Geometric Graph Matching Using Monte Carlo Tree Search

Miguel Amável Pinheiro, Jan Kybic, *Senior Member, IEEE*, and Pascal Fua, *Fellow, IEEE*

Abstract—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. 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.

Index Terms—Geometric graph matching, Monte Carlo tree search, image registration, curve descriptor

1 Introduction

Many linear structures can be naturally represented as geometric graphs. Examples include road and street networks in aerial images, blood vessels in retinal scans, bronchi in angiograms, and neuronal fibers in microscopy image stacks. The images often lack easily matchable features other than the linear structures themselves. This makes it impractical to use traditional registration techniques that rely on maximizing image similarity [1], [2], [3]. In many cases, standard feature-based methods cannot be used either, as local neighborhoods are not sufficiently distinctive, for example because many road intersections look alike. We therefore propose to register these images by matching the extracted geometrical graphs. For an example, see Fig. 1.

In earlier work, we described an *active testing search* (ATS) method [4], which involved progressively refining the geometrical transformation model as more correspondences are added. The model was used to explore the set of all possible correspondences starting with the most likely ones, which allowed convergence at an acceptable computational cost even though no appearance information is used. ATS outperforms a number of state-of-the-art techniques [5], [6], [7] and is usable for graphs of up to about 200 nodes. However, for larger graphs its computational cost becomes prohibitive.

Here, we propose a different approach that can handle graphs with thousands of vertices, even in the presence of substantial deformations. We formulate the problem as a single player game and use a Monte Carlo tree search (MCTS) [8] to automatically balance between *exploration* (finding new partial matches) and *exploitation* (extending previously found ones). We will refer to our method as *graph matching using Monte Carlo tree search* (GMMC). We follow the graph edges to grow the matches incrementally and we speed-up the computation by using curve descriptors [9] to distinguish compatible edges from incompatible ones and to prune the search tree. Finally, we use an implicit geometric deformation model, which is much faster to evaluate than the Gaussian processes of ATS.

We demonstrate the effectiveness of our approach using aerial images of roads networks, blood vessels in retinal scans, bronchi in angiograms, and neurites in microscopy image stacks. We also experimented with several synthetically generated datasets to test the method against various types of transformations, size of the graphs and dependence on the parameters of the algorithm.

Our key contributions are the use of the Monte Carlo tree search along with our fast and efficient curve descriptors and implicit transformation model. These ideas were first introduced in earlier conference papers [9], [10]. Here we brought them together, described the algorithms in more detail, and conducted much more extensive experiments on both synthetic and real data.

2 Related Work

Our method matches points and line-features across images. We briefly review other approaches for solving this task.

2.1 Point Matching

By neglecting the edges, we can reduce our problem to point cloud matching [11]. Optionally, points may be sampled along the edges, which makes the point cloud denser at the expense of the uniqueness of the matching. If the

- *M.A. Pinheiro and J. Kybic are with the Center for Machine Perception, Department of Cybernetics, Faculty of Electrical Engineering, Czech Technical University in Prague, Prague 6 166 36, Czech Republic. E-mail: amavemig@cmp.felk.cvut.cz, kybic@fel.cvut.cz.*
- *P. Fua is with the Computer Vision Laboratory, EPFL, Lausanne 1015, Switzerland. E-mail: pascal.fua@epfl.ch.*

Manuscript received 3 June 2016; revised 10 Oct. 2016; accepted 30 Nov. 2016. Date of publication 5 Dec. 2016; date of current version 10 Oct. 2017. Recommended for acceptance by T. Brox.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TPAMI.2016.2636200

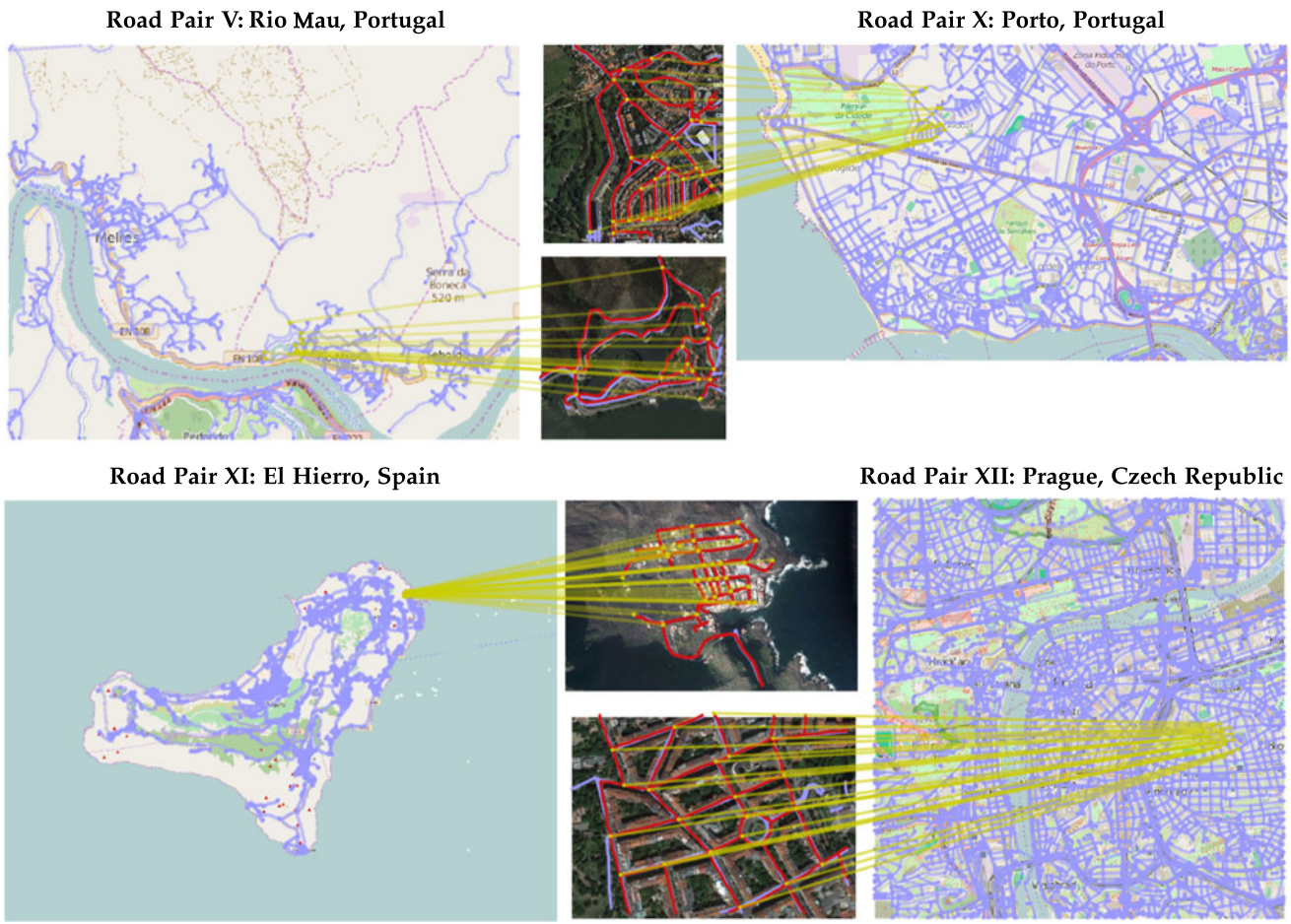

Fig. 1. Localizing a small satellite image within a map. Yellow lines link matched nodes of the road networks extracted from a satellite image (red) and from a map (blue). Transformed version of the map network (blue) after matching is shown overlaid on the satellite image.

number of degrees of freedom is small, its parameters can be recovered by RANSAC [12], based on randomly sampling the correspondences and fitting the geometrical model Many improvements to RANSAC have been proposed [13], such as Guided-MLESAC [14] or PROSAC [15]. The advantage of RANSAC 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) [16], which alternatively identifies closest points and updates the deformation parameters until convergence. There are variants of ICP that increase robustness [17], [18] or employ non-linear transformations [19]. The Coherent Point Drift (CPD) method [20] uses probabilistic assignment. These methods are fast and can handle large number of points but require a good initial estimate of the transformation.

2.2 Graph Matching

Appearance node attributes or descriptors [21], [22] are often used to prune matches between incompatible points in RANSAC-like methods. The next step is to consider compatibilities between edges or pairs of points, for example by comparing their Euclidean or geodesic distances [5], [7] or neighborhood similarity [23]. Matching can be approached as finding an approximate minimum vertex cover of pairwise consistent matches [24], or maximum weighted independent set [25]. The algorithm is fast as long as the number of consistent point pairs is low, i.e., for rigid transformations or for very discriminative appearance descriptors.

Binary compatibilities between point pairs can be relaxed to real-valued affinities, which leads to an integer quadratic program (IQP), which can be solved by spectral techniques [26], [27], spectral matching with affine constraints [28], iterative projections [29], dual decomposition [30], path following algorithms [31], or Sequential Monte Carlo search [32]. Unfortunately, none of these methods can handle more than a few tens of nodes.

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 [33] at the expense of increased computational complexity.

The active testing search method [4], [34] builds the match incrementally, adding node correspondences one by one. The probability of the match belonging to the solution is estimated from observable features and the most promising matches are added first, backtracking if necessary. This method does not require an initial estimate of the transformation and was applied to graphs with up to 200 nodes [4], which is better than all other previously known methods capable of handling non-linear transformations and unknown initial position.

2.3 Curve Matching

A simple but computationally demanding method for matching two curves in 2D or 3D consists of first aligning the curves and then calculating the residual Euclidean distance [4], [17] or the Frechet distance [35].

Descriptors of curve segments invariant to similarity transformations can be established [36], based on angle [37] or curvature [38], [39]. In [40], 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 a gradient [41].

Partial matches can be found by iterative closest point algorithm [17], dynamic programming [42] or the Hungarian algorithm [4].

2.4 Applications

Graph and point cloud matching has been used many times for image registration. In retinal fundus photography, it is possible to match the branching points [43] but also the complete vessel graphs [5], [31]. Tree matching has been used for matching 3D lung structures [44], [45] and also blood vascular systems [46]. Graph matching has been applied in the registration of volumes of neuronal networks and brain blood vessels [4].

The problem of car self-localization on a map was so far used in a setting, where the initial position is known and odometry information is available [47]; odometry was obtained from a set of images by modeling a map using landmarks [48]. This is unlike our graph-based method (Section 7.3.1) where no initial position estimate is needed.

3 Problem Definition

Let us consider a graph $\mathbf{G}^A = (\mathbf{V}^A, \mathbf{E}^A)$ 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* [49]. Each edge $\mathbf{e} \in \mathbf{E}^A$ is described by a continuous function $\zeta_\mathbf{e} : I \to \mathbb{R}^D$, where $I = [0, 1]$ is the unit interval, so that $\mathbf{e} = \big(\zeta_\mathbf{e}(0), \zeta_\mathbf{e}(1)\big)$. 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* [25] 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. 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 $\mathbf{G}^A$ 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

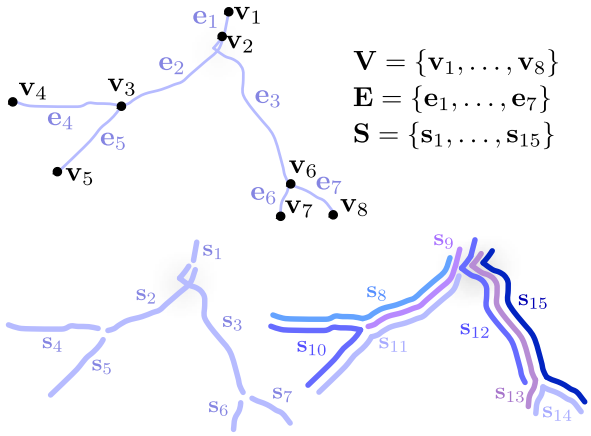

Fig. 2. Example of a geometrical graph *(top)* and its superedges of length one *(bottom left)* and two *(bottom left)*.

which is a concatenation of the curves associated with individual edges and is described by a function $\zeta_\mathbf{s} : I \to \mathbb{R}^D$.

The matching between two geometrical graphs $\mathbf{G}^A$ and $\mathbf{G}^B = (\mathbf{V}^B, \mathbf{E}^B)$ 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

(a) if no matched superedges overlap (contain the same edges),
(b) if it is *consistent* with some vertex matching $\mathbf{M}^\mathbf{V}$.

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

$$(\mathbf{u}, \mathbf{v}) \in \mathbf{M}^\mathbf{V} \Rightarrow \mathbf{v} = T(\mathbf{u}), \tag{1}$$

$$(\mathbf{r}, \mathbf{s}) \in \mathbf{M}^\mathbf{S} \Rightarrow \zeta_\mathbf{s}(I) = \big(T \circ \zeta_\mathbf{r}\big)(I). \tag{2}$$

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

$$l(\mathbf{M}^\mathbf{S}) = \sum_{(\mathbf{s}_k^A, \mathbf{s}_l^B) \in \mathbf{M}^\mathbf{S}} \frac{l(\mathbf{s}_k^A) + l(\mathbf{s}_l^B)}{2}. \tag{3}$$

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

$$Q(\mathbf{M}^\mathbf{V}, \mathbf{M}^\mathbf{S}) = l(\mathbf{M}^\mathbf{S}) + \kappa \, \overline{l(\mathbf{S}^A, \mathbf{S}^B)} \, |\mathbf{M}^\mathbf{V}|, \tag{4}$$

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 $\mathbf{G}^A$ and $\mathbf{G}^B$, which maximizes the criterion Q (4), and is consistent with some geometrical transformation T from a given class of allowed transformations $\mathscr{T}$.*

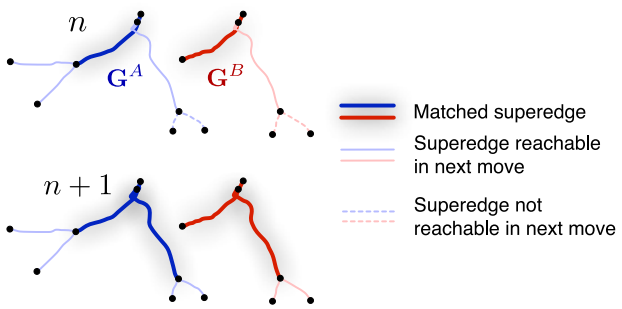

Fig. 3. Example for a possible next move (*top*—initial state, *bottom*—next state) for a simplified case of superedges with length one. The superedges reachable in each state are connected with already matched superedge pairs.

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 4.

3.1 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. 3 for illustration of possible moves and Section 5.1 for handling disconnected graphs.

To prune the search even further, we will use pre-computed path descriptors (Section 6). 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:

(i) $(\mathbf{r}, \mathbf{s})$ is adjacent to existing matches $\mathbf{M}_t^{\mathbf{S}}$,
(ii) $(\mathbf{r}, \mathbf{s})$ does not conflict with already matched superedges (i.e., no edge overlap or conflicting vertex assignment),
(iii) $\mathbf{r}$ and $\mathbf{s}$ have compatible descriptors (Section 6),
(iv) $(\mathbf{r}, \mathbf{s})$ together with previous matches $\mathbf{M}_t^{\mathbf{S}}$ are consistent with the transformation model (Section 4).

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

The superedges are considered in a *default order*: first the superedges with the least number of constituent edges and, in case of equality, longer superedges are considered first. For pairs of superedges, a maximum number of edges and the sum of their lengths is considered. This heuristic quickly constrains future matching choices without skipping vertices unless necessary.

4 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 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

$$\frac{1}{1+\varepsilon_T} d(\mathbf{x}, \mathbf{y}) \leq d(T(\mathbf{x}), T(\mathbf{y})) \leq (1+\varepsilon_T) d(\mathbf{x}, \mathbf{y}), \qquad (5)$$

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

To check condition (iv) in Section 3.1, 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 vertices $(\mathbf{p}, \mathbf{q}) \in M^{\mathbf{V}}$ matched so far

$$\frac{1}{1+\varepsilon_T} d(\mathbf{u}, \mathbf{p}) \leq d(\mathbf{v}, \mathbf{q}) \leq (1+\varepsilon_T)\, d(\mathbf{u}, \mathbf{p}). \qquad (6)$$

5 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 [8].

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 $Q_\nu = Q(M^{\mathbf{V}}, M^{\mathbf{S}})$ defined by (4) 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*

$$\tilde{Q}_\nu = \frac{Q_\nu^+}{Q_{\text{norm}}} + \gamma \sqrt{\frac{2 \log n}{n_\nu}}. \qquad (7)$$

The second term is the *upper confidence bound*(UCB) [8] and it balances between exploration of yet unvisited branches and exploitation of known good branches; n is the current

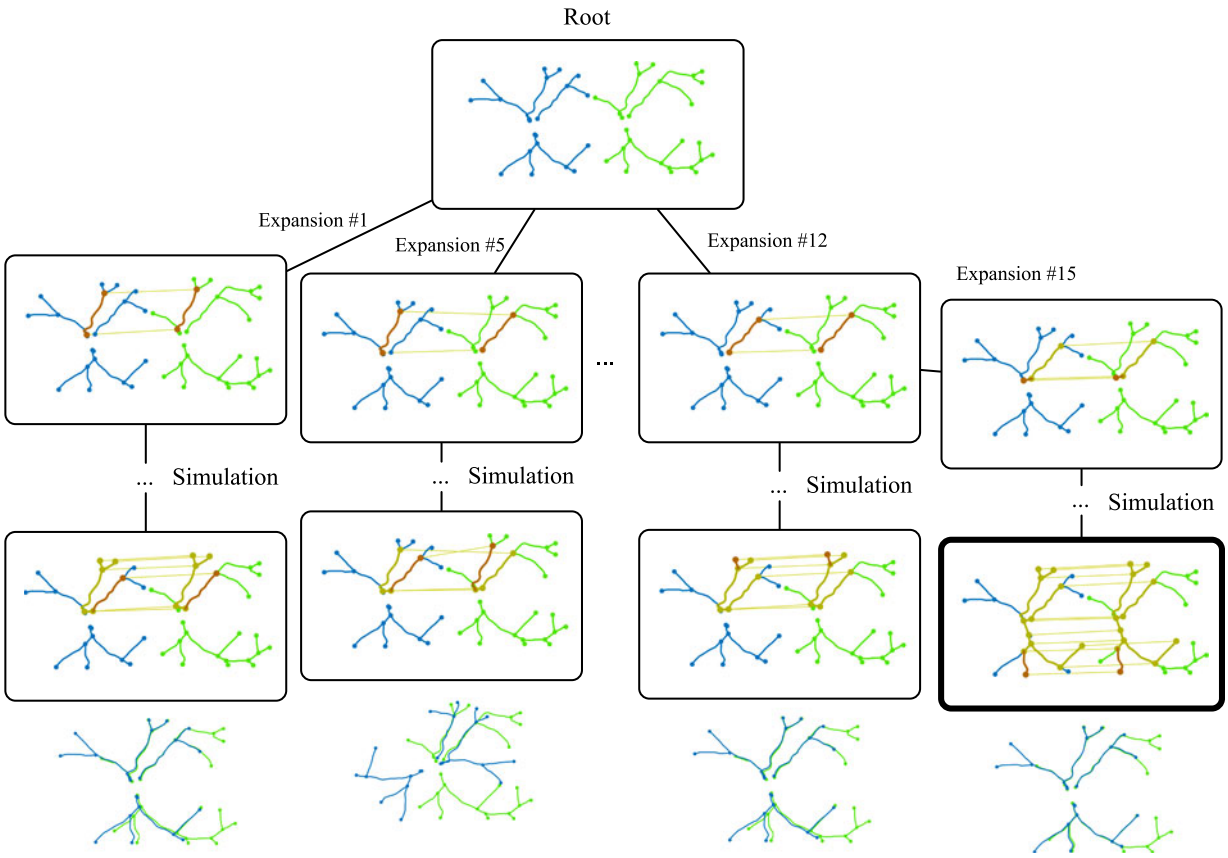

Fig. 4. Example of the GMMC algorithm exploring the search tree for a simple pair of graphs. A node corresponding to a superedge pair is selected and expanded and then extended greedily in the simulation step. For each pair of graphs in blue and green, $M^{\mathbf{V}}$ is represented by yellow lines 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.

iteration number, n_ν is the number of times the node ν has been *selected* (see below), γ is a user-defined parameter (in our experiments $\gamma = 0.01$), and Q_{norm} is an upper bound and a normalization factor for the node reward Q_ν

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

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. 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 3.1). 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 [50] 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 [4] should be used, which alternates between solving for the assignment between all points using the Hungarian algorithm [51] and fitting the Gaussian process model.

5.1 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* 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*: true if some node in the subtree rooted at ν can be expanded.

To allow exploring disconnected components of the graphs $\mathbf{G}^A$, $\mathbf{G}^B$, *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.

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.

5.2 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 3.1) and find the first geometrically compatible superedge in $\mathbf{S}^B$ (Section 6). 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 (i), (ii), and (iii) in Section 3.1. We then check the geometric compatibility with previous matches (condition (iv)) using (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 (iii)) are added to the list of possible moves. This can be done in linear time with respect to the number of adjacent edges.

6 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 [9].

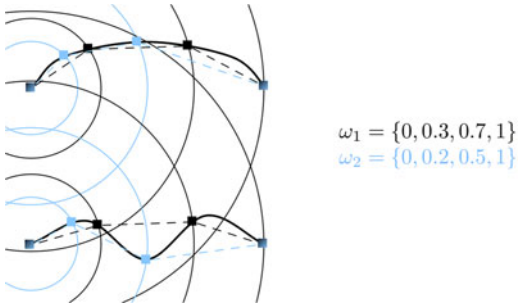

Fig. 5. Simple example of the sampling for calculating the path descriptor $h_{\boldsymbol{\omega}}(\mathbf{s}_k)$ for a superedge $\mathbf{s}_k$.

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

$$h_{\boldsymbol{\omega}}(\mathbf{s}) = \sum_{i=0}^{n_\omega} d\big(\zeta_{\mathbf{s}}(t_i), \zeta_{\mathbf{s}}(t_{i+1})\big), \tag{8}$$

where the vector $\mathbf{t} = (t_0, \ldots, t_{n_\omega+1})$ is chosen such that $d\big(\zeta_{\mathbf{s}}(0), \zeta_{\mathbf{s}}(t_i)\big) = \omega_i d\big(\zeta_{\mathbf{s}}(0), \zeta_{\mathbf{s}}(1)\big)$. 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). A vector descriptor

$$\mathbf{h}_{\boldsymbol{\Omega}}(\mathbf{s}) = (h_{\boldsymbol{\omega}_1}, \ldots, h_{\boldsymbol{\omega}_{|\boldsymbol{\Omega}|}}) \tag{9}$$

is created by evaluating (8) for a set $\boldsymbol{\Omega} = \big(\boldsymbol{\omega}_1, \ldots, \boldsymbol{\omega}_{|\boldsymbol{\Omega}|}\big)$ 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 (iii) in Section 3.1), we take inspiration from the Lipschitz condition (5). We shall consider the two superedges to be compatible iff

$$\frac{1}{1+\varepsilon_h} h_{\boldsymbol{\omega}_i}(\mathbf{r}) \le h_{\boldsymbol{\omega}_i}(\mathbf{s}) \le (1+\varepsilon_h) h_{\boldsymbol{\omega}_i}(\mathbf{r}), \quad \forall i \tag{10}$$

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 7.3.4.

7 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 7.2). We then show results on real bioimaging data and on satellite images of roads (Section 7.3).

7.1 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 [4]. We have chosen RANSAC [12] as an example of sampling methods. Active Testing Search [4], which uses incremental matching, is the best performing method so far. Graph matching techniques are represented by the Integer Projected Fixed Point method (IPFP) [29] and the Path Following Algorithm (PATH) [31]. Local matching approaches are represented by the Coherent Point Drift [20].

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

- RANSAC [12]: 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.
- ATS [4]: Synthetic training data (Section 7.2.3) 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 $\beta^{-1} = 0.05$.
- CPD [20]: 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).

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.

7.2 Synthetic Datasets

To generate a synthetic graph $\mathbf{G}^A$, the procedure from [4] is used—points are randomly sampled from a uniform distribution on the $[-1, 1]^3$ cube and then minimum spanning tree [52] edges are added. The branching points become vertices of $\mathbf{G}^A$ and the edges are resampled using a sampling step $\Delta = 0.025$. The graph $\mathbf{G}^B$ is obtained by transforming $\mathbf{G}^A$ in the desired way, as described below.

7.2.1 Evaluating Matching Accuracy

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

- *Deformation*: A cubic B-spline transformation [53] 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 $\{0, 10, \ldots, 90 \text{ percent}\}$.
- *Rotation*: The graph is rotated by a given angle from the interval $[-\pi, \pi]$ around a random rotation axis through the origin.
- *Noise*: A Gaussian random displacement with standard deviation in the range $[0, 0.1]$ was applied to each component of each point in $\mathbf{G}^B$, 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 percent of the size of $\mathbf{V}^A$.

We see in Fig. 6 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.

7.2.2 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 (6) and the allowed path descriptor tolerance (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 7.3.4.

We have generated 250 pairs of graphs by combining the effects from Section 7.2: a small nonlinear deformation, a random rigid motion, removal of randomly selected 40 percent 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. 7, 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 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 $\mathbf{G}^A$ 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. 8. The

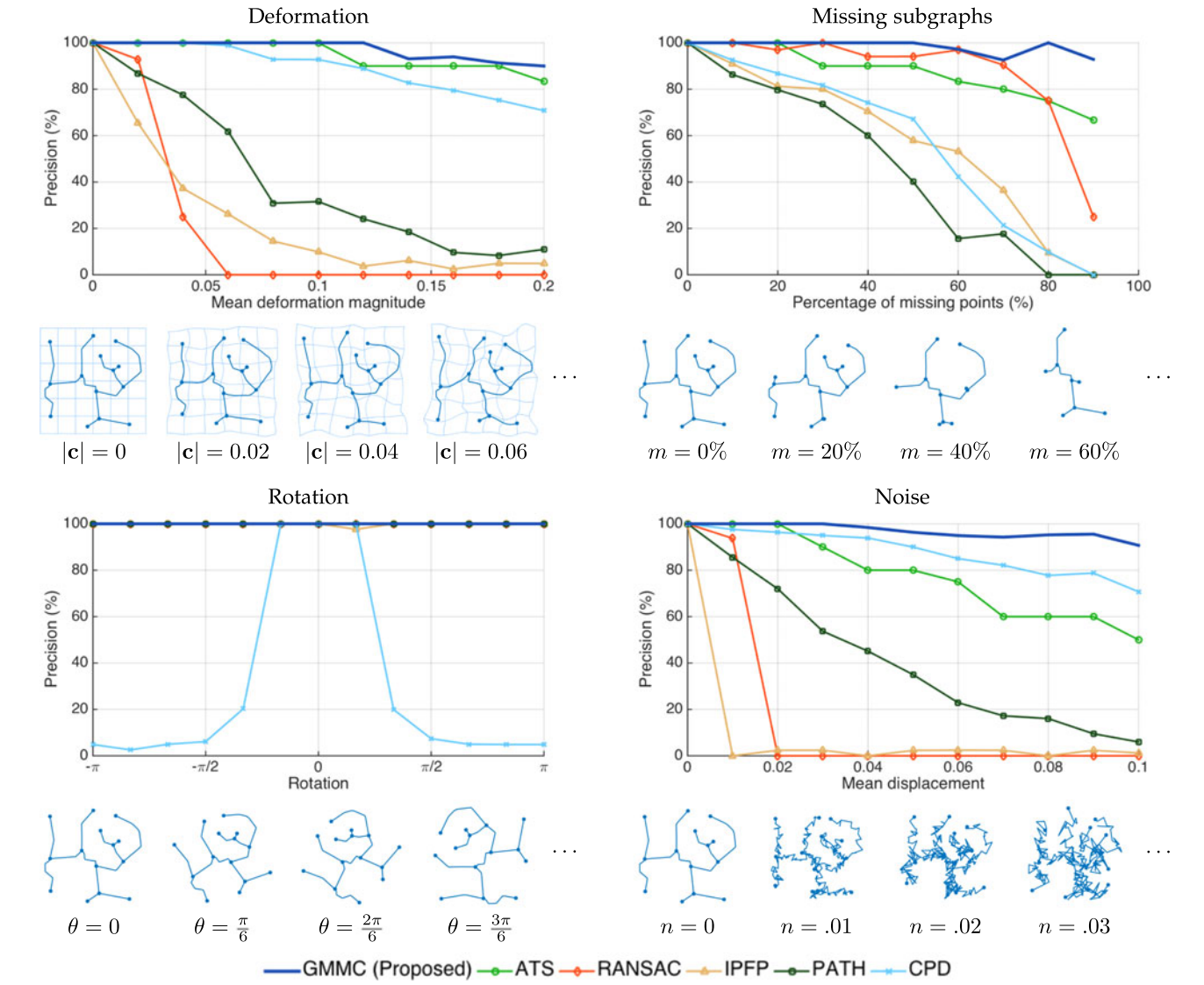

Fig. 6. Results for synthetic datasets in 3D showing the performance of the tested methods. The median correct correspondence percentage of 20 realizations for each parameter value is shown. Under each graph, we show 2D examples of the effects.

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.

7.2.3 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. 9 shows the median processing time for all the tested methods. The parameters for all methods were set so that at least 75 percent of the vertices

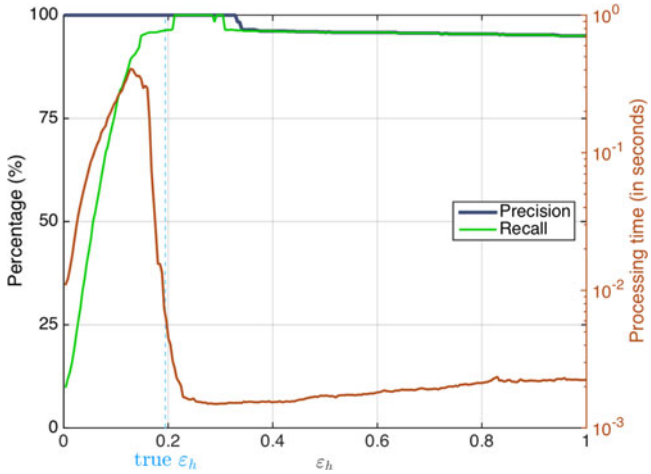

Fig. 7. The processing time, precision and recall as a function of ε_h, where $\varepsilon_T = \frac{1}{3}\varepsilon_h$. For different values of ε_h, we calculate the processing time taken to obtain the solution, the precision (positive predictive value) and the recall (true positive rate). The values shown are a median over 100 synthetically generated pairs of graphs. N_{match} was set to the size of the true match.

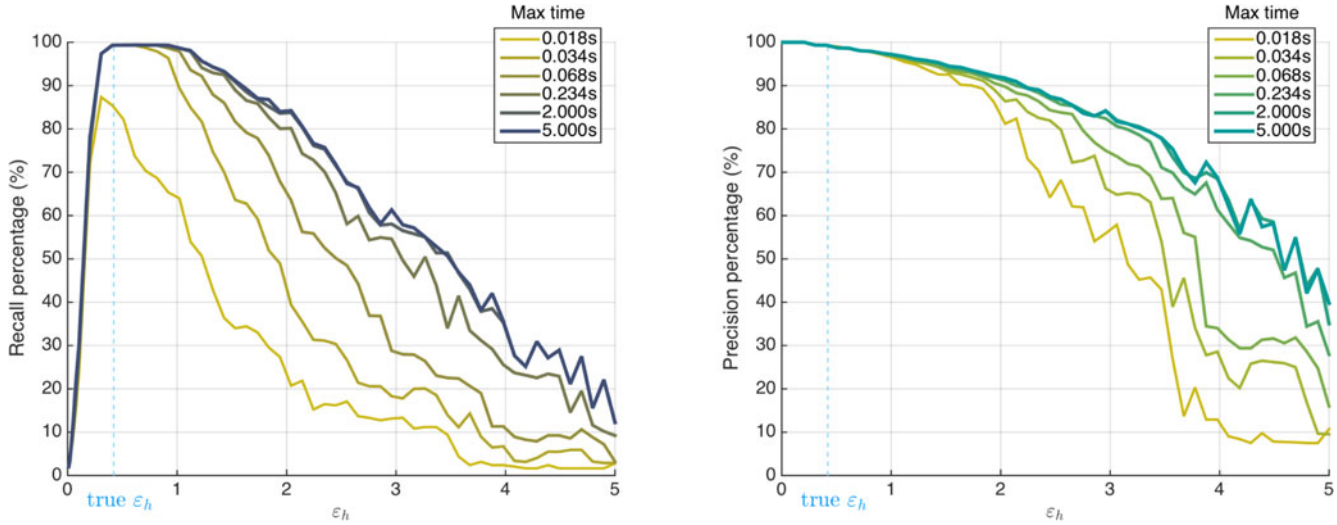

Fig. 8. Precision and recall as a function of the allowed processing time and ε_h, where $\varepsilon_T = \frac{1}{3}\varepsilon_h$. For this experiment, the maximum amount of time for the algorithm to stop was the only termination criterion used. The median true ε_h is shown on the horizontal axes.

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 five 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.

7.3 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 7.1 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.

7.3.1 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 [54] and we semi-automatically segmented [55], [56] 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.

We tested 10 different pairs of map graphs with up to 6,000 vertices, 7,500 edges and 75,000 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 Table 1 and a few visual examples are shown in Fig. 1. Some methods could not process all datasets as they exhausted the available 256 GB 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.

To validate our choice of the default order as described in Section 3.1, we have also tried to consider superedges in a random order instead. On this dataset, it resulted in an almost 40 times longer processing.

7.3.2 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.

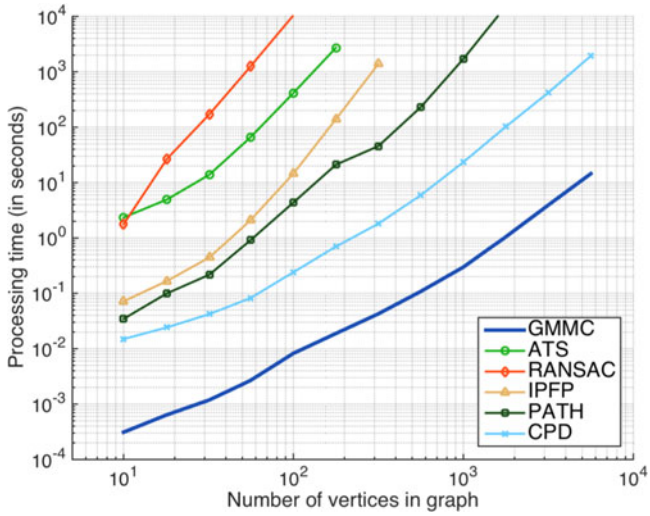

Fig. 9. Experiment evaluating the complexity of methods ATS [4], RANSAC [12], IPFP [29], PATH [31], CPD [20] and the proposed method (GMMC). Both axes use logarithmic scale.

1. http://maps.google.com

TABLE 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

Dataset		GMMC (Proposed)	ATS	IPFP	CPD
Road Pair I	Error	**0.004**	0.005	0.305	0.239
$\lvert\mathbf{V}^A\rvert$: 32	Precision (%)	**100.0**	**100.0**	0.0	3.1
$\lvert\mathbf{V}^B\rvert$: 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
$\lvert\mathbf{V}^A\rvert$: 25	Precision (%)	**92.9**	71.4	0.0	0.0
$\lvert\mathbf{V}^B\rvert$: 199	Recall (%)	**61.9**	23.8	0.0	0.0
	Time (s)	0.00	30,306.50	137.90	0.01
Road Pair III	Error	**0.003**	0.005	0.403	0.358
$\lvert\mathbf{V}^A\rvert$: 19	Precision (%)	90.9	**100.0**	0.0	0.0
$\lvert\mathbf{V}^B\rvert$: 306	Recall (%)	**76.9**	23.1	0.0	0.0
	Time (s)	0.01	10,946.63	1,679.98	0.01
Road Pair IV	Error	**0.009**	–	0.447	0.388
$\lvert\mathbf{V}^A\rvert$: 20	Precision (%)	**100.0**	–	0.0	0.0
$\lvert\mathbf{V}^B\rvert$: 1,344	Recall (%)	**71.4**	–	0.0	0.0
	Time (s)	0.96	–	1,606.86	0.01
Road Pair V	Error	**0.004**	–	0.429	0.303
$\lvert\mathbf{V}^A\rvert$: 29	Precision (%)	**73.3**	–	0.0	0.0
$\lvert\mathbf{V}^B\rvert$: 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
$\lvert\mathbf{V}^A\rvert$: 49	Precision (%)	**70.0**	–	0.0	0.0
$\lvert\mathbf{V}^B\rvert$: 1,989	Recall (%)	**42.4**	–	0.0	0.0
	Time (s)	10.95	–	83,619.15	0.17
Road Pair VII	Error	**0.007**	–	–	0.377
$\lvert\mathbf{V}^A\rvert$: 47	Precision (%)	**73.7**	–	–	0.0
$\lvert\mathbf{V}^B\rvert$: 6,050	Recall (%)	**43.8**	–	–	0.0
	Time (s)	363.74	–	–	0.29
Road Pair VIII	Error	**0.009**	–	0.464	0.363
$\lvert\mathbf{V}^A\rvert$: 24	Precision (%)	**85.7**	–	0.0	0.0
$\lvert\mathbf{V}^B\rvert$: 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
$\lvert\mathbf{V}^A\rvert$: 67	Precision (%)	**96.3**	–	0.0	0.0
$\lvert\mathbf{V}^B\rvert$: 1,693	Recall (%)	**57.8**	–	0.0	0.0
	Time (s)	414.36	–	36,056.12	0.20
Road Pair X	Error	**0.002**	–	0.268	0.577
$\lvert\mathbf{V}^A\rvert$: 51	Precision (%)	**95.8**	–	0.0	0.0
$\lvert\mathbf{V}^B\rvert$: 2,576	Recall (%)	**71.9**	–	0.0	0.0
	Time (s)	0.05	–	45,758.20	3.08
Road Pair XI	Error	**0.006**	–	0.489	0.571
$\lvert\mathbf{V}^A\rvert$: 41	Precision (%)	**100.0**	–	0.0	0.0
$\lvert\mathbf{V}^B\rvert$: 2,479	Recall (%)	**78.0**	–	0.0	0.0
	Time (s)	0.66	–	21,431.36	0.02
Road Pair XII	Error	**0.004**	–	–	0.668
$\lvert\mathbf{V}^A\rvert$: 36	Precision (%)	**95.2**	–	–	0.0
$\lvert\mathbf{V}^B\rvert$: 6,050	Recall (%)	**83.3**	–	–	0.0
	Time (s)	549.41	–	–	0.23

For each dataset, we present the number of vertices of each graph $|\mathbf{V}^A|$ and $|\mathbf{V}^B|$. The ATS and IPFP methods could not process some of the larger datasets without exhausting the available 256 GB of RAM. These cases are marked with '–'.

The aim of the registration of *retinal fundus* images [5], [57] (Fig. 11, $\varepsilon_T = 0.35$) 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. 13a, $\varepsilon_T = 0.2$) are sparse sets of fluorescently labeled neurons in the neocortex in 3D, which were obtained using large-scale two-photon laser scanning microscopy at two time instances in a living

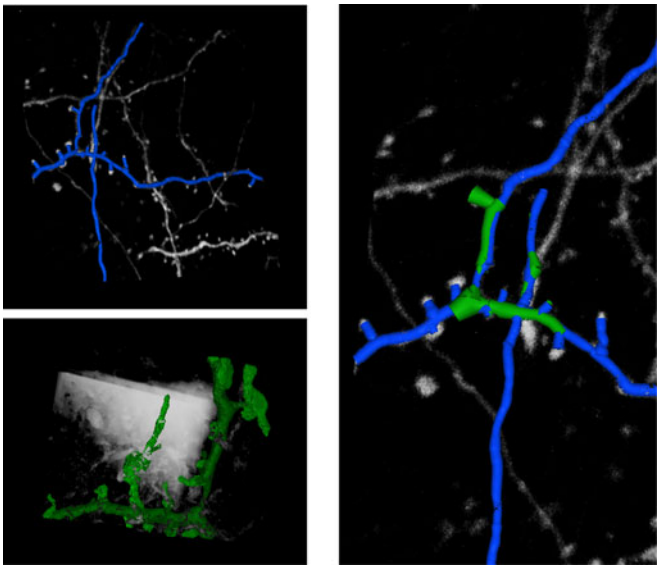

Fig. 10. *Top left:* Brain tissue acquired using two-photon light microscopy from live brain tissue at a $1\,\mu\mathrm{m}$ resolution and *bottom left:* A smaller area of the same tissue imaged using electron microscopy, at a $20\,\mathrm{nm}$ resolution. *Right:* The alignment of the two structures after matching using the proposed method.

mouse [58]. Registration is needed to detect and quantify the differences [59].

Multimodal registration is demonstrated on two 3D datasets. The *EM/LM* (Fig. 10, $\varepsilon_T = 0.35$) dataset shows brain tissue at two different scales using electron microscopy (EM) and light microcopy (LM). *Brain vessels* (Fig. 13b, $\varepsilon_T = 0.35$) are blood vessels acquired using optical and two-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. 13c, $\varepsilon_T = 0.25$). 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 2. 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.

7.3.3 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. 12, 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

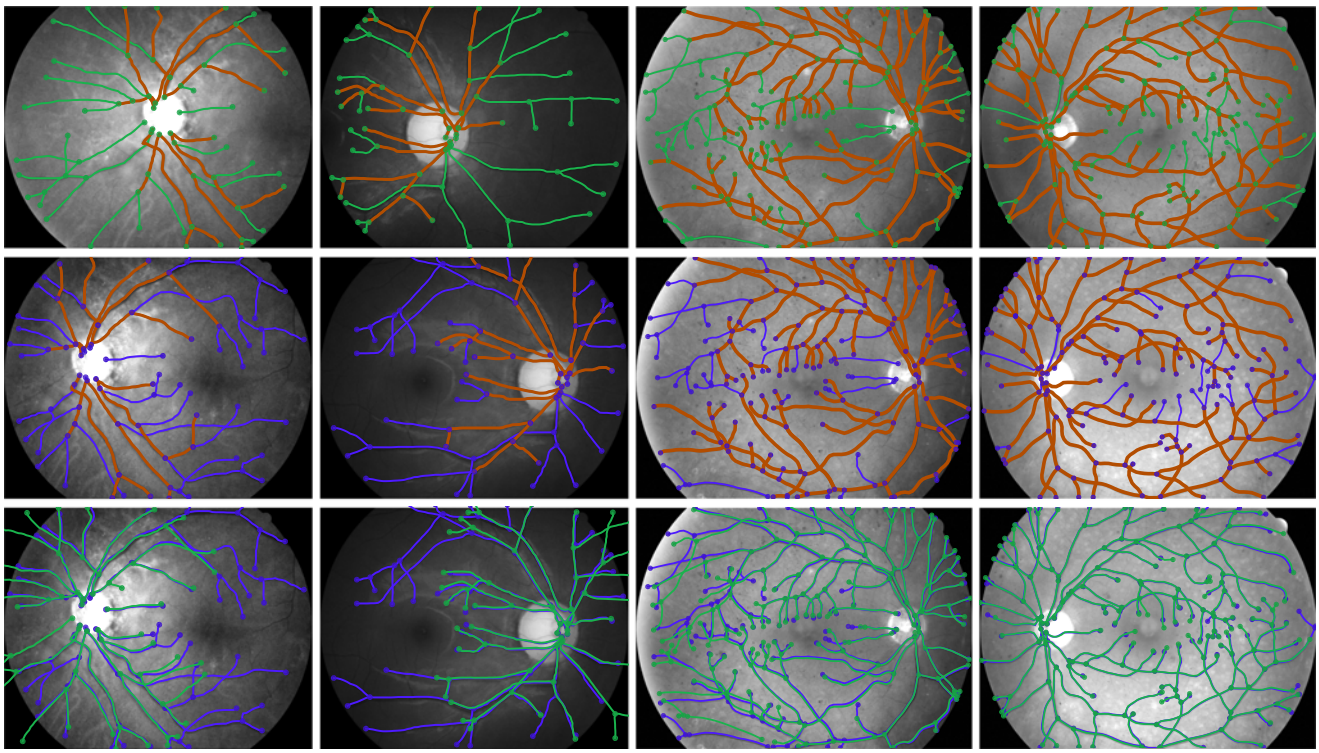

Fig. 11. Examples of results on the retinal dataset. The first and second rows are the input images with the extracted geometrical graphs superimposed. Matched superedges are shown in red. The third row depicts the resulting alignment of the first graph onto the second.

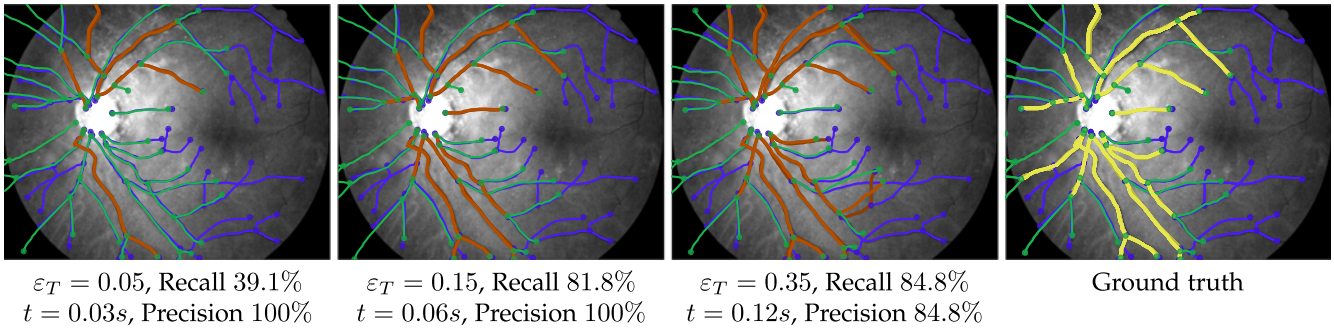

Fig. 12. The recall increases with increasing ε_T (with $\varepsilon_h = 3\varepsilon_T$) when matching two retinal fundus images. Consequently, the precision decreases and the processing time t increases. The graphs are drawn in green and blue, transformed based on the matched superedges (shown in red for both graphs), and overlaid on the original image. The same graph pair is visible as the left-most result in Fig. 11. The ground truth is shown in yellow in the right-most pair of images. Only entire edges are matched.

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 the shorter one [25], at the expense of more false matches.

7.3.4 Path Descriptors

We have tested the proposed path descriptors from Section 6 against four other methods: Similarity of Deformable Shapes [60], Curve Matching using the Fast Marching Method [61] and Similarity Invariants for 3D Space Curve Matching [36]. We have used all medical datasets from Section 7.3.2.

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 [60] and [61], thresholding a euclidean distance between the descriptors for [36], and varying ε_h in (10) for our method. We can see in Fig. 14, 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 percent percentile of the histogram while

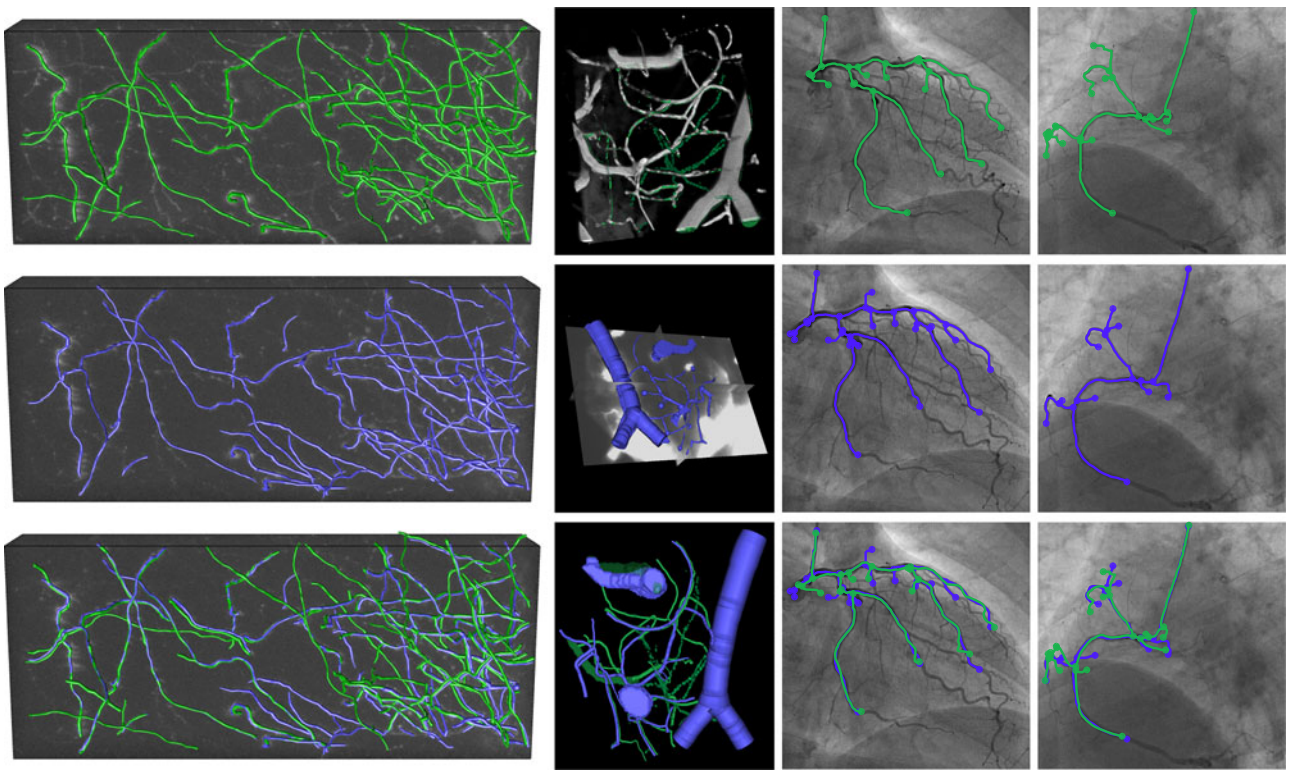

Fig. 13. Examples of several medical applications results. (a) is a neuronal network acquired at different time instances using two-photon microscopy. (b) are blood vessels acquired using *(top)* two-photon microscopy and *(middle)* bright-field optical microscopy. Only segmentation is shown for better visualization. (c) are angiography images from a beating heart taken at different time instances. The first and second rows are the initial images with the extracted geometrical graphs superimposed. The third row depicts the alignment of the proposed method of the first graph onto the space of the second.

TABLE 2
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

Dataset		GMMC (Proposed)	ATS	IPFP	CPD
	Error	**0.026**	0.030	0.669	0.027
Brain Circuits	Precision (%)	72.0	**100.0**	2.4	45.2
$\lvert\bar{\mathbf{V}}\rvert : 124$	Recall (%)	29.5	8.2	4.9	**91.8**
	Time (s)	0.16	1,104.41	59.42	0.11
	Error	**0.016**	**0.016**	0.179	0.128
EM/LM	Precision (%)	77.8	**100.0**	4.5	4.5
$\lvert\bar{\mathbf{V}}\rvert : 22$	Recall (%)	**70.0**	50.0	10.0	10.0
	Time (s)	0.00	97.14	0.78	0.01
	Error	0.045	**0.041**	0.409	0.048
Brain Vessels	Precision (%)	71.4	**100.0**	21.4	29.7
$\lvert\bar{\mathbf{V}}\rvert : 37$	Recall (%)	41.7	41.7	50.0	**91.7**
	Time (s)	0.01	4.84	3.26	0.01
	Error	**0.031**	0.052	0.057	0.078
Angiography	Precision (%)	**70.1**	70.0	24.7	19.1
$\lvert\bar{\mathbf{V}}\rvert : 24$	Recall (%)	**80.6**	47.2	66.7	61.1
	Time (s)	0.29	13.03	2.66	0.08
	Error	**0.017**	0.031	0.133	0.070
Retina	Precision (%)	**88.6**	86.7	62.1	66.6
$\lvert\bar{\mathbf{V}}\rvert : 141$	Recall (%)	73.8	4.2	76.1	**84.1**
	Time (s)	0.47	585.60	640.20	0.10

For each dataset, we also give the average number of vertices $|\bar{\mathbf{V}}|$.

$\varepsilon_T = 0.05$, corresponds to a 95.17 percent 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.

8 Conclusion

Geometric graph matching is a task, which has not received much attention so far. It is an extended version of standard graph matching (with geometry information) but also an extended version of point cloud matching (with edge connections).

We have presented a method that can match geometrical graphs extracted from 2D and 3D images. Its main application is fast template localization or registration of very large images with very different appearance, which have in common the extracted graph but not much else. It is applicable in cases where the geometric graph (linear) structures can be distinguished but standard keypoint detectors fail or keypoint descriptors are not discriminative enough and similarity criterion based registration would be too slow.

We have compared GMMC with other point cloud and graph matching methods, which all have their strengths and weaknesses. For example, CPD [20] works great when the transformation is small but fails when it is not. RANSAC is simple, fast and robust, but requires a low-dimensional deformation model. Optimization-based, linear programming, and spectral methods (IPFP,PATH), as well as earlier Monte Carlo methods, can often find globally optimal solutions but cannot handle even medium size graphs.

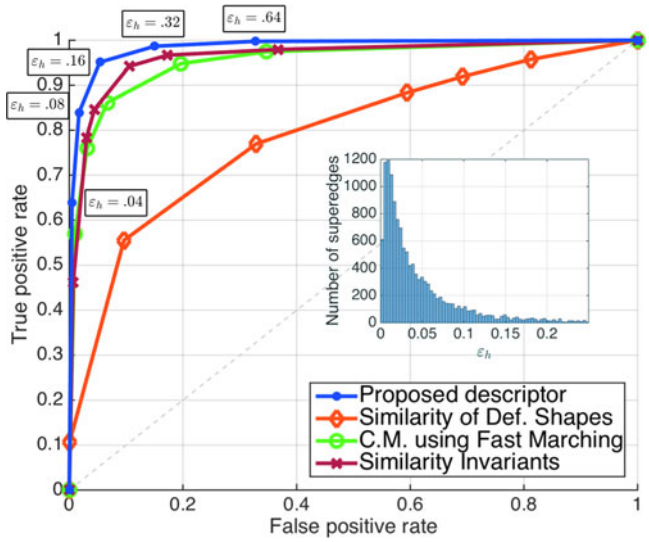

Fig. 14. ROC curve comparing the performance of the curve descriptors. in all medical datasets of different methods. For each point of the proposed method in the ROC curve, we show the used ε_h values. We also show a histogram of the true ε_h of the superedges in the data. The mean value of ε_h in this data is approximately 0.06.

As we have shown in the experiments, our method is unique in that it is fast, does not need an initial estimate of the transformation, yet it is robust to nonlinear deformation, noise and topological differences. It is the only method we know that can successfully match geometrical graphs of many thousands of vertices. This is mostly due to a novel formulation of the matching problem as a single player game, solved using a modified Monte Carlo tree search approach, which automatically balances between exploration and exploitation of the search space. Note that while we use the Monte Carlo tree search method, our method is in fact completely deterministic. The method is fast also thanks to the curve descriptors which we have introduced and which allow efficient pruning. We have shown experimentally that these descriptors outperform other previously known curve descriptors.

Our method is modular and could be extended in various ways. For example: (i) we could add appearance comparison to the superedge pair compatibility check. (ii) The deformation threshold ε_T could vary in space. (iii) The subtree match quality criterion Q_v^+ could be estimated more accurately from the properties of the partial matches using machine learning, similar to [4]. (iv) The search could be parallelized to yield further speedup [62]. The challenge remains how to perform the matching of truly large geometrical graphs, of millions of vertices or more.

Acknowledgments

This work has been supported by the Czech Science Foundation project 14-21421S and by the Ph.D. grant SFRH/BD/77134/2011 of the Fundação para a Ciência e Tecnologia.

References

[1] J. P. W. Pluim, J. B. A. Maintz, and M. A. Viergever, "Mutual information based registration of medical images: A survey," *IEEE Trans. Med. Imag.*, vol. 22, no. 8, pp. 986–1004, Aug. 2003.

[2] B. Zitová and J. Flusser, "Image registration methods: A survey," *Image Vis. Comput.*, vol. 21, pp. 977–1000, 2003.

[3] A. Sotiras, C. Davatzikos, and N. Paragios, "Deformable medical image registration: A survey," *IEEE Trans. Med. Imag.*, vol. 32, no. 7, pp. 1153–1190, Jul. 2013. [Online]. Available: http://www.ncbi.nlm.nih.gov/pmc/articles/PMC3745275/

[4] E. Serradell, M. A. Pinheiro, R. Sznitman, J. Kybic, F. Moreno-Noguer, and P. Fua, "Non-rigid graph registration using active testing search," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 3, pp. 625–638, Mar. 2015.

[5] K. Deng, J. Tian, J. Zheng, X. Zhang, X. Dai, and M. Xu, "Retinal fundus image registration via vascular structure graph matching," *J. Biomed. Imag.*, vol. 2010, 2010, Art. no. 14. [Online]. Available: http://dx.doi.org/10.1155/2010/906067

[6] S. Gold and A. Rangarajan, "A graduated assignment algorithm for graph matching," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 18, no. 4, pp. 377–388, Apr. 1996.

[7] D. Smeets, P. Bruyninckx, J. Keustermans, D. Vandermeulen, and P. Suetens, "Robust matching of 3D lung vessel trees," in *Proc. 3rd Int. Workshop Pulmonary Image Anal.*, 2010, pp. 61–70.

[8] C. Browne, et al., "A survey of Monte Carlo tree search methods," *IEEE Trans. Comput. Intell. AI Games*, vol. 4, no. 1, pp. 1–43, Mar. 2012.

[9] M. A. Pinheiro and J. Kybic, "Path descriptors for geometric graph matching and registration," in *Proc. 11th Int. Conf. Image Anal. Recognit.*, 2014, pp. 3–11.

[10] M. A. Pinheiro and J. Kybic, "Geometrical graph matching using Monte Carlo tree search," in *Proc. IEEE Int. Conf. Image Process.*, Sep. 2015, pp. 3145–3149. [Online]. Available: ftp://cmp.felk.cvut.cz/pub/cmp/articles/amavemig/Pinheiro-ICIP2015.pdf

[11] G. K. L. Tam, et al., "Registration of 3D point clouds and meshes: A survey from rigid to nonrigid," *IEEE Trans. Vis. Comput. Graph.*, vol. 19, no. 7, pp. 1199–1217, Jul. 2013.

[12] M. A. Fischler and R. C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Commun. ACM*, vol. 24, pp. 381–395, 1981.

[13] S. Choi, T. Kim, and W. Yu, "Performance evaluation of RANSAC family," in *Proc. Brit. Mach. Vis. Conf.*, 2009, pp. 1–12.

[14] B. Tordoff and D. W. Murray, "Guided sampling and consensus for motion estimation," in *Proc. 7th Eur. Conf. Comput. Vis.—Part I*, 2002, pp. 82–98. [Online]. Available: http://dl.acm.org/citation.cfm?id=645315.649190

[15] O. Chum and J. Matas, "Matching with PROSAC—Progressive sample consensus," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, 2005, pp. 220–226. [Online]. Available: http://dx.doi.org/10.1109/CVPR.2005.221

[16] P. J. Besl and N. D. McKay, "A method for registration of 3-D shapes," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 14, no. 2, pp. 239–256, Feb. 1992. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=121791

[17] T. Pajdla and L. V. Gool, "Matching of 3-D curves using semi-differential invariants," in *Proc. IEEE Int. Conf. Comput. Vis.*, 1995, pp. 390–395.

[18] D. Chetverikov, D. Stepanov, and P. Krsek, "Robust Euclidean alignment of 3D point sets: The trimmed iterative closest point algorithm," *Image Vis. Comput.*, vol. 23, no. 3, pp. 299–309, 2005. [Online]. Available: http://dx.doi.org/10.1016/j.imavis.2004.05.007

[19] H. Chui and A. Rangarajan, "A new point matching algorithm for non-rigid registration," *Comput. Vis. Image Understanding*, vol. 89, pp. 114–141, 2003. [Online]. Available: http://dx.doi.org/10.1016/S1077–3142(03)00009-2

[20] A. Myronenko and X. Song, "Point set registration: Coherent point drift," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 12, pp. 2262–2275, Dec. 2010.

[21] D. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comput. Vis.*, vol. 60, no. 2, pp. 91–110, 2004. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.2.8899

[22] H. Bay, T. Tuytelaars, and L. V. Gool, "SURF: Speeded up robust features," *Comput. Vis.*, vol. 3951, pp. 404–17, 2006. [Online]. Available: http://www.springerlink.com/index/E580H2K58434P02K.pdf

[23] S. Gold, A. Rangarajan, C. Lu, and E. Mjolsness, "New algorithms for 2D and 3D point matching: Point estimation and correspondence," *Pattern Recognit.*, vol. 31, no. 8, pp. 1019–1031, 1998.

[24] O. Enqvist, K. Josephson, and F. Kahl, "Optimal correspondences from pairwise constraints," in *Proc. IEEE 12th Int. Conf. Comput. Vis.*, 2009, pp. 1295–1302.

[25] E. Serradell, F. Moreno-Noguer, J. Kybic, and P. Fua, "Robust elastic 2D/3D geometric graph matching," *SPIE Med. Imag.*, vol. 8314, no. 1, 2012, Art. no. 831408.

[26] G. Scott and H. Longuet-Higgins, "An algorithm for associating the features of two patterns," *Proc. Royal Soc. London*, vol. B244, pp. 21–26, 1991.

[27] M. Leordeanu and M. Hebert, "A spectral technique for correspondence problems using pairwise constraints," in *Proc. 10th IEEE Int. Conf. Comput. Vis.*, 2005, vol. 2, pp. 1482–1489.

[28] T. Cour, P. Srinivasan, and J. Shi, "Balanced graph matching," in *Proc. Advances Neural Inf. Process. Syst. 19*, 2006, pp. 313–320.

[29] M. Leordeanu, M. Hebert, and R. Sukthankar, "An integer projected fixed point method for graph matching and MAP inference," in *Proc. Advances Neural Inf. Process. Syst. 22*, 2009, pp. 1114–1122.

[30] L. Torresani, V. Kolmogorov, and C. Rother, "Feature correspondence via graph matching: Models and global optimization," in *Proc. 10th Eur. Conf. Comput. Vis.: Part II*, 2008, pp. 596–609.

[31] M. Zaslavskiy, F. Bach, and J.-P. Vert, "A path following algorithm for the graph matching problem," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 31, no. 12, pp. 2227–2242, Dec. 2009.

[32] Y. Suh, M. Cho, and K. M. Lee, "Graph matching via sequential Monte Carlo," in *Proc. 12th Eur. Conf. Comput. Vis.—Vol. Part III*, 2012, pp. 624–637. [Online]. Available: http://dx.doi.org/10.1007/978–3-642-33712-3_45

[33] B. Messmer and H. Bunke, "A new algorithm for error-tolerant subgraph isomorphism detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 20, no. 5, pp. 493–504, May 1998.

[34] M. A. Pinheiro, R. Sznitman, E. Serradell, J. Kybic, F. Moreno-Noguer, and P. Fua, "Active testing search for point cloud matching," in *Proc. 23rd Int. Conf. Inf. Process. Med. Imag.*, 2013, pp. 572–583. [Online]. Available: ftp://cmp.felk.cvut.cz/pub/cmp/articles/amavemig/Pinheiro-IPMI2013.pdf

[35] K. Buchin, M. Buchin, and Y. Wang, "Exact algorithms for partial curve matching via the Fréchet distance," in *Proc. 20th Annu. ACM-SIAM Symp. Discr. Algorithms*, 2009, pp. 645–654. [Online]. Available: http://dl.acm.org/citation.cfm?id=1496770.1496841

[36] S. Z. Li, "Similarity invariants for 3D space curve matching," in *Proc. 1st Asian Conf. Comput. Vis.*, 1993, pp. 454–457.

[37] M. Al-Khaiyat and F. Kamangar, "Planar curve representation and matching," in *Proc. Brit. Mach. Vis. Conf.*, 1998, pp. 174–184.

[38] H. J. Wolfson, "On curve matching," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 12, no. 5, pp. 483–489, May 1990. [Online]. Available: http://dblp.uni-trier.de/db/journals/pami/pami12.html#Wolfson90

[39] M. Cui, J. Femiani, J. Hu, P. Wonka, and A. Razdan, "Curve matching for open 2D curves," *Pattern Recognit. Lett.*, vol. 30, no. 1, pp. 1–10, 2009. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167865508002572

[40] H. Fu, Z. Tian, M. Ran, and M. Fan, "Novel affine-invariant curve descriptor for curve matching and occluded object recognition," *IET Comput. Vis.*, vol. 7, no. 4, pp. 279–292, Aug. 2013.

[41] Z. Wang, H. Liu, and F. Wu, "Image content based curve matching using HMCD descriptor," in *Proc. 9th Asian Conf. Comput. Vis.—Vol. Part III*, 2010, pp. 448–455. [Online]. Available: http://dx.doi.org/10.1007/978–3-642-12297-2_43

[42] T. B. Sebastian, P. N. Klein, B. B. Kimia, and J. J. Crisco, "Constructing 2D curve atlases," in *Proc. IEEE Workshop Math. Methods Biomed. Image Anal.*, 2000, pp. 70–77. [Online]. Available: http://dl.acm.org/citation.cfm?id=823463.824303

[43] A. Can, C. V. Stewart, B. Roysam, and H. L. Tanenbaum, "A feature-based, robust, hierarchical algorithm for registering pairs of images of the curved human retina," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 3, pp. 347–364, Mar. 2002. [Online]. Available: http://dx.doi.org/10.1109/34.990136

[44] C. Pisupati, L. B. Wolff, W. Mitzner, and E. A. Zerhouni, "Geometric tree matching with applications to 3D lung structures," in *Proc. 12th Annu. Symp. Comput. Geometry*, 1996, pp. 419–420.

[45] M. W. Graham and W. E. Higgins, "Optimal graph-theoretic approach to 3D anatomical tree matching," in *Proc. IEEE Int. Symp. Biomed. Imag.*, Apr. 2006, pp. 109–112.

[46] A. Charnoz, V. Agnus, G. Malandain, L. Soler, and M. Tajine, "Tree matching applied to vascular system," in *Proc. 5th IAPR Int. Workshop Graph-Based Representations Pattern Recognit.*, 2005, pp. 183–192. [Online]. Available: http://dx.doi.org/10.1007/978–3-540-31988-7_17
[47] M. A. Brubaker, A. Geiger, and R. Urtasun, "Lost! leveraging the crowd for probabilistic visual self-localization," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2013, pp. 3057–3064.
[48] H. Durrant-Whyte and T. Bailey, "Simultaneous localization and mapping: Part I," *IEEE Robot. Autom. Mag.*, vol. 13, no. 2, pp. 99–110, Jun. 2006.
[49] J. Erickson, "Local polyhedra and geometric graphs," *Comput. Geometry*, vol. 31, no. 1/2, pp. 101–125, 2005. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0925772104001087
[50] D. J. C. MacKay, "Introduction to Gaussian processes," *Neural Netw. Mach. Learning*, vol. 168, pp. 133–165, 1998.
[51] J. Munkres, "Algorithms for the assignment and transportation problems," *J. Soc. Ind. Appl. Math.*, vol. 5, no. 1, pp. 32–38, 1957.
[52] O. Boruvka, "O Jistém Problému Minimálním (About a Certain Minimal Problem) (in Czech, German summary)," *Práce Mor. Prírodoved. Spol. v Brne III*, vol. 3, pp. 37–58, 1926.
[53] M. Unser, "Splines: A perfect fit for medical imaging," *Progress Biomed. Optics Imag.*, vol. 3, pp. 225–236, 2002.
[54] M. Haklay and P. Weber, "OpenStreetMap: User-generated street maps," *IEEE Pervasive Comput.*, vol. 7, no. 4, pp. 12–18, Oct. 2008.
[55] M. H. Longair, D. A. Baker, and J. D. Armstrong, "Simple neurite tracer: Open source software for reconstruction, visualization and analysis of neuronal processes," *Bioinformatics*, vol. 27, pp. 2453–2454, 2011. [Online]. Available: http://bioinformatics.oxfordjournals.org/content/early/2011/07/04/bioinformatics.btr390.abstract
[56] E. Türetken, F. Benmansour, and P. Fua, "Automated reconstruction of tree structures using path classifiers and mixed integer programming," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2012, pp. 566–573.
[57] R. Kolář, V. Harabiš, and J. Odstrčilík, "Hybrid retinal image registration using phase correlation," *Imag. Sci. J.*, vol. 61, no. 4, pp. 369–384, 2013.
[58] A. Holtmaat, J. Randall, and M. Cane, "Optical imaging of structural and functional synaptic plasticity in vivo," *Eur. J. Pharmacology*, vol. 719, no. 1–3, pp. 128–136, 2013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0014299913005426
[59] P. Głowacki, et al., "Reconstructing evolving tree structures in time lapse sequences," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2014, pp. 3035–3042.
[60] R. Basri, L. Costa, D. Geiger, and D. Jacobs, "Determining the similarity of deformable shapes," *Vis. Res.*, vol. 38, pp. 135–143, 1998.
[61] M. Frenkel and R. Basri, "Curve matching using the fast marching method," in *Proc. 4th Int. Workshop Energy Minimization Methods Comput. Vis. Pattern Recognit.*, 2003, pp. 35–51.
[62] G. M. J. B. Chaslot, M. H. M. Winands, and H. J. van den Herik, "Parallel Monte-Carlo tree search," in *Proc. 6th Int. Conf. Comput. Games*, 2008, pp. 60–71. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-87608-3_6

Miguel Amável Pinheiro received the MSc degree in electrical and computers engineering from the Faculty of Engineering, University of Porto, in 2010. He is currently working toward the PhD degree in the Center of Machine Perception, which is a part of the Department of Cybernetics, Faculty of Electrical Engineering, Czech Technical University in Prague.

Jan Kybic received the master's degree from the Czech Technical University in Prague, Czech Republic and the PhD degree from EPFL, Switzerland, in 1998 and 2001, respectively. He held a post-doc position with INRIA, France, in 2002-2003. Since 2003, he is with Czech Technical University in Prague, becoming a full professor in 2015 and currently serving as a department head. He is a senior member of the IEEE.

Pascal Fua received an engineering degree from École Polytechnique, Paris, in 1984, and the PhD degree in computer science from the University of Orsay, in 1989. He joined Swiss Federal Institute of Technology (EPFL) in 1996, where he is now a professor with the School of Computer and Communication Science. Before that, he worked with SRI International and with INRIA Sophia-Antipolis as a computer scientist. He is a fellow of the IEEE.