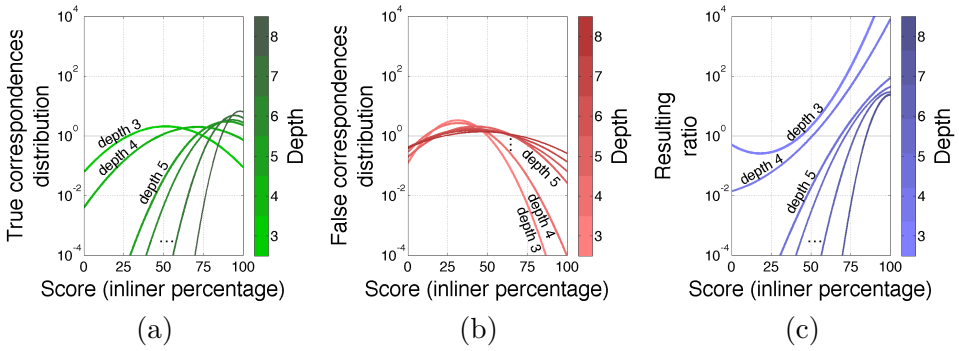


Figure 4.1: Coarse alignment steps. The initial graph structures are depicted in the top left-most figure, the model graph in red and the target in blue. Exploration of the search space starts by picking randomly two correspondences, highlighted in green, thus roughly fixing scale and orientation. Then, the next match candidate is chosen among the nodes located inside the bounded regions, which are a function of the GP predicted covariances, shown as black ellipses. Every correspondence added to the hypotheses set helps refining the mapping uncertainty. The final correspondence set defines a coarse alignment of the graphs. Best viewed in color.

match, the geodesic distances from the new corresponding points to the already matched nodes in both graphs have to be approximately proportional. We set the tolerance for geodesic distance variations depending on the level of deformations we expect to recover. Proceeding in this way, the algorithm gains robustness against outliers, while avoiding unnecessary checks, thus keeping a low complexity. Note that geodesic distances are invariant to rotations, to the bending of the branches, and to isometric changes.

4.3.2 Active Test Search

We have tested the algorithm described above on graphs containing up to 100 nodes, for which the computation takes more than 1000 seconds in Matlab on an 4-Core 2.3 GHz 64-bit processor. because the computational cost grows exponentially with the number of nodes, it becomes impractical for larger graphs.



(a) (b) (c)

Figure 4.2: Gaussian noise models for percentage of inliers. (a) Each curve depicts the Gaussian noise model $\mathcal{N}(\cdot;\boldsymbol{\omega}_1^u)$ for a given depth u of the tree. **(b)** Similarly, each curve depicts the noise models for $\mathcal{N}(\cdot;\boldsymbol{\omega}_0^u)$. **(c)** Likelihood ratio $r(\cdot)$ between $\mathcal{N}(\cdot;\boldsymbol{\omega}_1^u)$ and $\mathcal{N}(\cdot;\boldsymbol{\omega}_0^u)$ for each value of u.

We have therefore developed an alternative approach that relies on the Active Testing Search (ATS) [42, 94]. This involves progressively refining an approximate solution by making a budgeted number of observations and computing the posterior distribution over all potential solutions after each test. Hence, the algorithm proceeds iteratively and selects at each step the correspondence set expected to yield the highest information gain based on all previous ones.

In other words, our method does not perform either a depth-first search, such as the one described in Section 4.3.1, or a traditional breadth-first search, but a priority search. For this purpose, as new correspondences are added to partial solutions, it maintains a sorted list of which ones are most likely to lead to a correct solution. It then attempts to extend these first so that less likely candidates may never be extended at all, saving computation time.

In addition, our ATS approach is adaptive and allows for backtracking without hand-tuned pruning of the search space. It is summarized in Algorithm 2 and we describe it in more details below.

ATS—Coarse Alignment

ATS maintains a list of candidate correspondence sets, where we denote the probability that the correspondence set $\mathbf{M}$ is part of the correct mapping $\mathbf{M}^*$ as $\epsilon_{\mathbf{M}}$. This list of candidates is handled by means of a priority queue Q (line 1 of Algorithm 2), whose elements are $(\mathbf{M}, \epsilon_{\mathbf{M}})$ pairs sorted in order of decreasing probability.

At first, we form all possible sets of D pairs of correspondences that can be used to initialize a mapping $m_{\mathbf{M}}(\cdot)$. We assume that each $\mathbf{M}$ from this initial set has equal probability ϵ.

At each ATS iteration, $t = 0, ..., T$, we want to select the candidate set $\mathbf{M}_t$ that is most likely to provide a good mapping. We therefore select the first element $(\mathbf{M}_t, \epsilon_{\mathbf{M}_t})$ of Q, which is the one with highest likelihood and then evaluate the quality score $S_{\mathbf{M}_t}$ (line 5) to verify if it is indeed a good mapping. Given that $S_{\mathbf{M}_t}$ can be noisy, we consider it to be a random variable with a known noise model, i.e. the likelihood model $P(S_{\mathbf{M}_t}|\mathbf{M}^*)$ is assumed to be explicitly known and is described in the following section.

To aggregate the information provided by the quality score, we compute the posterior distribution of the correct correspondences given the newly observed score. We further refine our candidates by expanding the candidate set previously evaluated. In particular, from $\mathbf{M}_t$, we generate a new set of candidate correspondences $\mathcal{C}_{\mathbf{M}_t} = \{\mathbf{M}_t \cup \{\mathbf{v}_i^A \leftrightarrow \mathbf{v}_j^B\} | \mathbf{v}_j^B \notin \mathbf{M}_t\}$, which contains all children of the current node, where $\mathbf{v}_i^A \notin \mathbf{M}_t$ is an additional element of $\mathbf{V}^A$.

As in [42, 94], the posterior for any element $\mathbf{M} \in \mathcal{C}_{\mathbf{M}_t}$ can be computed as

$$\epsilon \propto \frac{r(S_t)\epsilon_{\mathbf{M}_t}}{|\mathcal{C}_{\mathbf{M}_t}|}, \tag{4.7}$$

where $r(S_t)$ is the likelihood ratio (defined explicitly in the following section). The complete derivation of this equation as well as why the normalization constant is not needed are shown in Appendix A. Intuitively, Eq. (4.7) is simply an application of Bayes rule, where $\epsilon_{\mathbf{M}_t}$ is a prior, $r(S_t)$ is the data-term and dividing by $|\mathcal{C}_{\mathbf{M}_t}|$ attributes equal a priori probability to all the expanded candidates of $\mathbf{M}_t$.

This process is repeated T times or until the likelihood ratio is higher than ψ. We then return the assignment $\mathbf{M}^*$ with the best score. In Appendix A, we give further details and describe the derivation of the proposed algorithm.

Quality Score Selection and Noise Model

To compute the quality score $S_{\mathbf{M}}$ for any set of correspondences $\mathbf{M}$ we use the previously described assigned distance of Eq. (4.5) and the number of inliers of Eq. (4.6). In particular, we compute $S_{\mathbf{M}}$ using the assigned distance when

Algorithm 2 ATS Alignment $(\mathbf{V}^A, \mathbf{V}^B; \boldsymbol{\Theta}, \beta; T, \psi)$

Initialize Priority Queue:
$Q \leftarrow \text{Push}\left(\{\mathbf{v}_{i_1}^A \leftrightarrow \mathbf{v}_{j_1}^B, \ldots, \mathbf{v}_{i_D}^A \leftrightarrow \mathbf{v}_{j_D}^B\}, \epsilon\right)$
for $t = 1 \ldots T$ **do**
  $\{\mathbf{M}_t, \epsilon_t\} = \text{Pop}(Q)$
  $\{m_{\mathbf{M}_t}, \sigma^2_{\mathbf{M}_t}\} = \text{ComputeMapping}(\mathbf{V}^A, \mathbf{V}^B, \boldsymbol{\Theta}, \mathbf{M}_t)$
  $S_{\mathbf{M}_t} = \text{QualityScore}(m_{\mathbf{M}_t})$
  if $r(S_{\mathbf{M}_t}) > \psi$ **then return** $\mathbf{M}^* = \mathbf{M}_t$ **end if**
  for $\mathbf{M}_z \in \mathcal{C}_{\mathbf{M}_t}$ **do**
    $Q \leftarrow \text{Push}\ (\mathbf{M}_t, \epsilon_{\mathbf{M}_t} r(S_{\mathbf{M}_t})/|\mathcal{C}_{\mathbf{M}_t}|\ )$
  end for
end forreturn $\mathbf{M}^* = \arg\max_{\{1,\ldots,T\}} S_{\mathbf{M}_t}$

the number of correspondences $|\mathbf{M}|$ is below a certain threshold γ, which we set to 5 in all experiments. Otherwise we compute $S_{\mathbf{M}}$ using the number of inliers. We found that combining two different quality functions provides more informative scores for small and large sets of correspondences. This is similar to the strategies employed in [21].

We consider the quality scores to be random noisy observations and assume the following observation model

$$P(S_{\mathbf{M}} = s|\mathbf{M}^*) = \begin{cases} \mathcal{N}(s; \boldsymbol{\omega}_1^u), & \text{if } \delta(\mathbf{M}, \mathbf{M}^*) = 1 \\ \mathcal{N}(s; \boldsymbol{\omega}_0^u), & \text{if } \delta(\mathbf{M}, \mathbf{M}^*) = 0 \end{cases}, \tag{4.8}$$

where $\mathbf{M}^*$ is the correct set of correspondence assignments, u is the number of correspondences in $\mathbf{M}$, $\mathcal{N}$ is a Gaussian probability distribution with parameters $\boldsymbol{\omega}_1^u$, $\boldsymbol{\omega}_0^u$ and $\delta(\mathbf{M}, \mathbf{M}^*) = 1$ if the correspondences of $\mathbf{M}$ respect $\mathbf{M}^*$ and 0 otherwise. From this model, the likelihood ratio can be computed as

$$r(S_{\mathbf{M}}) = \frac{\mathcal{N}(S_{\mathbf{M}}; \boldsymbol{\omega}_1^u)}{\mathcal{N}(S_{\mathbf{M}}; \boldsymbol{\omega}_0^u)}. \tag{4.9}$$

To learn the parameters of the Gaussian distributions $\mathcal{N}(\cdot; \boldsymbol{\omega}_1^u)$ and $\mathcal{N}(\cdot; \boldsymbol{\omega}_0^u)$, we proceed as follows:

- **True Distribution:** To estimate the parameters for $\mathcal{N}(\cdot; \boldsymbol{\omega}_1^u)$, we synthetically generate L point clouds $\mathbf{V}^A$ such that $n_B > n_A$ and fit a minimum spanning tree to obtain the graph representation. The point cloud $\mathbf{V}^B$ is generated by applying random affine transformations and a smaller amplitude non-linear deformation to $\mathbf{V}^A$. This allows us to know exactly the true correspondence $\mathbf{M}^*$ and generate a set $\{\{\mathbf{V}^A\}_l, \{\mathbf{V}^B\}_l, \mathbf{M}_l^*\}_{l=1}^L$. Then, we take subsets of the full set of true correspondences $\mathbf{M}^*$ and compute $S_{\mathbf{M}}$. Once all scores on all L generated sets are computed, we

estimate the Gaussian distribution parameters $\{\boldsymbol{\omega}_1^u\}_{u=1}^{n_A} = \{\mu_1^u, \sigma_1^u\}_{u=1}^{n_A}$. An example of the learned probability densities can be seen in Fig. 4.2(a).

- **False Distribution:** Second, to learn the parameters for $\mathcal{N}(\cdot; \boldsymbol{\omega}_0^u)$, we follow a similar sampling approach. Given the number of possible incorrect correspondences, for partial assignments that include many assignments, i.e. when u is large, we construct sets of incorrect correspondences $\mathbf{M}$ by starting from a subset of $\mathbf{M}^*$ and adding a few incorrectly matched points. Proceeding in this manner, we take false partial assignments which are close to the true correspondence $\mathbf{M}^*$ because we expect to only explore the higher depths of the search tree, that is, high values of u, when our previous hypotheses are correct. An example of such distribution is depicted in Fig. 4.2(b).

In practice, we have found the above the process for learning the parameters of the observation models to be effective and robust. If enough training data with known correspondences is available, we could learn more complex models for the real shape of the positive and negative distributions. In addition, even though we use synthetically generated datasets, the same learned parameters are good enough to be used across different experiments in Sec. 4.5 and indicate that the parameters are fairly robust and valid for different tasks.

4.4 Fine Alignment

Given two graphs $\mathcal{G}^A$ and $\mathcal{G}^B$, both coarse alignment algorithms described above in Sections 4.3.1 and 4.3.2 return a set of corresponding graph nodes $\mathbf{M}^*$, along with the corresponding mapping $m(\cdot) = m_{\mathbf{M}^*}$ and the covariance estimator $\sigma^2(\cdot) = \sigma^2_{\mathbf{M}^*}$.

This set of matches $\mathbf{M}^*$ relates the graph's nodes and is therefore coarse. Given that the nodes are connected by paths, we can refine the mapping by establishing correspondences between edge points that lie on these paths. We assume the node correspondences to be correct and only establish new ones between points lying on paths linking matching nodes. We proceed iteratively using the following steps:

1. For each pair of paths connected by corresponding graph nodes, we use the Hungarian algorithm [70], guided by the current mapping $m_{\mathbf{M}_r}(\cdot)$ and covariance estimator $\sigma^2_{\mathbf{M}_r}(\cdot)$, to establish new matches $\mathbf{M}_{r+1}$ between

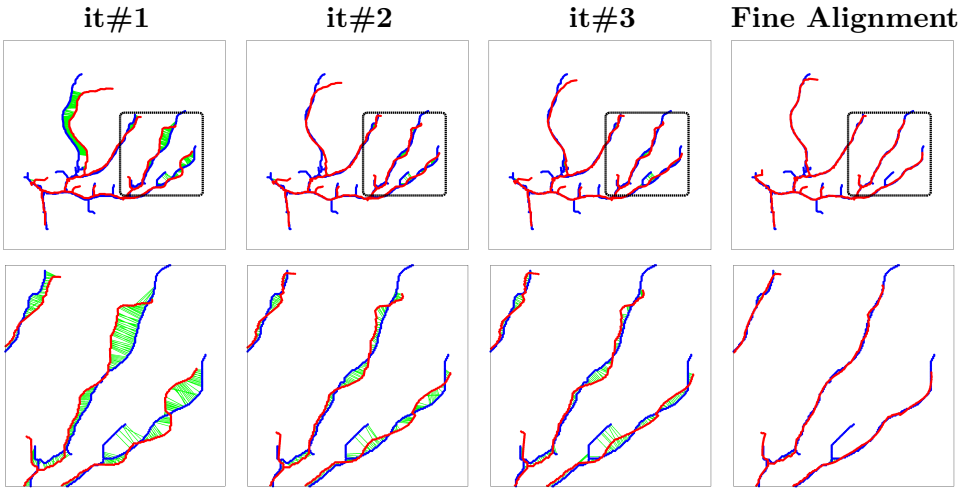


Figure 4.3: Fine alignment steps. Once a coarse alignment of the two graphs (model in red and target in blue) has been found, the algorithm starts matching points lying on the edges. The assignments (depicted in green) are computed using the Hungarian algorithm and constrained by the graph topology and GP predictions. After a few iterations, the warped structure **(top)** is completely aligned to the target graph. For each successive plot, we zoom to a smaller region **(bottom)** to better show the algorithm at work. Best viewed in color.

the edge points of the two paths. We constrain all the matches to have a consistent geodesic distance with their respective graph nodes.

2. Given these new correspondences $\mathbf{M}_{r+1}$, reestimate the values $m_{\mathbf{M}_{r+1}}(\cdot)$ and $\sigma^2_{\mathbf{M}_{r+1}}(\cdot)$.

3. Compute the quality of the resulting mapping $S_{\mathbf{M}_{r+1}}$ using the Assigned distance function defined in Eq. (4.5).

4. If $S_{\mathbf{M}_{r+1}} > S_{\mathbf{M}_r}$, iterate. Otherwise, terminate and return $\mathbf{M}_r$, $m_{\mathbf{M}_r}(\cdot)$, and $\sigma^2_{\mathbf{M}_r}(\cdot)$.

This yields a final expanded set of correspondences $\mathbf{M}_{fine}$, mapping $m_{\mathbf{M}_{fine}}(\cdot)$, and covariance estimator $\sigma^2_{\mathbf{M}_{fine}}(\cdot)$. Note that we use the same GP parameters $\boldsymbol{\Theta}$ as before. The whole process is illustrated by Fig. 4.3 and summarized in Algorithm 3.

4.5 Experiments

We evaluate our approach on both synthetic and real data against state-of-the-art methods. In the remainder of the paper we will refer to the

Algorithm 3 Fine Alignment $(\mathcal{G}^A, \mathcal{G}^B; \mathbf{\Theta}, \beta; \mathbf{M}^*)$

Initialize Correspondence Set from Coarse Alignment:
$\mathbf{M}_r = \mathbf{M}^*$
$m_{\mathbf{M}_r} = m_{\mathbf{M}_*},\ \sigma^2_{\mathbf{M}_r} = \sigma^2_{\mathbf{M}_*}$;
repeat
    $\mathbf{M}_{r+1} \leftarrow \text{OptimalAssignment}(\mathbf{V}^A, \mathbf{V}^B, \mathbf{\Theta}, \mathbf{M}_r)$
    $\{m_{\mathbf{M}_{r+1}}, \sigma^2_{\mathbf{M}_{r+1}}\} = \text{ComputeMapping}(\mathbf{V}^A, \mathbf{V}^B, \mathbf{\Theta}, \mathbf{M}_{r+1})$
    $S_{\mathbf{M}_{r+1}} = \text{QualityScore}(m_{\mathbf{M}_{r+1}})$
until $S_{\mathbf{M}_{r+1}} \leq S_{\mathbf{M}_r}$ **return** $\mathbf{M}_{fine} = \mathbf{M}_r$

methods presented in Sections 4.3.1 and 4.3.2 as Greedy-RGM (Greedy Search for Robust Graph Matching) and ATS-RGM (Active Testing Search for Robust Graph Matching), respectively. We will initially test all methods on synthetically generated data with increasing levels of noise, non-rigid deformation, missing points and different initial conditions. We will then show the performance of the algorithms on 2D and 3D biomedical images, including retinal images, neuronal structures and heart angiograms. The Gaussian Process parameters β and θ_i should ideally be estimated from training data. However, due to the limited amounts of available data in our applications, we will manually select these parameters once for all experiments. We keep the problem of estimating them from validation data as a possible future work direction.

We will compare our algorithm to several others for non-rigid point matching and shape recovery. As representative examples of *point set registration* we have chosen the original Iterative Closest Point [10] (ICP), the Thin Plate Splines-Robust Point Matching (TPS-RPM) [25], the Coherent Point Drift (CPD) [71] and the recent Gaussian Mixture Model Registration (GMMREG) proposed in [53]. As examples of *graph matching* approaches, we have considered Spectral Matching (SM) [59] and Integer Projected Fixed Point (IPFP) [60], which can be combined, as well as Path Following Algorithm (PATH) [108] [1]. The results of the coarse alignment obtained by ATS-RGM and our previous Greedy-RGM version are virtually the same, therefore only the results for ATS-RGM are shown when comparing accuracy.

We ran all the algorithms on a 2.3 GHz 4-Core 64-bit machine with 8 GB RAM. Most of the aforementioned algorithms are implemented using a combination of Matlab and Mex-C++ functions. Similarly, we implemented the skeleton of our approach in Matlab and used C++ for the most time-consuming parts: the Gaussian Process routines and the Active Testing Search.

[1] All implementations were made available by respective authors, except of ICP, which was made available by a third party and verified to be a correct implementation of the algorithm.

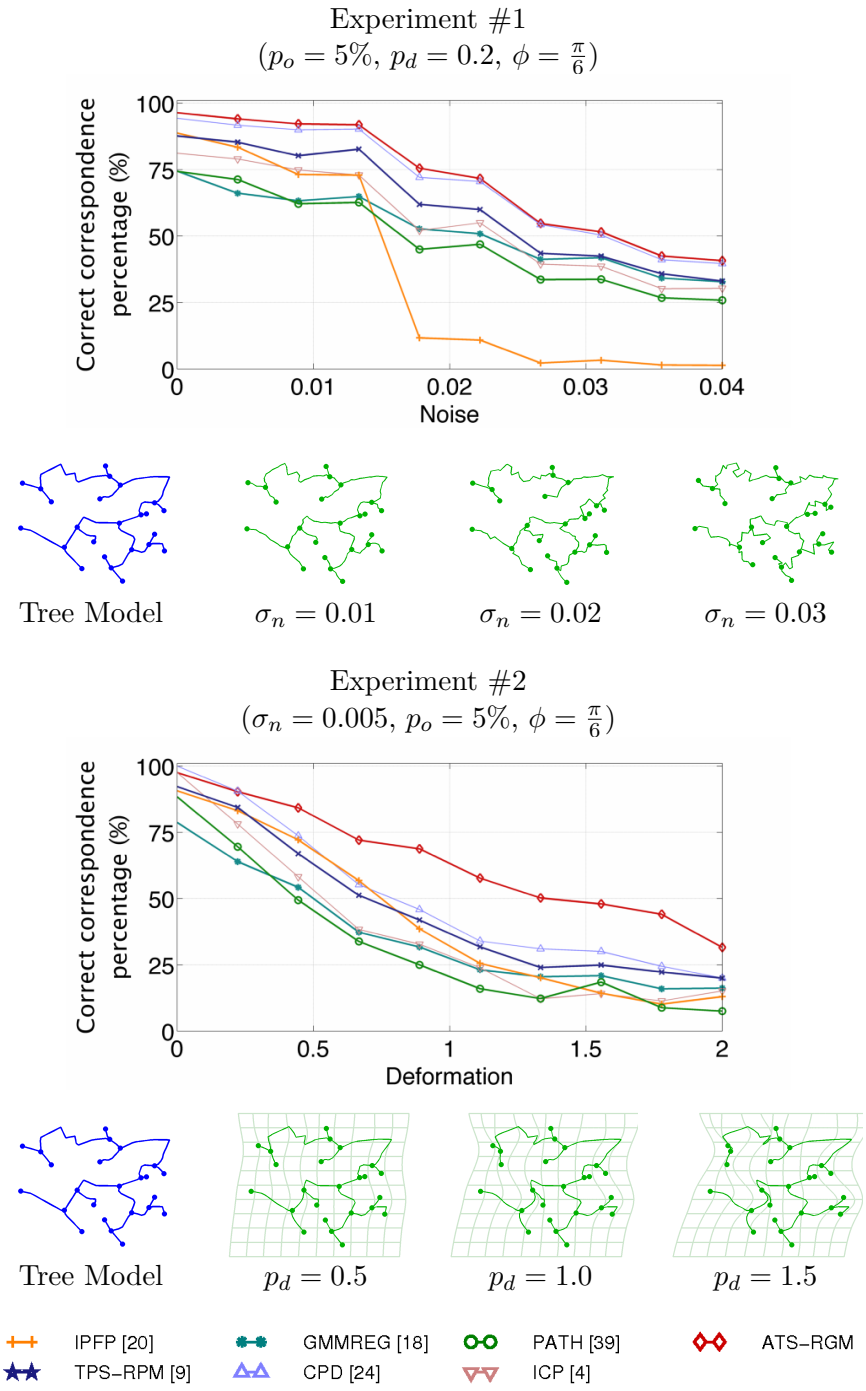


Figure 4.4: Quantitative evaluation on synthetic data. Performance tests of all competing methods in different configurations of noise (Exp. #1) and deformation (Exp. #2). Curves represent the median of the correct correspondences percentage achieved by each method. Below each result, we show the *tree model* used in the experiments (in blue) and corrupted illustrations of how each parameter affects the transformed graph (in green).

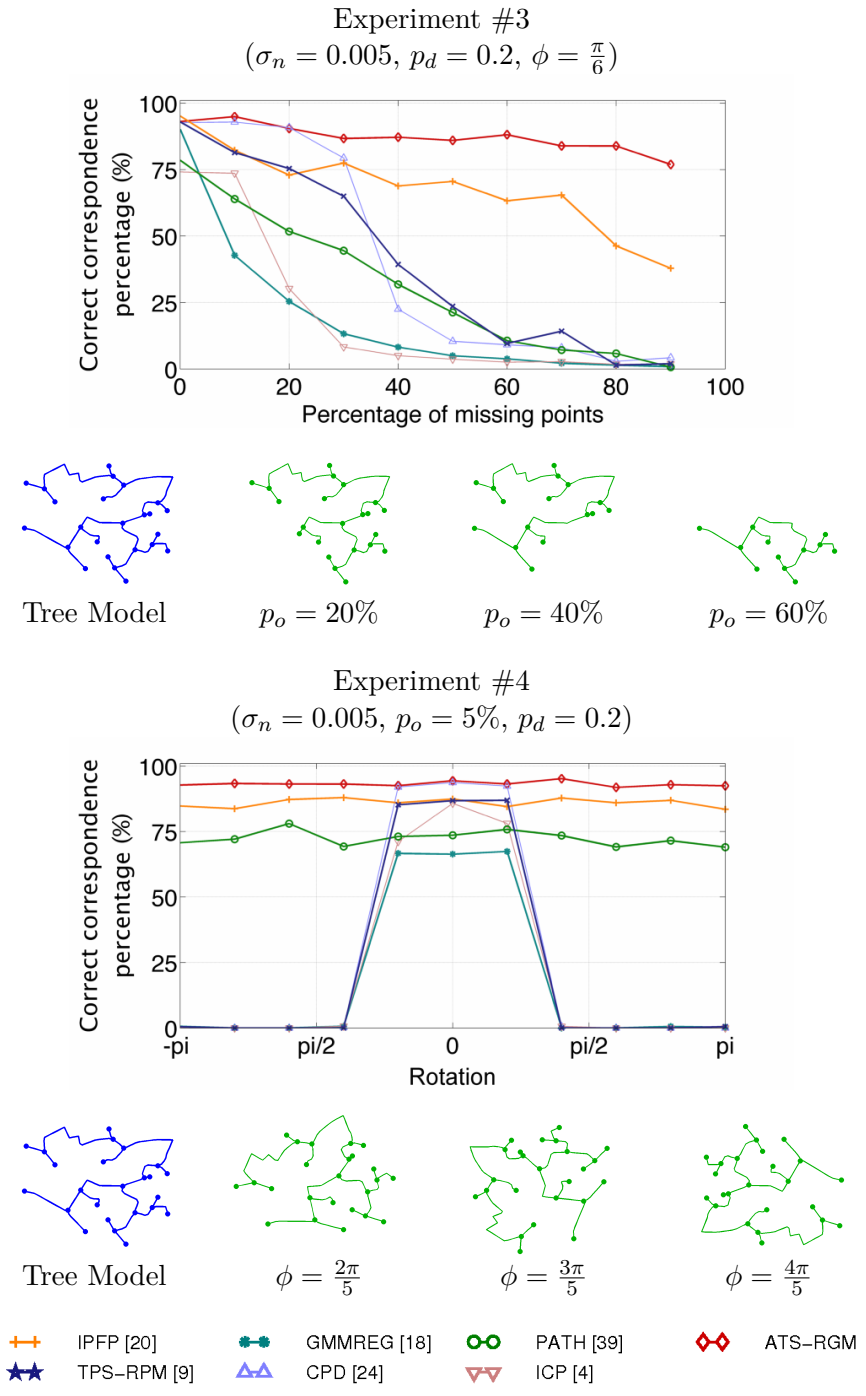


Figure 4.5: Quantitative evaluation on synthetic data. Performance tests of all competing methods in different configurations of outliers (Exp. #3) and rotation (Exp. #4). Curves represent the median of the correct correspondences percentage achieved by each method. Below each result, we show the *tree model* used in the experiments (in blue) and corrupted illustrations of how each parameter affects the transformed graph (in green).

4.5.1 Synthetic Experiments

To evaluate our approach against others, we generated pairs of trees composed of a *model tree* computed as the minimum spanning tree of $N = 50$ randomly selected 2D points in a 2×2 bounding box and *target tree* obtained by deforming it. More specifically, let $\mathbf{v}_i$ be the nodes of the model tree and $\mathbf{v}'_i$ those of the target tree. For all i from 1 to N, we write

$$\mathbf{v}'_i = \mathcal{R}(\phi) \begin{bmatrix} S_x & 0 \\ 0 & S_y \end{bmatrix} \mathbf{v}_i + \begin{bmatrix} T_x \\ T_y \end{bmatrix} + \mathcal{D}(p_d) + \xi(\sigma_n), \tag{4.10}$$

where the deformation includes the following components.

- Rotation, scaling, and translation; The target model is rotated by an angle ϕ, translated by T_x and T_y, and scaled by S_x and S_y
- Non-linear deformation: We add a non-linear warping $\mathcal{D}(p_d)$ defined as a linear combination of B-splines whose control points are uniformly distributed in the input space. Its magnitude is controlled by p_d, which specifies the amount of displacement of the B-spline coefficients.
- Noise: We perturb the node locations of the target graph by a zero mean Gaussian noise $\xi(\sigma_n)$, where σ_n is the standard deviation.
- Outliers: We produce outliers by randomly and independently removing a percentage p_o of the total number N of nodes from both the model and target trees. As a result, some branches appear in one tree and not the other.

We use the pairs of trees created in this manner as input to all algorithms. To ensure a fair comparison, we either set the required parameters to the values suggested in the corresponding papers, or manually modify them in case they did not provide good results. The exact values for these configuration parameters are as follows: [2].

- ICP: Does not require extra parameters.
- TPS-RPM: We set the initial temperature T_{init} to half the maximum Euclidean distance between the nodes of the model tree, and the final temperature to $T_{final} = 0.01 \cdot T_{init}$. The remaining parameters are set to $\lambda_1^{init} = 1$ and $\lambda_2^{init} = 0.01$, as suggested by the authors.

[2] Note that for ease of reference we are keeping the same parameter notation here as in the original papers. While some of these parameters are also used in our algorithm (i.e. : w, T, σ_d) their meaning are not necessarily the same.

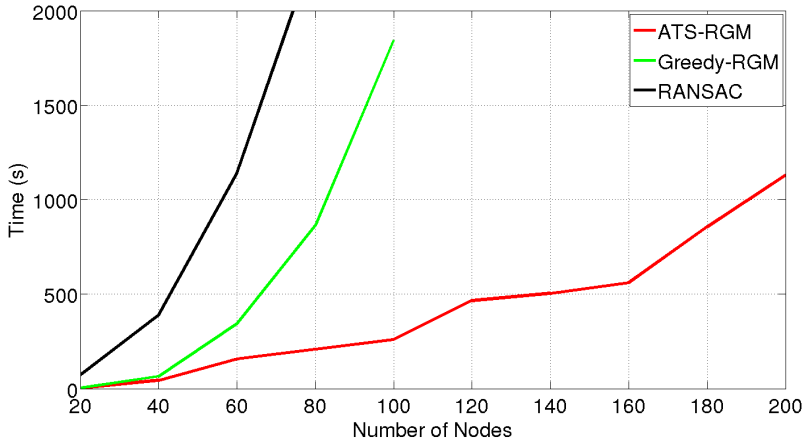


Figure 4.6: Computational Cost. Processing time required by RANSAC, Greedy-RGM and ATS-RGM as a function of the number of nodes. We computed the median of 20 experiments for each of the methods.

- GMMREG: We set the maximum number of iterations to 10000 with $\alpha = 1$ and $\beta = 2$.
- CPD: We use a non-rigid configuration of the algorithm for all the experiments. We set $\lambda = 3$, $\beta = 3$ and $outliers = 0.2$.
- SM and IPFP: We build the affinity matrix using the description provided in [59]. As we do not have appearance information, we set the label affinity term $M(a, a)$ to zero, making the matching score depend solely on the pairwise geometric information. The pairwise affinity is set to $M(a, b) = 4.5 - \frac{(d_{ij} - d_{i'j'})^2}{2\sigma_d^2}$ if $|d_{ij} - d_{i'j'}| < 3\sigma_d$ and zero otherwise.
- PATH: We build the graphs as suggested by the authors for the application of "alignment of vessel images". We connect each node $\mathbf{v}_i$ to all points $\mathbf{v}_j$ within a distance r, and each edge is assigned a weight $w_{i,j} = \exp(-||\mathbf{v}_i - \mathbf{v}_j||_2), \forall \mathbf{v}_i, \mathbf{v}_j \in \mathbf{V}, ||\mathbf{v}_i - \mathbf{v}_j||_2 < r$. This is done for all nodes of both model and target trees. In our case we set $r = 0.3 \cdot d_{max}$, where d_{max} is the maximum Euclidean distance between the points of the model tree.

Performance Evaluation

We tested all algorithms for robustness to rotation, deformation, and topology changes by varying the geometric deformation parameters of Eq. (4.10) as well as the percentage of missing nodes p_0. We performed four different independent experiments. For each we generated 50 pairs of model and target

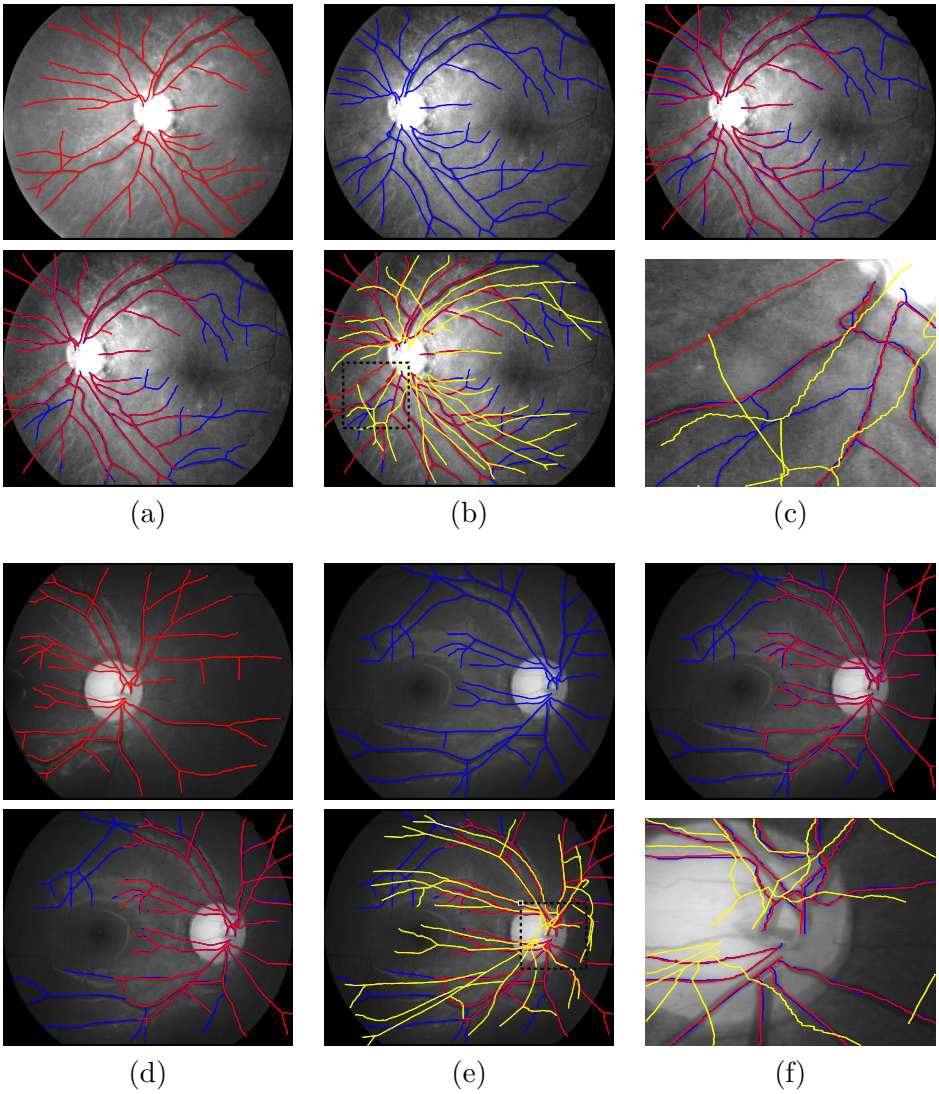


Figure 4.7: Retinal fundus images used in [29]. Each row contains a single experiment. **(a,b)** Two images of the same retina taken from different viewpoints, with the vascular trees overlaid in red and blue. **(c)** The first tree is overlaid in red over the second image after non-linear transformation, which corresponds to the output of the Greedy-RGM coarse alignment. **(d)** Final result of our non-rigid registration: the graph from the first image is overlaid in red over the second image. **(e)** Our result is superposed with the Coherent Point Drift alignment, in yellow. **(f)** Detail of the rectangle in (e). Our algorithm behaves well on this dataset, while CPD fails to recover the correct shape because there are too many non-corresponding branches. Best viewed in color.

trees using the same set of parameters and a fixed small change in scale and translation. In *Experiment #1* we evaluated the influence of noise on the points 2D locations by sweeping the range $\sigma_n \in [0, 0.04]$ and setting $p_o = 5\%$, $p_d = 0.2$ and $\phi = \frac{\pi}{6}$. In *Experiment #2* we tested the behavior of the algorithms against increasing levels of non-linear deformation, varying

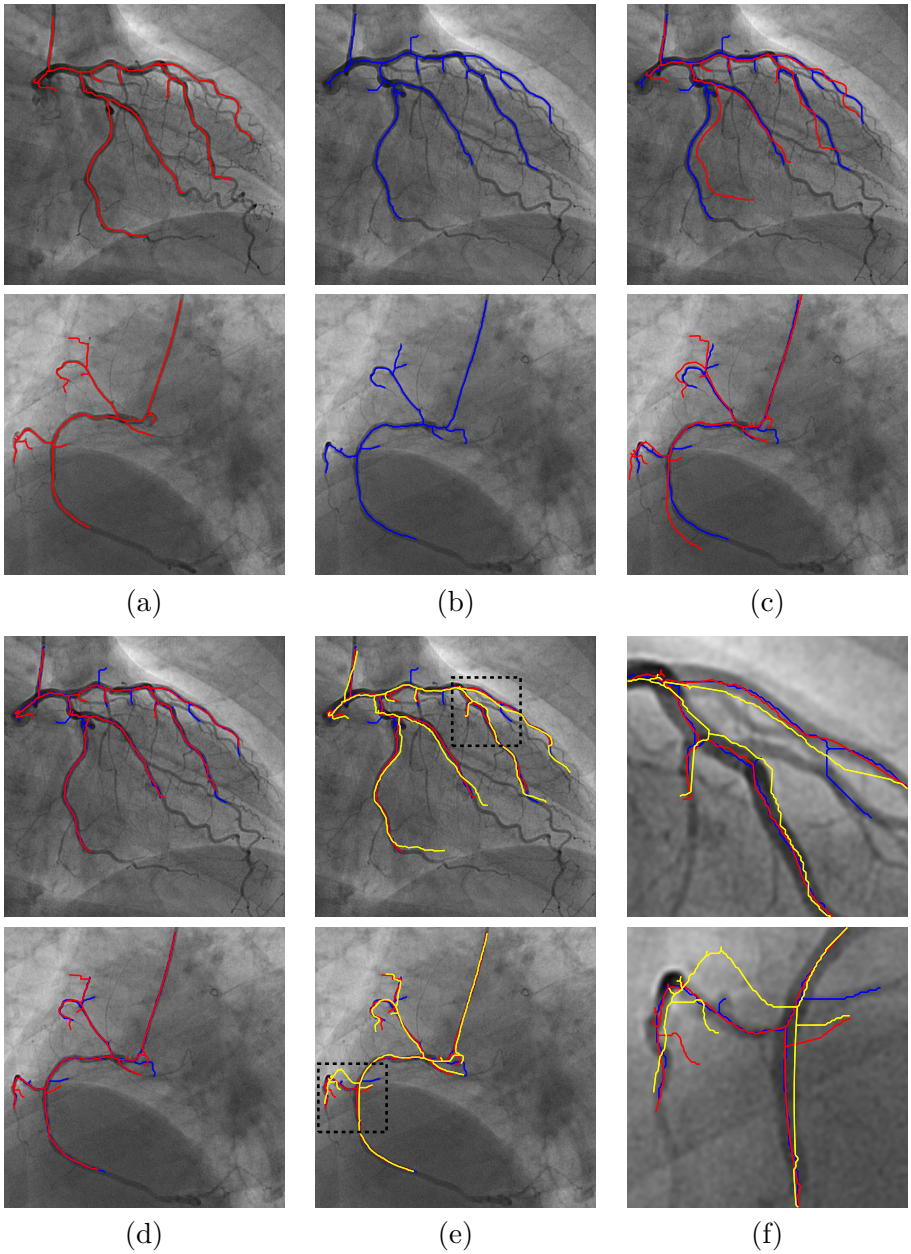


Figure 4.8: Angiography images from a beating heart. (a) Two different images with extracted vascular trees overlaid in red. **(b)** Two other images taken later in the heart cycle with extracted vascular trees overlaid in blue. **(c)** The original red trees are shown after the non-linear coarse alignment of the tree nodes, obtained using our Greedy-RGM. **(d)** The resulting warped trees are overlaid in red after non-linear registration. Note that the trees —in particular in the first example— have distinctly different topologies, which affects our algorithm very little. **(e)** Comparison with the result obtained using non-linear Coherent Point Drift, in yellow. **(f)** A zoom of a region of interest. Using the graph intrinsic geometry grants us robustness against vessel bendings and outliers, achieving a better registration of the two shapes.

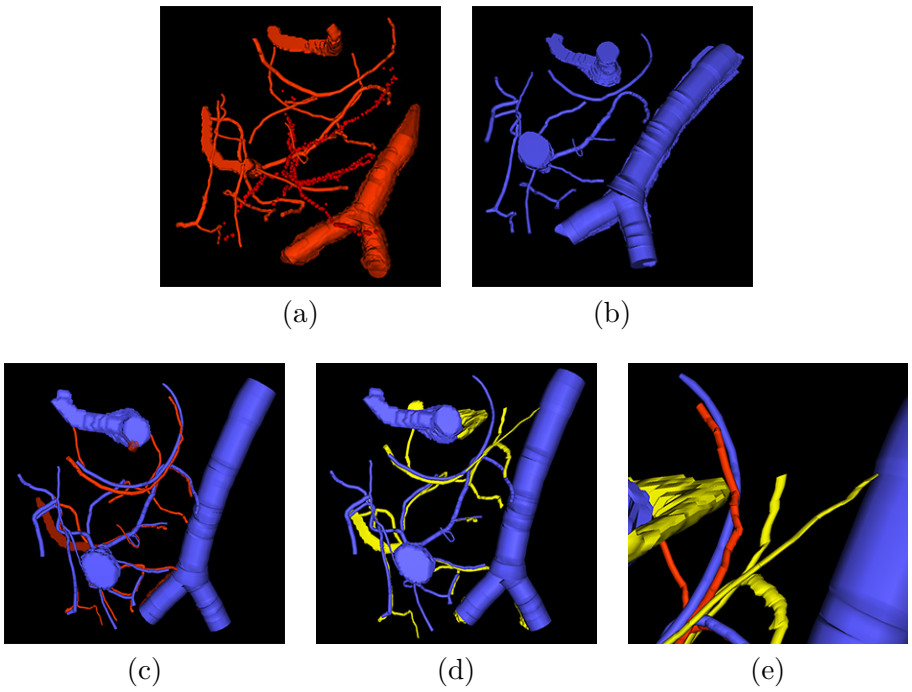


Figure 4.9: Blood vessels in brain tissue. **(a)** Segmented two photon microscopy data. **(b)** Segmented bright-field optical microscopy data. **(c)** Registration of structures using Active Testing Search, in red. **(d)** Alignment using CPD, in yellow. **(e)** A view in detail at the results of both ATS-RGM and CPD. Best viewed in color.

the deformation parameter within the range $p_d \in [0, 2]$, and setting the rest of parameters to constant values $\sigma_n = 0.005$, $p_o = 5\%$, and $\phi = \frac{\pi}{6}$. Similarly, in *Experiment #3*, we assessed the robustness of the algorithms to the presence of outliers by randomly deleting a percentage $p_o \in [0\%, 90\%]$ of nodes in both trees and setting $\sigma_n = 0.005$, $p_d = 0.2$ and $\phi = \frac{\pi}{6}$. Finally, in *Experiment #4* we tested the invariance of all the methods to initial conditions by changing the orientation of the target within $\phi \in [-\pi, \pi]$ and fixing the rest of the parameters to $\sigma_n = 0.005$, $p_o = 5\%$ and $p_d = 0.2$. To give significance to the magnitudes of the experimental parameters we consider, the images below each performance curve of Figures 4.4 and 4.5 show different samples of the same model graph and different target graphs generated by varying the levels of noise, deformation, outliers and rotation.

For each experimental parameter setting experiments and each algorithm, we computed the average percentage of correct matches over our 50 tree pairs and plot the results in Figures 4.4 and 4.5. Under favorable conditions, that is, relatively small graphs with less than 50 nodes, uncorrupted data and purely affine transformation, all methods exhibit similar performance. However, when we progressively introduce artifacts the differences become

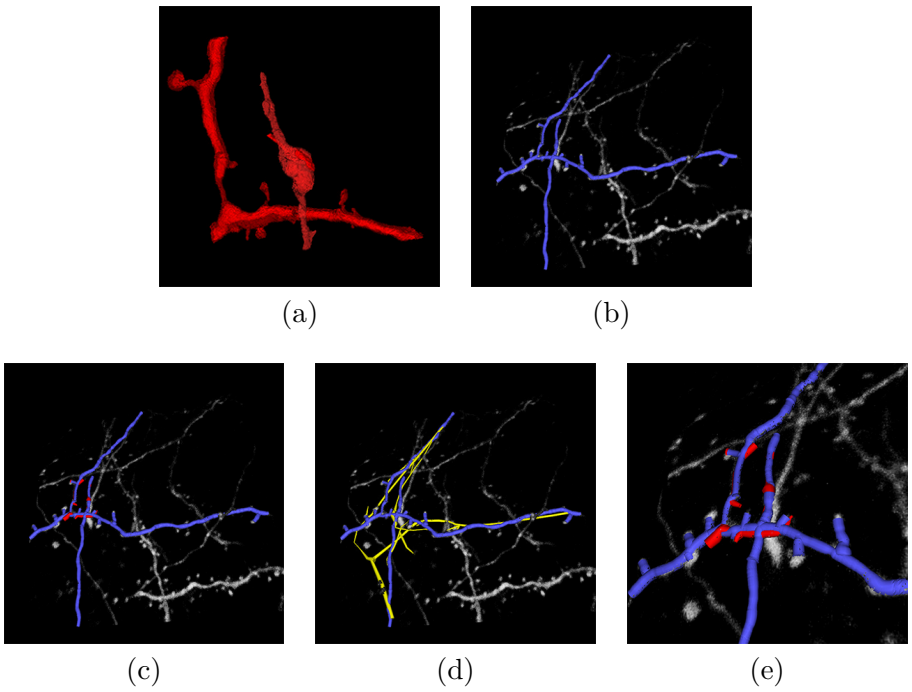


Figure 4.10: Light and electron microscopy neuronal trees. (a) Graph structure extracted from the electron microscopy image stack, in red. **(b)** Segmented light microscope neurons in blue. **(c)** After the non-linear registration process using ATS-RGM, the EM segmented neuron is deformed and aligned over the LM extracted neuron. **(d)** Registration using CPD, in yellow, which falls into a local minimum. **(e)** A zoom over the region where the EM stack has been extracted. The two neurons have been completely aligned. Best viewed in color.

clear. For instance, it can be seen that CPD deals poorly when there are missing parts or when the initial rotation is above 70 degrees, as stated in their paper. Similarly, the rigid ICP can only find local solutions, even when dealing with much smaller initial rotation angles. Graph methods (PATH, SM+IPFP) are able to find global solutions as they are invariant to initial conditions by construction. However, as shown in Fig. 4.4, they are very sensitive to modifications in the topology of the graph. TPS-RPM and GMMREG underperform our approach for each of the tests. In short, each one of the competing methods can address some of the difficulties, however only ours can handle all of them.

The results of Figures 4.4 and 4.5 also indicate the suitability of using our approach in sub-graph matching. The robustness of our approach stems from the randomized search strategy, that allows searching for a global minimum and makes the algorithm insensitive to initial rotations. This is true for both Greedy and Active Testing Search. On the other hand, the non-rigid transformation based on Gaussian Process regression provides robustness to

large amounts of noise and deformation.

Computational Cost

Finally, we compared the computational cost of the two versions of our algorithm and RANSAC [37]. We generated a new set of experiments in which the new model and target tree pairs are as before except for the fact that we varied the number of nodes from 20 to 200 and performed an affine transformation plus random noise of small amplitude. As illustrated in Fig. 4.6, the computation time grows much faster for both Greedy-RGM and RANSAC than for ATS-RGM, which remains manageable. Both versions of our algorithm (ATS-RGM and Greedy-RGM) yield similar performance, only differing in the time consumed to reach the global solution. Note that, since the transformations are almost affine, the absolute run-time value is lower than some of the real experiments we consider in the next section.

Note that the purely local methods, such as CPD and ICP, tend to be much faster and can deal with thousands of points in reasonable amounts of time. Yet, as we have seen in the fourth experiment of Fig. 4.4, these algorithms require accurate initializations. On the other hand, the graph matching methods, such as SM + IPFP or PATH, treat the problem as an Integer Quadratic Program (IQP) and hence are limited by the construction of the *pairwise score* matrix whose size grows quadratically with the number of nodes.

4.5.2 Real Data Experiments

We next present several examples of results obtained by ATS-RGM and Greedy-RGM on real biomedical datasets. The graphs were extracted semi-automatically using a plugin [8] for the Fiji platform [84].

To evaluate the accuracy of the different methods in the absence of ground truth, we assigned each node in the deformed graphs —overlaid in red for our method and yellow for CPD on Figs. 5.14 and 4.8 (d,e,f)— to its assumed match in the target graph overlaid in blue. To this end, we use the Hungarian algorithm to find it by taking the Euclidean distance as the cost to minimize, while rejecting outlier branches by setting a distance threshold defined *ad hoc* for each of the datasets. We use this error measure since there is no true

Dataset		ATS-RGM	Greedy-RGM	CPD [71]	ICP [10]	TPS-RPM [25]	GMMREG [53]
Retina I	Error (pix)	**2.68**	**2.68**	20.67*	20.04*	19.32*	21.30*
(Fig. 4.7 Top)	Time (s)	1293.3 + 406.4	5998.9 + 336.8	580.2	24.3	4236.8	139.7
Retina II	Error (pix)	2.94	**2.51**	20.45*	20.46*	17.79*	20.84*
(Fig. 4.7 Bot.)	Time (s)	280.0 + 301.4	16353.1 + 261.2	468.5	67.9	5791.1	147.1
Angio I	Error (pix)	1.16	**1.05**	2.95	9.77*	2.92	4.21
(Fig. 4.8 Top)	Time (s)	307.8 + 129.4	1240.9 + 162.8	144.3	8.1	726.7	31.9
Angio II	Error (pix)	**1.57**	1.81	3.42	4.84*	3.21	6.56
(Fig. 4.8 Bot.)	Time (s)	167.9 + 77.2	112.0 + 95.4	68.8	5.0	327.0	21.1
Brain vessels	Error (vox)	4.38	4.89	**4.19**	7.23	6.67	12.71
(Fig. 5.17)	Time (s)	593.7 + 55.5	15029.1 + 19.9	37.1	30.9	334.8	31.2
Neuronal	Error (μm)	**0.05**	0.07	0.25*	0.27*	0.20*	0.46*
(Fig. 4.10)	Time (s)	42.4 + 15.8	116.1 + 18.2	22.2	28.2	28.5	22.4

Table 4.1: Error: Geometric error on real datasets for the proposed approach and other state of the art methods. Failed experiments (producing incorrect alignment) are marked with an *, see Fig. 5.14(d) or Fig. 4.10(d) for examples. **Elapsed Time:** Processing time for each method, in seconds. For ATS-RGM and Greedy-RGM, we distinguish the times required for coarse and fine alignment.

correspondence between the sampled points along the edges of the graphs. This error measure gives an idea of the quality of the alignment.

In Table 4.1, we show these errors and the corresponding computation times. For ATS-RGM and Greedy-RGM, we distinguish the times required for coarse and fine alignment. We have not provided the error for IPFP and PATH because these methods only give correspondence hypotheses, and are unable to define a valid transformation without an outlier rejection step.

In Fig. 5.14 we show registration results for retinal fundus vascular graphs that are deformed from one image to the next because the camera is looking from different viewpoints. This produces distortions of the curved retinal surface's projection, that are well modeled by an affine transform. Thus, there is very little non-linearity in the deformation and these results are similar to those of [29], even though the trees only partially overlap. However, as the amount of spurious branches is quite large, CPD fails to recover the correct shape. In contrast, our approach can naturally handle such artifacts.

In the 2D X-ray angiography images of Fig. 4.8 the non-linearities of the transformation are much more apparent. As shown in the zoomed-in area, our algorithm nevertheless does a good job of recovering this more complex deformation and aligning the trees. Again, we assessed the performance of the CPD on these images and observed that it could not retrieve a correct solution unless a relatively accurate initialization was provided. Even when we supplied with our coarse transformation estimate, CPD could only deal with small non-linearities.

Next, we register two 3D datasets. A blood vessels network in brain tissue is shown in Fig. 5.17. One of the 3D image stacks is acquired using a two-photon microscope and the other using bright-field microscopy after excision and fixation. As the resulting segmentations are partially aligned, the experiment's difficulty consists in identifying the non-linearities of the deformation. Our algorithm clearly outperforms other state-of-art methods and provides results similar to the CPD. In the zoomed image of Fig. 5.17(e) it is possible to appreciate it. Although CPD works well when the initial conditions are favorable, it completely misaligns one of the branches while our result respects the topology.

Finally, we register the 3D neuronal stacks extracted from the brain tissue of Fig. 1.1 using light (LM) and electron (EM) microscopy, where the EM block is a small section of the LM volume and has been non-linearly deformed due to the extracting process. The intended application is to automatically localize the EM volume in the corresponding part of the LM volume. Even though the two images look extremely different, our algorithm returns a valid deformation as shown in Fig. 4.10. No other method was able to recover the correct alignment.

Chapter 5

Graph Matching using Monte Carlo Tree Search

In this chapter, we present an efficient matching method for generalized geometric graphs. Such graphs consist of vertices in space connected by curves and can represent many real world structures such as road networks in remote sensing, or vessel networks in medical imaging. Graph matching can be used for very fast and possibly multimodal registration of images of these structures. Other applications include recognition of graph-like patterns or camera-based localization for UAVs. We formulate the matching problem as a single player game solved using Monte Carlo Tree Search, which automatically balances exploring new possible matches and extending existing matches. Our method can handle partial matches, topological differences, geometrical distortion, does not use appearance information and does not require an initial alignment. Moreover, our method is very efficient – it can match graphs with thousands of nodes, which is an order of magnitude better than the best competing method, and the matching only takes a few seconds.

This chapter is strongly based on the paper [76], which is an extended publication of the work developed on the papers [74, 75].

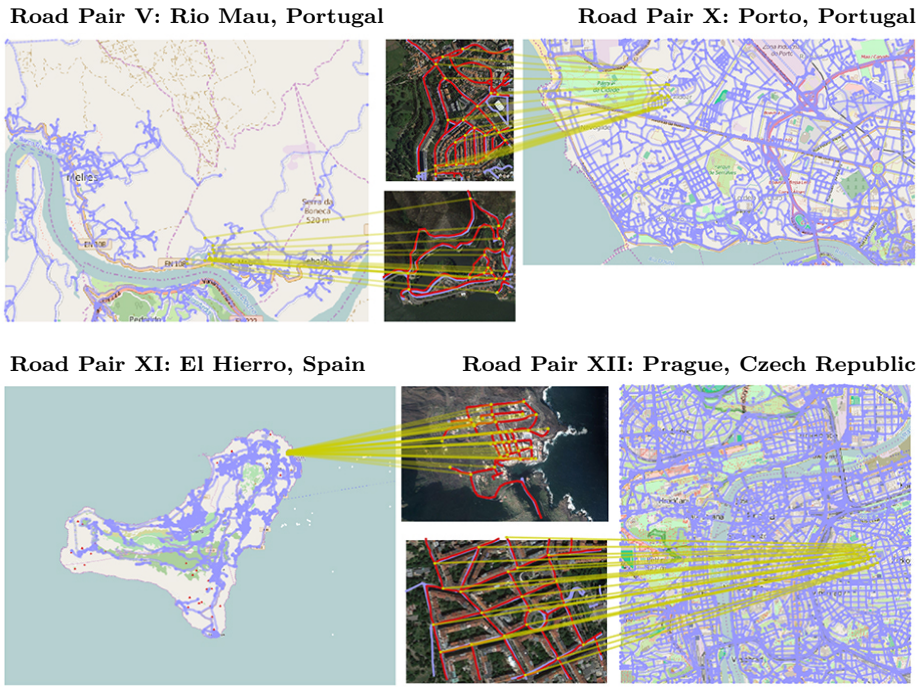


Figure 5.1: Localizing a small satellite image within a map. Yellow lines link matched nodes of the road networks extracted from a satellite image (red) and from a map (blue). Transformed version of the map network (blue) after matching is shown overlaid on the satellite image.

5.1 Problem definition

Let us consider a graph $\mathcal{G}^A = (\mathbf{V}^A, \mathbf{E}^A)$ where the vertices $\mathbf{V}^A$ are points in $\mathbb{R}^D$, and the edges $\mathbf{E}^A \subseteq \mathbf{V}^A \times \mathbf{V}^A$ are associated with curves connecting the two incident vertices. This is a generalization of a *geometric graph* [35]. Each edge $\mathbf{e} \in \mathbf{E}^A$ is described by a continuous function $\zeta_\mathbf{e} \colon I \to \mathbb{R}^D$, where $I = [0, 1]$ is the unit interval, so that $\mathbf{e} = (\zeta_\mathbf{e}(0), \zeta_\mathbf{e}(1))$. The curve is an image of this function, $\zeta_\mathbf{e}(I) = \{\zeta_\mathbf{e}(t); t \in I\}$ and has length $l(\mathbf{e}) = \int_0^1 \|\dot{\zeta}_\mathbf{e}(t)\| \, \mathrm{d}t$. We choose a constant speed parameterization, implying $\|\dot{\zeta}_\mathbf{e}(t)\| = l(\mathbf{e})$ for all $t \in I$. We assume that for each edge $(\mathbf{u}, \mathbf{v})$ in $\mathbf{E}^A$, the graph contains also its reverse $(\mathbf{v}, \mathbf{u})$, with the same curve reversed. The total length of all edges is denoted $l(\mathbf{E}^A)$ and the number of vertices is $|\mathbf{V}^A|$.

To handle segmentation errors, we want to allow for some vertices and edges to be present only in one of the graphs, i.e. to perform *partial matching*. Moreover, two or more edges in one graph may correspond to one longer edge in the other graph. To handle these topological differences, we define *superedges* [88] as sequences of up to K consecutive edges, $\mathbf{s} = (\mathbf{e}_1, \ldots, \mathbf{e}_{K_\mathbf{s}})$,

$K_{\mathbf{s}} \leq K$. We mostly use $K = 2$ or 3 and give examples in Fig. 5.2. Matching superedges instead of edges effectively allows some nodes to be skipped in one of the graphs. The set of all *superedges* in the graph $\mathcal{G}^A$ will be denoted $\mathbf{S}^A$. The length of a superedge is the sum of the lengths of the constituent edges, $l(\mathbf{s}) = \sum_i l(\mathbf{e}_i)$. To each superedge, we associate a curve which is a concatenation of the curves associated with individual edges and is described by a function $\zeta_{\mathbf{s}} \colon I \to \mathbb{R}^D$.

The matching between two geometrical graphs $\mathcal{G}^A$ and $\mathcal{G}^B = (\mathbf{V}^B, \mathbf{E}^B)$ can be described by a set of superedge pairs $\mathbf{M}^{\mathbf{S}} \subseteq \mathbf{S}^A \times \mathbf{S}^B$. The superedge matching $\mathbf{M}^{\mathbf{S}}$ determines uniquely a vertex matching $\mathbf{M}^{\mathbf{V}} \subseteq \mathbf{V}^A \times \mathbf{V}^B$, such that matched superedge end-vertices are matched in $\mathbf{M}^{\mathbf{V}}$ but no other vertices are matched. A matching $\mathbf{M}^{\mathbf{S}}$ is *feasible* only

i) if no matched superedges overlap (contain the same edges),

ii) if it is *consistent* with some vertex matching $\mathbf{M}^{\mathbf{V}}$.

The matching is said to be *consistent* with a geometrical transformation $T \colon \mathbb{R}^D \to \mathbb{R}^D$, iff the vertices and superedges are transformed version of each other,

$$(\mathbf{u}, \mathbf{v}) \in \mathbf{M}^{\mathbf{V}} \implies \mathbf{v} = T(\mathbf{u}), \tag{5.1}$$

$$(\mathbf{r}, \mathbf{s}) \in \mathbf{M}^{\mathbf{S}} \implies \zeta_{\mathbf{s}}(I) = (T \circ \zeta_{\mathbf{r}})(I). \tag{5.2}$$

We want the matching to be as large as possible, so we measure its quality using the total length of the matched superedges, averaged over the two graphs

$$l(\mathbf{M}^{\mathbf{S}}) = \sum_{(\mathbf{s}_k^A, \mathbf{s}_l^B) \in \mathbf{M}^{\mathbf{S}}} \frac{l(\mathbf{s}_k^A) + l(\mathbf{s}_l^B)}{2}. \tag{5.3}$$

Since we also want to discourage skipping nodes that exist in both graphs, we will reward the number of matched nodes $|\mathbf{M}^{\mathbf{V}}|$. The combined objective function is

$$Q(\mathbf{M}^{\mathbf{V}}, \mathbf{M}^{\mathbf{S}}) = l(\mathbf{M}^{\mathbf{S}}) + \kappa \, \overline{l(\mathbf{S}^A, \mathbf{S}^B)} \, |\mathbf{M}^{\mathbf{V}}|, \tag{5.4}$$

where $\overline{l(\mathbf{S}^A, \mathbf{S}^B)}$ is the average length of the superedges of both graphs, and κ is a user-defined parameter (in our experiments $\kappa = 0.8$). The task can be now defined as follows:

Problem 1. Find a feasible matching $\mathbf{M}^{\mathbf{S}}$ between the superedges of the graphs $\mathcal{G}^A$ and $\mathcal{G}^B$, which maximizes the criterion Q (5.4), and is consistent with some geometrical transformation T from a given class of allowed transformations $\mathscr{T}$.

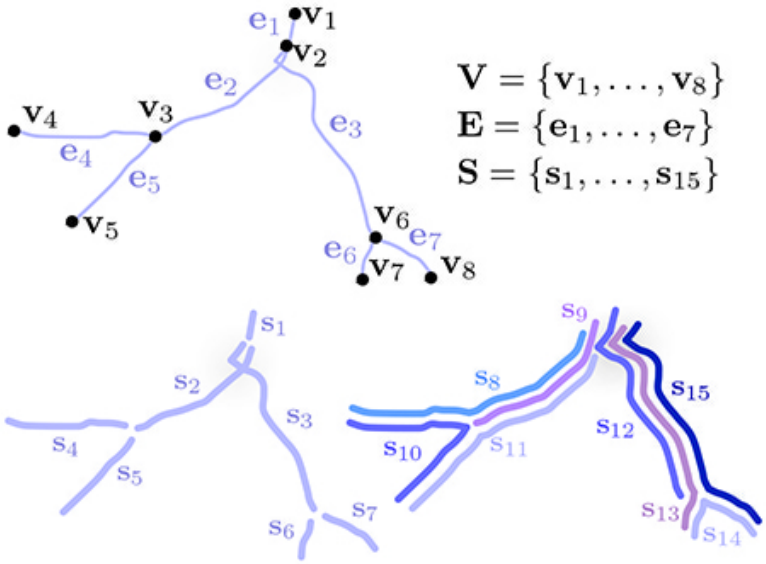


Figure 5.2: Example of a geometrical graph (**top**) and its superedges of length one (**bottom left**) and two (**bottom right**).

Note that the matching may be partial, i.e. not all edges and vertices are necessarily matched. Our choice of the class of allowed transformations $\mathscr{T}$ will be defined in Section 5.2.

5.1.1 Proposed approach

The key idea of our approach is to formulate the combinatorial Problem 1 as a single player game with the following rules:

1. Start with an empty matching $\mathbf{M}_0^{\mathbf{S}} = \emptyset$.
2. The game consists of a sequence of valid moves. In each move, a superedge pair $(\mathbf{r}, \mathbf{s}) \in \mathbf{S}^A \times \mathbf{S}^B$ is added to $\mathbf{M}^{\mathbf{S}}$, i.e. $\mathbf{M}_{t+1}^{\mathbf{S}} = \mathbf{M}_t^{\mathbf{S}} \cup (\mathbf{r}, \mathbf{s})$. A move is *valid*, if the resulting matching $\mathbf{M}_{t+1}^{\mathbf{S}}$ is feasible and consistent with some geometrical transformation $T \in \mathscr{T}$, where T can vary in time.
3. A move consisting of adding a pair $(\mathbf{r}, \mathbf{s})$ can be taken only if $(\mathbf{r}, \mathbf{s})$ is adjacent to $\mathbf{M}_t^{\mathbf{S}}$. The move is called *adjacent*, if there is a superedge pair $(\mathbf{r}', \mathbf{s}') \in \mathbf{M}_t^{\mathbf{S}}$, such that the first vertex of $\mathbf{r}$ and $\mathbf{s}$ is equal to the last vertex of $\mathbf{r}'$ and $\mathbf{s}'$, respectively.
4. The goal of the game is to find a sequence of valid moves such that the final matching maximizes the criterion Q.

The rules require the matching to be built incrementally (rule 2) and contiguously (rule 3) and thus limiting the number of possibilities to consider in each step. See Fig. 5.3 for illustration of possible moves and Section 5.3.1 for handling disconnected graphs.

To prune the search even further, we will use pre-computed path descriptors (Section 5.4). Only *compatible* superedge pairs $(\mathbf{r}, \mathbf{s})$ will be considered as possible moves. To summarize, $(\mathbf{r}, \mathbf{s})$ is a *possible move*, if the following conditions are fulfilled:

a) $(\mathbf{r}, \mathbf{s})$ is adjacent to existing matches $\mathbf{M}_t^{\mathbf{S}}$,

b) $(\mathbf{r}, \mathbf{s})$ does not conflict with already matched superedges (i.e. no edge overlap or conflicting vertex assignment),

c) $\mathbf{r}$ and $\mathbf{s}$ have compatible descriptors (Section 5.4),

d) $(\mathbf{r}, \mathbf{s})$ together with previous matches $\mathbf{M}_t^{\mathbf{S}}$ are consistent with the transformation model (Section 5.2).

The conditions are tested in this order so that the geometrical consistency, which is the most time consuming, is tested last.

The *search space* can be described as a directed acyclic graph, with nodes being the partial matches $\mathbf{M}^{\mathbf{S}}$ and edges corresponding to possible moves. Each node has an associated reward Q. Starting in the root $\mathbf{M}_0^{\mathbf{S}} = \emptyset$, we find the maximum reward using the Monte Carlo Tree Search method (Section 5.3).

The superedges are considered in a *default order*: first the superedges with the least number of constituent edges; in case of equality, longer superedges are considered first. For pairs of superedges, a maximum number of edges and a sum of the lengths is considered. This heuristic leads to quickly constraining future matching choices while not skipping vertices unless necessary.

5.2 Transformation model

Any geometrical transformation model can be used, as long as it allows efficient test of consistency with a set of points. We represent the curves by a set of regularly sampled points with a sampling step Δ, so no special test

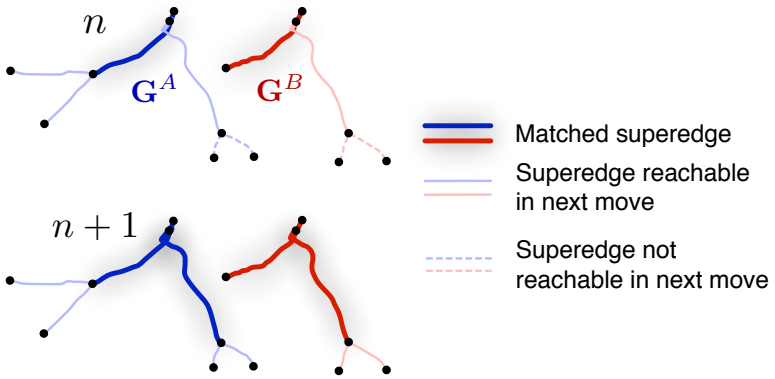


Figure 5.3: Example for a possible next move (**top** — initial state, **bottom** — next state) for a simplified case of superedges with length one. The superedges reachable in each state are connected with already matched superedge pairs.

for curves is needed. Note that for the matching itself, the deformation does not need to be explicitly evaluated. We shall therefore define the model only implicitly, which is computationally advantageous.

In our applications, the scale is usually known, so we need to model mainly a rigid body transformation with a small nonlinear component due to a tissue deformation, optical distortion, measurement inaccuracies or perspective distortion. We are therefore modeling the transformation as bi-Lipschitz, which means that for any two points $\mathbf{x}, \mathbf{y} \in \mathbb{R}^D$, their relative distance after applying the transformation should not change by more than some small constant ε_T:

$$\frac{1}{1+\varepsilon_T} d(\mathbf{x},\mathbf{y}) \leq d(T(\mathbf{x}),T(\mathbf{y})) \leq (1+\varepsilon_T) d(\mathbf{x},\mathbf{y}), \tag{5.5}$$

where $d(\mathbf{x},\mathbf{y})$ is the Euclidean distance between $\mathbf{x}$ and $\mathbf{y}$.

To check condition d) in Section 5.1.1, i.e. whether a new superedge pair $(\mathbf{r},\mathbf{s})$ is geometrically compatible with already matched pairs $\mathbf{M}_t^{\mathbf{S}}$, we should take all pairs of points $(\mathbf{x},\mathbf{y})$ from all edges in $\mathbf{M}_t^{\mathbf{S}} \cup (\mathbf{r},\mathbf{s})$. However, this would be computationally very costly. Instead, we shall only test the yet unmatched end-vertices $\mathbf{u} \in \mathbf{V}^A$ and $\mathbf{v} \in \mathbf{V}^B$ of $\mathbf{r}$ and $\mathbf{s}$, respectively, with all other matched vertices so far:

$$\frac{1}{1+\varepsilon_T} d(\mathbf{u},\mathbf{p}) \leq d(\mathbf{v},\mathbf{q}) \leq (1+\varepsilon_T)\ d(\mathbf{u},\mathbf{p}) \quad \forall (\mathbf{p},\mathbf{q}) \in M^{\mathbf{V}}. \tag{5.6}$$

This approximate test is very quick and on our data it rejects a sufficient number of incorrect matches.

5.3 Monte Carlo tree search

To find the optimum matching $M^{\mathbf{S}}$, we will use an algorithm inspired by the Upper Confidence Bound on Trees (UCT), a variant of the Monte Carlo Tree Search (MCTS) [13].

A search tree is built incrementally. Each node ν stores the matched superedges $M^{\mathbf{S}}$ and vertices $M^{\mathbf{V}}$. It contains the node reward $Q_\nu = Q(M^{\mathbf{V}}, M^{\mathbf{S}})$ and the estimated maximum reward Q_ν^+ for the subtree rooted in ν; Q_ν^+ is calculated in the *simulation* step described below. For each node, we also calculate the *urgency*

$$\tilde{Q}_\nu = \frac{Q_\nu^+}{Q_{\text{norm}}} + \gamma\sqrt{\frac{2\log n}{n_\nu}}. \tag{5.7}$$

The second term is the *upper confidence bound* (UCB) [13] and it balances between exploration of yet unvisited branches and exploitation of known good branches; γ is a user-defined parameter (in our experiments $\gamma = 0.01$), Q_{norm} is an upper bound and a normalization factor for the reward Q_ν, i.e.

$$Q_{\text{norm}} = \frac{l(\mathbf{E}^A) + l(\mathbf{E}^B)}{2} + \kappa\,\overline{l(\mathbf{S}^A, \mathbf{S}^B)}\,\min(|\mathbf{V}^A|, |\mathbf{V}^B|),$$

n is the current iteration number, and n_ν is the number of times the node ν has been *selected* (see below).

A node can be *expanded* by performing a possible move and adding the resulting state as a child node of the current node. We keep track of the expandability (existence of unused possible moves) of each node.

The algorithm repeatedly performs the following four steps (Fig. 5.4), until the computational budget is exhausted or until no nodes can be expanded:

1. **Selection** — The *most urgent* expandable node is selected by a greedy top-down tree search. We start in the root and always select among the expandable children the one with the highest urgency $\tilde{Q}_\nu$. We return the expandable node with the highest $\tilde{Q}_\nu$ found.

2. **Expansion** — N_{exp} children of the selected node are added to the tree. The children nodes to be expanded are taken in the default order (Section 5.1.1). In our experiments $N_{\text{exp}} = 2$.

3. **Simulation** — We estimate the maximum possible reward Q_ν^+ of the subtree rooted in the selected node by greedily and recursively adding N_{sim} children in a depth-first manner, adding always the first node in the default ordering. The greedy approach is fast, so N_{sim} can be set to a large number (in our experiments $N_{\text{sim}} = 25$). We insert the added nodes into the search tree.

4. **Backpropagation** — We update Q_ν^+ in the nodes along the path from the current node back to the root, taking the maximum of the children values.

The algorithm stops after reaching a predefined match size N_{match}, maximum number of iterations N_{it} or maximum processing time $T_{\max}$. The result is a matching $(M^{\mathbf{V}}, M^{\mathbf{S}})$ between the graphs with the highest reward Q_ν.

When the matching is complete, we use a Gaussian process regression [66] to fit a smooth non-linear transformation T given the matched vertices. This step is fast and provides sufficiently good results on our data. For more demanding applications, the iterative procedure [89] should be used, which alternates between solving for the assignment between all points using the Hungarian algorithm [70] and fitting the Gaussian process model.

In Appendix B, we detail the implementation of the method, giving some extra information about the algorithm. In Appendix C, we give some details over the publicly distributed code of the algorithm.

5.3.1 Implementation details

Several additional fields are stored for each node besides $M^{\mathbf{S}}$, $M^{\mathbf{V}}$, Q_ν and Q_ν^+ :

- **Counter n_ν:** Number of times the node has been selected.
- **Skipped vertices**: a list of skipped vertices in the match. Skipped vertices are vertices which are part of superedges in $M^{\mathbf{S}}$ but which are not in $M^{\mathbf{V}}$. They are not considered for addition in the future, as they overlap with already matched superedges.
- **Expandable flag**: false if all elements in the subtree under the node (including itself) have been fully expanded and true otherwise.

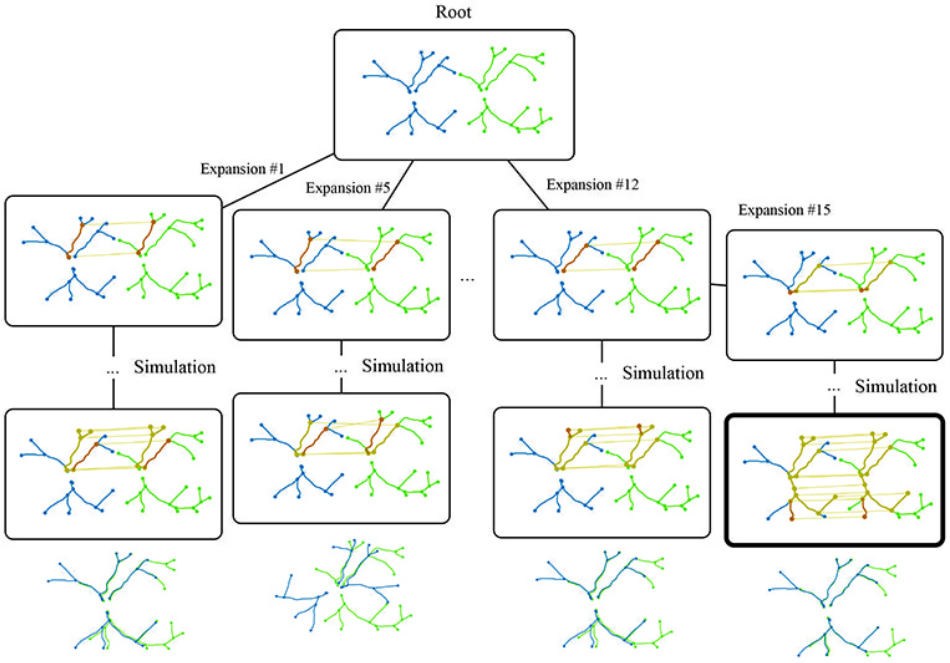


Figure 5.4: Example of the GMMC algorithm exploring the search tree for a simple pair of graphs. A node corresponding to a superedge pair is selected and expanded and then extended greedily in the simulation step. For each pair of graphs in blue and green, $M^{\mathbf{V}}$ is represented by yellow lines connecting vertices, matched superedges from $M^{\mathbf{S}}$ are shown in yellow and the newest match is shown in red. Below each simulation, we show the alignment such matching would produce. The optimal solution (highlighted on the tree) was found after 6 milliseconds in 44 iterations.

To allow exploring disconnected components of the graphs $\mathcal{G}^A$, $\mathcal{G}^B$, *virtual superedges* composed of a single straight edge are added between vertices closer than a distance d_c which belong to different graph components (for graphs normalized s.t. $\mathbf{V}^A, \mathbf{V}^B \in [-1, 1]^D$, we use $d_c = 0.15$). Only superedges of the same type (virtual and non-virtual) can be matched to each other.

If more severe topological differences are expected in a particular application, which cannot be handled by superedges and virtual superedges, it is possible to extend the set of possible moves even further and to allow with some small probability a match between any yet unmatched nodes, regardless of other constraints. This is in the spirit of Markov chain Monte Carlo (MCMC) methods and in theory allows to explore the whole space of possible matchings at the cost of much increased computational complexity. However, this turned out not to be necessary in any of our datasets.

Unlike in standard MCTS, the search space is a directed acyclic graph, not a tree, because several sequences of moves may lead to the same state.

To save the computational effort, the nodes representing the same state are shared. Each time a new state $M^{\mathbf{S}}$ is to be added to the search tree, we check a list of already encountered states. If the node already exists, it is shared. The test is fast using binary search, as the list is kept sorted by defining a topological ordering between matches.

In Appendix B, we further detail the implementation of the method, including several pseudo algorithms for each step of the approach.

5.3.2 Adding a child node

There is a special procedure for expanding the root node, since it would be wasteful to enumerate all possible superedge pairs. If the *root* is selected for expansion, a single superedge pair is added. We start with the first superedge in $\mathbf{S}^A$ (in the default order, i.e. the longest edge, see Section 5.1.1) and find the first geometrically compatible superedge in $\mathbf{S}^B$ (Section 5.4). Only superedge pairs with the same number of constituent edges are considered for addition. The lists of possible superedge pairs and endpoints are then created (as for all other nodes, see below). If the root needs to be expanded again, the search continues where it left off.

For a non-root node, the procedure of adding a child node is the same for both the expansion and simulation steps:

- We pick an unexplored superedge pair from the list of possible superedge pairs, sorted in default order. This pair will automatically satisfy conditions a)–c) in Section 5.1.1. We then check the geometric compatibility with previous matches (condition d)) using (5.6).
- The new node starts as a copy of its parent. We then add the newly matched superedge pair to $M^{\mathbf{S}}$ and the yet unmatched end-vertices to $M^{\mathbf{V}}$. The list of skipped vertices is updated. Infeasible matches are removed.
- We search in $\mathbf{S}^A$ and $\mathbf{S}^B$ for superedges incident with the newly matched vertices but with none of the already matched edges. Geometrically compatible superedge pairs (condition c)) are added to the list of possible moves. This can be done in linear time with respect to the number of adjacent edges.

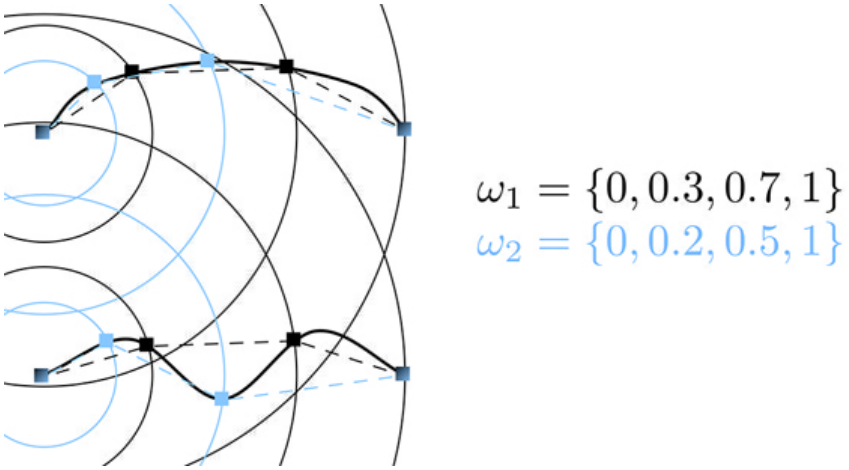


Figure 5.5: Simple example of the sampling for calculating the path descritptor $h_{\boldsymbol{\omega}}(\mathbf{s}_k)$ for a superedge $\mathbf{s}_k$

5.4 Path descriptors

The path descriptors characterize a curve by a fixed-length real vector, invariant to rigid body transformations, allowing efficient curve matching. An earlier version of the descriptors was used in [74].

Let us have a *sampling vector* $\boldsymbol{\omega} = (\omega_0, \cdots, \omega_{n_\omega+1})$ such that $0 = \omega_0 < \omega_1 < \ldots < \omega_{n_\omega} < \omega_{n_\omega+1} = 1$. For a given superedge $\mathbf{s}$ and its associated curve $\zeta_{\mathbf{s}}$, we shall define a scalar value

$$h_{\boldsymbol{\omega}}(\mathbf{s}) = \sum_{i=0}^{n_\omega} d\big(\zeta_{\mathbf{s}}(t_i), \zeta_{\mathbf{s}}(t_{i+1})\big), \tag{5.8}$$

where the vector $\mathbf{t} = (t_0, \ldots, t_{n_\omega+1})$ is chosen such that $d(\zeta_{\mathbf{s}}(0), \zeta_{\mathbf{s}}(t_i)) = \omega_i d(\zeta_{\mathbf{s}}(0), \zeta_{\mathbf{s}}(1))$. If there are more possible t_i, the smallest one is taken. In other words, we find the first intersection of the curve with concentric circles of relative radii given by ω_i, and calculate the total length of the line segments, connecting these points (see Fig. 5.5). A vector descriptor

$$\mathbf{h}_{\boldsymbol{\Omega}}(\mathbf{s}) = (h_{\boldsymbol{\omega}_1}, \cdots, h_{\boldsymbol{\omega}_{|\boldsymbol{\Omega}|}}) \tag{5.9}$$

is created by evaluating (5.8) for a set $\boldsymbol{\Omega} = (\boldsymbol{\omega}_1, \ldots, \boldsymbol{\omega}_{|\boldsymbol{\Omega}|})$ of randomly generated sampling vectors $\boldsymbol{\omega}$. The number of sampling vectors $|\boldsymbol{\Omega}|$ and their length n_ω are user defined fixed parameters. In our experiments $n_\omega = 5, |\boldsymbol{\Omega}| = 50$.

The descriptors are precalculated for all superedges. To test the compatibility of two superedges $(\mathbf{r}, \mathbf{s}) \in \mathbf{S}^A \times \mathbf{S}^B$ (condition c) in Section 5.1.1), we

take inspiration from the Lipschitz condition (5.5). We shall consider the two superedges to be compatible iff

$$\frac{1}{1+\varepsilon_h} h_{\boldsymbol{\omega}_i}(\mathbf{r}) \leq h_{\boldsymbol{\omega}_i}(\mathbf{s}) \leq (1+\varepsilon_h) h_{\boldsymbol{\omega}_i}(\mathbf{r}), \quad \forall i \tag{5.10}$$

for all sampling vectors $\boldsymbol{\omega}_i$ from $\boldsymbol{\Omega}$. We use $\varepsilon_h = 3\varepsilon_T$, which has been confirmed to be a reasonable choice in an experiment in Section 5.5.3.

5.5 Experiments

We evaluate the performance of our proposed method GMMC on synthetic and real data and compare its performance against that of state-of-the-art methods. We first quantify the relationship between accuracy and time complexity given the amount of deformation and noise, the initial position and topological differences using synthetic data (Section 5.5.2). We then show results on real bioimaging data and on satellite images of roads (Section 5.5.3).

5.5.1 Tested methods

The methods to compare with were chosen to cover the range of known approaches to point cloud and graph matching. We have preferred methods that scored well in our previous experimental comparison in [89]. We have chosen RANSAC [37] as an example of sampling methods. Active Testing Search (ATS) [89], which uses incremental matching, is the best performing method so far. Graph matching techniques are represented by the Integer Projected Fixed Point method (IPFP) [60] and the Path Following Algorithm (PATH) [108]. Local matching approaches are represented by the Coherent Point Drift (CPD) [71].

The methods were implemented as described in their respective papers and using provided implementation when available, with the following parameters:

- RANSAC [37]: Four points are sampled at each iteration and an affine transform is fitted. This disadvantages RANSAC somewhat but to fit more general transforms would not be feasible due to the exponential time complexity.

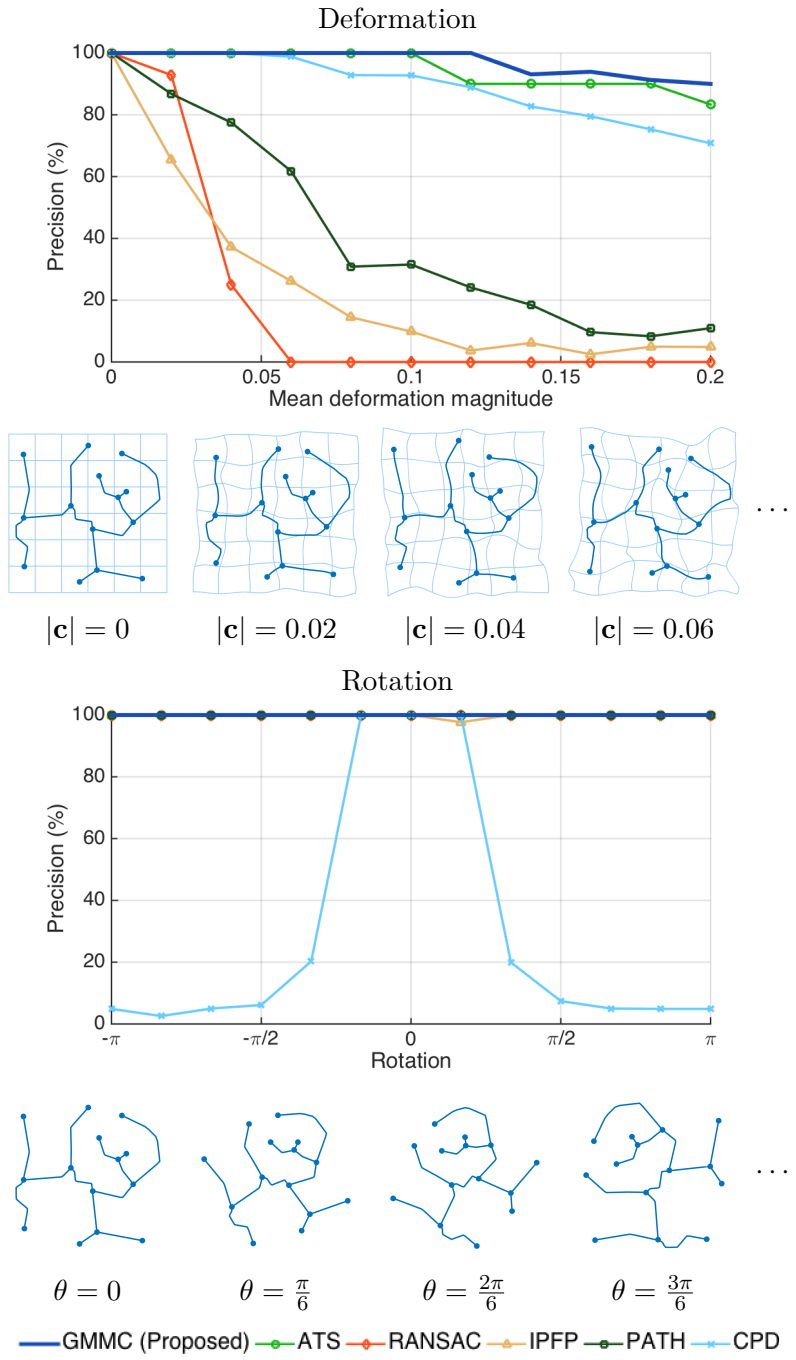


Figure 5.6: Results for synthetic datasets testing deformation and rotation in 3D showing the performance of the tested methods. The median correct correspondence percentage of 20 realizations for each parameter value is shown. Under each graph, we show 2D examples of the effects.

- ATS [89]: Synthetic training data (Section 5.5.2) was used to learn the score function. The Gaussian process regression hyperparameters were chosen as $\theta_0 = 1$, $\theta_1 = 10$, $\theta_2 = 0.1$, $\theta_3 = 1$ and $\beta^{-1} = 0.05$.

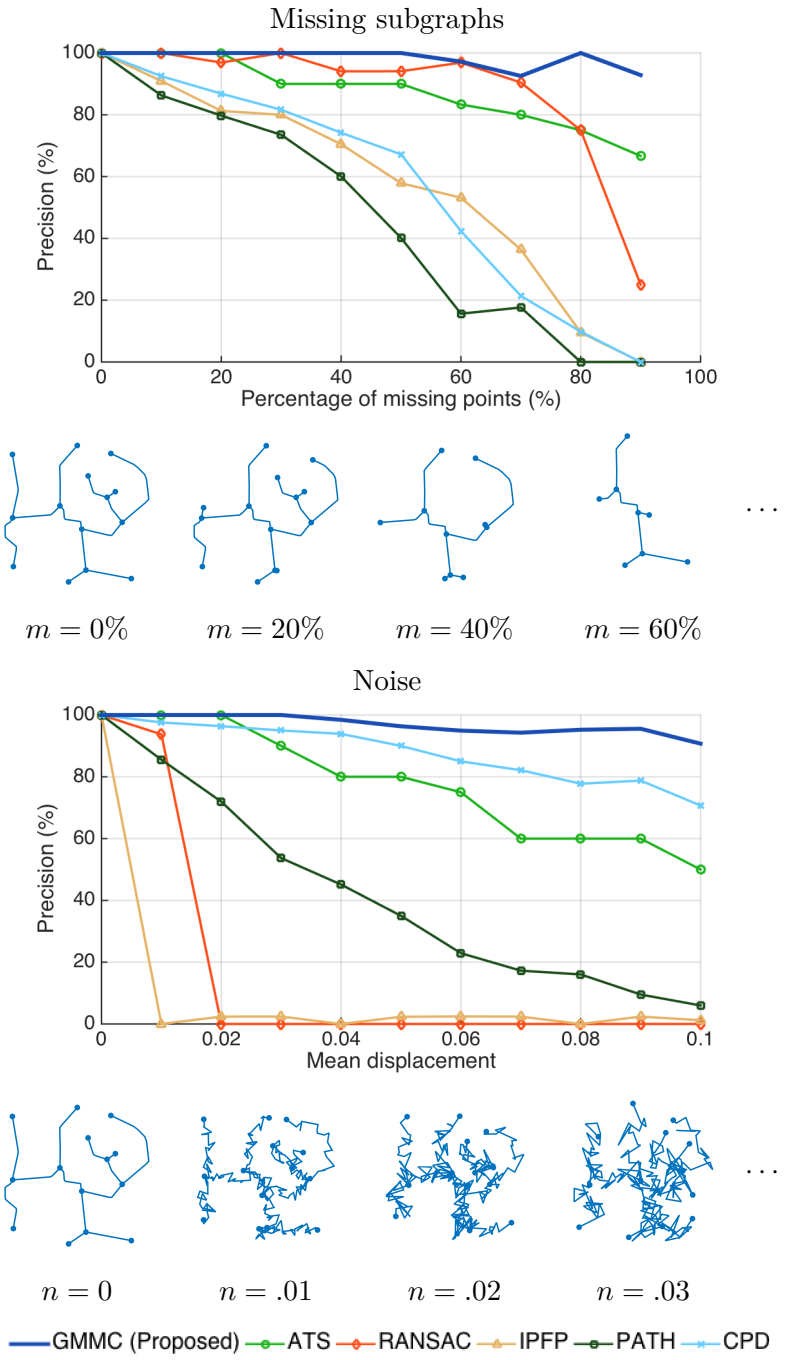


Figure 5.7: Results for synthetic datasets testing missing subgraphs and white noise in 3D showing the performance of the tested methods. The median correct correspondence percentage of 20 realizations for each parameter value is shown. Under each graph, we show 2D examples of the effects.

- CPD [71]: A point cloud is obtained by considering also the point representation of the edges with a sampling step of $\Delta = 0.025$. We used the non-rigid configuration with $\beta = 3$ and $\lambda = 3$. The initial

transformation was identity.

- For the proposed method (GMMC), the parameters used for all experiments were $K = 3, n_\omega = 5$ and $|\mathbf{\Omega}| = 50$.

All tested methods, with the exception of CPD, only provide a *coarse matching*, i.e. a matching between the graph nodes or their subsets. For the remaining methods, we use the same approach as in GMMC (as described in Section 5.3).

A correspondence $(\mathbf{x}, \mathbf{y})$ is considered to be correct, if the distance $d(\mathbf{y}, \mathbf{y}^*)$ to the true match $\mathbf{y}^*$ of $\mathbf{x}$ is smaller than the sampling step Δ.

5.5.2 Synthetic datasets

To generate a synthetic graph $\mathcal{G}^A$, the procedure from [89] is used — points are randomly sampled from a uniform distribution on the $[-1, 1]^3$ cube and then minimum spanning tree [12] edges are added. The branching points become vertices of $\mathcal{G}^A$ and the edges are resampled using a sampling step $\Delta = 0.025$. The graph $\mathcal{G}^B$ is obtained by transforming $\mathcal{G}^A$ in the desired way, as described below.

Evaluating matching accuracy

Random graphs $\mathcal{G}^A$ with approximately $|\mathbf{V}| \approx 40$ nodes were generated as described above and the following effects were applied to them to obtain the graphs $\mathcal{G}^B$:

- *Deformation*: A cubic B-spline transformation [99] was applied with random coefficients of a given standard deviation in the range $[0, 0.2]$.
- *Missing subgraphs*: Randomly selected branches of the graph were removed so that the number of nodes is decreased by $\{0\%, 10\%, \ldots, 90\%\}$.
- *Rotation*: The graph is rotated by a given angle from the interval $[-\pi, \pi]$ around a random rotation axis through the origin.

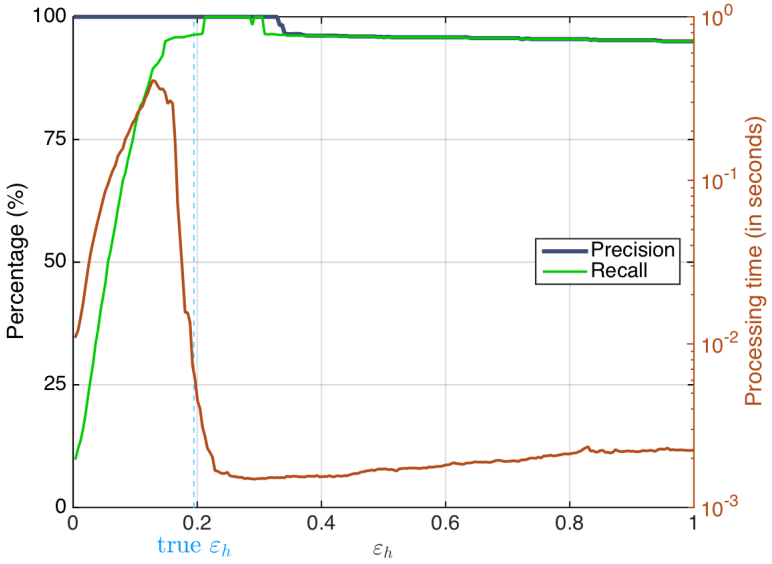


Figure 5.8: The processing time, precision and recall as a function of ε_h, where $\varepsilon_T = \frac{1}{3}\varepsilon_h$. For different values of ε_h, we calculate the processing time taken to obtain the solution, the precision (positive predictive value) and the recall (true positive rate). The values shown are a median over 100 synthetically generated pairs of graphs. N_{match} was set to the size of the true match.

- *Noise*: A Gaussian random displacement with standard deviation in the range $[0, 0.1]$ was applied to each component of each point in $\mathcal{G}^B$, including the edge points.

For each effect and each parameter value, 20 pairs of random graphs were generated and matched by the tested method, and median correct correspondence percentage was calculated over all vertices. The termination criteria were set so that the number of matched vertices is 80% of the size of $\mathbf{V}^A$.

We see in Figures 5.6 and 5.7 that the proposed method (GMMC) is among the best in all cases. CPD cannot handle large rotations and missing subgraphs, RANSAC and IPFP cannot handle deformation and noise, PATH performs badly for missing branches, deformation and noise. This leaves only ATS and GMMC fulfilling our requirements, with GMMC performing better than ATS in all cases except the rotation test, where their performance is similar.

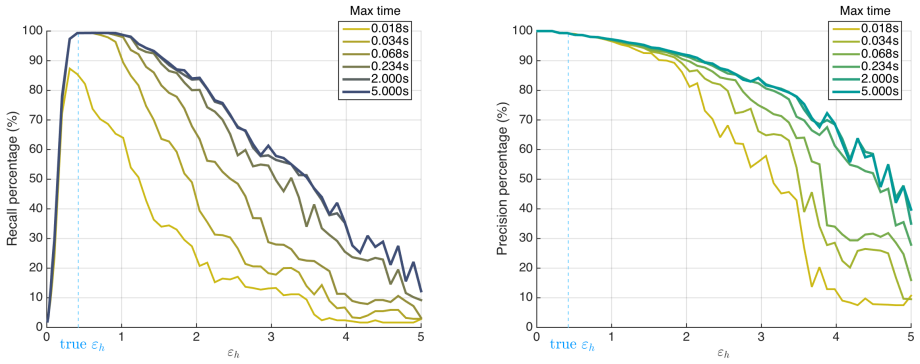


Figure 5.9: Precision and recall as a function of the allowed processing time and ε_h, where $\varepsilon_T = \frac{1}{3}\varepsilon_h$. For this experiment, the maximum amount of time for the algorithm to stop was the only termination criterion used. The median true ε_h is shown on the horizontal axes.

Lipschitz parameters

We analyzed the performance of the algorithm with respect to the Lipschitz parameters ε_h and ε_T, which determine the allowed deformation (5.6) and the allowed path descriptor tolerance (5.10). The parameter ε_T is set according to how much deformation can be expected in specific datasets and we take $\varepsilon_h = 3\varepsilon_T$, as discussed in Section 5.5.3.

We have generated 250 pairs of graphs by combining the effects from Section 5.5.2: a small nonlinear deformation, a random rigid motion, removal of randomly selected 40% of the vertices. The true median value of the Lipschitz constant ε_h for this dataset is approximately 0.2 and $\varepsilon_T \approx 0.07$. We ran the algorithm with different values of ε_h until either the search tree was completely explored or until a solution with as many matched points $M^{\mathbf{V}}$ as in the true solution was found.

As we can see in Fig. 5.8, for small values of ε_h, the tests are very strict, the search tree is rather small and is completely explored. The running time and the number of true correspondences found (recall) increases with ε_h. Above the optimal value of ε_h, which is close to the true median ε_h, the recall starts to decrease slowly. Note that by experiment design, the precision equals recall in this regime, since the number of returned correspondences is fixed. The running time first drops sharply, as the solution is found quickly and the search tree does not need to be fully explored; it then increases slowly with increasing ε_h.

We conclude that it is important to set ε_h to a large enough value, so

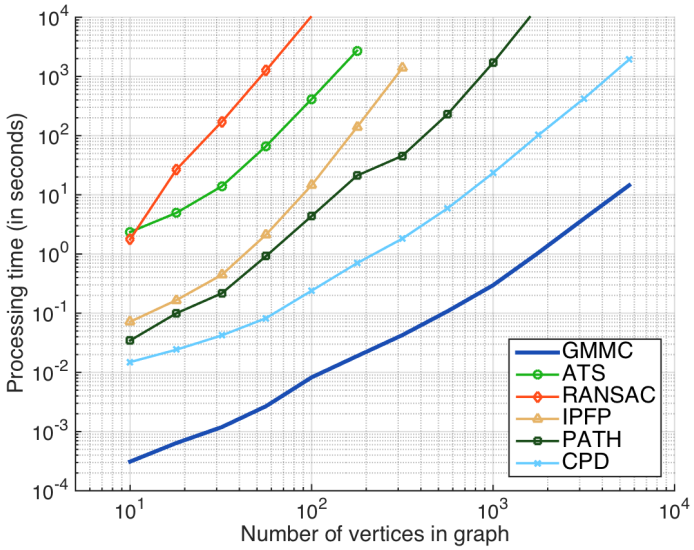


Figure 5.10: Experiment evaluating the complexity of methods ATS [89], RANSAC [37], IPFP [60], PATH [108], CPD [71] and the proposed method (GMMC). Both axes use the logarithmic scale.

that the true solution is not missed. Increasing it further will deteriorate the performance only slightly. Note that in most realistic applications, the Lipschitz constant ε_T is small, usually much smaller than 1.

To understand the performance of the algorithm with respect to the maximum allowed time $T_{\max}$, we generated 250 synthetic graph pairs as in the previous experiment, only increasing the size of the undecimated graph $\mathcal{G}^A$ to 250 vertices to make the differences more visible. In this experiment, we do not set the maximum number of iterations N_{it} nor the minimum stopping match size N_{match}; the algorithm stops only after the time budget $T_{\max}$ is exhausted or if the search tree is completely explored. Recall and precision for $\varepsilon_h \in [0, 5]$ and $T_{\max} \in [0.018, 5.0]\,\text{s}$ are shown in Fig. 5.9. The median true ε_h for this dataset is approximately 0.42 and the median true $\varepsilon_T \approx 0.14$.

As expected from the previous experiment, for small ε_h, the recall increases sharply with increasing ε_h and the precision is high. After exceeding the optimal value, which is close to the true median Lipschitz constant, false positives start appearing and the performance decreases slowly with increasing ε_h. Allowing more processing time always improves the results, as the algorithm has the time to explore a larger part of the search tree. This effect is more pronounced for higher values of ε_h.

Time complexity

Using the procedure as described at the beginning of this Section, with a random rotation and a small nonlinear deformation, we have generated a synthetic dataset of graph pairs with an increasing number of vertices $|\mathbf{V}| \in \{10^1, 10^{1.25}, 10^{1.5}, 10^{1.75}, \ldots, 10^{3.75}\}$, with 20 pairs of graphs for each graph size. Fig. 5.10 shows the median processing time for all the tested methods. The parameters for all methods were set so that at least 75% of the vertices of the graphs were matched. We can see that the proposed method is the fastest of the methods tested. For larger graphs, it is about 5 orders of magnitude faster than ATS, the only other method meeting our requirements. Moreover, GMMC also has a lower asymptotic complexity (see the slope of the curve) than all other competing methods except CPD. The times are given for the coarse matching only, while CPD matches the edge points, too.

5.5.3 Real datasets

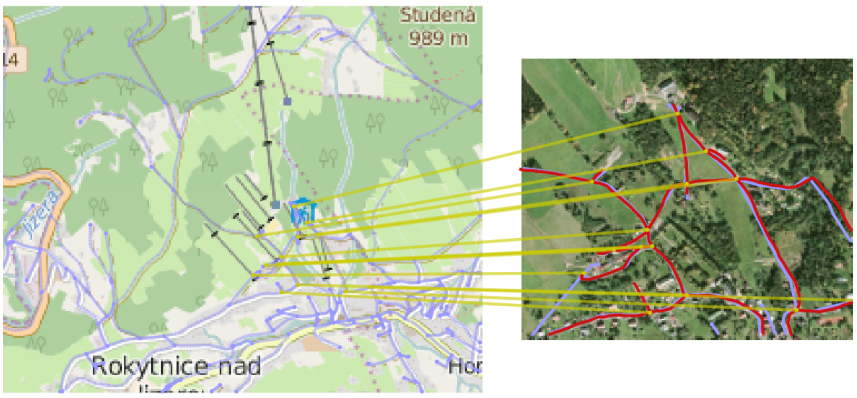
The algorithm was also tested on real datasets. All segmented graphs were normalized such that $\mathbf{V}^A, \mathbf{V}^B \in [-1, 1]^D$. The ground truth correspondence between the graph vertices was obtained manually. For all methods except CPD, the Gaussian process regression as described in Section 5.5.1 was used to obtain the fine alignment. Then, we calculated the *alignment error* as the mean Euclidean distance between the corresponding vertices, once the graphs were aligned. We also show the precision and recall of identifying the true matches between vertices, which were annotated manually, as well as the total processing time. As in the synthetic case, we choose ε_T according to how much deformation we expect to find in the dataset and set $\varepsilon_h = 3\varepsilon_T$. The chosen value of ε_T is reported for each dataset.

Roads

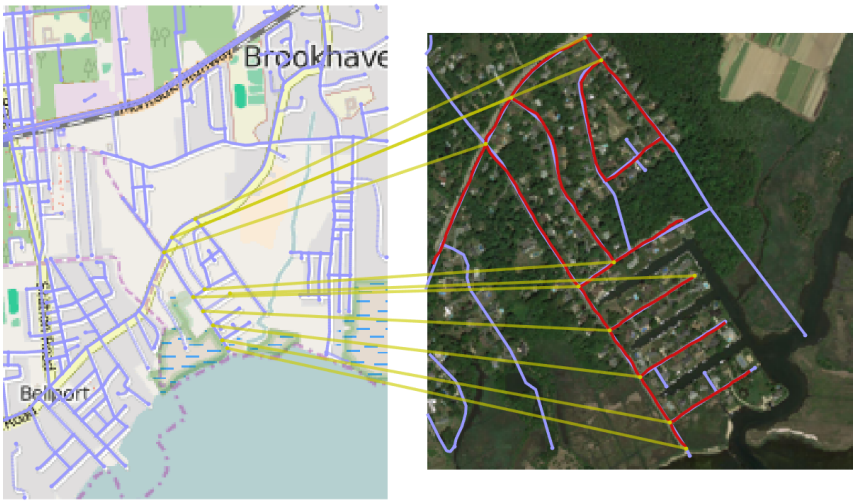
We matched a segmented road network from an aerial view image to a graph extracted from a road map. The map was obtained from OpenStreetMap [50] and we semi-automatically segmented [65, 97] roads from satellite images obtained from Google Maps[1]. The satellite image corresponds to only a small subset of the graph obtained from the map.

[1] `http://maps.google.com`

Road Pair II: Rokytnice nad Jizerou, Czech Republic



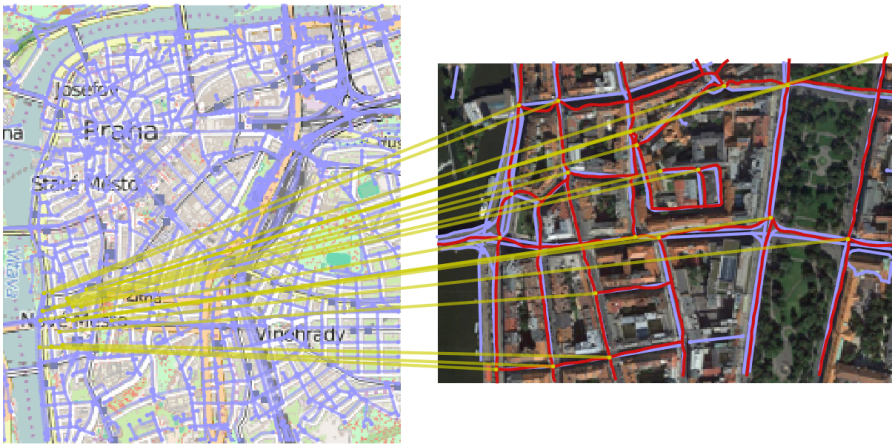
Road Pair III: Brookhaven, New York



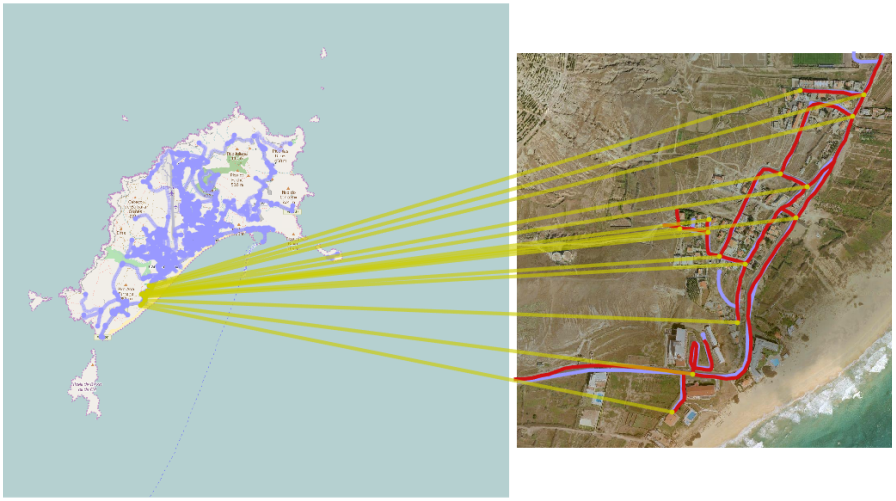
Road Pair IV: Ilha do Faial, Portugal

Figure 5.11: Examples of results for the roads dataset. On the left the graph extracted from the map is shown overlaid on a map image. On the right the graph extracted from the road network is shown (in red) aligned with the graph extracted from the map (in blue). The correspondences are depicted by the yellow lines connecting vertices of the graphs.

Road Pair VI: Prague, Czech Republic



Road Pair VIII: Porto Santo, Portugal



Road Pair IX: Porto, Portugal

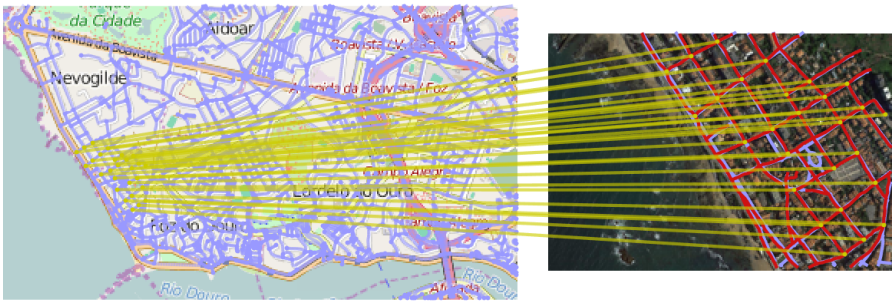


Figure 5.12: Examples of results for the roads dataset. On the left the graph extracted from the map is shown overlaid on a map image. On the right the graph extracted from the road network is shown (in red) aligned with the graph extracted from the map (in blue). The correspondences are depicted by the yellow lines connecting vertices of the graphs.

Dataset		GMMC (Proposed)	ATS	IPFP	CPD
Road Pair I	Error	**0.004**	0.005	0.305	0.239
$\|\mathbf{V}^A\|$: 32	Precision (%)	**100.0**	**100.0**	0.0	3.1
$\|\mathbf{V}^B\|$: 38	Recall (%)	**83.3**	33.3	0.0	8.3
	Time (s)	0.03	55.17	9.16	0.92
Road Pair II	Error	**0.005**	0.011	0.252	0.070
$\|\mathbf{V}^A\|$: 25	Precision (%)	**92.9**	71.4	0.0	0.0
$\|\mathbf{V}^B\|$: 199	Recall (%)	**61.9**	23.8	0.0	0.0
	Time (s)	0.00	30306.50	137.90	0.01
Road Pair III	Error	**0.003**	0.005	0.403	0.358
$\|\mathbf{V}^A\|$: 19	Precision (%)	90.9	**100.0**	0.0	0.0
$\|\mathbf{V}^B\|$: 306	Recall (%)	**76.9**	23.1	0.0	0.0
	Time (s)	0.01	10946.63	1679.98	0.01
Road Pair IV	Error	**0.009**	–	0.447	0.388
$\|\mathbf{V}^A\|$: 20	Precision (%)	**100.0**	–	0.0	0.0
$\|\mathbf{V}^B\|$: 1344	Recall (%)	**71.4**	–	0.0	0.0
	Time (s)	0.96	–	1606.86	0.01
Road Pair V	Error	**0.004**	–	0.429	0.303
$\|\mathbf{V}^A\|$: 29	Precision (%)	**73.3**	–	0.0	0.0
$\|\mathbf{V}^B\|$: 711	Recall (%)	**61.1**	–	0.0	0.0
	Time (s)	0.00	–	509.53	0.02
Road Pair VI	Error	**0.005**	–	0.782	0.790
$\|\mathbf{V}^A\|$: 49	Precision (%)	**70.0**	–	0.0	0.0
$\|\mathbf{V}^B\|$: 1989	Recall (%)	**42.4**	–	0.0	0.0
	Time (s)	10.95	–	83619.15	0.17

Table 5.1: Results for road datasets: alignment error (graphs were normalized s. t. $\mathbf{V}^A, \mathbf{V}^B \in [-1, 1]^D$), percentage of correct matches in the solution (precision), percentage of ground truth matches retrieved (recall) and processing time in seconds. For each dataset, we present the number of vertices of each graph $|\mathbf{V}^A|$ and $|\mathbf{V}^B|$.

We tested 10 different pairs of map graphs with up to 6000 vertices, 7500 edges and 75000 superedges, which were matched to templates with up to 60 vertices and 600 superedges. We used $\varepsilon_T = 0.1$.

The alignment error, precision, recall and elapsed time are presented in Tables 5.1 and 5.2 and a few visual examples are shown in Figures 5.1, 5.11 and 5.12. Some methods could not process all datasets as they exhausted the available 256GB of RAM memory in our computer. These cases are marked with '—'. We see that the proposed GMMC method is the best in almost all criteria (shown in bold). The ATS method has a better precision but a much lower recall and much higher computational complexity. The tradeoff between precision and recall can be influenced by parameter setting but the computational complexity remains. The other tested methods simply do not find any correct matches.

Dataset		GMMC (Proposed)	ATS	IPFP	CPD
Road Pair VII	Error	**0.007**	–	–	0.377
$\|\mathbf{V}^A\|$: 47	Precision (%)	**73.7**	–	–	0.0
$\|\mathbf{V}^B\|$: 6050	Recall (%)	**43.8**	–	–	0.0
	Time (s)	363.74	–	–	0.29
Road Pair VIII	Error	**0.009**	–	0.464	0.363
$\|\mathbf{V}^A\|$: 24	Precision (%)	**85.7**	–	0.0	0.0
$\|\mathbf{V}^B\|$: 803	Recall (%)	**60.0**	–	0.0	0.0
	Time (s)	0.02	–	824.31	0.05
Road Pair IX	Error	**0.002**	–	0.526	0.675
$\|\mathbf{V}^A\|$: 67	Precision (%)	**96.3**	–	0.0	0.0
$\|\mathbf{V}^B\|$: 1693	Recall (%)	**57.8**	–	0.0	0.0
	Time (s)	414.36	–	36056.12	0.20
Road Pair X	Error	**0.002**	–	0.268	0.577
$\|\mathbf{V}^A\|$: 51	Precision (%)	**95.8**	–	0.0	0.0
$\|\mathbf{V}^B\|$: 2576	Recall (%)	**71.9**	–	0.0	0.0
	Time (s)	0.05	–	45758.20	3.08
Road Pair XI	Error	**0.006**	–	0.489	0.571
$\|\mathbf{V}^A\|$: 41	Precision (%)	**100.0**	–	0.0	0.0
$\|\mathbf{V}^B\|$: 2479	Recall (%)	**78.0**	–	0.0	0.0
	Time (s)	0.66	–	21431.36	0.02
Road Pair XII	Error	**0.004**	–	–	0.668
$\|\mathbf{V}^A\|$: 36	Precision (%)	**95.2**	–	–	0.0
$\|\mathbf{V}^B\|$: 6050	Recall (%)	**83.3**	–	–	0.0
	Time (s)	549.41	–	–	0.23

Table 5.2: Results for road datasets: alignment error (graphs were normalized s. t. $\mathbf{V}^A, \mathbf{V}^B \in [-1, 1]^D$), percentage of correct matches in the solution (precision), percentage of ground truth matches retrieved (recall) and processing time in seconds. For each dataset, we present the number of vertices of each graph $|\mathbf{V}^A|$ and $|\mathbf{V}^B|$.

Medical images

We tested our method on five types of datasets of medical images or volumes. The datasets are examples of different applications in medical imaging and were obtained using different acquisition techniques.

Tem aim of the registration of **retinal fundus** images [29, 55] (Fig. 5.14 and 5.15, $\varepsilon_T = 0.25$) is to combine images with limited fields of view or to detect changes in time. In this case the segmented graphs are two-dimensional.

Images of **brain circuits** (Fig. 5.18 $\varepsilon_T = 0.2$) are sparse sets of fluorescently labeled neurons in the neocortex in 3D, which were obtained using large-scale 2-photon laser scanning microscopy at two time instances in a living mouse [51]. Registration is needed to detect and quantify the differences [44].

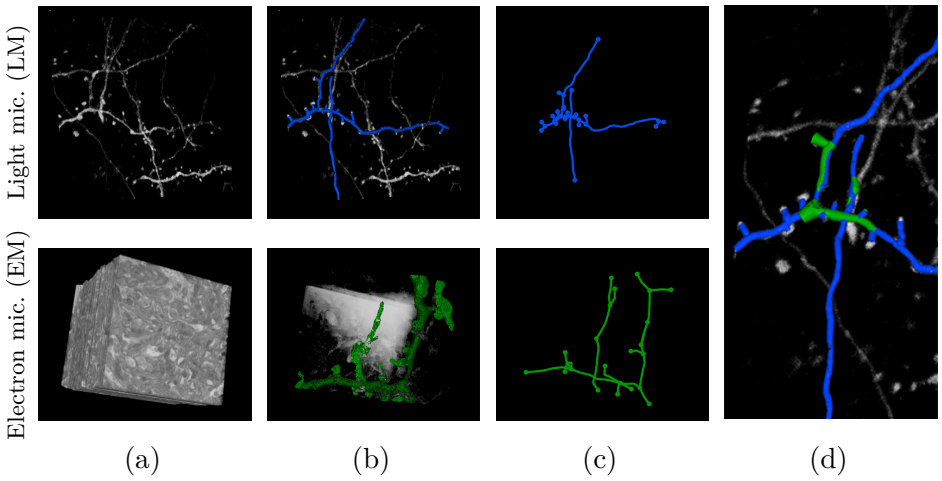


Figure 5.13: (a) top: Brain tissue acquired using two-photon light microscopy from live brain tissue at a 1 μm resolution and **bottom:** a smaller area of the same tissue imaged using electron microscopy, at a 20 nm resolution. **(b):** The neuronal fibers are segmented. **(c):** The segmentation is converted to graphs $\mathcal{G}^A = (\mathbf{V}^A, \mathbf{E}^A)$ and $\mathcal{G}^B = (\mathbf{V}^B, \mathbf{E}^B)$. **(d):** The alignment of the two structures after matching using the proposed method.

Multimodal registration is demonstrated on two 3D datasets. The **EM/LM** (Fig. 5.13, $\varepsilon_T = 0.35$) dataset shows brain tissue at two different scales using electron microscopy (EM) and light microcopy (LM). **Brain vessels** (Fig. 5.17, $\varepsilon_T = 0.35$) are blood vessels acquired using optical and 2-photon microscopy. In both cases the registration is needed to fuse images from the two modalities.

Finally, heart images from the **angiography** dataset are two-dimensional angiograms taken at different time instances (Fig. 5.19, $\varepsilon_T = 0.25$). The aim of this application is to track the displacement of the heart vessels over time.

The alignment error, precision, recall and elapsed time for these datasets is shown in Table 5.3. Both the proposed method (GMMC) and ATS are able to successfully match all the graphs, however GMMC is much faster. CPD can solve some of the tasks but fails when large parts of the graph are missing (EM/LM) or when the deformation is large (angiography). CPD sometimes presents a high recall since it tries to match all points. However, the precision is lower, as many of these points are incorrect, resulting in a higher alignment error in most cases.

Dataset		GMMC (Proposed)	ATS	IPFP	CPD
Brain Circuits $\lvert\bar{\mathbf{V}}\rvert$: 124	Error	**0.026**	0.030	0.669	0.027
	Precision (%)	72.0	**100.0**	2.4	45.2
	Recall (%)	29.5	8.2	4.9	**91.8**
	Time (s)	0.16	1104.41	59.42	0.11
EM/LM $\lvert\bar{\mathbf{V}}\rvert$: 22	Error	**0.016**	**0.016**	0.179	0.128
	Precision (%)	77.8	**100.0**	4.5	4.5
	Recall (%)	**70.0**	50.0	10.0	10.0
	Time (s)	0.00	97.14	0.78	0.01
Brain Vessels $\lvert\bar{\mathbf{V}}\rvert$: 37	Error	0.045	**0.041**	0.409	0.048
	Precision (%)	71.4	**100.0**	21.4	29.7
	Recall (%)	41.7	41.7	50.0	**91.7**
	Time (s)	0.01	4.84	3.26	0.01
Angiography $\lvert\bar{\mathbf{V}}\rvert$: 24	Error	**0.031**	0.052	0.057	0.078
	Precision (%)	**70.1**	70.0	24.7	19.1
	Recall (%)	**80.6**	47.2	66.7	61.1
	Time (s)	0.58	2.84	4.03	0.15
Retina $\lvert\bar{\mathbf{V}}\rvert$: 141	Error	**0.017**	0.030	0.128	0.070
	Precision (%)	**88.2**	86.7	61.9	66.4
	Recall (%)	77.9	4.7	76.4	**84.1**
	Time (s)	0.66	330.56	182.40	0.09

Table 5.3: Results for medical images: average distance between true matches of aligned graphs (graphs were normalized s. t. $\mathbf{V}^A, \mathbf{V}^B \in [-1, 1]^D$), percentage of correct matches in the solution (precision), percentage of ground truth matches retrieved (recall) and processing time in seconds. For each dataset, we also give the average number of vertices $|\bar{\mathbf{V}}|$.

Discussion — what should be matched

Let us comment on several aspects, which are well illustrated on the medical datasets. First, if the deformation is smooth, as for the retinal dataset, it is faster and sufficiently accurate to match just a part of the graph for example by reducing the desired number of matched vertices N_{match}. Second, it is up to the user to provide the parameters ε_h or ε_T, determining how much deformation is allowed between the two input graphs. If this parameters are set conservatively, strongly deformed parts will not be matched. In Fig. 5.16, we show that by increasing ε_T and ε_h, a large part of the graph can be matched at the expense of computation time.

Finally, note that leaf nodes are sometimes not matched, as they do not correspond to well defined physical points in the tissue, but rather to points where the annotator or the segmentation algorithm decided to stop the edge (see the angiography or retinal datasets), and these points may not coincide in the two graphs. If desired, leaf edge matches can be treated specially, allowing partial matches by shortening the longer superedge to the length of

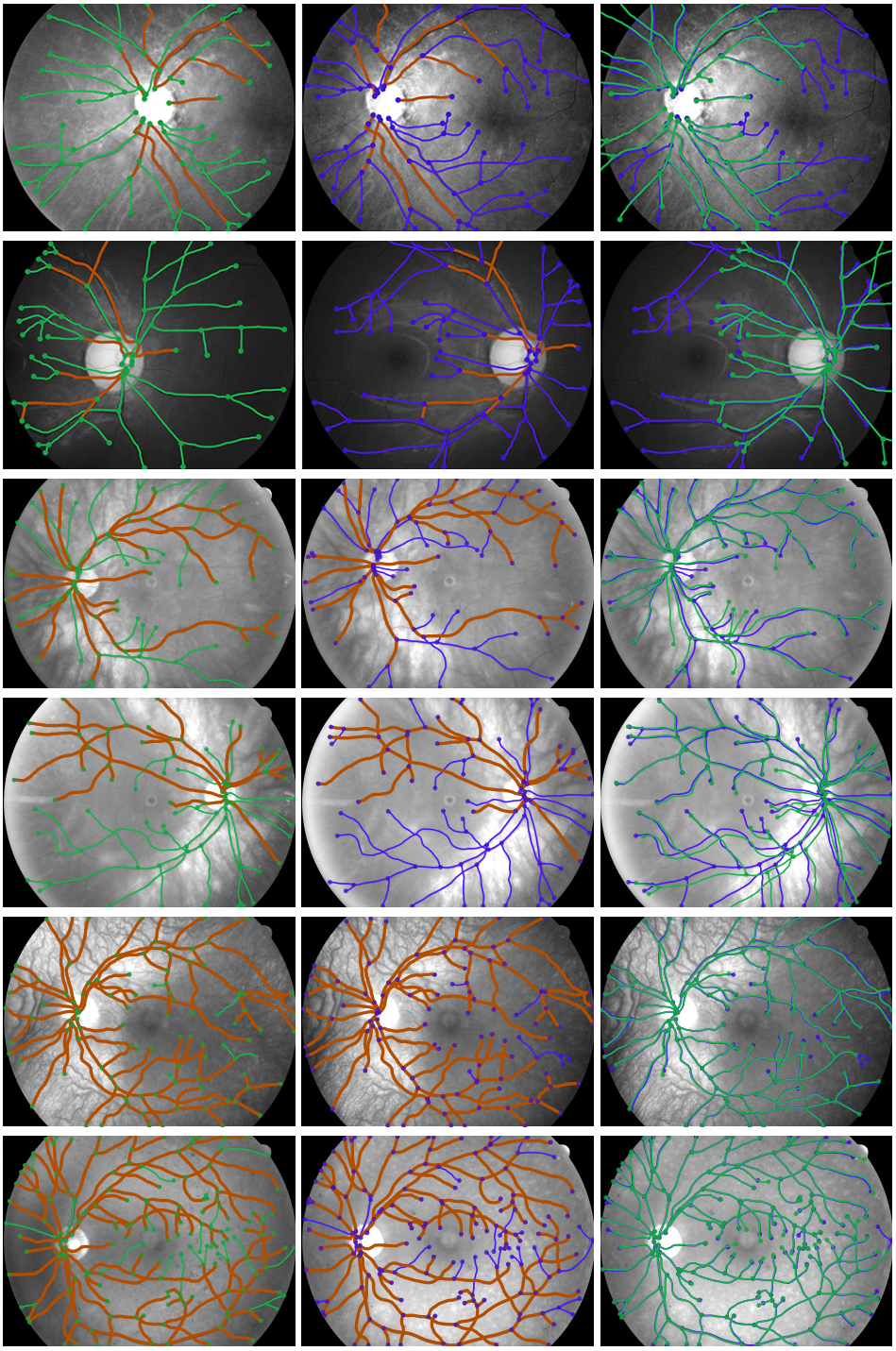


Figure 5.14: Examples of results on the retinal dataset. The first and second columns are the input images with the extracted geometrical graphs superimposed. Matched superedges are shown in red. The third column depicts the resulting alignment of the first graph onto the second.

the shorter one [88], at the expense of more false matches.

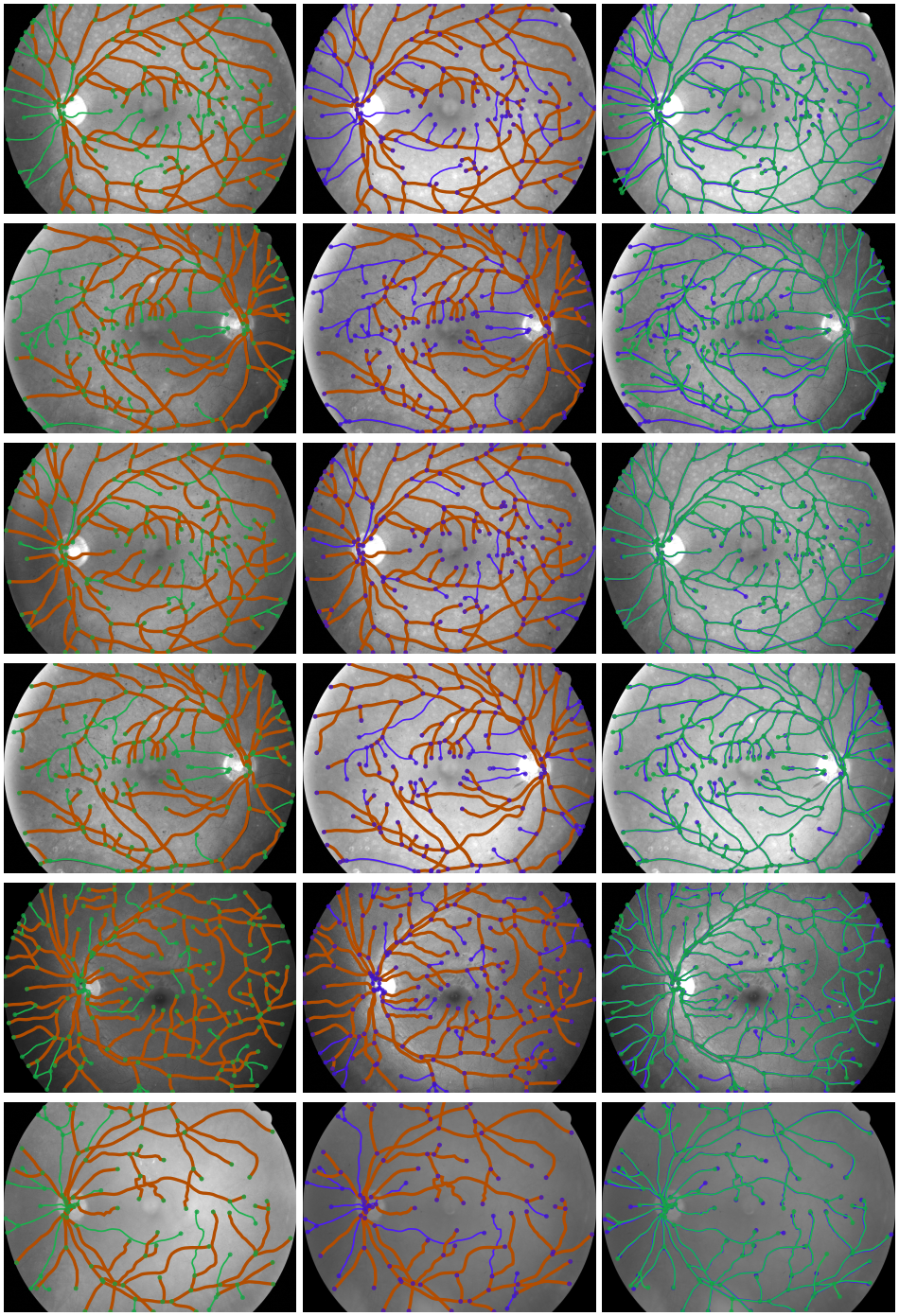
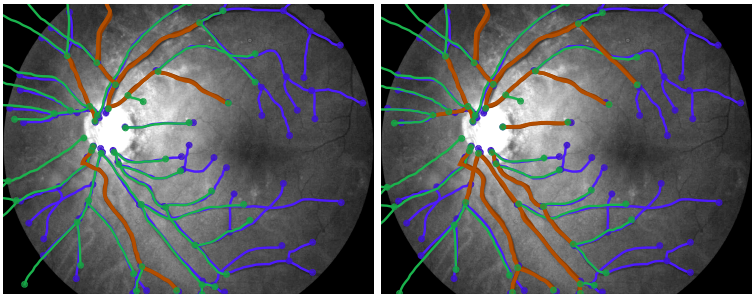
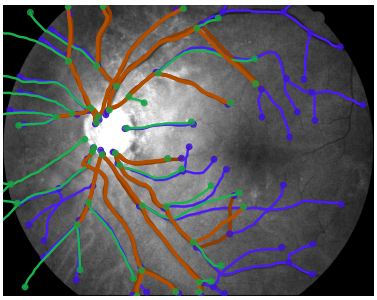


Figure 5.15: Examples of results on the retinal dataset. The first and second columns are the input images with the extracted geometrical graphs superimposed. Matched superedges are shown in red. The third column depicts the resulting alignment of the first graph onto the second.

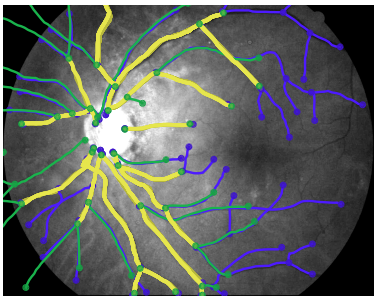


$\varepsilon_T = 0.05$, Recall 39.1%
$t = 0.03s$, Precision 100%

$\varepsilon_T = 0.15$, Recall 81.8%
$t = 0.06s$, Precision 100%



$\varepsilon_T = 0.35$, Recall 84.8%
$t = 0.12s$, Precision 84.8%



Ground truth

Figure 5.16: The recall increases with increasing ε_T (with $\varepsilon_h = 3\varepsilon_T$) when matching two retinal fundus images. Consequently, the precision decreases and the processing time t increases. The graphs are drawn in green and blue, transformed based on the matched superedges (shown in red for both graphs), and overlaid on the original image. The same graph pair is visible as the left-most result in Fig. 5.14. The ground truth is shown in yellow in the right-most pair of images. Only entire edges are matched.

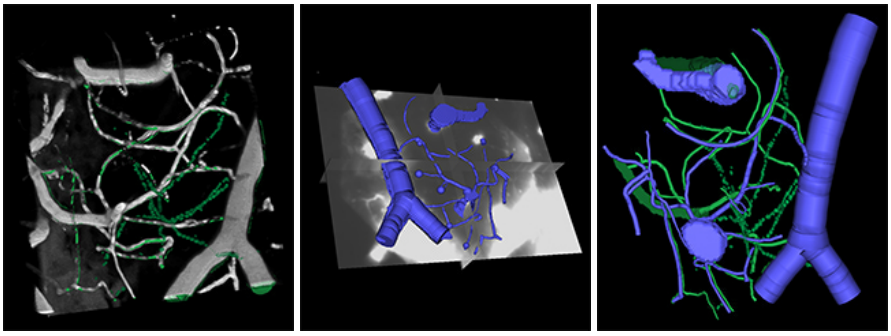


Figure 5.17: Example of medical applications results for blood vessels acquired using **(left)** two-photon microscopy and **(middle)** bright-field optical microscopy. Only segmentation is shown for better visualization. The first and second volumes are the initial images with the extracted geometrical graphs superimposed. The third depicts the alignment of the proposed method of the first graph onto the space of the second.

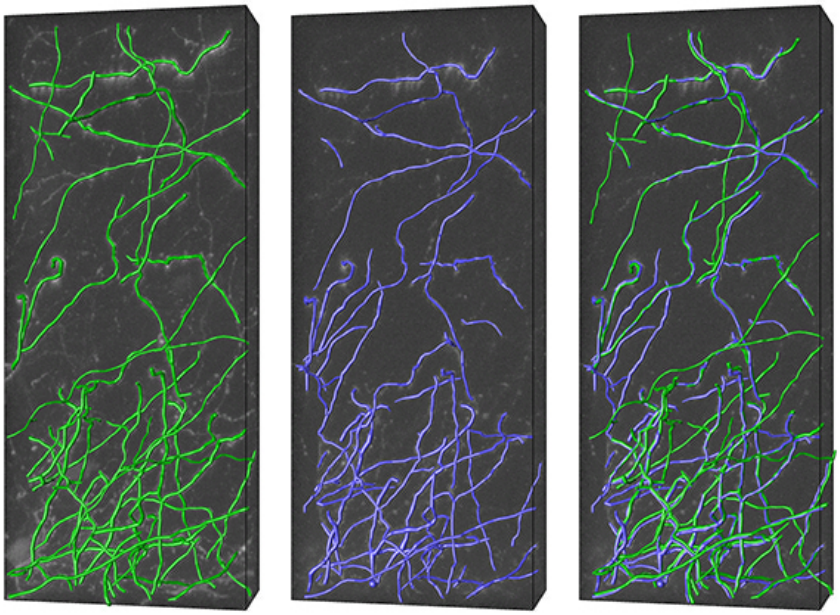


Figure 5.18: Example of medical applications results for a neuronal network acquired at different time instances using 2-photon microscopy. The first and second volumes are the initial images with the extracted geometrical graphs superimposed. The third depicts the alignment of the proposed method of the first graph onto the space of the second.

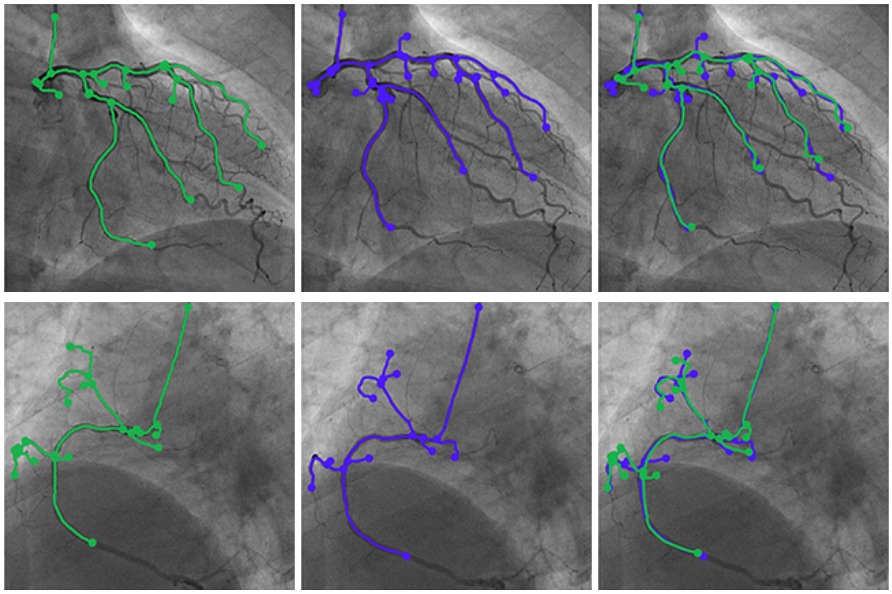


Figure 5.19: Example of medical applications results for angiography images from a beating heart taken at different time instances. The first and second volumes are the initial images with the extracted geometrical graphs superimposed. The third depicts the alignment of the proposed method of the first graph onto the space of the second.

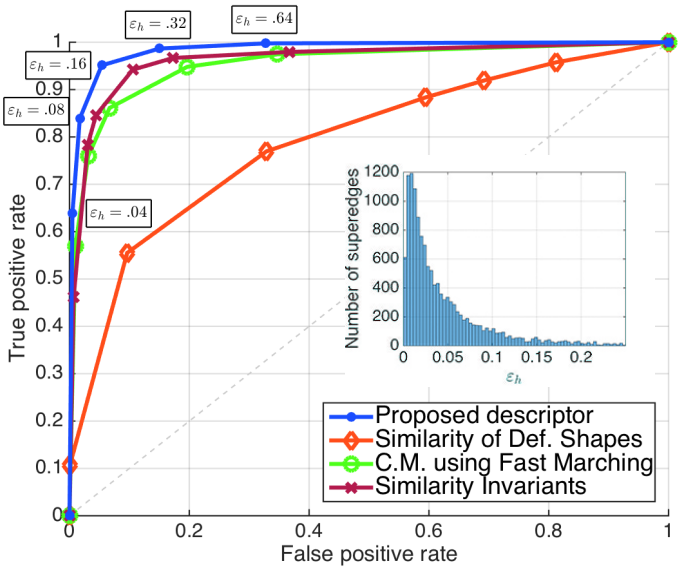


Figure 5.20: ROC curve comparing the performance of the curve descriptors. in all medical datasets of different methods. For each point of the proposed method in the ROC curve, we show the used ε_h values. We also show a histogram of the true ε_h of the superedges in the data. The mean value of ε_h in this data is approximately 0.06.

Path descriptors

We have tested the proposed path descriptors from Section 5.4 against four other methods: Similarity of Deformable Shapes [6], Curve Matching using the Fast Marching Method [40] and Similarity Invariants for 3D Space Curve Matching [64]. We have used all medical datasets from Section 5.5.3.

We took all possible superedge pairs and tried to determine whether two superedges matched. ROC curves were calculated by thresholding a provided curve distance measure for [6] and [40], thresholding a Euclidean distance between the descriptors for [64], and varying ε_h in (5.10) for our method. We can see in Fig. 5.20, that our path descriptors outperform all other methods.

The points on the ROC curve are annotated with the Lipschitz constant ε_h and we also show a histogram of the true ε_h values for all matching superedge pairs. We can see that the equal error rate point is reached for $\varepsilon_h = 0.16$, which corresponds to the 93.5% percentile of the histogram while $\varepsilon_T = 0.05$, corresponds to a 95.17% percentile of its respective histogram. We observe similar behavior in other datasets, including synthetically generated. Hence our choice of $\varepsilon_h = 3\varepsilon_T$, with ε_T sufficiently large to allow the expected deformation.

Chapter 6

Conclusion

Throughout this thesis we focused on the problem of image registration using graph matching. We proposed to match images by first extracting the geometrical information of graph shaped structures present in the images, and consequently matching them to obtain an alignment between the images.

We proposed a segmentation approach that enforces time consistency constraints to improve the extraction of the structures. The method successfully segments trees from neuronal data and aerial images of road networks. We showed that applying these constraints improves the segmentation accuracy as opposed to considering the images individually.

We then proposed two different algorithms for matching of geometrical graphs. The first, Active Testing Search (ATS) uses learning to estimate the likelihood to determine whether a partial matching is correct, given several matching quality features. The transformation is obtained by a Gaussian Processes Regression given a partial matching between points. We showed the method is able to accurately match real datasets from different medical applications. The approach is faster than other search methods, however it is still slower than several point matching techniques when matching sets with a large number of points with high deformation.

Second, we proposed an approach to considerably improve the time complexity of the graph matching, called Graph Matching using Monte Carlo Tree Search (GMMC). The method aggregates adjacent compatible edges using a search algorithm inspired in the Monte Carlo tree search algorithm. Compatible edges are obtained using a curve descriptor to compare the geo-

metrical curves. As stated as an objective in the introduction of this thesis, the approach is able to match graph structures with unknown initial position, presenting non-linear deformation and disregarding appearance features. Although ATS is also capable of matching most of this type of structures, it uses significantly more processing time, as GMMC presents the best time complexity from the tested methods. We show that the method is accurate in all tested real datasets in medical imaging and aerial image localization, and that the time complexity is the lowest considering other competitive methods.

The method is fast also thanks to the curve descriptors which we have introduced and which allow efficient pruning. We show experimentally that these descriptors outperform other previously known curve descriptors.

Our method is modular and could be extended in various ways. For example, we do not use any appearance information at the moment but it would be easy to add an appearance comparison to the superedge pair compatibility check. Alternative transformation models could be used to improve robustness and for a further speedup. The subtree match quality criterion could be estimated more efficiently from the properties of the partial matches using machine learning, similar to [89]. The search could be parallelized to yield further speedup. The challenge remains how to perform the matching of truly large geometrical graphs, of millions of vertices or more.

Appendix A

Active Testing Search derivation

In this appendix, we first introduce some notations and then derive our Active Testing Search (ATS) algorithm, mentioned in Section 4.3.2.

Notation and Assumptions

Let n_A and n_B be the number of nodes, respectively, in $\mathbf{X}^A$ and $\mathbf{X}^B$ and assume that $n_A \leq n_B$. We impose a fixed ordering for all the points in $\mathbf{X}^A$ and denote $Z_i \in \{-1, 1, \ldots, n_B\}$ to be the index of the node $\mathbf{x}^B_{Z_i}$ that matches $\mathbf{x}^A_i$, where $Z_i = -1$ if there is no such match. Let $\mathbf{M} = \left\{\mathbf{x}^A_l \leftrightarrow \mathbf{x}^B_{Z_l}\right\}_{1 \leq l \leq L}$ be a candidate set of correspondences where the first L points in $\mathbf{X}^A$ are assigned. Let $Z = (Z_1, ..., Z_{n_A}) \in \mathcal{Z}$, where $\mathcal{Z}$ is the space of all possible solutions that map $\mathbf{X}^A$ to $\mathbf{X}^B$ be a discrete random vector with probability distribution $P(Z) = P(Z_1, ..., Z_{n_A})$.

To organize the space of solutions, we construct a hierarchy over the space of $\mathcal{Z}$ by means of decomposition tree Λ, whose nodes are taken to be $\Lambda_{u,v} = (Z_1 = z_1, ..., Z_u = z_u), v \in \{1, ..., V_u\}$, where V_u is the number of assignments of length u. Note that each $\Lambda_{u,v}$ corresponds to a unique set of correspondences $\mathbf{M}$. The children of $\Lambda_{u,v}$ are formed by adding one more assignment

$$\mathcal{C}_{\Lambda_{u,v}} = \{(Z_1 = z_1, \ldots, Z_u = z_u, Z_{u+1} = z) \mid \forall z \notin \{z_1, \ldots z_u\}\} \qquad \text{(A.1)}$$

and note that because the elements of $\mathcal{C}_{\Lambda_{u,v}}$ are disjoint then

$$P(\Lambda_{u,v}) = \sum_{c \in \mathcal{C}_{\Lambda_{u,v}}} P(c). \tag{A.2}$$

We define a quality score $S_{\mathbf{M}} \in \mathbb{R}^+$ to be a random observation of the quality of the mapping defined by a set of correspondences $\mathbf{M}$ and assume that it is normally distributed

$$P(S_{\mathbf{M}} = s|Z) = \begin{cases} \mathcal{N}(s; \boldsymbol{\omega}_1^u), & \text{if } \delta(\mathbf{M}, Z) = 1 \\ \mathcal{N}(s; \boldsymbol{\omega}_0^u), & \text{if } \delta(\mathbf{M}, Z) = 0 \end{cases}, \tag{A.3}$$

where u is number of correspondences in $\mathbf{M}$, $\mathcal{N}$ is a Gaussian probability distribution with parameters $\boldsymbol{\omega}_1^u = (\mu_1^u, \sigma_1^u)$, $\boldsymbol{\omega}_0^u = (\mu_0^u, \sigma_0^u)$, i.e. the mean and variance of each of the Gaussian distribution, and $\delta(\mathbf{M}, Z) = 1$ if the correspondences of $\mathbf{M}$ are compatible with a given Z and 0 otherwise. In addition, we assume that

$$P(\{S_{\Lambda_{u,v}}; \Lambda_{u,v} \in \Lambda\}|Z) = \prod_{\Lambda_{u,v} \in \Lambda} P(S_{\Lambda_{u,v}}|Z). \tag{A.4}$$

which, together with (A.3) allows for a computationally tractable solution without compromising the quality of the solution too much.

Finally, the likelihood ratio is

$$R(s) = \frac{\mathcal{N}(s; \boldsymbol{\omega}_1^u)}{\mathcal{N}(s; \boldsymbol{\omega}_0^u)}, \tag{A.5}$$

and T is the number of iterations of the ATS optimization.

Objective

Our objective is to infer Z from some set of observations $\{S_{\mathbf{M}_1}, ..., S_{\mathbf{M}_T}\}$. To do this, we consider solving the MAP

$$\begin{aligned} Z^* &= \arg\max_{z \in \mathcal{Z}} P(Z|S_{\mathbf{M}_1}, \ldots, S_{\mathbf{M}_T}) && \text{(A.6)} \\ &= \arg\max_{z \in \mathcal{Z}} \{P(Z)P(S_{\mathbf{M}_1}, \ldots, S_{\mathbf{M}_T}|Z)\} \\ &= \arg\max_{z \in \mathcal{Z}} \left\{ P(Z) \prod_{t=1}^{T} P(S_{\mathbf{M}_t}|Z) \right\}. && \text{(A.7)} \end{aligned}$$

The ATS approach approximates the MAP by iteratively making observations $S_{\mathbf{M}_t}$. If at time step t, the correspondence set $\mathbf{M}_t$, associated with $\Lambda_{u,v}$, is evaluated and $\mathbf{S}_t = \{S_{\mathbf{M}_1} = s_1, \ldots, S_{\mathbf{M}_t} = s_t\}$ then for any set of correspondences $Z = z$

$$\begin{aligned} P(Z = z|\mathbf{S}_t) &= \frac{P(S_{\mathbf{M}_t} = s_t|Z = z)P(Z = z|\mathbf{S}_{t-1})}{P(S_{\mathbf{M}_t} = s_t)} \\ &= \frac{B(s_t, z)P(Z = z|\mathbf{S}_{t-1})}{P(S_{\mathbf{M}_t} = s_t)} \end{aligned}$$

where $B(s_t, z) = \mathcal{N}(s_t; \boldsymbol{\omega}_1^u)^{\delta(\mathbf{M}_t, z)} + \mathcal{N}(s_t; \boldsymbol{\omega}_0^u)^{1-\delta(\mathbf{M}_t, z)}$. Using the likelihood ratio, we can therefore write

$$P(Z = z|\mathbf{S}_t) = \frac{1}{C} R(s_t)^{\delta(\mathbf{M}_t, z)} P(Z = z|\mathbf{S}_{t-1}), \tag{A.8}$$

where $C = R(s_t)P(Z = z|\mathbf{S}_{t-1}) + P(Z = z|\mathbf{S}_{t-1}) - 1$.

Using the property of Eq (A.2), we can further specify that

$$P(\Lambda_{u,v}|\mathbf{S}_t) = \frac{1}{C} R(s_t)^{\delta(\mathbf{M}_t, \Lambda_{u,v})} P(\Lambda_{u,v}|\mathbf{S}_{t-1}), \tag{A.9}$$

which describes the posterior probability of any nodes in Λ and allows the ATS to grow the tree dynamically as iterations proceed in a top-down manner.

Consider that at iteration t of the ATS algorithm we have computed the score $S_{\mathbf{M}_t} = s_t$ after evaluating the set of correspondences $\mathbf{M}_t$ (associated with $\Lambda_{u,v}$). We must now compute the posterior distribution for each node in the queue. Hence, in one case the node to update is the one evaluated at iteration t (i.e. $\delta(\mathbf{M}_t, \Lambda_{u,v}) = 1$) and in the other it is not (i.e. $\delta(\mathbf{M}_t, \Lambda_{u,v}) = 0$). In the former, the posterior can be computed as

$$P(\Lambda_{u,v}|\mathbf{S}_t) = \frac{1}{C'} R(s_t) P(\Lambda_{u,v}|\mathbf{S}_{t-1}) \tag{A.10}$$

while in the latter, the posterior is computed as

$$P(\Lambda_{u',v'}|\mathbf{S}_t) = \frac{1}{C'} P(\Lambda_{u',v'}|\mathbf{S}_{t-1}), \tag{A.11}$$

where

$$C' = R(s_t)P(\Lambda_{u,v}|\mathbf{S}_{t-1}) + P(\Lambda_{u,v}|\mathbf{S}_{t-1}) - 1. \tag{A.12}$$

In conclusion, the exact posterior distribution of each node in the tree can be computed at each iteration using Eq. (A.10) and Eq. (A.11): the former for the node $\Lambda_{u,v}$ which has been queried at that iteration and the later for all other nodes in the tree. Therefore at each iteration each node must be normalized by C' and only $\Lambda_{u,v}$ (or equivalently it's children) is multiplied by $R(s_t)$.

Observation Selection

In our ATS algorithm we avoid computing the normalization constant of Eq. (A.12) for the evaluated candidates and all other elements of the queue $\mathcal{Q}$. We now show why this is appropriate.

At each time step of the ATS process, the queue $\mathcal{Q}$ maintains a set of candidates which are frontier nodes from the tree Λ expended thus far. That is, at time step t, the queue maintains tuples of $\Lambda_{u,v}$ and $P(\Lambda_{u,v}|\mathbf{S}_t)$, where none of the $\Lambda_{u,v}$ nodes have any ancestors in the queue.

The ATS selects the next node to evaluate (and expand from the queue) by selecting the node which maximizes the information gain,

$$\begin{aligned} \mathbf{M}_t &= \arg\max_{\mathbf{M}\in Q} MI(Z; S_{\mathbf{M}}|\mathbf{M}, \mathbf{S}_{t-1}) \\ &= \arg\max_{\mathbf{M}\in Q} H(Z|\mathbf{M}, \mathbf{S}_{t-1}) - H(Z|S_{\mathbf{M}}, \mathbf{M}, \mathbf{S}_{t-1}) \end{aligned} \tag{A.13}$$

where $H(\cdot)$ is the Shannon entropy and $MI(Z; S_{\mathbf{M}}|\mathbf{M}, \mathbf{S}_{t-1})$ is the mutual information between the true correspondences and the observation to make when evaluating the set $\mathbf{M}$ and having already observed $\mathbf{S}_{t-1}$. Recall that $\mathbf{M}$ is associated with unique tree node $\Lambda_{u,v}$ and by letting $\epsilon_{u,v} = P(\Lambda_{u,v}|\mathbf{S}_{t-1})$, Eq. (A.13) can be rewritten as

$$\begin{aligned} \mathbf{M}_t = \arg\max_{\mathbf{M}\in Q} \{ & H\left(\epsilon_{u,v}\mathcal{N}(s;\boldsymbol{\omega}_1^u) + (1-\epsilon_{u,v})\mathcal{N}(s;\boldsymbol{\omega}_0^u)\right) \\ & -\epsilon_{u,v} H\left(\mathcal{N}(s;\boldsymbol{\omega}_1^u)\right) - (1-\epsilon_{u,v}) H\left(\mathcal{N}(s;\boldsymbol{\omega}_0^u)\right)\}. \end{aligned} \tag{A.14}$$

Solving this optimization exactly is non trivial and in [42, 94] the Gini Index was used to approximate the Shannon entropy, leading to

$$\mathbf{M}_t \approx \arg\max_{\mathbf{M}\in Q} \{\epsilon_{u,v}(1-\epsilon_{u,v})\delta_u\} \tag{A.15}$$

where δ_u is a constant which only depends on the depth of node evaluated, u. Using the result of [107], this optimization can be further approximated without great impact on accuracy by

$$\mathbf{M}_t \approx \arg\max_{\mathbf{M}\in Q} \{\epsilon_{u,v}\}, \tag{A.16}$$

suggesting that the node with the highest posterior distribution be queried during the following iteration and can be achieved by selecting the element at the top of the queue.

Finally, given that we select nodes using Eq. (A.16), whether the queue has elements whose values sum to 1 or not does not affect the order in which the nodes are organised, since all elements in the queue have the same normalization constant, C'. For this reason, we can avoid normalizing each node at each iteration, which would be otherwise computationally intractable in large trees.

Appendix B

Graph Matching using Monte Carlo tree search implementation

In this appendix, we describe in detail the implementation of the approach Graph Matching using Monte Carlo Tree Search (GMMC), proposed in Chapter 5.

B.1 Input

The assumed input is two graphs $\mathcal{G}^A = (\mathbf{V}^A, \mathbf{E}^A)$ and $\mathcal{G}^B = (\mathbf{V}^B, \mathbf{E}^B)$ segmented from 2D or 3D images. These graphs are the result of a segmentation of graph-like structures performed pixel or voxel-wise, where their branching points and endpoint of the structure are represented by vertices $\mathbf{V}$ connected by edges $\mathbf{E}$. These edges are not only connections between vertices, but also have a specific path in the Euclidean space, and which can be mapped into this space in a continuous way.

B.2 Graph representation

We represent the graphs mentioned in the previous section by calculating the parameters described in Table B.1 for both $\mathcal{G}^A$ and $\mathcal{G}^B$. We define *superedges*

Parameters	Description
$\mathbf{V}$	Position of the nodes on $\mathbb{R}^D$
$f_{\mathbf{V}}$	Type of node
$\mathbf{E}$	Edges
path	Projection of the edge onto $\mathbb{R}^D$ as a continuous function
length	Geodesic length of the edge
$\mathbf{S}$	Superedges of the graph
$\{\mathbf{e}_1, \cdots \mathbf{e}_m\}$	Edges which compose the superedge
length	Geodesic length of the superedge
$\mathbf{h}_\Omega(\mathbf{s}_k)$	Descriptor of each superedge
$\mathbf{dA}$	Adjacency matrix of the graph with size $\lvert\mathbf{V}\rvert \times \lvert\mathbf{V}\rvert$
$\mathbf{H}$	Hop matrix of the graph with size $\lvert\mathbf{V}\rvert \times \lvert\mathbf{V}\rvert$

Table B.1: Information kept for the graph $\mathcal{G}^A$ and similarly for $\mathcal{G}^B$. All the information for each graph can be calculated independently.

as sets of at most K consecutive edges. For each superedge in $\mathbf{S}^A$ and $\mathbf{S}^B$, we calculate its respective path descriptor $\mathbf{h}_\Omega(\mathbf{s}_k)$ with the user-input parameters n_ω and $|\Omega|$.

We consider *virtual superedges* composed of a single straight edge connects two vertices of a graph, when these two graphs belong to two independent subgraphs and are separated by an Euclidean distance of less than d_c. This allows for the matching of all the graph, when the graph is composed by multiple independent subgraphs. Only superedges of the same type can be matched to each other.

This representation is done for each graph independently and can therefore be done offline.

In order to obtain a continuous function $\xi_{\mathbf{s}_k} : [0, 1] \to \mathbb{R}^D$ for each superedge $\mathbf{s}_k \in \mathbf{S}$, we maintain a discrete path of the edge, obtained from the original segmentation, and linearly interpolate to obtain continuous information in $\mathbb{R}^D$.

B.3 Algorithm

B.3.1 Tree contents and parameters

The search tree $\mathcal{T}$ introduced in the algorithm is explored using an algorithm inspired in the Monte Carlo tree search approach as introduced in Section 5.3, and here we further describe its implementation. The parameters for the algorithm are listed in Table B.2.

Each node ν of the tree $\mathcal{T}$ is composed by the values listed in Table B.3. We refer to a specific reward Q of a node ν as Q_ν and similarly for the remaining fields. The root ν_0 is a node with empty $\mathbf{M^V}$ and $\mathbf{M^S}$. Each node ν has the information of $\tilde{Q}_\nu$ and Q^+_ν, where Q^+_ν is the greatest value Q_ν in the branch below ν and

$$Q_\nu = \sum_{(\mathbf{s}^A_k, \mathbf{s}^B_l) \in \mathbf{M}^\mathbf{S}_\nu} \frac{l(\mathbf{s}^A_k) + l(\mathbf{s}^B_l)}{2} + \kappa \, \overline{l(\mathbf{S}^A, \mathbf{S}^B)} \; |\mathbf{M}^\mathbf{V}_\nu|, \tag{B.1}$$

where $\overline{l(\mathbf{S}^A, \mathbf{S}^B)}$ is the average length of the superedges and

$$\tilde{Q}_\nu = \frac{Q^+_\nu}{Q_\text{norm}} + \gamma \sqrt{\frac{2 \ln n}{n_\nu}}, \tag{B.2}$$

where n is the current iteration number and $\gamma \in [0, 1]$ is a tunable parameter and Q_norm is the maximum possible reward calculated as

$$Q_\text{norm} = \frac{l(\mathbf{E}^A) + l(\mathbf{E}^B)}{2} + \kappa \, \overline{l(\mathbf{S}^A, \mathbf{S}^B)} \; \min(|\mathbf{V}^A|, |\mathbf{V}^B|). \tag{B.3}$$

The children of the root will have precisely one element (one mapping) in $\mathbf{M^S}$ and a pair of matches in $\mathbf{M^V}$ correspondent to the endpoints of the superedges of the match in $\mathbf{M^S}$. The nodes in the all depths except the root ($d = 0$) will have $|\mathbf{M^S}| = d$ and $|\mathbf{M^V}| = d + 1$, for a depth d of the search tree. The superedge pairs represented in the mapping $\mathbf{M^S}$ have a direct relationship with $\mathbf{M^V}$ as the endpoints of the edges or superedges in $\mathbf{M^S}$ have their endpoints matched in $\mathbf{M^V}$.

For each node ν, we keep only the compatible superedge pairs in $\mathbf{S}_\nu$, which have common endpoints with the elements in $\mathbf{M}^\mathbf{S}_\nu$ or have endpoints at the vertices of $\mathbf{M}^\mathbf{V}_\nu$. However, their compatibility with the transformation model (see Section 5.2) is only checked if the superedge pair is selected to be expanded

Parameters	Description
K	Maximum number of edges in each superedge
n_ω	Number of elements of $\boldsymbol{\omega}$
$\lvert\boldsymbol{\Omega}\rvert$	Number of vectors $\boldsymbol{\omega}$ in $\mathbf{h}_\Omega(\mathbf{s}_k)$
ε_T	Maximum deformation of the transformation (see Eq. (5.5))
ε_h	Maximum deformation between path descriptors (see Eq. (5.10))
κ	Factor in the calculation of the reward Q for the number of matched nodes
γ	Factor in the calculation of the reward $\tilde{Q}$ for the importance of the number of times visited
d_c	Maximum distance for which two vertices in different graphs may be considered as a virtual superedge
N_{sim}	Number of iterations in the Simulation step
N_{exp}	Number of children to add in Expansion step
N_{it}	Maximum number of iterations

Table B.2: Summary of parameters used in the implementation of the Monte Carlo tree search algorithm, using graphs parameterized with superedges and path descriptors.

into the search tree. This check can be computationally expansive depending on the transformation model, and ignoring doing so for the superedge pairs that are never selected for expansion results on speeding up the algorithm. For simplicity we keep a list of variables $\mathbf{F}_\nu$ indicating whether each element of $\mathbf{S}_\nu$ has been yet explored.

We call ν *expandable* if the size of $\mathbf{S}_\nu$ is higher than zero (there are compatible superedges which can be added to $\mathbf{M}_\nu^{\mathbf{V}}$ and $\mathbf{M}_\nu^{\mathbf{S}}$) and if there is at least one element of $\mathbf{F}_\nu$ equal to zero (not all superedge pairs were explored and can be added as children of the node).

Algorithm 4 Graph matching using Monte Carlo Tree Search

$n \leftarrow 0$
while $n < N_{\text{it}}$ **and** $t < T_{\max}$ **and** $|\mathbf{M}_{\nu_E}^{\mathbf{V}}| < N_{\text{match}}$ **do**
  $\nu_E \leftarrow \text{Select}(\nu_0, n)$ // select most urgent node (see Alg. 5). ν_0 is the root
  if $\nu_E = \emptyset$ **or** $x(\nu_0) = 1$ **then break** // nothing to expand
  $\{\nu_B\} \leftarrow \text{Expand}(\mathcal{T}, \nu_E, N_{\text{exp}})$ // (see Alg. 6). $\{\nu_B\}$ are expanded nodes
  $\text{Simulate}(\mathcal{T}, \{\nu_B\}, N_{\text{sim}})$ // simulate node (see Alg. 7)
end while

Variables	Description
$\mathbf{M}_\nu^{\mathbf{V}}$	Nodes matched
$\mathbf{M}_\nu^{\mathbf{S}}$	Superedges matched
Q_ν	Reward for node ν
Q_ν^+	Best reward reachable from node
$\tilde{Q}_\nu$	see eq. (B.2)
$\mathbf{S}_\nu$	expandable superedge pairs connected with $M_{\mathbf{V}}$ in both graphs
$\mathbf{F}_\nu$	$f_\nu^i = \begin{cases} 1 & \text{if corresponding element of } \mathbf{S}_\nu \text{ is expanded} \\ 0 & \text{otherwise} \end{cases}, \forall f_\nu^i \in \mathbf{F}_\nu$
$\mathbf{V}_\nu$	endpoint vertices of $\mathbf{S}$
$\mathbf{K}_\nu$	skipped nodes in both graphs $\mathcal{G}^A$ and $\mathcal{G}^B$
n_ν	number of times the node has been visited
$x(\nu)$	$\begin{cases} 1 & \text{if } \nu \text{ if fully expanded} \\ 0 & \text{otherwise} \end{cases}$
$\mathbf{C}_\nu$	children of ν in the search tree

Table B.3: Information kept in each node ν of the search tree $\mathcal{T}$ used in the proposed algorithm.

Steps

The search algorithm loops through four steps: **Selection**, **Expansion**, **Simulation** and **Backpropagation**. The algorithm is described in the following paragraphs and is also is pseudocode in Algorithms 4–9.

Selection. In each iteration, we want to select the most promising node to explore. We recursively select the node with the highest $\tilde{Q}$ from the children of each node, starting from the root. We greedily descend the tree to the leaf with highest $\tilde{Q}$, and backtrack to the highest expandable $\tilde{Q}$ in case the resulting node is not expandable. We keep a flag (expanded) to indicate nodes which do not have expandable nodes under them, so that we can exclude them in step in future iterations. The pseudo-code of this step is presented in Algorithm 5.

Expansion. In the Expansion step, we add children to the selected node. The number of children added in this step is bounded by either the number of expandable superedge pairs of the node or the parameter N_{exp}.

In the selection of which superedge pair to expand into the search tree (function selectSuperedgePair($\mathbf{S}^A, \mathbf{S}^B$) in the pseudocode), the compatibility

Algorithm 5 $\nu \leftarrow \text{Select}(\nu_p, n)$

if $|\mathbf{C}_{\nu_p}| = 0$ **or** $\forall c_i \in \mathbf{C}_{\nu_p}, x(c_i) = 1$ **then** // $\mathbf{C}_{\nu_p}$ are the children of ν_p in the tree
  $x(\nu_p) \leftarrow 1$ // ν_p is fully expanded
  return $\emptyset$
end if
for $\nu_i \in \mathbf{C}_{\nu_p}$ **do**
  $\tilde{Q}_{\nu_i} \leftarrow \frac{Q^+_{\nu_i}}{Q_{\text{norm}}} + \gamma\sqrt{\frac{2\log n}{n_{\nu_i}}}$
end for
for $\nu_i \in \mathbf{C}_{\nu_p}$ in desc. order of $\tilde{Q}_{\nu_i}$, where $x(\nu_p) = 0$ **do**
  $\nu_{\text{cur}} \leftarrow \text{Select}(\nu_i)$
  if $\nu_{\text{cur}} \neq \emptyset$ **then**
    if $\tilde{Q}_{\nu_{\text{cur}}} \geq \tilde{Q}_{\nu_p}$ **or** ν_p fully expanded **then return** ν_{cur}
    else return ν_p
  end if
end for
if $\exists c_i \in \mathbf{C}_{\nu_p}, x(c_i) = 0$ **then** // there are expandable children but $\tilde{Q}_{\nu_p}$ is larger
  return ν_p
else
  $x(\nu_p) \leftarrow 1$
  return $\emptyset$
end if

of the superedge pair with the global transformation model is checked. In the proposed method that corresponds to checking Eq. 5.5 for all elements of $\mathbf{M^V}$ together with the new endpoints of the considered superedge pair i.e.

$$\frac{1}{1+\varepsilon_T} d(\mathbf{v}_i^A, \mathbf{v}_c^A) \leq d(\mathbf{v}_j^B, \mathbf{v}_c^B) \leq (1+\varepsilon_T) d(\mathbf{v}_i^A, \mathbf{v}_c^A), \quad \forall (\mathbf{v}_i^A, \mathbf{v}_j^B) \in \mathbf{M^V}, \tag{B.4}$$

for a new match $(\mathbf{v}_c^A, \mathbf{v}_c^B)$. The order in which the possible superedge pairs are considered is referred as *default order* defined in Section 5.1.1.

To create the new node based on the selected superedge pair, we take the parent node ν_E as basis for the node's children. The creation of this child node is described in Algorithm 8. The fields $\mathbf{M^V}$ and $\mathbf{M^E}$ are updated based on the selected superedge pair and using $\mathbf{M^S}$ the algorithm checks whether there exist any nodes which already consider the same $\mathbf{M^S}$. To optimize this duplicate search, the $\mathbf{M^S}$ is kept in an index ascending order. The adjacent and non-overlapping superedge pairs are updated, where some may be removed, which are incompatible with the chosen pair (because of overlapping, containing of the chosen superedges or one of the skipped vertices). The set of skipped nodes is also updated, based on previously and new skipped nodes.

Algorithm 6 $\mathcal{S} \leftarrow \text{Expand}(\mathcal{T}, \nu_E, N_{\text{exp}})$

$\mathcal{S} \leftarrow \emptyset$ // The expansion is different depending if ν_E is the root or not
if $\nu_E = \nu_0$ **then**
  for $k \in \{1, N_{\text{exp}}\}$ **do**
    $\mathbf{S}^i_{\nu_0} = (\mathbf{s}^A_k, \mathbf{s}^B_l) \leftarrow \text{selectSuperedgePairFromRoot}(\mathbf{S}^A, \mathbf{S}^B)$ // in *default* order
    $\nu \leftarrow \text{cloneNode}(\nu_0)$ // new child based on parent
    $M^{\mathbf{V}}_\nu \leftarrow$ endpoints of the pair $\mathbf{S}^i_{\nu_0}$
    $M^{\mathbf{S}}_\nu \leftarrow \mathbf{S}^i_{\nu_0}$
    $n_\nu \leftarrow 1$
    $\nu \leftarrow \text{updateAdjacentPaths}(\nu)$ // update paths adjacent to superedges in $\mathbf{M^S}$
    $\text{skipped}_\nu \leftarrow \text{getSkippedNodes}(M^{\mathbf{S}}_\nu, \mathcal{G}^A, \mathcal{G}^B)$ // refresh skipped nodes
    $Q_\nu \leftarrow \frac{l(\mathbf{M^S})}{2} + \kappa\ \overline{l(\mathbf{S})}\ |M^{\mathbf{V}}_\nu|$
    $Q^+_\nu \leftarrow Q_\nu$
    $\tilde{Q}_\nu \leftarrow \frac{Q^+_\nu}{Q_{\text{norm}}} + \gamma\sqrt{\frac{2\log n}{n_\nu}}$
    $n \leftarrow n + 1$
    $\mathcal{T} \leftarrow \text{add}(\mathcal{T}, \nu)$
    $\mathcal{T} \leftarrow \text{Backpropagate}(\mathcal{T}, \nu)$
    $\mathcal{S} \leftarrow \{\mathcal{S}, \nu\}$ // nodes to be used in Simulation step
  end for
else
  for $k \in \{1, N_{\text{exp}}\}$ **do**
    $\mathbf{S}^i_{\nu_E} = (\mathbf{s}^A_k, \mathbf{s}^B_l) \leftarrow \text{selectSuperedgePair}(\mathbf{S}_{\nu_E}, e_{\nu_E})$ // in *default* order
    $e^i_{\nu_E} \leftarrow 1$ // explored superedge pair
    $\nu \leftarrow \text{cloneNode}(\nu_E)$ // new child based on parent
    $\nu \leftarrow \text{CreateChild}(\mathcal{T}, \nu, \mathbf{s}^A_k, \mathbf{s}^B_l)$ // defined in Alg. 8
    if $|\mathbf{C}_\nu| = 0$ **then** $x(\nu) \leftarrow 1$ // avoid future selection if not expandable
    if not duplicate **then** $\mathcal{T} \leftarrow \text{add}(\mathcal{T}, \nu)$
    $\mathcal{T} \leftarrow \text{Backpropagate}(\mathcal{T}, \nu)$
    $n \leftarrow n + 1$
    $\mathcal{S} \leftarrow \{\mathcal{S}, \nu\}$ // nodes to be used in Simulation step
  end for
end if

We then calculate $Q(\nu)$ associated to each child and set the number of times the node has been visited to 1, and add the node to the search tree. A list of the nodes added to the search tree is saved, so that the next step (Simulation) is performed on each of the expanded nodes.

The pseudocode of this step is presented in Algorithm 6.

Simulation. This step is done to obtain an estimate of the value of $\tilde{Q}$ for the selected search node. This is done in a greedy way, by iteratively (N_{sim} times) selecting the superedge pair in default order, creating the corresponding new child and recursively repeating the procedure with the new node. This results in a greedy depth-first search exploration of the initial node, so that an estimate of $\tilde{Q}$ can be obtained.

The procedure the new node is added to the search tree is similar to the procedure in the expansion step, and it is described in Algorithm 8.

The pseudocode of this step is presented in Algorithm 7.

Algorithm 7 Simulate($\mathcal{T}, \mathcal{S}, \mathcal{P}_s, N_{\text{sim}}$) – Creates nodes in depth first fashion from ν_E

for $\nu \in \mathcal{S}$ **do**
  for $k \in \{1, N_{\text{sim}}\}$ **do**
    if $x(\nu) = 1$ **then break** // no superedge pairs available for expansion
    $\mathbf{S}_\nu^i = (\mathbf{s}_k^A, \mathbf{s}_l^B) \leftarrow$ selectSuperedgePair(s_ν, e_ν) // longest pair with least number of edges in superedges
    $\nu \leftarrow$ CreateChild$(\mathcal{T}, \nu, \mathbf{s}_k^A, \mathbf{s}_l^B)$
    $e_\nu^i \leftarrow 1$ // explored superedge pair
    if not duplicate **then** $\mathcal{T} \leftarrow$ add$(\mathcal{T}, \nu)$
    $\mathcal{T} \leftarrow$ Backpropagate$(\mathcal{T}, \nu)$
    $n \leftarrow n + 1$
  end for
end for

Algorithm 8 $\nu \leftarrow$ CreateChild$(\mathcal{T}, \nu, \mathbf{s}_k^A, \mathbf{s}_l^B)$

$\mathbf{v}_s^A, \mathbf{v}_t^B \leftarrow$remainingEndpoints$(\mathbf{s}_k^A, \mathbf{s}_l^B)$
$M_\nu^{\mathbf{V}} \leftarrow M_\nu^{\mathbf{V}} \cup (\mathbf{v}_s^A, \mathbf{v}_t^B)$ // append the endpoints of the pair s
$M_\nu^{\mathbf{S}} \leftarrow M_\nu^{\mathbf{S}} \cup (\mathbf{s}_k^A, \mathbf{s}_l^B)$
if $\nu_D \leftarrow$ alreadyInTree$(\mathcal{T}, M_\nu^{\mathbf{V}})$ **then return** ν_D // checks for duplicate nodes
$\nu \leftarrow$ updateAdjacentPaths(ν) // update paths adjacent to superedges in $\mathbf{M}^{\mathbf{S}}$
skipped$_\nu \leftarrow$ getSkippedNodes$(M_\nu^{\mathbf{S}}, \mathcal{G}^A, \mathcal{G}^B)$ // refresh skipped nodes
$Q_\nu \leftarrow \sum_{(\mathbf{s}_k^A, \mathbf{s}_l^B) \in M_\nu^{\mathbf{S}}} \frac{l(\mathbf{s}_k^A) + l(\mathbf{s}_l^B)}{2} + \kappa \; \overline{l(\mathbf{S})} \; |M_\nu^{\mathbf{V}}|$
$Q_\nu^+ \leftarrow Q_\nu$
$n_\nu \leftarrow 1$
$\tilde{Q}_\nu \leftarrow \frac{Q_\nu^+}{Q_{\text{norm}}} + \gamma \sqrt{\frac{2 \log n}{n_\nu}}$

Backpropagation. This step done when adding a node in both expansion and simulation steps consists in simply replacing Q^+ in the nodes in the path

from the new node to the root, when the value is below the new node's Q. The pseudocode of this step is presented in Algorithm 9.

Algorithm 9 Backpropagate($\mathcal{T}, \nu$)

```
Q_ν^+ ← Q_ν
ν_P ← Parent(ν)
while ν_P ≠ ∅ do
    if Q_{ν_P}^+ < Q_ν then
        Q_{ν_P}^+ ← Q_ν
    end if
    ν_P ← Parent(ν_P)
end while
```

The algorithm loops through Selection, Expansion and Simulation until either all nodes have empty list of available expandable superedge pairs, after reaching a predefined match size N_{match}, or the maximum number of iterations N_{it} or maximum processing time T_{max} is reached. The output is the node corresponding to the highest reward Q obtained.

Appendix C

Graph Matching using Monte Carlo tree search code

In this appendix, we describe in detail the available code for the approach Graph Matching using Monte Carlo Tree Search (GMMC), proposed in Chapter 5.

The code is available at `https://gitlab.fel.cvut.cz/amavemig/gmmc_code`. To clone the repository, use the command

```
git clone git@gitlab.fel.cvut.cz:amavemig/gmmc_code.git
```

The implementation is written in C++, with a wrapper in MATLAB.

C.1 Compilation

The only dependency of the code is the Eigen library. The library is used for matrix operations.

To compile the code use the script `compile_gmmc_mex.m`. The variable `eigen_dir` must be changed to the path pointing to the location of the Eigen headers.

C.2 Usage

An example of how to run the function is provided in the script `run_gmmc_mex.m`.

The current code loads the graphs from SWC files with a single tree in each file. SWC files are commonly used in medical imaging. They define tree structures, usually neuronal data. Different implementations must be created if the input is from a different format. The internal graph representation is a MATLAB structure with the following fields:

- `nodes`: $n \times D$ matrix with vertices of the graph
- `edges`: $1 \times |\mathbf{E}|$ array of structures with the following format:
  - `endpoints`: 1×2 array with indices of respective endpoint vertices
  - `path`: $D \times m$ matrix with path of the edge
  - `cumd`: $1 \times m$ array with cumulative distance of the path
  - `length`: total length of the path
- `dA`: $n \times n$ sparse adjacent matrix of the graph
- `fnodes`: $n \times 1$ array with labels for each vertex
- `hm`: $n \times n$ hop matrix of the graph
- `sdA`: $n \times n$ matrix with the superedge which connects the vertices with given indices
- `superedges`: $1 \times |\mathbf{S}|$ array of structures with the following format:
  - `path`: $1 \times (s+1)$ array with indices of the vertices through which the superedge has its path
  - `edgeid`: $1 \times s$ array with indices of the respective edges of the superedge
  - `length`: total length of the superedge
  - `desc`: $1 \times |\Omega|$ path descriptor

where n is the number of vertices of the graph, D is the number of dimensions, m is the length of a given path of an edge and s is the number of edges of a given superedge.

Parameter name	Corresponding variable	Description
	Graph pre-processing	
`n_w`	n_ω	Number of sampling points on a superedge
`Omega`	Ω	Number of descriptor elements
`K`	K	Maximum size of superedge (number of edges)
`sr`	—	Sampling rate for input data
`d_c`	d_c	Maximum virtual edge size (graphs are normalized)
`interpolfun`	—	Interpolation function of the superedges (linear recommended)
`kappa`	κ	Weight on reward or objective function
`gamma`	γ	Weight on urgency (for number of times node is visited)
`N_exp`	N_{exp}	Number of children added in Expansion
`N_sim`	N_{sim}	Number of search tree nodes added in Simulation
`epsilon_T`	ε_T	Allowed deformation in transformation model
`epsilon_h`	ε_h	Allowed deformation between superedges
`N`	N_{it}	Maximum number of iterations
`termDepth`	N_{match}	Number of matches at which the method stops
`timeLimit`	T_{max}	Maximum time in seconds for the method to stop
`simulateAll`	—	Simulate each of the expanded nodes
`modelcheck`	—	0 – Check model for each possible superedge pair 1 – Check model only when selecting node to explore
`transfmodel`	—	Transformation model considered (0 for the proposed method)
`verbose`	—	Verbose of the method (0 for reduced verbose, 1 for more details)

Table C.1: List of parameters which can be changed in the implementation of the GMMC algorithm. For more details on each parameter, refer to Chapter 5.

The parameters of the algorithm can be tuned, using the structure `params`. The list of available parameters including the respective descriptions is available in Table C.1.

C.3 Implementation

The core of the implementation of GMMC is done in C++. Below is an overview of the functions and classes implemented with each file:

- `Graph.h`: Class for the graph structure.
- `MatrixOp.h`: Implements several matrix operations.
- `MCT.h`: Class for the search tree and for the nodes in the search tree.
- `mex_run_gmmc.cpp`: Interface between MATLAB and the core code.
- `Superedge.h`: Class for a superedge and a single edge. Also implements the code for edge interpolation.
- `TransModel.h`: Implements several functions with respect to the transformation model: checking whether a new match is compatible, updating the model and removing incompatibilities. Some functions are only used depending on the value of `modelcheck` of the parameters (see Table C.1).
- `TreeSearch.h`: Implements the main functions of the Monte Carlo tree search: Selection, Expansion, Simulation and Backpropagation, as well as the main loop in the function `runTreeSearch`.
- `UniqueMv.h`: Functions handling a unique matching, so that the algorithm easily detects duplicates.

The files are also commented, so that further details can be obtained for each function and variable used in the code.

Bibliography

[1] Maher Al-Khaiyat and Farhad Kamangar. Planar curve representation and matching. In *British Machine Vision Conference (BMVC)*, 1998.

[2] Khalid A. Al-Kofahi, Sharie Lasek, Donald H. Szarowski, Christopher J. Pace, George Nagy, James N. Turner, and Badrinath Roysam. Rapid Automated Three-Dimensional Tracing of Neurons from Confocal Image Stacks. *IEEE Transactions on Information Technology in Biomedicine*, 6(2):171–187, 2002.

[3] Brian Amberg, Sami Romdhani, and Thomas Vetter. Optimal step nonrigid ICP algorithms for surface registration. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8, 2007.

[4] G. Ascoli, K. Svoboda, and Y. Liu. Digital Reconstruction of Axonal and Dendritic Morphology DIADEM Challenge, 2010.

[5] Erhan Bas and Deniz Erdogmus. Principal Curves as Skeletons of Tubular Objects - Locally Characterizing the Structures of Axons. *Neuroinformatics*, 9(2-3):181–191, 2011.

[6] Ronen Basri, Luiz Costa, Davi Geiger, and David Jacobs. Determining the similarity of deformable shapes. *Vision Research*, 38:135–143, 1998.

[7] Serge Belongie, Jitendra Malik, and Jan Puzicha. Shape matching and object recognition using shape contexts. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 24:509–522, 2001.

[8] Fethallah Benmansour, Engin Türetken, and Pascal Fua. Tubular Geodesics: Fiji plugins, 2012. http://cvlab.epfl.ch/software/delin/index.php.

[9] H. BenShitrit, J. Berclaz, F. Fleuret, and P. Fua. Multi-Commodity Network Flow for Tracking Multiple People. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2013.

[10] Paul J. Besl and Neil D. McKay. A method for registration of 3-D shapes. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 14(2):239–256, 1992.

[11] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[12] Otakar Borůvka. O Jistém Problému Minimálním (About a Certain Minimal Problem) (in Czech, German summary). *Práce Mor. Prírodoved. Spol. v Brne III*, 3, 1926.

[13] Cameron Browne, Edward J. Powley, Daniel Whitehouse, Simon M. Lucas, Peter I. Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez, Spyridon Samothrakis, and Simon Colton. A survey of Monte Carlo tree search methods. *IEEE Trans. Comput. Intellig. and AI in Games*, 4(1):1–43, 2012.

[14] Kevin Buchin, Maike Buchin, and Yusu Wang. Exact algorithms for partial curve matching via the Fréchet distance. In *Proceedings of the Twentieth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '09, pages 645–654, Philadelphia, PA, USA, 2009. Society for Industrial and Applied Mathematics.

[15] Hongmin Cai, Xiaoyin Xu, Ju Lu, Jeff W. Lichtman, S.P. Yung, and Stephen T.C. Wong. Repulsive Force Based Snake Model to Segment and Track Neuronal Axons in 3D Microscopy Image Stacks. *NeuroImage*, 32(4):1608–1620, August 2006.

[16] Ali Can, Hong Shen, James N. Turner, Howard L. Tanenbaum, and Badrinath Roysam. Rapid Automated Tracing and Feature Extraction from Retinal Fundus Images Using Direct Exploratory Algorithms. *IEEE Trans. on Information Technology in Biomedicine*, 3(2):125–138, June 1999.

[17] Ali Can, Charles V. Stewart, Badrinath Roysam, and Howard L. Tanenbaum. A feature-based, robust, hierarchical algorithm for registering pairs of images of the curved human retina. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 24:347–364, 2002.

[18] Arnaud Charnoz, Vincent Agnus, Grégoire Malandain, Luc Soler, and Mohamed Tajine. Tree matching applied to vascular system. *Graph-Based Representations in Pattern Recognition*, pages 183–192, 2005.

[19] Dmitry Chetverikov, Dmitry Stepanov, and Pavel Krsek. Robust Euclidean alignment of 3D point sets: the trimmed iterative closest point algorithm. *Image Vision Computing*, 23(3):299–309, 2005.

[20] Tat-Jun Chin, Jin Yu, and David Suter. Accelerated hypothesis generation for multistructure data via preference analysis. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 34(4):625–638, April 2012.

[21] Margarita Chli and Andrew J. Davison. Active matching. *European Conference on Computer Vision (ECCV)*, pages 72–85, 2008.

[22] Minsu Cho and Kyoung Mu Lee. Progressive graph matching: Making a move of graphs via probabilistic voting. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 398–405, 2012.

[23] Sunglok Choi, Taemin Kim, and Wonpil Yu. Performance Evaluation of RANSAC Family. *British Machine Vision Conference*, pages 1–12, 2009.

[24] Paarth Chothani, Vivek Mehta, and Armen Stepanyants. Automated Tracing of Neurites from Light Microscopy Stacks of Images. *Neuroinformatics*, 9:263–278, 2011.

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

[26] Ondřej Chum and Jiří Matas. Matching with PROSAC – progressive sample consensus. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 220–226, 2005.

[27] Timothee Cour, Praveen Srinivasan, and Jianbo Shi. Balanced graph matching. *Neural Information Processing Systems*, pages 313–320, 2006.

[28] Ming Cui, John Femiani, Jiuxiang Hu, Peter Wonka, and Anshuman Razdan. Curve matching for open 2D curves. *Pattern Recognition Letters*, 30(1):1 – 10, 2009.

[29] Kexin Deng, Jie Tian, Jian Zheng, Xing Zhang, Xiaoqian Dai, and Min Xu. Retinal fundus image registration via vascular structure graph matching. *Journal of Biomedical Imaging*, 2010.

[30] Thomas Dietenbeck, François Varray, Jan Kybic, Olivier Basset, and Christian Cachard. Neuromuscular fiber segmentation through particle filtering and discrete optimization. In *SPIE Medical Imaging*, pages 90340B–10, San Diego, USA, 2014.

[31] Luke Domanski, Changming Sun, Raquibul Hassan, Pascal Vallotton, and Dadong Wang. Linear Feature Detection on GPUs. In *Digital Image Computing: Techniques and Applications (DICTA)*, 2010.

[32] D. Donohue and G. Ascoli. Automated Reconstruction of Neuronal Morphology: An Overview. *Brain Research Reviews*, 67:94–102, 2011.

[33] Christophe Duhamel, Luis Gouveia, Pedro Moura, and Mauricio Souza. Models and Heuristics for a Minimum Arborescence Problem. *Networks*, 51(1):34–47, 2008.

[34] Olof Enqvist, Klas Josephson, and Fredrik Kahl. Optimal correspondences from pairwise constraints. *IEEE International Conference in Computer Vision (ICCV)*, 2009.

[35] Jeff Erickson. Local polyhedra and geometric graphs. *Computational Geometry*, 31(1–2):101 – 125, 2005. Special Issue on the 19th Annual Symposium on Computational Geometry (SoCG).

[36] Denis Che Keung Fan. *Bayesian Inference of Vascular Structure from Retinal Images*. PhD thesis, Dept. of Computer Science, U. of Warwick, Coventry, UK, 2006.

[37] Martin A. Fischler and Robert C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24:381–395, 1981.

[38] Martin A. Fischler, Jay M. Tenenbaum, and Helen C. Wolf. Detection of Roads and Linear Structures in Low-Resolution Aerial Imagery Using a Multisource Knowledge Integration Technique. *Computer Graphics and Image Processing*, 15(3):201–223, March 1981.

[39] A.F. Frangi, W.J. Niessen, K.L. Vincken, and M.A. Viergever. Multiscale Vessel Enhancement Filtering. *Lecture Notes in Computer Science*, 1496:130–137, 1998.

[40] Max Frenkel and Ronen Basri. Curve matching using the fast marching method. In *Energy Minimization Methods in Computer Vision and Pattern Recognition*, pages 35–51, 2003.

[41] Huijing Fu, Zheng Tian, Maohua Ran, and Ming Fan. Novel affine-invariant curve descriptor for curve matching and occluded object recognition. *IET Computer Vision*, 7(4):279–292, August 2013.

[42] Donald Geman and Bruno Jedynak. An active testing model for tracking roads in satellite images. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 18(1):1–14, 1996.

[43] Przemysław Głowacki, Miguel Amável Pinheiro, Agata Mosinska, Engin Türetken, Daniel Lebrecht, Raphael Sznitman, Anthony Holtmaat, Jan Kybic, and Pascal Fua. Reconstructing evolving tree structures in time lapse sequences by enforcing time-consistency. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 2015 (submitted).

[44] Przemysław Głowacki, Miguel Amável Pinheiro, Engin Türetken, Raphael Sznitman, Daniel Lebrecht, Jan Kybic, Anthony Holtmaat,

and Pascal Fua. Reconstructing evolving tree structures in time lapse sequences. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3035–3042, June 2014.

[45] Steven Gold and Anand Rangarajan. A graduated assignment algorithm for graph matching. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 18:377–388, April 1996.

[46] Steven Gold, Anand Rangarajan, Chien Ping Lu, and Eric Mjolsness. New algorithms for 2D and 3D point matching: Pose estimation and correspondence. *Pattern Recognition*, 31:957–964, 1997.

[47] Steven Gold, Anand Rangarajan, Chien Ping Lu, and Eric Mjolsness. New Algorithms for 2D and 3D Point Matching: Point Estimation and Correspondence. *Pattern Recognition*, 31(8):1019–1031, 1998.

[48] G. Gonzalez, F. Aguet, F. Fleuret, M. Unser, and P. Fua. Steerable Features for Statistical 3D Dendrite Detection. In *International Conference on Medical Image Computing and Computer Assisted Intervention (MICCAI)*, pages 625–32, September 2009.

[49] Michael W. Graham and William E. Higgins. Optimal graph-theoretic approach to 3D anatomical tree matching. In *International Symposium on Biomedical Imaging: Nano to Macro (ISBI)*, pages 109–112, April 2006.

[50] Mordechai Haklay and Patrick Weber. Openstreetmap: User-generated street maps. *Pervasive Computing, IEEE*, 7(4):12–18, 2008.

[51] Anthony Holtmaat, Jerome Randall, and Michele Cane. Optical imaging of structural and functional synaptic plasticity in vivo. *European Journal of Pharmacology*, 719(1–3):128 – 136, 2013.

[52] M. Jacob and M. Unser. Design of Steerable Filters for Feature Detection Using Canny-Like Criteria. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(8):1007–1019, August 2004.

[53] Bing Jian and B.C. Vemuri. Robust point set registration using gaussian mixture models. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 33(8):1633–1645, 2011.

[54] C. Kirbas and F. Quek. Vessel Extraction Techniques and Algorithms: a Survey. In *IEEE Symposium on BioInformatics and BioEngineering*, page 238, 2003.

[55] Radim Kolář, Vratislav Harabiš, and Jan Odstrčilík. Hybrid retinal image registration using phase correlation. *Imaging Science Journal*, 61(4):369–384, 2013.

[56] Max W. K. Law and Albert C. S. Chung. Three Dimensional Curvilinear Structure Detection Using Optimally Oriented Flux. In *European Conference on Computer Vision*, 2008.

[57] Max W. K. Law and Albert C. S. Chung. An Oriented Flux Symmetry Based Active Contour Model for Three Dimensional Vessel Segmentation. In *European Conference on Computer Vision*, pages 720–734, 2010.

[58] Ta-Chih Lee, Rangasami L. Kashyap, and Chong-Nam Chu. Building Skeleton Models via 3D Medial Surface Axis Thinning Algorithms. *CVGIP: Graphical Models and Image Processing*, 56(6):462–478, 1994.

[59] Marius Leordeanu and Martial Hebert. A Spectral Technique for Correspondence Problems Using Pairwise Constraints. *IEEE International Conference in Computer Vision (ICCV)*, 2:1482–1489, 2005.

[60] Marius Leordeanu, Martial Hebert, and Rahul Sukthankar. An integer projected fixed point method for graph matching and MAP inference. *Conference on Neural Information Processing Systems (NIPS)*, pages 1114–1122, 2009.

[61] Hao Li, Robert W Sumner, and Mark Pauly. Global correspondence optimization for non-rigid registration of depth scans. *Computer graphics forum*, 27(5):1421–1430, 2008.

[62] Hua Li and Anthony Yezzi. Vessels as 4-D Curves: Global Minimal 4D Paths to Extract 3-D Tubular Surfaces and Centerlines. *Trans. in Medical Imaging*, 26(9):1213–1223, 2007.

[63] Qing Li, Zhigang Deng, Yong Zhang, Xiaobo Zhou, U. Valentin Nagerl, and Stephen T. C. Wong. A Global Spatial Similarity Optimization Scheme to Track Large Numbers of Dendritic Spines in Time-Lapse Confocal Microscopy. *Trans. in Medical Imaging*, 30(3):632–641, 2011.

[64] Stan Z. Li. Similarity Invariants for 3D Space Curve Matching. In *Asian Conference on Computer Vision (ACCV)*, pages 454–457, 1993.

[65] Mark H. Longair, Dean A. Baker, and J. Douglas Armstrong. Simple neurite tracer: Open source software for reconstruction, visualization and analysis of neuronal processes. *Bioinformatics*, 2011.

[66] David J.C. MacKay. Introduction to gaussian processes, 1998.

[67] E. Meijering. Neuron Tracing in Perspective. *Cytometry Part A*, 77(7):693–704, 2010.

[68] Bruno T. Messmer and Horst Bunke. A New Algorithm for Error-Tolerant Subgraph Isomorphism Detection. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 20:493–504, 1998.

[69] I. Mikic, S. Krucinski, and J. Thomas. Segmentation and tracking in echocardiographic sequences: Active contours guided by optical flow estimates. *Trans. in Medical Imaging*, 17:274–284, 1998.

[70] James Munkres. Algorithms for the Assignment and Transportation Problems. *Journal of the Society of Industrial and Applied Mathematics*, 5(1):32–38, March 1957.

[71] Andriy Myronenko and Xubo Song. Point set registration: Coherent point drift. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 32(12):2262–2275, 2010.

[72] Tomáš Pajdla and Luc Van Gool. Matching of 3-D curves using semi-differential invariants. In *IEEE International Conference on Computer Vision*, pages 390–395, 1995.

[73] Hanchuan Peng, Fuhui Long, and Gene Myers. Automatic 3D Neuron Tracing Using All-Path Pruning. *Bioinformatics*, 27(13):239–247, 2011.

[74] Miguel Amável Pinheiro and Jan Kybic. Path descriptors for geometric graph matching and registration. *International Conference on Image Analysis and Recognition*, pages 3–11, 2014.

[75] Miguel Amável Pinheiro and Jan Kybic. Geometrical Graph Matching using Monte Carlo Tree Search. *IEEE International Conference in Image Processing (ICIP)*, pages 3145–3149, September 2015.

[76] Miguel Amável Pinheiro, Jan Kybic, and Pascal Fua. Graph matching using Monte Carlo tree search. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 2016 (early access).

[77] Miguel Amável Pinheiro, Raphael Sznitman, Eduard Serradell, Jan Kybic, Francesc Moreno-Noguer, and Pascal Fua. Active testing search for point cloud matching. *Information Processing in Medical Imaging*, pages 572–583, 2013.

[78] Chandrasekhar Pisupati, Lawrence B. Wolff, Wayne Mitzner, and Elias A. Zerhouni. Geometric Tree Matching with Applications to 3D Lung Structures. In *Computational Geometry*, 1996.

[79] Josien P. W. Pluim, J. B. Antoine Maintz, and Max A. Viergever. Mutual-information-based registration of medical images: a survey. *IEEE Transactions on Medical Imaging*, 22(8):986–1004, Aug 2003.

[80] Madeline Pool, Joachim Thiemann, Amit Bar-Or, and Alyson E. Fournier. Neuritetracer: A Novel ImageJ Plugin for Automated Quantification of Neurite Outgrowth. *Journal of Neuroscience Methods*, 168(1):134–139, 2008.

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

[82] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press, 2005.

[83] Y. Sato, S. Nakajima, H. Atsumi, T. Koller, G. Gerig, S. Yoshida, and R. Kikinis. 3D Multi-Scale Line Filter for Segmentation and Visualization of Curvilinear Structures in Medical Images. *Medical Image Analysis*, 2:143–168, June 1998.

[84] Johannes Schindelin, Ignacio Arganda-Carreras, Erwin Frise, Verena Kaynig, Mark Longair, Tobias Pietzsch, Stephan Preibisch, Curtis Rueden, Stephan Saalfeld, Benjamin Schmid, Jean-Yves Tinevez, Daniel James White, Volker Hartenstein, Kevin Eliceiri, Pavel Tomancak, and Albert Cardona. Fiji: an open-source platform for biological-image analysis. *Nat Methods*, 9(7):676–682, July 2012.

[85] Guy L. Scott and H. Christopher Longuet-Higgins. An algorithm for associating the features of two patterns. In *Proc. Royal Society London*, volume B244, pages 21–26, 1991.

[86] Thomas B. Sebastian, Philip N. Klein, Benjamin B. Kimia, and Joseph J. Crisco. Constructing 2D Curve Atlases. *IEEE Workshop on Mathematical Methods in Biomedical Image Analysis*, pages 70–77, 2000.

[87] Eduard Serradell, Przemysław Głowacki, Jan Kybic, Francesc Moreno-Noguer, and Pascal Fua. Robust non-rigid registration of 2D and 3D graphs. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 996–1003, 2012.

[88] Eduard Serradell, Francesc Moreno-Noguer, Jan Kybic, and Pascal Fua. Robust elastic 2D/3D geometric graph matching. *SPIE Medical Imaging*, 8314(1):831408–831408–8, 2012.

[89] Eduard Serradell, Miguel Amável Pinheiro, Raphael Sznitman, Jan Kybic, Francesc Moreno-Noguer, and Pascal Fua. Non-rigid graph registration using active testing search. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 37(3):625–638, 2015.

[90] Eduard Serradell, Adriana Romero, Ruben Leta, Carlo Gatta, and Francesc Moreno-Noguer. Simultaneous correspondence and non-rigid 3D reconstruction of the coronary tree from single x-ray images. In *IEEE International Conference on Computer Vision*, pages 850–857, Nov 2011.

[91] Amos Sironi, Vincent Lepetit, and Pascal Fua. Multiscale Centerline Detection by Learning a Scale-Space Distance Transform. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.

[92] Dirk Smeets, Pieter Bruyninckx, Johannes Keustermans, Dirk Vandermeulen, and Paul Suetens. Robust matching of 3D lung vessel trees. In *The Third International Workshop on Pulmonary Image Analysis*, pages 61–70, 2010.

[93] Kaiqiong Sun, Nong Sang, and Tianxu Zhang. Marked Point Process for Vascular Tree Extraction on Angiogram. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 467–478, 2007.

[94] Raphael Sznitman and Bruno Jedynak. Active testing for face detection and localization. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 32(10):1914–1920, 2010.

[95] Ben Tordoff and David W. Murray. Guided sampling and consensus for motion estimation. In *European Conference on Computer Vision*, pages 82–98, 2002.

[96] Lorenzo Torresani, Vladimir Kolmogorov, and Carsten Rother. Feature correspondence via graph matching: Models and global optimization. In *European Conference on Computer Vision*, pages 596–609, 2008.

[97] Engin Türetken, Fethallah Benmansour, and Pascal Fua. Automated reconstruction of tree structures using path classifiers and mixed integer programming. *IEEE International Conference in Computer Vision*, pages 566–573, 2012.

[98] Engin Türetken, Germán González, Christian Blum, and Pascal Fua. Automated Reconstruction of Dendritic and Axonal Trees by Global Optimization with Geometric Priors. *Neuroinformatics*, 9(2-3):279–302, 2011.

[99] Michael Unser. Splines: A perfect fit for medical imaging. *Progress in Biomedical Optics and Imaging*, 3:225–236, 2002.

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

[101] Yu Wang, Arunachalam Narayanaswamy, and Badrinath Roysam. Novel 4D Open-Curve Active Contour and Curve Completion Approach for Automated Tree Structure Extraction. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1105–1112, 2011.

[102] Zhiheng Wang, Hongmin Liu, and Fuchao Wu. Image Content Based Curve Matching Using HMCD Descriptor. *Asian Conference on Computer Vision (ACCV)*, pages 448–455, 2010.

[103] Susan L. Wearne, Alfredo Rodriguez, Douglas B. Ehlenberger, Anne B. Rocher, Scott C. Henderson, and Patrick R. Hof. New Techniques for Imaging, Digitization and Analysis of Three-Dimensional Neural Morphology on Multiple Scales. *Neuroscience*, 136(3):661–680, 2005.

[104] Jan D. Wegner, Javier A. Montoya-Zegarra, and Konrad Schindler. A Higher-Order CRF Model for Road Network Extraction. *IEEE*

Conference on Computer Vision and Pattern Recognition (CVPR), 2013.

[105] Haim J. Wolfson. On curve matching. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 12(5):483–489, 1990.

[106] Xin Yu, Jinwen Tian, and Jian Liu. Transformation model estimation for point matching via gaussian processes. *World Congress on engineering*, 1, 2007.

[107] Alan L. Yuille and James Coughlan. Twenty questions, focus of attention, and A*: A theoretical comparison of optimization strategies. In *International Workshop on Energy Minimization Methods in CVPR*, pages 197–212, 1997.

[108] Mikhail Zaslavskiy, Francis Bach, and Jean-Philippe Vert. A path following algorithm for the graph matching problem. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 31(12):2227–2242, 2009.

[109] Ting Zhao, Jun Xie, Fernando Amat, Nathan Clack, Parvez Ahammad, Hanchuan Peng, Fuhui Long, and Eugene Myers. Automated Reconstruction of Neuronal Morphology Based on Local Geometrical and Global Structural Models. *Neuroinformatics*, 9:247–261, May 2011.

[110] Barbara Zitová and Jan Flusser. Image registration methods: a survey. *Image and Vision Computing*, 21:977–1000, 2003.