Tracing Retinal Blood Vessels by Matrix-Forest Theorem of Directed Graphs

Supplementary Material

Abstract. This paper aims to trace retinal blood vessel trees from fundus images. This task is far from being trivial as the *crossover* of vessels are commonly encountered in image-based vessel networks, meanwhile it is often crucial to separate the vessel tree structures in applications such as diabetic retinopathy. In this work, a novel and systematic directed graph based approach is proposed to cast the task as label propagation over directed graphs, such that the graph is to be partitioned into disjoint sub-graphs, or equivalently, each of the vessel trees is traced and separated from the rest of the vessel network. This enables us to address the tracing problem by adopting the recent development of matrix-forest theorem in algebraic graph theory. Empirical experiments on synthetic as well as publicly available fundus image datasets demonstrate the applicability of our approach.

1 Introduction

Topological and geometrical properties of retinal blood vessels from fundus images can provide valuable clinical information in diagnosing diseases. In particular, vascular anomaly in retina is one of the clinical manifestations of retinal diseases such as diabetic retinopathy, glaucoma, and hypertensive retinopathy. Take diabetic retinopathy as an example, it is a leading cause of blindness in the working-age population of most developed countries. Diabetic retinopathy is the result of progressive damage to the network of tiny blood vessels that supply blood to the retina. It is classified into two major groups in clinics according to the severity of the disease: non-proliferative and proliferative. Proliferative diabetic retinopathy is characterized by the formation of new-formed vessels in the retina, while non-proliferative diabetic retinopathy refers to the absence of abnormal new blood vessels [1]. The description of blood vessel tree structure is therefore essential in clinical diagnosis of eye diseases such as diabetic retinopathy. It is thus necessary to trace and separate each of the vessel trees from the retinal vessel network. Unfortunately, commercial softwares still largely rely on manual tracing of the blood vessel trees, which is usually a tedious and time-consuming effort.

Existing efforts in retinal vessel analysis can be roughly categorized into two groups, namely, segmentation-based and tracking-based. The segmentation-based methods often use pixel classification [2–10] to produce a binary segmentation, where a pixel is classified into vessel or non-vessel. The tracking-based methods [11–20], on the other hand, usually start with a seed and track the intended vessel based on local intensity or texture information. It has been observed that segmentation-based methods tend to produce many disconnected

and isolated segments [21], which are less favourable towards retaining the important topological properties of vessel networks [22]. Vessel tracking methods, on the other hand, often preserve the connectivity structure of vessel segments. Nonetheless they encounter great difficulties with the occurrence of *crossover* [15] at the junction points. Current methods often fail to trace properly, as it is non-trivial to predict whether the vessel segments touching at a junction point belong to one tree, or two and more trees, and for the later case, to which tree each segment belongs. In this paper, we dedicate our attention to addressing this bottleneck issue, which is referred to as the *crossover* issue. One important observation is that local and global contextual information is crucial to resolve the crossover issue. For example, at a junction point, it is very helpful to go beyond the current vessel brunch and examine the angular properties of the other vessel brunches of the junction. These information is unfortunately ignored by current tracing methods.

This observation inspires us to consider a different tracing approach that can take into account both local and global contextual information of the vessel network: After initial steps of pixel-based segmentation and skeleton extraction, a novel *graph representation* is formed, where each segment in the skeleton map becomes a node, and a direct contact between two adjacent segments is translated to an edge of the two corresponding nodes. The segments in the skeleton map touching the optic disk area are considered as the root nodes. The number of trees to-be-found in the the vessel network thus equals the number of root nodes. The directed graph (or interchangeably, *digraph*) representation is obtained via a segment ordering method adopted from [23]. Note different from undirected graphs that delineate symmetric weights between adjacent nodes, in digraphs, edge or link directions are well preserved in its weight matrix. This information is particularly useful and natural for tracing retinal vessels, which coincides with the physical principle governing how blood flows in vessels. The tracing problem is now formulated as label propagation [24] on digraphs: The goal becomes that of propagating the tree labels from known root nodes to the rest of the graph, such that the graph will be split into disjoint sub-graphs, which corresponds to trees of the vessel network. This allows us to consider and make novel usage of the recent development of matrix-forest theorem [25] studied in algebraic digraph theory [26], which is to be described in the following sections.

The main contributions of this paper are three-fold. First, our approach offers a principled way of addressing the tracing with crossover problem. By connecting to the well-established algebraic directed graph theory [25], local and global contextual information can be both considered explicitly. It also bears strong connections to a number of established theories including the absorbing random walks, graph Laplacians, and graph kernels. We expect the graph representation, and the algebraic graph theory connection can open the door to some insightful understanding of the characteristics of crossover sections in vessel networks. Second, a novel graph representation is proposed, which can be regarded as an equivalent *dual* representation of the original vessel network, and is essential for establishing the graph-theoretical and the learning-based connections. Third, A

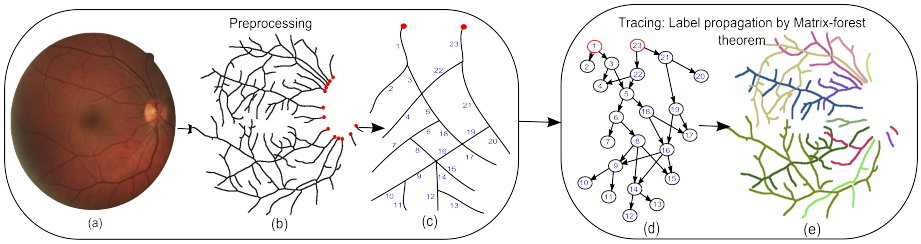

Fig. 1. Overview of the proposed tracing pipeline where the focus of this paper is the tracing step. Best view in color.

large dataset of synthetic retinal blood vessel images are generated to mimic various vessel spread and overlap scenarios. Our algorithm is simple and easy to implement. We present its intimate connections to a number of existing methods, including graph kernels and graph Laplacian based methods. Its computational complexity and the generalization error are also studied.

Related Work of Label Propagation on Directed Graphs Very few label propagation methods are specifically dedicated to directed graphs, including in particular [27, 28]. The most related work is that of [27], which generalizes their earlier framework of regularized risk minimization on undirected graphs [29] to directed graphs by discriminatively normalizing in-links and out-links, as well as adopting the directed graph Laplacian of [30]. One major difficulty in learning with directed graphs lies in the asymmetric nature of the weight matrix introduced by these directed edges or links. This is often regarded as cumbersome when aligning with the key concepts developed for undirected graphs that are symmetric by nature, such as graph Laplacians. It thus leads to the widely adopted symmetrization trick in e.g. the construction of graph Laplacians [27], or co-link similarity matrices [31]. Unfortunately important information conveyed by edge directions are still lost. In contrast, we directly work with asymmetric matrices, which is the key in preserving edge directions. In addition, the construction in both methods [27, 28] rely on *strongly connected* digraphs, that is, there is a directed path from any node to any other node of the graph. Since in practice the directed graphs are usually not necessarily strongly connected, a teleporting random walk trick (e.g [32]) is adopted by inserting bi-directional edges between all node pairs with an equal weight. The resulting method thus works only with non-sparse matrices, instead our algorithm works with a much broader set of digraphs called *weakly connected* digraphs, i.e. for an arbitrary partitioning of the digraph node set into two subsets, there is at least an edge connecting them regardless of the edge direction. This enable our algorithm to well-preserve the sparse graph structures and edge directional information of the input.

2 Our Approach

The problem of vessel tracing is to trace blood vessels by separating them into disjoint vessel trees, each starting from a unique root segment within the op-

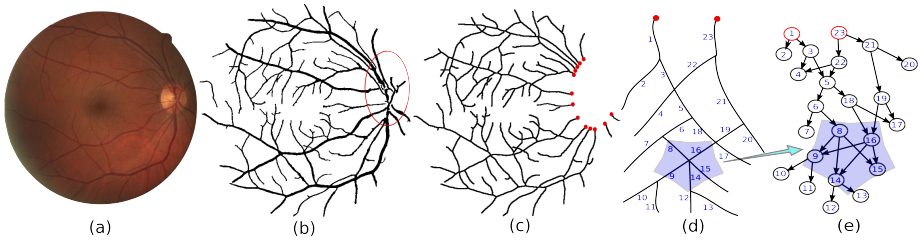

Fig. 2. Preprocessing Step. (a) An input image from DRIVE. (b) Binary image after segmentation. (c) Image after skeleton extraction and optic disk removal. The red elliptical area in (a) and (b) is the optic disk. The red dots in (c) are tips of the *root segments* identified as those directly contacting the optic disk. Note each root segment will induce a distinct vessel tree from the graph with itself being the tree root, due to the nature of blood flow in vessels. (d) A illustrative skeleton map, can be regarded as a subset from (c). (d) $\rightarrow$ (e) presents the transformation from skeleton to digraph. (e) The corresponding digraph G. The highlighted zone of nodes are shown as an example where the corresponding directed subgraph is formed. The segments marked with red dots at their tips are the root segments, with each being regarded as the labeled node for its class. In other words, each class (corresponds to a vessel tree) has exactly its root node labeled, which corresponds to a source node in graph.

tic disk, as illustrated in Figure 1(b). The primary difficulty is to resolve the challenging cross-over issues that are common in the retinal datasets. Figure 1 describes the pipeline of our approach that consists of *two* main steps: the *preprocessing* step for segmentation and skeleton extraction, as well as the *tracing* step where the digraph is made ready and the tracing task is cast as digraph-based label propagation using Matrix-forest theorem — the main focus of this paper. In what follows, we will briefly mention the preprocessing step while leaving the details to appendix, and will describe the tracing step in length.

2.1 The Preprocessing Step

The preprocessing step is comprised of the following three modules: *Segmentation*: As illustrated in Figure 2 (a)$\rightarrow$(b), an input retinal image is segmented into a binary image, with vessel pixels being foreground and the rest as background. Note this step is skipped for synthetic retinal images as they are already binary images. *Skeleton map*: Build a skeleton map from the binary image, and remove the optic disk area as marked within red ellipse in Figure 2(c). The cusps attached to the removed optic disk are the tips of root segments, which are also presented as red dots in Figure 2(d). *Skeleton to digraph*: A segment is defined in the skeleton as the group of connected pixels that ends in either a junction or a tip. This segment corresponds to a node in the resulting digraph, as shown in Figure 2(d)$\rightarrow$(e). Two nodes are then linked with a directed edge, when the two coinciding segments from the skeleton map contact and satisfy the ordering criteria detailed in appendix.

2.2 The Tracing Step by Matrix-Forest Theorem on Digraphs

The tracing problem becomes that of separating the vessel trees with only tree roots known, which can be equivalently formulated as a digraph-based label propagation problem with one labeled node per class (vessel tree). That is, all the source nodes are labeled in this problem, each with a unique label (tree), and the task is to make predictions on the rest unlabeled nodes by propagating the class labels following the underlining digraph structure. Different from undirected graphs that delineate symmetric weights between adjacent nodes, in digraphs, edge or link directions are well preserved in its weight matrix. This information is particularly useful and natural for tracing retinal vessels, which coincides with the physical principle governing how blood flows in vessels.

Problem Set-up Let $G = (\mathcal{V}, \mathcal{E}, W)$ denote a digraph, where $\mathcal{V} = \{v_1, v_2, \dots v_n\}$ is the set of nodes, $\mathcal{E}$ is the set of directed edges each connecting two adjacent nodes, and $W = [w_{ij}] \in \mathbb{R}^{n \times n}$ is the asymmetric non-negative matrix with $w_{ij} \geq 0$ denoting the weight associated with the directed edge from v_i to v_j. The out-degree of each node v_i is computed as $d_i^+ = \sum_{j=1}^{n} w_{ij}$. Denote the out-degree matrix $D = \mathrm{Diag}(d_1^+, \cdots, d_n^+)$, that is, $D = \mathrm{Diag}(W\mathbf{1})$, with $\mathbf{1}$ being an all-one column vector. Define the digraph Laplacian $L = D - W$. A row stochastic transition probability matrix, $P = [p_{ij}]$, can be constructed as $p_{ij} = \frac{w_{ij}}{d_j^+}$, or equivalently as $P = D^{-1}W$. It is noteworthy that undirected graphs can be regarded as a special subset in digraphs characterized algebraically by their symmetric weight matrix W, that is, the symmetric pair w_{ij} and w_{ji} correspond to bi-directional edges with the same weights. Indeed the digraph representation is a natural form for tracing retinal vessels, which coincides with the physical principle governing how blood flows in vessels. In this paper, we focus on a transductive inference scenario where labels from the set of few labeled nodes $\mathcal{V}_l$ are to be propagated to the rest nodes, namely the set of unlabeled nodes $\mathcal{V}_u$, with $\mathcal{V} = \mathcal{V}_l \cup \mathcal{V}_u$. The labels are multiclass, each corresponds to a separate vessel tree. To simplified the notation we assume $\mathcal{V}_l$ contains the first l nodes, $\mathcal{V}_l = \{v_1, \dots, v_l\}$. To accommodate label information, define a label matrix Y of size $n \times K$ (assuming there are K class labels available), with each entry Y_{ik} containing 1 provided the node i belongs to $\mathcal{V}_l$ and is labeled with class k, and 0 otherwise. Besides, define the ground-truth label vector $\mathbf{y}$, a vector of length n including two disjoint parts $\mathbf{y}_l$ and $\mathbf{y}_u$: $\mathbf{y}_l$ is the input label vector of length l over the set of labeled nodes, with each entry y_i for the input class assignment of node $v_i \in \mathcal{V}_l$; $\mathbf{y}_u$ is the hold-out ground-truth label for the unlabeled nodes, i.e. a vector of length $n - l$. Similarly, define the initial label vector $\hat{\mathbf{y}}$ containing also two parts, $\hat{\mathbf{y}}_l := \mathbf{y}_l$ and $\hat{\mathbf{y}}_u = \mathbf{0}$, where $\mathbf{0}$ is an all zero vector of length $n - l$. Define the prediction vector $\mathbf{y}^*$ with also two parts $\mathbf{y}_l^* := \mathbf{y}_l$, as well as $\mathbf{y}_u^*$ of length $n - l$, containing the prediction results, where each y_i^* denotes the predicted class assignment for a node $v_i \in \mathcal{V}_u$.

The proposed label propagation algorithm (shown in Algorithm 1 and referred to as MFTD) is derived based on matrix-forest theorem [25] of algebraic

digraph theory [26], as follows. Let w_{max} denote the entry in W containing the strongest signal, i.e. $w_{\max} = \max_{i,j} \left|w_{ij}\right|$. The forest matrix is defined as

$$S_1 := (I + \alpha L)^{-1}, \tag{1}$$

a normalized forest matrix where each (i, j)-th entry denotes the number of trees rooted at node i that also include the j-th node, as in Theorem 4 of [25]. It can be viewed as a generalization of the celebrated matrix-tree theorem (e.g. [33]) for undirected graphs to digraphs. Further, let $\tilde{L} := \lim_{\alpha\to\infty}(I + \alpha L)^{-1}$, which is a matrix of normalized spanning forests. Both S_1 and $\tilde{L}$ has a number of interesting properties [34]: Each entry of both matrices is non-negative, and both matrices are row-stochastic; $\tilde{L}$ resides in the null space of digraph Laplacian L, as $L\tilde{L} = \tilde{L}L = 0$; $\text{rank}(L) = n - \text{rank}(\tilde{L})$; $L + \beta\tilde{L}$ is non-singular for any $\beta > 0$, and is the "complementary perturbation of L" [35]. Indeed, this brings forward the second forest matrix:

$$S_2 := (L + \beta\tilde{L})^{-1}, \tag{2}$$

which is also termed the matrix of dense forest in [25]. As presented in what follows, varying the preprocessing schemes of normalizing W, we have two variants:

MFTD$_{\text{a}}$**:** It starts with a preprocessing effort to normalize W, $W \leftarrow \frac{W}{w_{\max}}$.
MFTD$_{\text{b}}$**:** Consider a different normalization of W as $W \leftarrow D^{-1}W$ instead, i.e. $P = W$.

Proposition 1. *Under normalization scheme of $W \leftarrow D^{-1}W$, the forest matrix becomes $S_1 = (1 - \tau)(I - \tau P)^{-1}$, with $\tau = \frac{\alpha}{1+\alpha}$.*

Proof. By normalization we know $P = W$. Apply the fact $L = D - W = I - P$, we have $I + \alpha L = (1 + \alpha)(I - \frac{\alpha}{1+\alpha}P) = \frac{1}{1-\tau}(I - \tau P)$.

When applied on the second forest matrix S_2 using (2), this clearly leads to two additional variants, denoted as MFTD$_{\text{c}}$ & MFTD$_{\text{d}}$, respectively.

One can interpret the (i, j)-th entry S_{ij} of the forest matrix S (being either $S := S_1$ or $S := S_2$) as quantifying the accessibility of a particle from a node v_i to visit node v_j along the digraph structure. This provides a notion of affinity from state i to j. The intuition is, if a state j is close to the initial state i in terms of graph structure, it will be visited by the particle more often than if it is far away from initial state, i.e., we visit our close relatives more often than our distant ones. Now define the affinity matrix as

$$A = SY. \tag{3}$$

It is a matrix of size $n \times K$, with each entry a_{ik} being associated with an affinity score of state i belonging to class k. To infer $\mathbf{y}_u^*$ of the unlabeled states $\mathcal{V}_u$, our algorithm predicts each entry's class assignment by identifying a label with the largest affinity score, namely

$$\mathrm{y}_i^* = \arg\max_k a_{ik}, \qquad \forall v_i \in \mathcal{V}_u. \tag{4}$$

Our algorithm is intuitive to understand, easy to implement, and by working with the matrix-forest theorem [25], we are able to retain the sparse and directed nature of the blood vessel tracing problem in the presence of crossovers.

Algorithm 1 Label Propagation by Matrix-Forest Theorem of Digraphs (MFTD)

Input: A digraph $G = (\mathcal{V}, \mathcal{E}, W)$, label information Y, $\mathbf{y}_l$, and $\alpha \in (0, 1)$.
Output: $\mathbf{y}_u^*$
Compute the out-degree matrix D.
Compute the affinity matrix by (3) and (1) (or (2)).
Produce prediction $\mathbf{y}_u^*$. The i-th entry is computed by (4), for an unlabeled node $v_i \in \mathcal{V}_u$.

Computational complexity As the forest matrix S is invertible, denote its inverse as $E := S^{-1}$. Usually it is computationally much cheaper to evaluate E than S, since $E = I + \alpha L$ or $E = L + \beta \tilde{L}$. Let us investigate the complexity of Algorithm 1, which is clearly dominated by the cost of computing the affinity matrix A in (3). In fact, it can be accomplished by instead solving the linear system of

$$EA = Y,$$

thus eliminating the need to explicitly compute any matrix inverse. For a general dense matrix P, the computational time is $O(n^3 + n^2 K)$. This is e.g. about the same complexity of [27], one of our main competing methods. Fortunately, E (as well as S) is usually a sparse matrix in our context, which can be exploited to reduce the computational time. There are many efficient solvers for large sparse linear systems, including both direct [36, 37] and iterative methods [38, 39]. In our implementation, we adopt the direct solver UMFPACK [36] which exists as a built-in routine (for LU, backslash, and forward slash functions) in MATLAB. The specific complexity depends on the size (n), the number of non-zero entries and the sparsity pattern of E, which remains a challenging task to provide a tighter complexity measure dedicated to our context. Nevertheless, our approach is practically much more efficient comparing to state-of-the-art methods including [27], as is empirically verified in experiments.

PAC-Bayesian Label-propagation Bound An learning error bound is provided here for our algorithm. This bound is an adaptation of the PAC-Bayes bound for label propagation developed in [40]. We start by reformulating our algorithm (i.e. both (1) and (2)) as an equivalent representation

$$\mathbf{h} = S\hat{\mathbf{y}}, \tag{5}$$

where $\hat{\mathbf{y}}$ is the initial label vector with partial labels $\hat{\mathrm{y}}_i \in \{\pm 1\}$ for $v_i \in \mathcal{V}_l$, and $\hat{\mathrm{y}}_i = 0$ otherwise. The obtained $\mathbf{h}$ is the "soft" label vector with h_i being the

"soft" label for node v_i, which will be assigned with class label $\text{sign}(\mathrm{h}_i)$ when making predictions. The hypothesis space is defined as

$$\mathcal{H} := \left\{ \mathbf{h} \;\middle|\; \mathbf{h} = S\hat{\mathbf{y}}, \|\hat{\mathbf{y}}\|_2 \leq \sqrt{l} \right\}. \tag{6}$$

Besides, for any label vector $\mathbf{h}$, define the *test error* as $\mathcal{L}_{l,n}(\mathbf{h}) := \frac{1}{n-l}\sum_{i=l+1}^{n} \ell(\mathrm{h}_i, \mathrm{y}_i)$ with respect to its 0/1 loss function ℓ satisfying $\ell(\mathrm{h}_i, \mathrm{y}_i) = 1$ if $\mathrm{h}_i \neq \mathrm{y}_i$ and $\ell(\mathrm{h}_i, \mathrm{y}_i) = 0$ otherwise, and define the *empirical error* of $\mathbf{h}$ as $\hat{\mathcal{L}}_{l,n}(\mathbf{h}) := \frac{1}{l}\sum_{i=1}^{l} \ell(\mathrm{h}_i, \mathrm{y}_i)$.

Theorem 1. *For any $\delta \in (0,1)$, with probability at lest $1-\delta$ over random draws of $\mathcal{V}_l$ from $\mathcal{V}$, the following bound holds for any $\mathbf{h} \in \mathcal{H}$*

$$\begin{aligned} \mathcal{L}_{l,n}(\mathbf{h}) \leq & \hat{\mathcal{L}}_{l,n}(\mathbf{h}) \\ & + \sqrt{\left(\frac{2\hat{\mathcal{L}}_{l,n}(\mathbf{h})n}{n-l}\right)\frac{ln\frac{l}{\delta} + 7ln(n+1)}{l-1}} \\ & + \frac{2\left(ln\frac{l}{\delta} + 7ln(n+1)\right)}{l-1}. \end{aligned} \tag{7}$$

Notice that when $\hat{\mathcal{L}}_{l,n}(\mathbf{h}) = 0$ (i.e. the "realizable case") the bound converges to 0 as $l \to \infty$ and the number of unlabelled instances is a t-th ordered polynomial of l (i.e. $n - l = O(l^t)$) for any t.

2.3 Connections to Existing Methods

Graph Kernels, and Random Walks with Restart Algebraically our algorithm is similar to several graph kernels, including the Von Neumann Kernel $K_{VND} = \sum_{k=0}^{\infty} \alpha^k W^k = (I - \alpha W)^{-1}$ [41], and the regularized commute time kernel $K_{RCT} = (D - \alpha W)^{-1}$ [29, 42]. These kernel functions are however constructed specifically from undirected graphs (i.e. within the cone of symmetric positive definite matrices) and based on considerably different motivations and derivations. Our approach is also related to PageRank [32], which resolves the issue of source nodes by teleporting random walks that introduce bi-directional edges to all node pairs with equal weights, i.e. $\bar{P} = (1-\eta)P + \frac{\eta}{n}\mathbf{e}\mathbf{e}^T$ with $\mathbf{e}$ a $n \times 1$ vector of all ones, and η a tiny positive real. A very similar idea is also used in random walks with restart (RWR) and their variants [43]: Instead of $\frac{1}{n}\mathbf{e}$ as in PageRank, they uses a vector of all zeros except for one being 1 to indicate the node where the random walk will be restarted, or a more generic probability distribution. Note that PageRank and RWR are different from our approach. First, both operate on graphs with *irreducible* Markov chains rather than the *absorbing* Markov chains considered in our context. By definition irreducibility requires each node can be reached from any other node, i.e. a strongly connected graph – algebraically this often gives rise to a dense matrix, as shown in the teleporting operation. Second, as side-effects of introducing the teleporting operation, the input graph structure is not well preserved, and weak edge signals also tend

to be washed away. In contrast our approach is able to retain the input graph structure as well as weak signals. This is more pronounced when $\tilde{Q}$ is sparse and of large size: The matrices in both methods are dense and thus demand significant amount of computation and storage efforts, while our approach is able to maintain the induced sparse matrix structure and is much more efficient in term of computation and storage.

Graph Laplacian in Undirected Graphs Our algorithm also works with *undirected* graphs as a special case (i.e. equivalent to bi-directional edges with equal weights). An interesting observation is that here our algorithm can be shown as a scaled variant of the graph Laplacian based method in [29], which has been specifically developed for undirected graphs. This is discussed in details in appendix.

Partially Absorbing Random Walks (PARW) [44] It can be shown that the absorbing Markov chains considered in our context is quite general: The random walks of [44] correspond to a very special kind of such absorbing Markov chains where the submatrix of W concerning transient nodes forms a symmetric non-negative matrix. In other words, the transient nodes are inter-connected with undirected edges, while the edges from transient to absorbing nodes are still directed. Details are relegated to appendix.

3 Synthetic Datasets

Three synthetic datasets are constructed for systematic performance evaluation under controlled setting, that have in total $17,000$ images. The Shreve's ordering is also used here to quantify the complexity of a tree, measured by the largest ordering value assigned to a tree segment (i.e. the root segment). To start with, thirty tree-like structures are hand-drawn, serving as building-blocks for simulating these synthetic images. These hand-drawn trees can be categorized into three groups: (1) Those with low complexity (root segment ordering value of 2-5); (2) Medium complexity (root value of 5-10); (3) High complexity (root value more than 10). Some examples are presented in Figure 3 subplots (a) and (b), where the tree axis (shown in dotted blue line) is defined in the direction of its length. To generate a synthetic image, the angular distance between two adjacent trees are defined as the spread angle γ between the two tree axes in counter-clockwise direction, as shown in subplot (c). Now, a synthetic image can be generated with three parameters: (1) Complexity of the trees in the image; (2) Number of trees; (3) Spread angle. Three synthetic image datasets are thus generated by varying these above parameters, where in each dataset, only one parameter varies while the rest two parameters are fixed.

- Dataset 1: In this dataset, the tree complexity is varying while the other two parameters are fixed, namely, the number of trees are fixed to 8 and the spread angle is set to $30\,^{\circ}$. This gives rise to 5 subsets of images within this

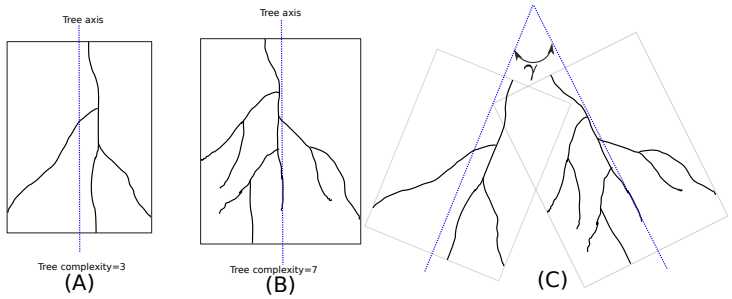

Fig. 3. Illustration on generating synthetic images. (A) An exemplar tree with low complexity. (B) An exemplar tree with medium complexity. (C) Illustration on how the synthetic image is generated.

dataset: (1) All trees are of low complexity; (2) Four out of the eight trees are with low complexity, and the rest four trees are with medium complexity; (3) All trees are of medium complexity; (4) Four out of the eight trees are with medium complexity, and the result four are with high complexity; (5) All trees are of high complexity. Two examples of this dataset are shown in Figure 4. 100 images are produced for each subset, in all $5,000$ images are generated in this dataset.

- Dataset 2: In this dataset, the number of trees are varying from the set $\{3, 5, 7, 9, 11, 12\}$, while the other two parameters are fixed: 1/3 from each complexity group, and the angles between trees is set to $30\,^{\circ}$. Two examples are displayed in Figure 5. Each subset contains 1000 images, in all we have $6,000$ images for this dataset.
- Dataset 3: In this dataset, the spread angle varies 60° – 360° with 60° step-size. while and the number of trees is set to 8, and for the tree complexity, the same strategy is used as that in dataset 2 (1/3 from each complexity group). Two examples are presented in Figure 6. Each subset contains 1000 images, in all we have $6,000$ images for this dataset.

4 Experiments

Our approach is evaluated in our in-house synthetic datasets, as well as two standard testbeds, DRIVE [45] and STARE [46]. The synthetic dataset is constructed in house that contains $17,000$ synthesized retinal images with varying densities of blood vessels (which strongly correlate with the frequency of cross-over occurrences among vessel branches), with its detail presented previously. Meanwhile, DRIVE dataset contains 40 retinal fundus images, and STARE has 20 fundus images. Exemplar images of the three datasets are illustrated in the first row of Figure 8.

Our approach is compared with the following state-of-the-art label propagation methods in machine learning:

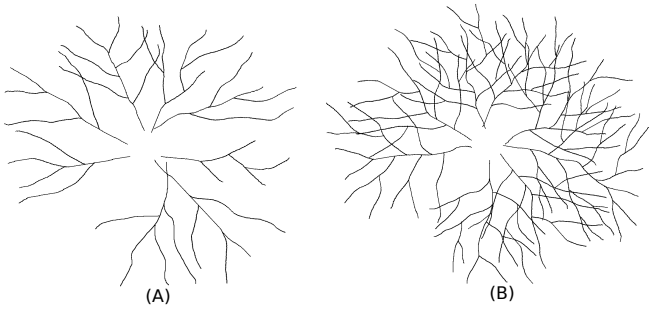

Fig. 4. Tracing: Synthetic dataset 1. Synthetic images with fixed number of trees and spread angle (angular distance between two adjacent trees), but with varying tree complexity. (a) Less complex image. (b) Complex image.

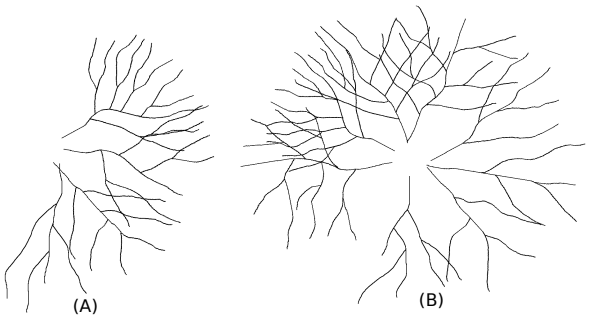

Fig. 5. Tracing: Synthetic dataset 2. Synthetic images with fixed tree complexity and spread angle, but with varying tree numbers. (a), (b) are two synthetic images composed with different number of trees.

- Network-only Bayes Classifier (NBC) [47].
- Network-only Link Based classifier (NLB) [48].
- Class Distribution Relational Neighbor classifier (CDRN) [49].
- Weighted Vote Relational Neighbor classifier (WVRN) [49].
- Digraph variant of the Commute Time Kernel classifier ($\mathrm{CTK_d}$) [42].
- (Original) Commute Time Kernel classifier for undirected graphs ($\mathrm{CTK_u}$) [42].
- Symmetrized Graph Laplacian (SGL) [27].
- Sum Over Path covariance kernel (SOP) [50].
- Von Neumann Diffusion Kernel on undirected graph(VNDK) [42].

Out of these methods, four (NBC, NLB, CDRN, and WVRN) are implemented by NetKit [49] in Java, SOP [50] is obtained from the authors, while the rest ($\mathrm{CTK_d}$, SGL, and Ours) are implemented by ourselves in MATLAB. Note the original Commute Time Kernel classifier (or $\mathrm{CTK_u}$) only works with undirected graphs, where the kernel is essentially L^+, the pseudo-inverse of the normalized

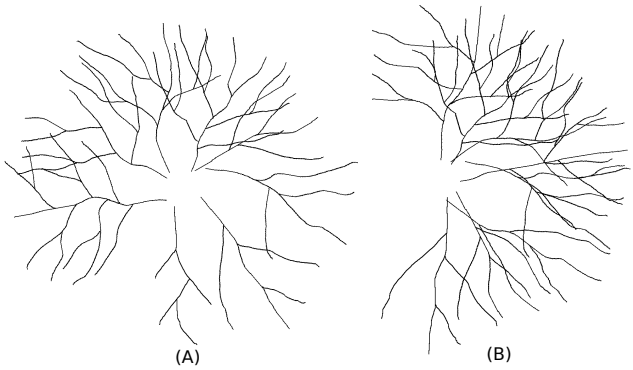

Fig. 6. Tracing: Synthetic dataset 3. Synthetic images with fixed tree complexity and tree numbers, but with varying spread angle. (a), (b) are two synthetic images composed with different spread angles.

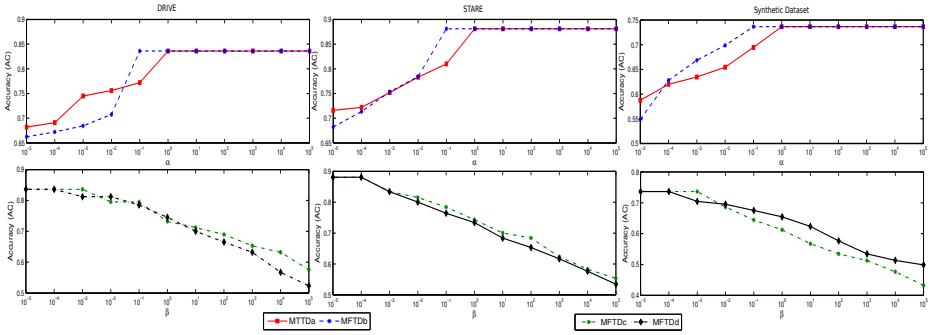

Fig. 7. Performance (AC) as a function of varying parameter value (α in col. 1 & β in col. 2) of our proposed algorithms $\text{MFTD}_{\text{a}-\text{d}}$. x-axis is in log-scale. See text for details.

graph Laplacian L. To work with digraphs, the graph Laplacian of [27] is used instead of the original undirected graph Laplacian of [29]. As a result, this variant is referred to as CTK_{d} in this paper. To summarize, CTK_{u} and $VNDK$ are undirected graph-based methods, CTK_{d}, SGL, SOP, and the proposed MFTD are digraph-based methods, while the rest methods are not graph-theoretical. To ensure fair evaluations, the internal parameters of the comparison methods are either set to as is from the authors' original source code, or as suggested by their respective authors: For example, according to [27], the regularization parameter is set to 0.1 and the jumping factor used in teleporting random walk is set to 0.01. In particular, for the four methods implemented by NetKit, the uniform local-classifier is used as the "local" model, and for collective inference relaxation labeling has been used for NBC, CDRN, and WVRN, while iterative

classification is used for NLB. This setting is reported [49] to deliver the best performance.

In term of evaluation metric, the micro-averaged accuracy (AC) [51] is adopted in all experiments as the accuracy measure, which is the sum of all true positive counts divided by the total number of instances. Besides, the DIADEM score (DS) [52] is also employed for the vessel tracing problem, which is a dedicated measure widely used by the biological tracing community.

Effect of Varying α (or β) of Our Approach Our first experiment is to evaluate the effect of varying the value of our algorithmic parameter, namely α in $\text{MFTD}_{\text{a}-\text{b}}$, and β in $\text{MFTD}_{\text{c}-\text{d}}$. This is performed on the standard testbeds: DRIVE and STARE datasets, and is presented in Figure 7, where the performance (AC) is displayed as a function of varying parameter value (α in row 1 & β in row 2) of our proposed algorithms $\text{MFTD}_{\text{a}-\text{d}}$. x-axis is in log-scale. Surprisingly, all four variants of our MFTD framework produces exactly the same results when $\alpha \geq 1$ and $\beta \leq 10^{-4}$, which is also verified under the AC criterion as in the figure. This is very interesting as despite their differences in algebraic forms and graph-theoretical interpretations, effectively these variants are *equivalent* characterized by their ability of tracing retinal blood vessels. To avoid redundancies, we will collectively refer to the performance of all these four variants as MFTD, and fix $\alpha = 1$ and $\beta = 10^{-4}$ throughout the experiments.

Synthetic Dataset

	NBC	NLB	CDRN	WVRN	CTK_{d}	CTK_{u}	SGL	SOP	VNDK	MFTD
AC	0.68	0.67	0.63	0.65	0.63	0.71	0.71	0.61	0.46	0.74
DS	0.63	0.61	0.62	0.62	0.61	0.68	0.64	0.59	0.21	0.71

DRIVE [45]

	NBC	NLB	CDRN	WVRN	CTK_{d}	CTK_{u}	SGL	SOP	VNDK	MFTD
AC	0.74	0.73	0.69	0.67	0.73	0.79	0.76	0.69	0.51	0.83
DS	0.72	0.71	0.68	0.64	0.72	0.74	0.75	0.67	0.32	0.77

STARE [46]

	NBC	NLB	CDRN	WVRN	CTK_{d}	CTK_{u}	SGL	SOP	VNDK	MFTD
AC	0.77	0.75	0.71	0.73	0.75	0.83	0.79	0.70	0.56	0.88
DS	0.75	0.74	0.68	0.69	0.74	0.78	0.76	0.67	0.39	0.82

Table 1. The micro-averaged accuracy (AC) and the average DIADEM score (DS) are reported for the synthetic dataset, as well as DRIVE and STARE. Note CTK_{u} and $VNDK$ are undirected graph-based methods, while CTK_{d}, SGL, SOP, and the proposed MFTD are digraph specific methods.

Comparison with the State-of-the-art Label Propagation Methods The performance of our approach is evaluated on three scenarios and is compared with nine state-of-the-art label propagation methods in machine learning, as reported in Table 4. Overall our approach consistently outperforms the other methods by a

large margin. The second best performer is usually $\mathrm{CTK_u}$, which are followed by SGL and $\mathrm{CTK_d}$, on the other side, VNDK tends to produce least accurate predictions. Another observation is that some of the undirected graph based methods (such as $\mathrm{CTK_u}$) may achieve better results when comparing with digraph based methods, while some might not doing very well (such as VNDK).

To facilitate visual inspection, Figure 8 presents exemplar images and comparison results. It suggests that empirically our approach delivers visually plausible tracing results when comparing to the ground-truths side-by-side, and the errors occur at those challenging spots that are often also difficult for human observers. This is in contrast with the comparison methods where the prediction mistakes are often not easy to be interpreted.

	DRIVE [45]				
	GOGP	$\mathrm{MFTD_a}$	$\mathrm{MFTD_b}$	$\mathrm{MFTD_c}$	$\mathrm{MFTD_d}$
Max. DS	0.71	0.77	0.77	0.77	0.77
Min. DS	0.15	0.68	0.66	0.58	0.52

Table 2. Comparison of maximum and minimum DIADEM scores (DS) between our approach and that of Engin et al. [18] on DRIVE dataset.

Comparison with State-of-the-art Tracing Systems The performance of our approach is compared with Global Optimization with Geometric Prior (GOGP), the state-of-the-art vessel tracing method by Turetken et al. [18] on DRIVE in Table 4. In [18], the authors have varied the cardinality of their k-minimum spanning tree algorithm and plotted the DIADEM scores (DS), where the min and max scores describe the robustness of the concerned system performing on the DRIVE dataset. For our case we varied our α (or β) parameter values, and show the comparison min and max scores in Table 4. Our algorithm clearly outperforms the state-of-the-art method, as we have the best DIADEM score of 0.77 which is better than the best DIADEM score of 0.71 of [18]. Besides, the performance of [18] varies dramatically from 0.15 to around 0.71 when changing the internal parameter, which suggests a sense of non-robust behaviour for their system. In comparison, the performance of our approach varies rather mildly, ranging overall from 0.52 to 0.77 over all four variants, i.e. $\mathrm{MFTD_{a-d}}$.

5 Conclusion and Outlook

A novel approach is proposed for tracing vessels in fundus images. The tracing problem is solved by utilizing matrix-forest theorem of digraphs. Practically our tracing approach performs very well in resolving many crossover scenarios and various complex situations. As a future direction we also plan to extend to work with broader range of applications such as neurite tracing and vesicle traffic tracing.

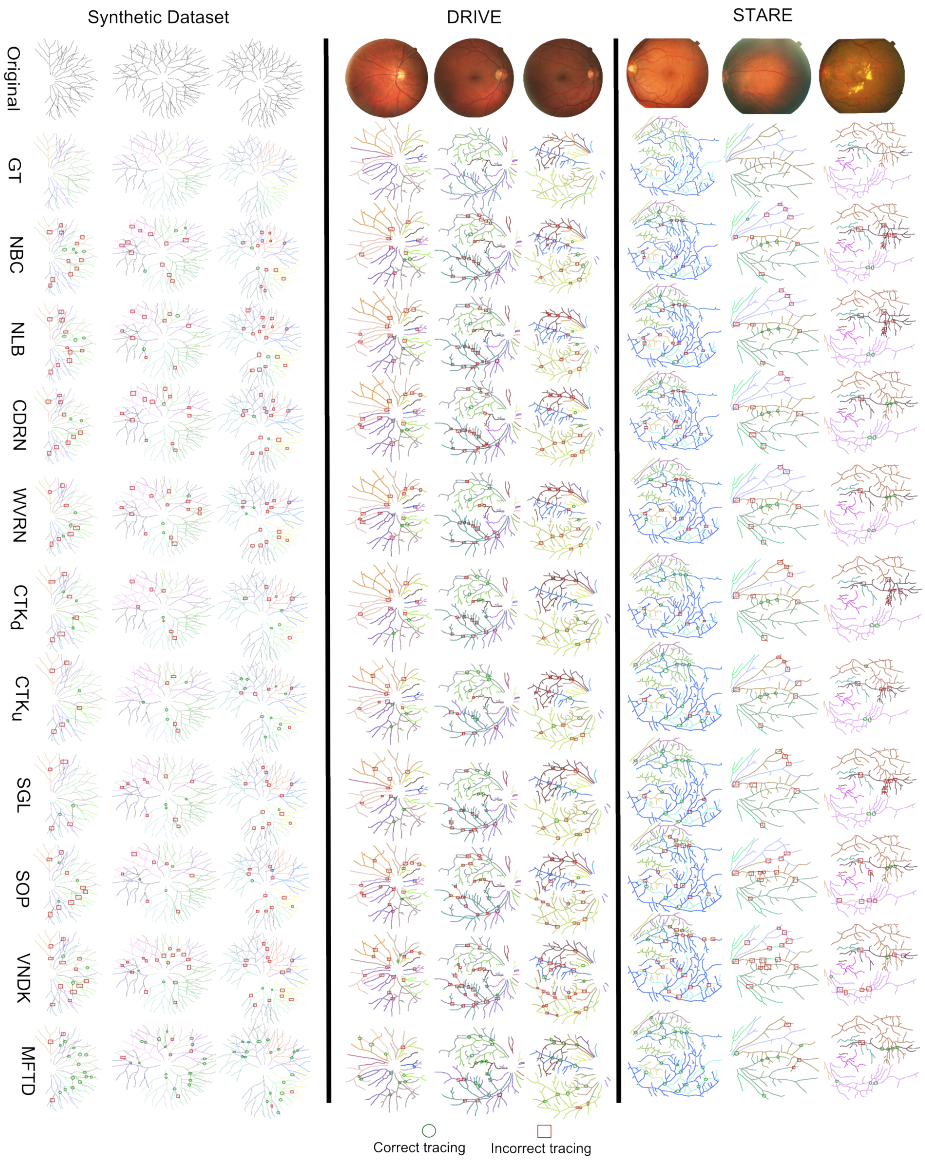

Fig. 8. Each row presents two exemplar retinal tracing results on Synthetic dataset, DRIVE, and STARE, respectively. Segments with the same color form a distinct vessel tree. Thus the number of colors equal to the number of classes (vessel trees). Selected correct (wrong) tracing segments are shown in green circles (red squares).

References

1. Viswanath, K., McGavin, D.: Diabetic retinopathy: Clinical findings and management. Community Eye Health **16**(46) (2003) 21–24
2. Mendonca, A., Campilho, A.: Segmentation of retinal blood vessels by combining the detection of centerlines and morphological reconstruction. IEEE Trans. Med. Imag. **25**(9) (2006) 1200–1213
3. S, G., Sivaswamy, J., Chandra, S.: Unsupervised curvature-based retinal vessel segmentation. In: ISBI. (2007) 1200–1213
4. Espona, L., Carreira, M., Penedo, M., Ortega, M.: Retinal vessel tree segmentation using a deformable contour model. In: ICPR. (2008)
5. Martinez-Perez, M., Hughes, A., Thom, S., Bharath, A., Parker, K.: Segmentation of blood vessels from red-free and uorescein retinal images. Med. Image Anal. **11**(1) (2007) 47–61
6. Soares, J., Leandro, J., Cesar, R., Jelinek, H., Cree, M.: Retinal vessel segmentation using the 2d gabor wavelet and supervised classification. IEEE Trans. Med. Imag. **25**(9) (2007) 1214–1222
7. Marin, D., Aquino, A., Gegundez-Arias, M., Brav, J.: A new supervised method for blood vessel segmentation in retinal images by using gray-level and moment invariants-based features. IEEE Trans. Med. Imag. **30**(1) (2011) 146–158
8. Bankhead, P., Scholfield, C., McGeown, J., Curtis, T.: Fast retinal vessel detection and measurement using wavelets and edge location refinement. PLoS ONE **7**(3) (2012) e32435
9. Wang, L., Bhalerao, A.: Model based segmentation for retinal fundus images. In: Scandinavian conf. on Image analysis. (2003) 422–429
10. Becker, C., Rigamonti, R., Lepetit, V., Fua, P.: Supervised feature learning for curvilinear structure segmentation. In: MICCAI. (2013)
11. Xu, X., Niemeijer, M., Song, Q., Sonka, M., Garvin, M., Reinhardt, J., Abramoff, M.: Vessel boundary delineation on fundus images using graph-based approach. (2011) 1184–1191
12. Can, A., Shen, H., Turner, J., Tanenbaum, H., Roysam, B., Roysam, B.: Rapid automated tracing and feature extraction from retinal fundus images using direct exploratory algorithms. (1999) 125–138
13. Grisan, E., Pesce, A., Giani, A., Foracchia, M., Ruggeri, A.: A new tracking system for the robust extraction of retinal vessel structure. In: IEEE EMBS. (2004)
14. Bekkers, E., Duits, R., ter Haar Romeny, B., Berendschot, T.: A new retinal vessel tracking method based on orientation scores. CoRR **abs/1212.3530** (2012)
15. Al-Diri, B., Hunter, A., Steel, D.: An active contour model for segmenting and measuring retinal vessels. IEEE Trans Med. Imaging **28**(9) (2009) 1488–1497
16. Chothani, P., Mehta, V., Stepanyants, A.: Automated tracing of neurites from light microscopy stacks of images. Neuroinformatics **9**(2-3) (2011) 263–278
17. Turetken, E., Benmansour, F., P.Fua: Automated reconstruction of tree structures using path classifiers and mixed integer programming. In: CVPR. (2012)
18. Turetken, E., Gonzalez, G., Blum, C., Fua, P.: Automated reconstruction of dendritic and axonal trees by global optimization with geometric priors. Neuroinformatics **9** (2011) 279–302
19. Turetken, E., Blum, C., Gonzalez, G., Fua, P.: Reconstructing geometrically consistent tree structures from noisy images. In: MICCAI. (2010)
20. Gonzalez, G., Turetken, E., Fleuret, F., Fua, P.: Delineating trees in noisy 2d images and 3d image stacks. In: Conference on Computer Vision and Pattern Recognition. (2010)

21. Ricci, E., Perfetti, R.: Retinal blood vessel segmentation using line operators and support vector classification. IEEE Trans. Med. Imag. **26**(10) (2007) 1357–1365
22. Martinez-Perez, M., ahd A. Stanton, A.H., Thom, S., Chapman, N., Bharath, A., Parker, K.: Retinal vascular tree morphology: a semi-automatic quantification. IEEE Trans Biomed Eng. **49**(8) (2002) 912–917
23. Shreve, R.L.: Infinite topologically random channel networks. Journal of Geology **75**(2) (1967) 178–186
24. Chapelle, O., Scholkopf, B., Zien, A.: Semi-Supervised Learning. MIT Press (2006)
25. Agaev, R.P., Chebotarev, P.Y.: Spanning forests of a digraph and their applications. Automation and Remote Control **62**(3) (2001) 443–466
26. Harary, F., Norman, R., Cartwright, D.: Structural models : an introduction to the theory of directed graphs. Wiley (1965)
27. Zhou, D., Huang, J., Schölkopf, B.: Learning from labeled and unlabeled data on a directed graph. In: ICML. (2005)
28. Cai, X., Wang, H., Huang, H., Ding, C.: Simultaneous image classification and annotation via biased random walk on tri-relational graph. In: ECCV. (2012)
29. Zhou, D., Bousquet, O., Lal, T., Weston, J., Scholkopf, B.: Learning with local and global consistency. In: NIPS. (2004)
30. Chung, F.: Laplacians and the cheeger inequality for directed graphs. Annals of Combinatorics **9**(1) (2005) 1–19
31. Wang, H., Ding, C., Huang, H.: Directed graph learning via high-order co-linkage analysis. In: ECML. (2010)
32. Page, L., Brin, S., Motwani, R., Winograd, T.: The pagerank citation ranking: Bringing order to the web. In: WWW. (1998)
33. Brualdi, R., Ryser, H.: Combinatorial Matrix Theory. Cambridge Uni. Press (1991)
34. Chebotarev, P.Y., Agaev, R.P.: Forest matrices around the laplacian matrix. Linear Algebra and its Applications **356**(1-3) (2002) 253–247
35. Meyer, C., Stadelmaier, M.: Singular m-matrices and inverse positivity. Linear Algebra and its Applications **22** (1978) 139–156
36. Davis, T.: Algorithm 832: Umfpack v4.3—an unsymmetric-pattern multifrontal method. ACM Trans. Math. Softw. **30** (2004) 196–199
37. Demmel, J., Eisenstat, S., Gilbert, J., Li, X., Liu, J.: A supernodal approach to sparse partial pivoting. SIAM J. Matr. Anal. App. **20** (1999) 720–755
38. Paige, C., Saunders, M.: Lsqr: An algorithm for sparse linear equations and sparse least squares. ACM Trans. Math. Softw. **8** (1982) 43–71
39. Saad, Y.: Iterative Methods for Sparse Linear Systems. 2nd edn. SIAM (2003)
40. Derbeko, P., El-Yaniv, R., Meir, R.: Explicit learning curves for transduction and application to clustering and compression algorithms. J. Artif. Intell. Res. **22** (2004) 117–142
41. Schölkopf, B., Smola, A.: Learning with Kernels. MIT Press, Cambridge, MA (2002)
42. Fouss, F., Francoisse, K., Yen, L., Pirotte, A., Saerens, M.: An experimental investigation of kernels on graphs for collaborative recommendation and semisupervised classification. Neural Network **31** (2012) 53–72
43. Tong, H., Faloutsos, C., Pan, J.: Fast random walk with restart and its applications. In: ICDM. (2006)
44. Wu, X., Li, Z., So, A., Wright, J., Chang, S.: Learning with partially absorbing random walks. In: NIPS. (2012)
45. Staal, J., Abramoff, M., Niemeijer, M., Viergever, M., van Ginneken, B.: Ridge based vessel segmentation in color images of the retina. IEEE Trans. Med. Imag. **23**(4) (2004) 501–509

46. Hoover, A., Kouznetsova, V., Goldbaum, M.: Locating blood vessels in retinal images by piecewise threshold probing of a matched filter response. IEEE Trans. Medical Imaging **19**(3) (2000) 203 –210
47. Chakrabarti, S., Dom, B., Indyk, P.: Enhanced hypertext categorization using hyperlinks. In: SIGMOD. (1998)
48. Lu, Q., Getoor, L.: Link-based classification. In: ICML. (2003)
49. Macskassy, S., Provost, F.: Classification in networked data: A toolkit and a univariate case study. JMLR **8** (2007) 935–983
50. Mantrach, A., Yen, L., Callut, J., Françoisse, K., Shimbo, M., Saerens, M.: The sum-over-paths covariance kernel: A novel covariance measure between nodes of a directed graph. IEEE Trans. PAMI **32**(6) (2010) 1112–1126
51. Sen, P., Getoor, L.: link based classification. Technical report, CS department, Univ of Maryland (2007) CS-TR-4858.
52. Gillette, T.A., Brown, K.M., Ascoli, G.A.: The diadem metric: comparing multiple reconstructions of the same neuron. Neuroinformatics **9**(2-3) (2011) 233–245

Appendix

Details of Our Segmentation Step

To facilitate the tracing step of our pipeline, the goal of the segmentation step here is to extract the vessel skeleton while maintaining the structural connections, as well as the corresponding point-wise radii along the skeleton (A point radius is to measure the thickness of a skeleton point in the orthogonal direction), based on which the retinal vessels can be faithfully reconstructed. This differs notably from the usual aim of most existing segmentation work, where the emphasis is to achieve a high classification accuracy. As the number of vessel pixels are much fewer comparing to the number of background pixels, often a high accuracy is achieved by missing many vessel pixels – a situation we try to avoid. In fact, our goal can be better described as *segmentation with a high recall*. It is critical for us to retain the vessel pixels that keep the local vein and artery branches from being broken or entirely missing. To achieve this, we resort to a cascade of two segmentation modules for producing our final segments. The first one in the cascade is a supervised segmenter as being described next, and the second one is an unsupervised segmenter that is specialized at recovering parallel thin branches, which often tend to be merged into one thick branch by the first module.

The First Segmentation Module: Supervised Segmenter In the first module, we implement a supervised segmenter using Gabor filters and GMMs, which is inline with existing supervised methods used for segmenting retinal vessels [6]. For each pixel in the training set, the Gabor response feature of 18 directions are computed and normalized to form the input features [6]. Two GMMs, each having 20 Gaussian components, with one for vessel and the other for non-vessel background pixels, are trained on these features as a pixel classifier. Then for a test image, by applying the trained GMMs we obtain the probability

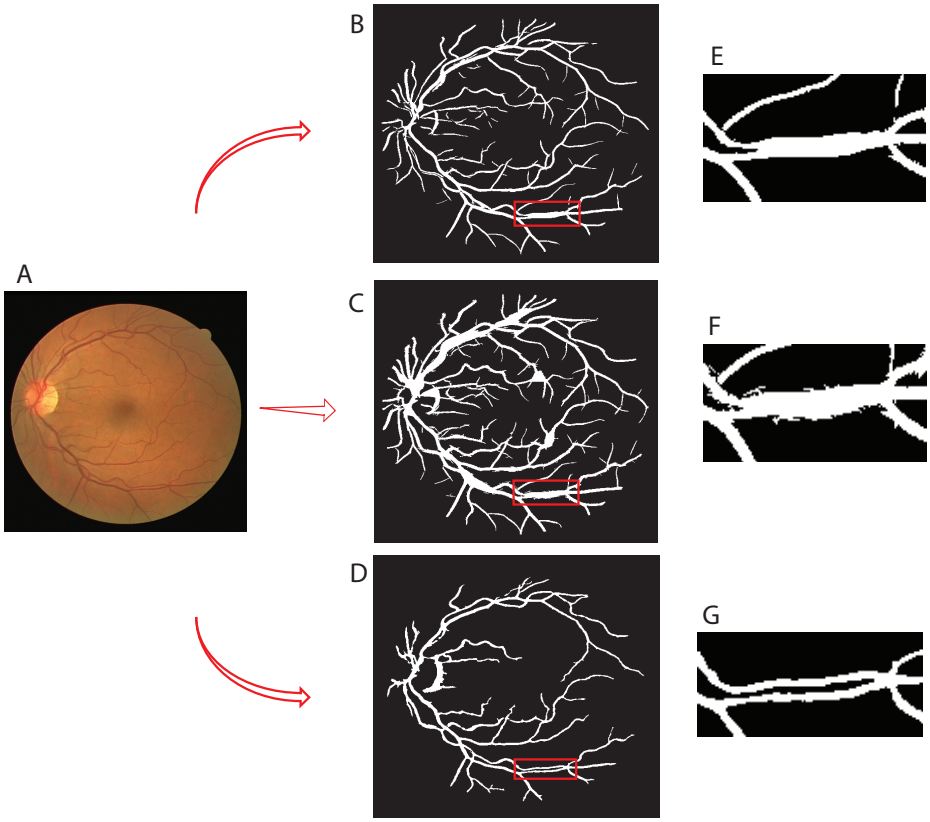

Fig. 9. Our segmenter: Based on existing methods (a) Original color fundus image. (b) Supervised segmentation result with the highest F1 score. (c) Supervised segmentation result with the highest recall. (d) Unsupervised segmentation method. (e) Two branches are wrongly segmented into one, small branches are disconnected. (f) Small branches are often connected back here. (g) Two branches are often kept separated from each other.

of a pixel being vessel or not. A probability map of the image is produced by maximizing over these two probabilities for each of the pixels.

We have observed that the desired segmentation in our context often stems from the result with highest F1 score, as e.g. demonstrated in Figure 9(b). This is reasonable, since with a high recall (i.e. less false negatives), we are still after a result with high precision (i.e. less false positives). Together they can be characterized by a single F1 score. As demonstrated in Figure 9(b) and (e), the output is reasonably clean, but many small branches of the retinal network are either entirely missing, or not connected. Moreover, the parallel thin branches tend to merge to form a thick branch, which is clearly not desirable for the tracing purpose. Therefore, we further adjust the threshold over a suitable range and obtain a second output with the highest recall: As illustrated in Figure 9(c)

and (f), the result tends to be very noisy and much thicker than what it should be. Nevertheless, by starting with the skeleton of the result with highest F1, followed by merging with the small branches from the one with highest recall, we are able to retrieve and reconnect those small branches that otherwise might be missed out. The point-wise radii issue is avoided by simply sticking to the highest F1's result.

Unfortunately, the result after merging the highest F1 and the highest recall outputs are still not satisfactory: It seems a characteristic of the supervised methods is that they tend to merge very close parallel branches into one branch, undesirable for our purpose of tracing. So we need to consider using a second module from unsupervised segmentation.

The Second Module: Unsupervised Segmenter We have attempted with a few existing methods and observed that the segmenter of [8], the one using Isotropic Undecimated Wavelet Transform (IUWT), empirically produces the best segmentation for the close and parallel branches, as illustrated in Figure 9(d) and (g). As a second add-on module of the cascade, based on the current partial result from supervised segmentation, the wrongly-merged thick branches are identified and replaced by the parallel branches from the second module.

Combining Supervised and Unsupervised Method of Segmentation For combining the images from supervised and unsupervised method of segmentation (total 3 images) we have followed these steps.

- We have used the binary segmented images and extracted the skeleton from them.
- Depending on the number of neighbours, we have marked the skeleton pixels as body pixels (those pixels with 2 neighbours), branching pixels (those with 3 or more neighbours) or terminal pixels (those with one neighbour). We define a vessel segment as a group of body pixels which are connected together.
- We calculate the median diameter of each vessel segment by estimating the diameter on each point of the vessel segment skeleton by following the method described in [8]. Then we calculate the mean diameter (d_m) of all the vessel segments for a particular image.
- We have replaced the segments which have diameter less d_m from Figure 9(b) by the same segments from Figure 9(c). While replacing we always took care about the continuity of the connected segments and we have always preferred thin and longer segments than thick and shorter segments.

Computing the Weight Matrix W

The weight matrix W is a symmetric matrix of size $n \times n$, with each entry contains a non-negative real value. Clearly W is of central importance in our approach as it is assumed to encode sufficient information from the input image

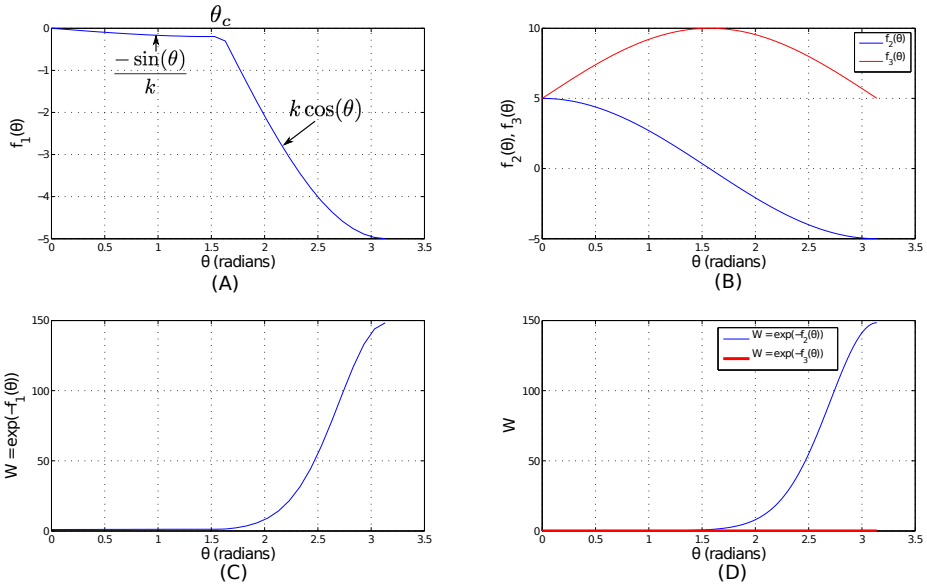

Fig. 10. Influence of θ on the values of functions f_1, f_2, f_3 and their corresponding W: (a) Variation of $f_1(\theta)$ with θ. (b) Variation of $f_2(\theta)$ and $f_3(\theta)$ with θ. (c) W corresponding to sub-figure (a). (d) W corresponding to sub-figure (b).

data. In this paper, the orientation-based features are proposed as the sufficient statistics toward computing W as below:

– *Segment orientation and angle between segments:* For each skeleton point, the first eigenvector of the Hessian matrix of local image patch determines an orientation. A (usually curved) vessel segment comes with two ends, thus has two local orientations. For each end, its orientation is computed by taking an average of the eigenvector of the last ten skeleton points from this end. It is then used to compute $\theta \in [0, \pi)$, the angle between two adjacent segments.

We also devise three functions that will be used in constructing W. They are

$$f_1(\theta) = \begin{cases} -\frac{\sin(\theta)}{k} & \text{if } \theta \in [0, \theta_c) \\ -\frac{sin(\theta_c)}{k} & \text{if } \theta \in [\theta_c, \arccos(\frac{-\sin\theta_c}{k^2})) \\ k\ \cos(\theta) & \text{if } \theta \in [\arccos(\frac{-\sin\theta_c}{k^2}), 180\,^\circ), \end{cases} \tag{8}$$

$$f_2(\theta) = k\, cos(\theta), \tag{9}$$

$$f_3(\theta) = k + k\, sin(\theta), \tag{10}$$

where the parameters k and θ_c are experimentally fixed as $k = 5$ and $\theta_c = 80\,^\circ$.

The diagonal elements of the weight matrix W are always zero, $W_{ii} = 0, \forall i$. The computation of the weight matrix elements W_{ij} for two different segments $i \neq j$ involves the following rules:

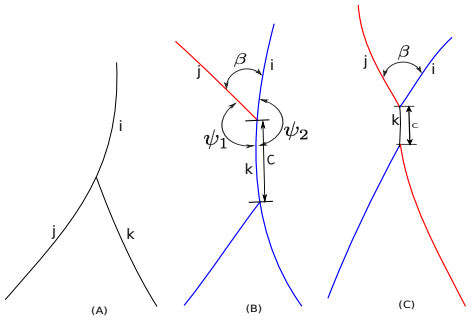

Fig. 11. Three exemplar 3-cliques: (a) A simple 3-clique. (b) A 3-clique created by the termination of one segment (red one) on another segment (blue one). (c) Two 3-cliques created by the wrong skeletonization of a 4-clique.

– *3-Clique:* i.e. there are three adjacent segments in the junction.
In this scenario we use the following equation:

$$W_{ij} = \exp(-f_1(\theta)). \tag{11}$$

The rationale behind choosing this function comes from the intuition that we want to encourage small changes of local curvatures between two connected segments, while punishing those with larger curvature changes. Figure 10 (a) and (c) shows the effects of varying θ in f_1 and W.
Figure 11 presents three exemplar 3-cliques: subplot (a) (case-A) is the standard branching situation where the segments (marked as i, j and k) belong to the same label. In subplot (b) (case-B), the red segment terminates on a blue segment and creates a 3-clique. Here the segment i should have a label different that of segment j and k. In subplot (c) (case-C), a crossover between the blue and the red branches is converted into two 3-cliques due to error in skeletonization. We differentiate case-A from case-B,C with the help of the segment ordering algorithm explained previously. Case-B and case-C and further differentiated by checking the length of segment k (C pixels). If $C \leq C_{critical}$ then we consider it as case-C, otherwise it is case-B. The value of $C_{critical}$ is experimentally set as $C_{critical} = 10$.
Then we find the orientation difference for two segment pairs (i, k) and (j, k). Those two angles are ψ_2 and ψ_1 (shown in Figure 11(b)). Now we calculate the W_{ik} and W_{jk} as following:
If $\psi_1 \geq \psi_2$ then,

$$\begin{aligned} W_{jk} &= \exp(-f_1(\theta)) \\ W_{ik} &= \exp(f_1(\theta)) \end{aligned} \tag{12}$$

else

$$\begin{aligned} W_{ik} &= \exp(-f_1(\theta)) \\ W_{jk} &= \exp(f_1(\theta)) \end{aligned} \tag{13}$$

– *4-Clique:* i.e. there are four fully-connected segments in the junction. As displayed in Figure 12, if A, B, C and D are four pixels connecting the four segments X, Y, W and Z in a crossover setting, intuitively we can see that only $\overline{AC}$ and $\overline{BD}$ line should intersect with each other. The other pairs $(\overline{AB}, \overline{CD})$ and $(\overline{AD}, \overline{BC})$ are not able to intersect within the convex hull of the four points (A, B, C, D). Hence from the set of feasible line segments $\{(\overline{AB}, \overline{CD}), (\overline{AD}, \overline{BC}), (\overline{AC}, \overline{BD})\}$ we can easily identify the $(\overline{AC}, \overline{BD})$ pair that are able to crossover. As a result, higher weight should be assigned to the segment pair (X, Z) (the segments which contains the points A and C) and (Y, W) (the segments which contains the points B and D). The subplots of Figure 10(b) and (d) suggest to define the following function form

$$W_{ij} = \exp(-f_2(\theta)) \tag{14}$$

for these pairs of interest, as well as the function form

$$W_{ij} = \exp(-f_3(\theta)) \tag{15}$$

for the rest less favourable pairs.

– *5/6-Clique:* As presented in Figure 13(a) for 5-clique scenario, one segment (the blue ones) crosses over another segment (the red ones) at the branching point. Here the goal is to divide them in two groups, one having 2 segments (j and k in subplot (a)) and the other having 3 segments (i, l and m). From segment ordering we know (i, j) are assigned larger values than the rest, so we already have one member from each group. The rest of the members can be assigned by the usual "smooth curve around the junction point" assumption and employing $f_1(\theta)$ in the same way as in 3-clique. As presented in Figure 13(b), for 6-clique two crossovers happen at the same location. The target here is also to divide them in two groups: one having 3 segments (i, l and m) and the other having 3 segments (j, n and k). Similarly, from the segment ordering we already know that the two nodes with large index values (i, j), belong to different groups, so we employ $f_1(\theta)$ in a same way as in 5-clique.

Connections to Graph Laplacian in undirected Graphs

Here we shown that when operating as random walks on undirected graphs, our algorithm is equivalent to a scaled variant of the graph Laplacian method of [29]. For an undirected graph G, denote $S = D^{-\frac{1}{2}} W D^{-\frac{1}{2}}$, and define $P = WD^{-1}$, where $W = [w_{ij}]$ is a symmetric matrix and $D = \mathrm{diag}(d_1, \cdots, d_n)$, with $d_i = \sum_j w_{ij}$. Now consider applying our algorithm (i.e. (3) and (4)) on undirected graph G. Since

$$A = (I - \alpha P^T)^{-1} Y = D^{-\frac{1}{2}} (I - \alpha S)^{-1} D^{\frac{1}{2}} Y,$$

we have

$$D^{\frac{1}{2}} A = (I - \alpha S)^{-1} \Big(D^{\frac{1}{2}} Y \Big).$$

Notice that since the goal is to choose the best element from the current row i as in (4), the result will not change by multiplying an additional constant $d_i^{\frac{1}{2}}$ to all elements in the row. Define $\hat{\mathrm{A}} := D^{\frac{1}{2}}\mathrm{A}$, and let $\hat{Y} := D^{\frac{1}{2}}Y$ we now have

$$\hat{A} = (I - \alpha S)^{-1}\hat{Y},$$

which recovers the update formula of [29], with the only difference that instead of Y, $\hat{Y}$ is used here as a row-wise scaled variant.

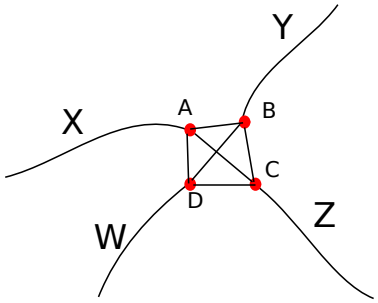

Fig. 12. A 4-clique. Here we know the (x, y) coordinates of points A, B, C, D in the image plane, so we can find the equation of straight lines $\overline{AC}, \overline{BD}, \overline{AB}, \overline{BC}, \overline{CD}, \overline{AD}$. We take the pair of straight line equations $(\overline{AC}, \overline{BD})$, $(\overline{AB}, \overline{CD})$ and $(\overline{AD}, \overline{BC})$ and find the intersecting points between the pairs. Only one pair $(\overline{AC}, \overline{BD})$ intersect within the convex hull of points (A, B, C, D) while others pairs are either parallel or will intersect outside the convex hull. By using this we understand that A pairs with C and B pairs with D. Accordingly we assign (X, Z) (The segments containing points A and C) and (W, Y) (The segments containing points B and D) pair with weight from Equation 14 (Blue curve in Figure 10(d)) and for other pairs we assign from Equation 15 (Red curve in Figure 10(d)).

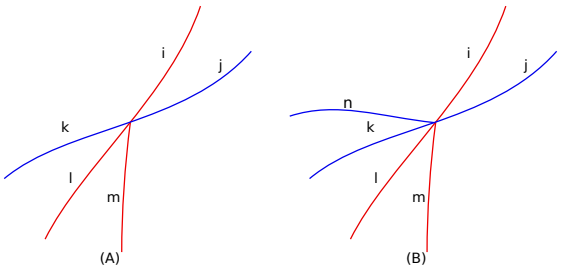

Fig. 13. 5- and 6-cliques. (a) A 5-clique. (b) A 6-clique. In both the scenarios weights are assigned using the same function as 3-cliques.