BMC Bioinformatics

**METHODOLOGY ARTICLE**

**Open Access**

# Scalable robust graph and feature extraction for arbitrary vessel networks in large volumetric datasets

Dominik Drees[1], Aaron Scherzinger[2], René Hägerling[4], Friedemann Kiefer[3] and Xiaoyi Jiang[1*]

*Correspondence:
xjiang@uni-muenster.de
[1] Faculty of Mathematics
and Computer Science,
University of Münster,
Münster, Germany
Full list of author information
is available at the end of the
article

## Abstract

**Background:** Recent advances in 3D imaging technologies provide novel insights to researchers and reveal finer and more detail of examined specimen, especially in the biomedical domain, but also impose huge challenges regarding scalability for auto-mated analysis algorithms due to rapidly increasing dataset sizes. In particular, existing research towards automated vessel network analysis does not always consider memory requirements of proposed algorithms and often generates a large number of spuri-ous branches for structures consisting of many voxels. Additionally, very often these algorithms have further restrictions such as the limitation to tree topologies or relying on the properties of specific image modalities.

**Results:** We propose a scalable iterative pipeline (in terms of computational cost, required main memory and robustness) that extracts an annotated abstract graph rep-resentation from the foreground segmentation of vessel networks of arbitrary topol-ogy and vessel shape. The novel iterative refinement process is controlled by a single, dimensionless, a-priori determinable parameter.

**Conclusions:** We are able to, for the first time, analyze the topology of volumes of roughly 1 TB on commodity hardware, using the proposed pipeline. We demonstrate improved robustness in terms of surface noise, vessel shape deviation and anisotropic resolution compared to the state of the art. An implementation of the presented pipe-line is publicly available in version 5.1 of the volume rendering and processing engine Voreen.

**Keywords:** Graph extraction, Vessel network, Scalability, Large volume processing

## Background

As one of the important problems in biomedical imaging, segmentation provides the foundation for tasks like quantification and diagnostics [1]. In particular, the study of vascular networks using modern 3D imaging techniques has become an increasingly popular topic of interest in biomedical research [2–7], as 2D slice analysis is limited to a small subset of the available data and cannot capture the 3D structure of vessel net-works [8]. At the same time, analysis of 3D datasets by visual inspection is error prone
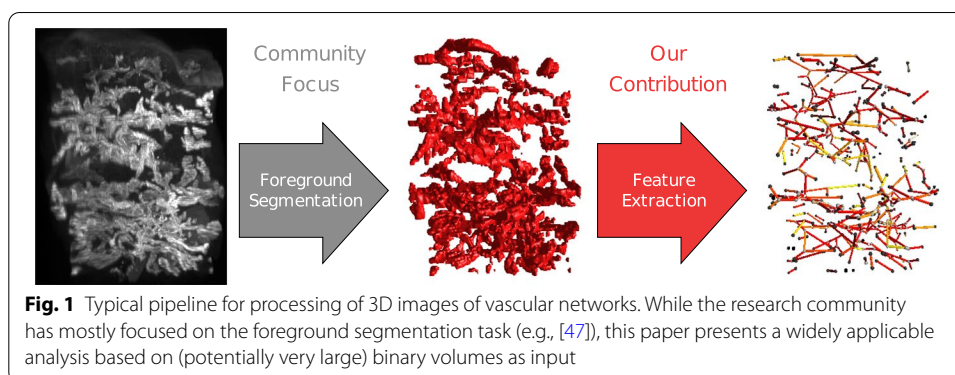
[9]. Thus, for quantifiable results or novel insights into data properties invisible to the human observer, automatic image processing techniques are required. In light of different imaging techniques, modalities and problem domains, the generation of a voxel-wise foreground segmentation can be seen as a sensible intermediate step for the automatic processing of vascular network images (see Fig. 1). However, the next step of calculating topological, morphological or geometric features—a key requirement for the application in biomedical research and beyond—has not received sufficient attention from the research community [10], which mostly focused on developing novel methods [11–13] and software [14, 15] in the segmentation domain.

In recent years, improvements in imaging technology and procedures are providing images of increasing resolution (in terms of the number of voxels per unit volume) and thus offer new chances for finer grained analysis, but also pose new challenges for image analysis algorithms. The analysis of fine structures down to the level of capillary networks promises great insights. At the same time it is desirable to analyze large networks at once in order to obtain as much topological information as possible and avoid artifacts at the boundary of the volume. Modern microscopy hardware generates a single volumetric image file of hundreds of GBs or even TBs (the same order of magnitude as hard drive capacity) per sample which existing vessel analysis approaches are not able to process. The problem of rapid advances in technology and increasing data sizes is also present in other domains such as cell type identification [16], but is especially pressing in 3D image analysis where datasets exceed the main memory of not only commodity hardware but also specialized workstations. Consequently, for a vessel network analysis pipeline to be useful now and in the future, it has to be *scalable* both in terms of memory and runtime, and it should be invariant with regard to increases in image resolution. For application in a wide range of biomedical domains, a pipeline should further not rely on specific imaging conditions, dataset-dependent parameters, a cylindrical vessel shape, isotropic image resolution or a specific network topology.

In this paper we present a pipeline designed to fulfill these requirements while extracting the topology, centerlines and edge associated features from binary volumetric images. This is achieved by our *main contribution*, a novel iterative refinement approach and careful design of all pipeline stages.

Generally, analyzing ultra large datasets by scalable algorithms is still a huge challenge. For instance, even in the very recent image segmentation tool Biomedisa [17] (based on the popular random walker) a dataset of $900 \times 1303 \times 4327$ voxels had to



**Fig. 1** Typical pipeline for processing of 3D images of vascular networks. While the research community has mostly focused on the foreground segmentation task (e.g., [47]), this paper presents a widely applicable analysis based on (potentially very large) binary volumes as input

be split for processing, despite testing on a system with amply resources (750 GB RAM (random access memory), 4 NVIDIA Tesla V100) compared to the standard PC (personal computer) as intended in our work. In [18] we presented a hierarchical segmentation algorithm. High-quality segmentation could be demonstrated on datasets of $9070 \times 12732 \times 1634$ voxels (377 GB, 37 times the size of the dataset processed by Biomedisa) on a standard PC. This paper represents another work on scalable algorithm development towards robust graph and feature extraction for arbitrary vessel networks in large volumetric datasets.

### Related work

Chen et al. [19] use the neighborhood-based voxel classification definitions of [20] for analysis of hepatic vasculature. They also discuss other alternative voxel-based skeleton extraction methods, but note that they are not suitable: Distance transformation-based methods [21] do not guarantee connectedness of the extracted skeleton, while Voronoi-skeleton methods [22] are comparatively time consuming. Furthermore, Chen et al. [19] perform extensive analysis to denote a single voxel of the skeleton as a branch point. They extract a graph by following skeleton voxels in a tree search, breaking up cycles in the graph. They do not perform any operations to remove erroneous branches—likely because for the processed low resolution volumes the effect of noise is negligible.

Drechsler and Laura [23] extend [19] by decomposing the generated skeleton into segments, computing a number of properties for each segment and generating a labeled graph structure from the segments. The authors observe that their algorithm is very sensitive to noise in the foreground segmentation, but do not propose any means to reduce this effect.

Palágyi [24, 25] proposes isthmus-based [26] thinning algorithms and shows that they reduce the sensitivity compared to other approaches, although they do not completely eliminate the problem.

Chen et al. [27] present an alternative thinning-based skeletonization method for analysis of hepatic vasculature as part of the previously published pipeline [19]. They use the voxel classification methods of [20], but present a different algorithm. As a preprocessing step they suggest filling-hole techniques and morphological operations to remove cavities and irregularities on the surface of the object.

Chothani et al. [28] propose a pipeline for tracing of neurites from light microscopy image stacks. It comprises foreground segmentation, skeletonization using a voxel coding algorithm [29], tree construction using the scalar field of the skeletonization step and refinement including pruning based on length as well as separation of branches that appear to be connected due to limited image resolution. The separation of branches in 3D should not be necessary using higher resolution images. The authors note that in this case due to the increased volume size, advances in processing capabilities are required.

Almasi et al. [30] present a method of extracting graphs of microvascular networks from fluorescence microscopy image stacks. In contrast to other methods operating on binary images they use imaging method specific information to improve the detection of vascular branches with disturbed image signal.

Cheng et al. [31] analyze connected 3D structures such as vessel systems and metal foams in binary volumes. They use topological thinning [32] and merge resulting

branching points using an algorithm based on regional maximal spheres and maximal inscribed spheres with regard to the object surface. While their algorithm is linear in runtime complexity, even for volumes of $10^9$ voxels (a gigabyte assuming one byte per voxel) it requires hours of computation time.

In [33], the authors use a non-topology preserving distance field guided voxel thinning algorithm. They construct a graph from an intermediate representation based on voxel neighborhood, but do not describe an efficient implementation. The thinning step creates holes in irregularly shaped vessels which is used to detect arteriovenous malformations, but makes is unsuitable for irregularly shaped (e.g., lymphatic) vessels.

*Rayburst sampling* [34] extracts precise measurements of radius, volume and surface of spatially complex histopathologic structures (given a centerline representation of a vessel). The method operates on grey value images and detects the surface by sampling along rays originating from the measurement point. For each point a large, but configurable number of rays is emitted to estimate the local morphology.

Finally, as many vessel network analysis pipelines include a skeletonization step, the work of Bates et al. [35] is worth of note. The authors demonstrate binary vessel segmentation using convolutional recurrent neural networks, including a variant for centerline extraction. However, when compared to traditional skeletonization algorithms, the complex decision process of artifical neural networks has disadvantages: No hard guarantees of connectedness or the thickness of skeletal branches are available (without further processing). Additionally, this approach may have problems with high resolution datasets where the vessels are wider than the field of view of the network, thus making it impossible to determine the centerline position.

All of the above works either do not mention processing of very large data samples (and thus likely do not support it) or explicitly mention that large datasets pose a problem for their method. In this work we want to fill this gap by presenting a widely applicable pipeline that is designed to handle very large input volumes and fulfil requirements that are outlined below.

### Requirements for broad biomedical application

For the discussion in the remainder of this paper, in this section we will structure and motivate the requirements that a vessel network analysis pipeline should fulfill in order to be applicable to a wide range of datasets and circumstances encountered in practice.

#### *Primary requirements*

The method should be scalable with regard to **runtime** (P1). As all voxels of a volume can be expected to be taken into account by the method, the runtime can be expected to be at least linear in the number of voxels *n*. In order to be applicable to increasingly large datasets, a runtime of $\mathcal{O}(n^2)$ would be unacceptable, but quasilinear algorithms (e.g., in $\mathcal{O}(n \log n)$) can still be executed. Beyond computational complexity, special care has to be taken for example in terms of memory access locality to avoid an increase in computation time by a large constant factor.

The method should be scalable with regard to **memory requirements** (P2). We can expect the input datasets to fit onto (commodity) non-volatile storage (disk), but not necessarily in main memory. While the price per capacity of both disks and main

Drees *et al. BMC Bioinformatics*    (2021) 22:346

Page 5 of 28

memory fall exponentially, the rate of (exponential) decrease is higher for disks than for memory [36, 37]. Consequently, the ratio of average disk size to average RAM size grows over time. The exact relationship is not easy to establish as it changes over time [37]. Here we assume that disk and volatile memory size *roughly* grow in a relationship of $m$ to $m^{\frac{2}{3}}$ for a disk size of $m$ which also corresponds to the common practice of loading a 2-dimensional slice of a large 3-dimensional volume. Therefore the required main memory size of the method should not exceed $\mathcal{O}(m^{\frac{2}{3}})$.

The method should exhibit **invariance with regard to image resolution** (P3). For a fixed size specimen, an increase in image resolution can often result in an increase in surface noise-related artifacts. This is not handled well by simple topological thinning-based methods (e.g., [23]) which is unacceptable for analysis of large datasets.

### *Secondary requirements*

From the desire to apply the method in practice we derive a set of further requirements that are not directly related to scalability:

The method should be **unbiased** (S1), i.e., not dependent on a set of parameters that are required to be selected carefully and *correctly* depending on the input image.

The method should be **robust in terms of deviations from the cylindrical shape** (S2) of the analyzed network structure. While blood vessels are cylindrical, lymphatic vasculature as an example is often highly irregular (see Fig. 1), but should still be processed correctly.

The method should handle volumes of **anisotropic resolution** (S3). Volumetric imaging techniques often create datasets with voxel spacing that differs between coordinate axes. As an example, in light sheet microscopy datasets are constructed from a series of (2D) slices for which the spacing is independent from the x-y-resolution. Methods operating on the voxel grid of a volume have to consider this.

The method should be able to fully analyze networks of **arbitrary topology** (S4). Existing methods often assume a tree structure and either fail to operate on other topologies or drop information [19]. However, for lymphatic vessels and capillaries, or even larger structures (*cerebral arterial circle*) the assumption of a tree topology is invalid.

The method should be able to analyze images **independently of the imaging conditions** (S5). Concrete imaging conditions (i.e., distribution of gray values or fluorescent staining techniques in microscopy imaging) vary between domains. In order to be widely applicable, the method should not be dependent on concrete models of imaging conditions. Our method achieves this by operating solely on binary input data and leveraging the broad range of other research on vessel image segmentation [11, 12].

## Results

### Pipeline overview

The proposed pipeline comprises four stages which are evaluated iteratively (see Fig. 2) which are summarized briefly here and described in more detail in Methods. In the *Skeletonization* stage the foreground of the binary input volume is reduced to a voxel skeleton using a topological thinning-based algorithm, similar to [23]. We use a modified version of [20] which can be implemented efficiently both in terms of computational complexity and disk access to the out-of-core dataset. Next, we build a graph

**Fig. 2** A schematic overview of the proposed pipeline
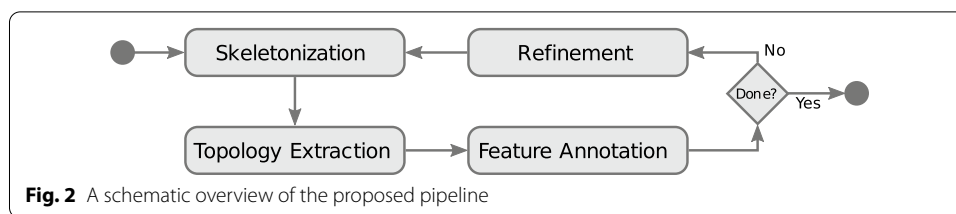
**Table 1** Datasets used for evaluation of the proposed method

| Dataset | Voxel dimensions | Voxel spacing | Size | Origin |
|---|---|---|---|---|
| Lymphatic 1 | $135 \times 160 \times 213$ | $32 \times 32 \times 16\,\mu m^3$ | 4.4 MiB | [5] |
| Lymphatic 2 | $135 \times 160 \times 117$ | $16 \times 16 \times 32\,\mu m^3$ | 2.4 MiB | [5] |
| Lymphatic 3 | $135 \times 160 \times 205$ | $16 \times 16 \times 16\,\mu m^3$ | 4.2 MiB | [5] |
| Synthetic 1–10 | $201 \times 201 \times 201$ | $200 \times 200 \times 200\,\mu m^3$ | 7.7 MiB | [39] |
| Kidney 1 | $9070 \times 12732 \times 1634$ | $0.76 \times 0.76 \times 3\,\mu m^3$ | 176 GiB | [40] |
| Kidney 2 | $5153 \times 4791 \times 1213$ | $0.76 \times 0.76 \times 3\,\mu m^3$ | 28 GiB | [40] |

representation from the voxel skeleton. In contrast to [23] this Topology Extraction stage is implemented as a single sweep over volume, engineered for memory locality, avoiding random disk access. In the Feature Annotation stage, we compute a set of geometrical [23] and additional morphological properties for all edges of the graph. This stage is subdivided into two steps. First, the Voxel-Branch Assignment determines for each foreground voxel which branch it is associated to. The Feature Extraction step uses this mapping to efficiently calculate edge properties in single sweep over the volume. Then, in the Refinement stage, the graph is pruned using the previously calculated scale invariant property *bulge size*, which defines how far a (potential) branch has to protrude from its parent vessel relative to its and the parent vessel's radius in order for it not to be removed. Since pruning invalidates the Voxel-Branch Assignment, the first three stages of the pipeline have to be reevaluated. This Extraction-Refinement cycle is repeated until a fixed point is reached. We want to stress that the novel iterative refinement approach is essential to obtain meaningful results from very large datasets, as demonstrated below.

Below we evaluate and discuss the proposed pipeline in terms of the primary and secondary Requirements for Broad Biomedical Application. Where applicable, we compare the proposed pipeline with a version without iterative refinement, which corresponds to the state-of-the-art method by Drechsler and Laura [23] except for additional support of extraction of cyclical structures. All experiments were conducted on a consumer grade PC (AMD Ryzen 7 2700X (3,7 GHz), 32 GiB RAM, and 1 TB hard disk (Samsung NVMe SSD 960 EVO)). For evaluation we use the binary foreground segmentation of 3D lymphatic vessel images from samples of human calf skin (datasets *Lymphatic 1/2/3*) which were obtained for and used as part of the study in [5], which used a preliminary version of the method proposed here as part of the pipeline. The full datasets were obtained using light sheet microscopy and subsequently downsampled to the dimensions and resolution reported in Table 1 and semiautomatically segmented into foreground and background components using the standard random walker method [38]. Additionally we use artificial binary blood vessel datasets generated using the open-source tool

Vascusynth [39]. Finally, a small case study demonstrates the application of the proposed method on datasets up to 176 GiB in size. For this, the binary mask covering the arterial vessel tree of a mouse liver [40] is used. The original dataset was obtained using light sheet microscopy and the segmentation was performed using a hierarchical variant of the random walker method [18].
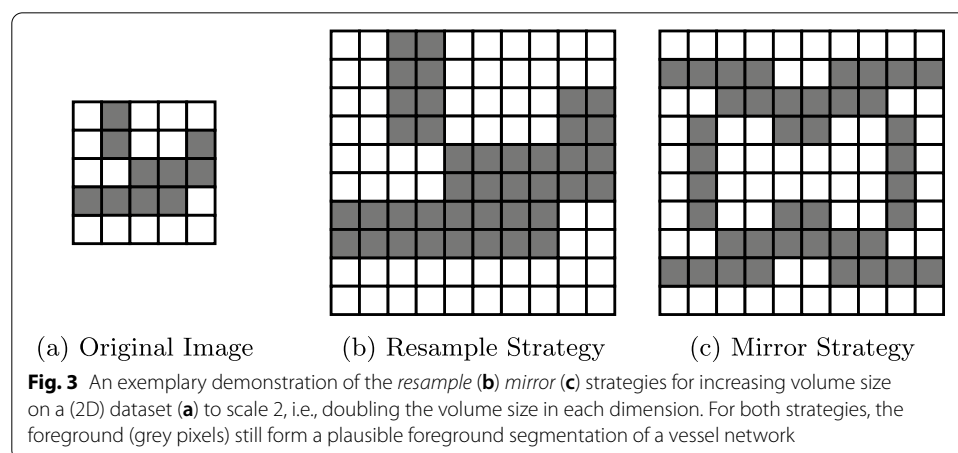
### Runtime scalability

The demonstration of scalability requires some way of modifying a *scale* parameter for a given dataset without changing other characteristics. One possibility would be to use an already large dataset, scaling it down or clipping it to a smaller region. However, besides the problem of availablility of a very large volume (representing the results of *future* microscopes), this could cause artifacts and loss of detail due to downsampling, making the comparison of generated graphs very difficult. Instead we use a *small* (real world) volume and artificially increase its size using two strategies (see Fig. 3).
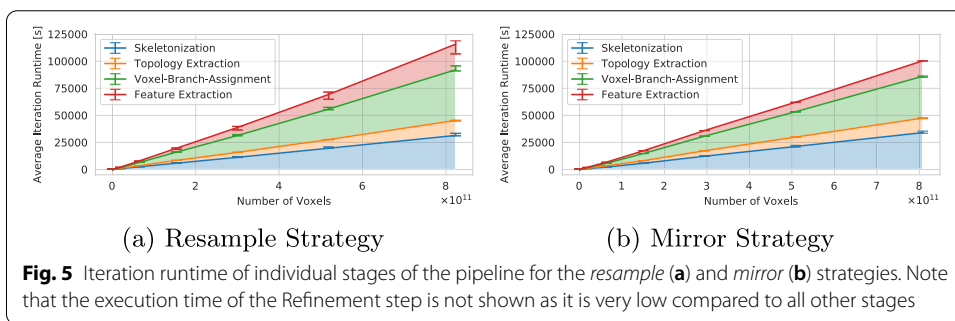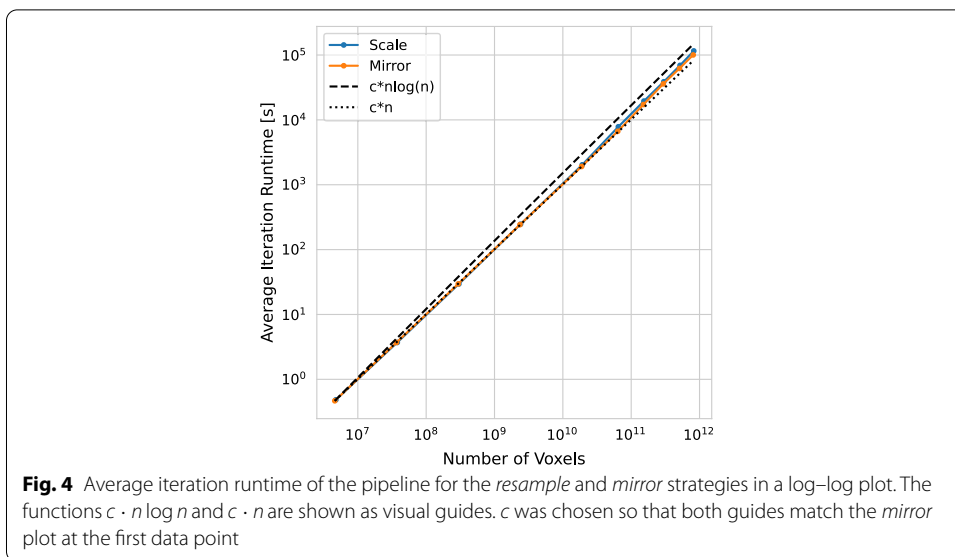
The *resample* strategy: Resampling the volume with larger resolution, using nearest-neighbor sampling to avoid changing topology near very thin connections in the original volume.

The *mirror* strategy: Repeating the volume in each dimension until the desired (integer) scale is achieved. In order to generate a mostly connected network, in each dimension the $2i + 1$'th volumes are mirrored.

In a real world scenario, improved acquisition technology would result in a combination of both increasing (voxel) size of previously visible vessels *and* revealing previously undetected finer networks of larger complexity. When not specified otherwise, the dataset *Lymphatic 1* (Fig. 14(a)) was used for evaluation, as it has a non-tree-like topology, anisotropic voxel resolution and an irregular vessel shape. In the following, *scale* denotes the factor that each dimension of the dataset was multiplied with using one of the above strategies.

Figure 4 shows the average runtime of a single refinement iteration of the pipeline depending on the number of voxels for the *mirror* and *resample* strategies in a log–log plot. As shown, the runtime is only slightly worse than linear, as should be expected in the light of the derived runtime complexity of $\mathcal{O}(n \log n)$. Furthermore, it should be



| (a) Original Image | (b) Resample Strategy | (c) Mirror Strategy |

**Fig. 3** An exemplary demonstration of the *resample* (**b**) *mirror* (**c**) strategies for increasing volume size on a (2D) dataset (**a**) to scale 2, i.e., doubling the volume size in each dimension. For both strategies, the foreground (grey pixels) still form a plausible foreground segmentation of a vessel network

**Fig. 4** Average iteration runtime of the pipeline for the *resample* and *mirror* strategies in a log–log plot. The functions $c \cdot n \log n$ and $c \cdot n$ are shown as visual guides. $c$ was chosen so that both guides match the *mirror* plot at the first data point



(a) Resample Strategy  (b) Mirror Strategy

**Fig. 5** Iteration runtime of individual stages of the pipeline for the *resample* (**a**) and *mirror* (**b**) strategies. Note that the execution time of the Refinement step is not shown as it is very low compared to all other stages
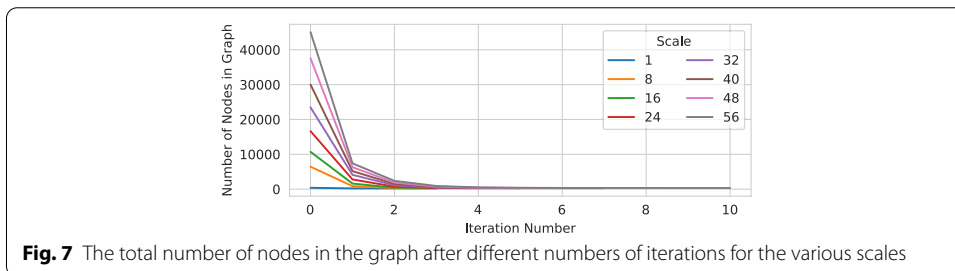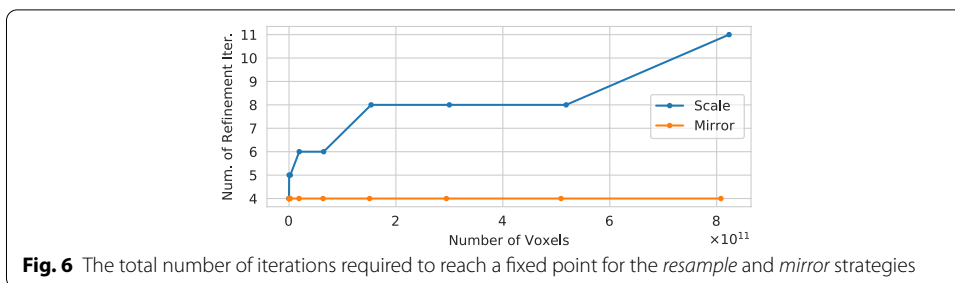
noted that the increase in slope around $10^{11}$ coincides with the exhaustion of main memory and thus a larger number of disk accesses (cf. Fig. 8).

Figure 5 once again shows the average iteration runtime, but using a linear scale and with details of individual steps of the pipeline. It can be observed that the runtime of all steps increases roughly linearly with respect to the number of voxels. Here, it becomes apparent that for the *resample* strategy, an iteration appears to take slightly more time than with the *mirror* strategy. This appears to be mostly due to more time spent in the Voxel-Branch-Assignment and Feature Extraction steps—likely because larger vessels result in more time spent in searching the k–d-trees in both steps.

The total runtime of the pipeline, however, is not only dependent on a single refinement iteration, but also on the number of iterations required to compute the final result. The number of iterations required to reach a fixed point for an increasing number of voxels is depicted in Fig. 6 for both scaling strategies. While the total number of iterations using the *mirror* strategy can expected to be constant for scale 4 and higher (since scale 8 corresponds to 8 copies of the dataset at scale 4), the number of iterations remains at 4 independent of the scale in this experiment. While we do not derive a maximum for the number of iterations for the *resample* strategy, we can observe the required

**Fig. 6** The total number of iterations required to reach a fixed point for the *resample* and *mirror* strategies



**Fig. 7** The total number of nodes in the graph after different numbers of iterations for the various scales

iterations to increase only moderately with the number of voxels in the experiment. While we acknowledge that the (total) processing time of longer than 1 week for a 1 TB dataset is certainly far from interactive use, this does not pose a problem in biomedical research since the time required for the preparation, image acquisition and postprocessing of a sample is of the same magnitude, and the presented pipeline is to the best of our knowledge the *first* to make analysis of these samples possible *at all*. Furthermore, as our pipeline can be considered unbiased (see Influence of the bulge size Parameter), interactive parameter tuning is not required.

Figure 7 shows that the total number of nodes in the graph rapidly decreases even for higher scales. The final number of nodes for all scales is very close to scale 1, but increases slightly for higher iterations. Section Image Resolution Invariance and Fig. 9 demonstrate that the resulting graphs are actually similar. Thus, if constant runtime (for different datasets of a given scale) is required, a relatively low maximum number of iterations can be specified to obtain a good approximation of running until a fixed point is reached. At the same time, Fig. 7 shows that a *single* refinement step is not sufficient to obtain meaningful results for very large datasets, as a large number of spurious branches remain. This underlines the necessity of the presented iterative refinement approach.

**Main memory scalability**

The detailed description of the pipeline in Methods shows that all steps of the pipeline asymptotically allocate $m^{\frac{2}{3}}$ memory or less. For that reason and because total allocated heap memory is difficult to measure efficiently, we demonstrate how the implementation of the presented pipeline behaves with regard to the maximum *resident set size (RSS)*, which specifies the portion of the memory map of a process that is actually held in main memory. This excludes allocated memory that has been swapped out to disk, but includes (sections of) files that have been copied to memory.

**Fig. 8** Maximum resident set size (RSS) of the graph extraction process for different problem sizes (number of voxels in the volume) for the *resample* and *mirror* strategies



(a) Scale 1, it. refinement  (b) Scale 16, no refinement  (c) Scale 16, it. refinement

**Fig. 9** A demonstration of the effect of increased image resolution on the intermediate result and how the refinement iterations mitigate this effect, using dataset Lymphatic 1: An increased image resolution highly increases the number of erroneous branches (in this example the total number of branches is increased roughly 60-fold). After 5 refinement iterations the resulting graph is very similar to the graph extracted from the original volume without artificially increased resolution. Centerlines are rendered as red lines. Nodes in the graph (end or branching points) are rendered as black spheres, which in (b) due to the extremely large number of spurious branches obscure most of the centerlines
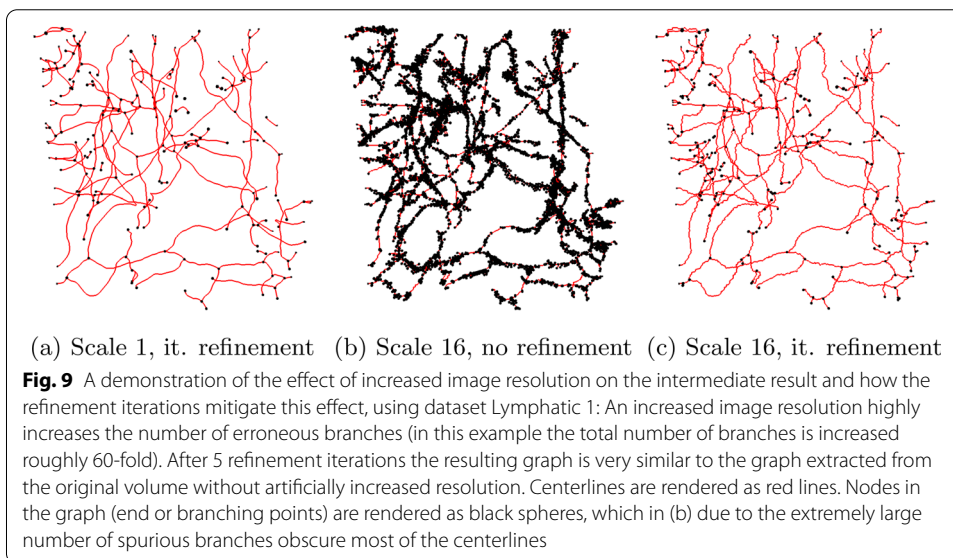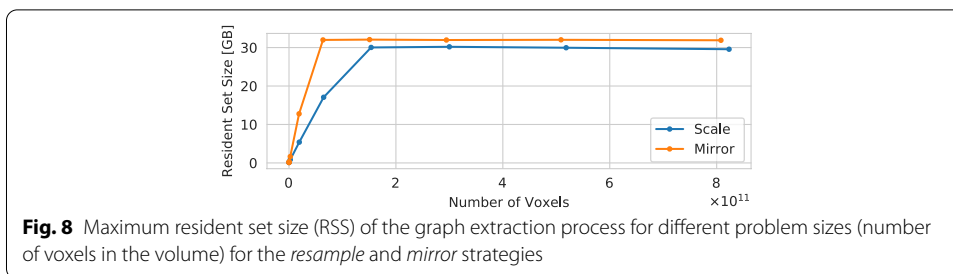
Figure 8 shows the maximum RSS of the process performing the graph extraction for different problem sizes (number of voxels in the volume). It can be observed that the RSS increases for higher problem sizes, but does not exceed a limit near the total available main memory of the test machine (32GiB). This demonstrates the aforementioned property of the pipeline to be scalable in terms of memory, but to use all of the available memory resources. Notably, the *mirror* strategy reaches the 32GiB limit earlier than the *resample* strategy. This may be due to the fact that by repeating the graph $m$-times in *each* dimension, the total number of centerline points in the graph (and thus the required memory to store it) increases roughly by a factor of $m^3$, while for the *resample* strategy, the number of centerline points is (roughly) multiplied by $m$.
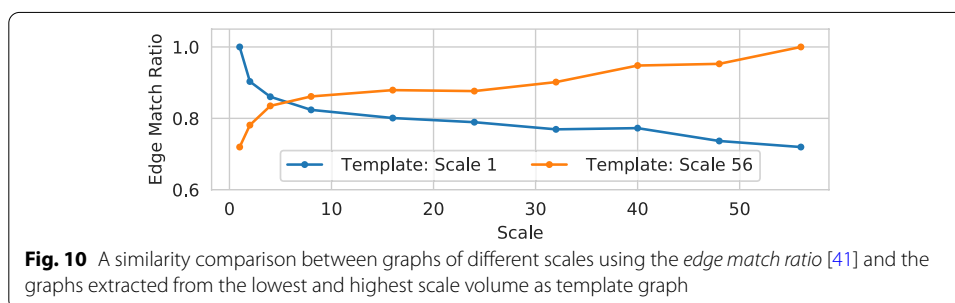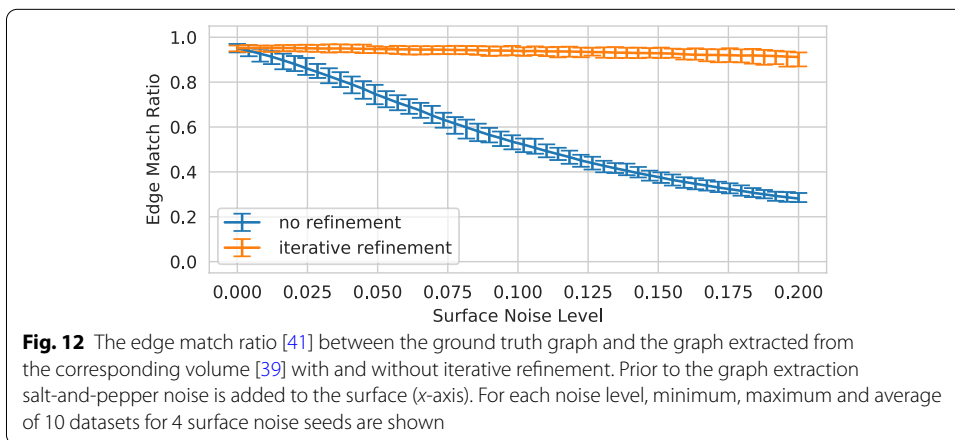
### Image resolution invariance

As a qualitative evaluation, Fig. 9 demonstrates the effect of increased image resolution on current skeletonization/graph extraction approaches and how the iterative refinement approach solves this problem. For a 16-fold resolution increase in each of the coordinate axes (using the resample strategy), without any refinement of the resulting graph, the number of branches and nodes increases sharply (Fig. 9(b)) compared to the the final graph extracted from the original volume (60-fold increase for the depicted example, Fig. 9(a)). After 5 refinement iterations, the number of

branches is reduced (Fig. 9(c)) to a number comparable to the simple case (scale 1, Fig. 9(a)). As the increase in resolution allows for finer grained skeletonization, the graphs still differ in some details, but most of the structure is the same. This is confirmed by biomedical domain experts who consider the refined version appropriate for further analysis, but reject the graph without refinement.

Figure 10 shows how an increase in resolution (using the *resample* strategy) affects the extracted graph with refinement until fixed point. Since no ground truth graph is available for the dataset, we compare all available graphs with the graph extracted using the *lowest* and *highest* scale, respectively. As a quantitative measure we use the edge match ratio [41] which can be understood as a DICE score $\frac{2|E_1 \cap E_2|}{|E_1|+|E_2|}$ where $E_1$ and $E_2$ are the sets of edges in the two graphs and $E_1 \cap E_2$ denotes the set of matches that could be matched when considering node positions and edge properties. When using the graph of scale 1 as a template, the edge match ratio drops relatively sharply for increasing, but smaller scales. For higher scales the difference in similarity is not as pronounced. Direct examination and comparison of the extracted graphs reveals that the difference in edge match ratio is mainly due to small differences in topology: Relatively small branches with a bulge size very close the selected threshold of 1.5 were removed from one graph during the extraction (due to the bulge size being just below 1.5), but not in the other. In this sense, the differences are in fact due to discretization and resulting small differences in properties, which are less pronounced in the higher resolution volumes and where the extracted graphs are actually very similar. In this case the effect is amplified due to the irregular surface of the lymphatic vessel and the relatively low selected bulge size threshold. This also once again underlines the need to process volumes at high resolution to avoid these discretization errors whenever possible.

As an additional way of evaluating the image resolution invariance (P3), we want to focus on one of the primary problem of increased image resolution: The increase in surface noise on the individual voxel level. For this we generate 10 datasets using Vascusynth [39], a tool for generating volumetric images of vascular trees as well as the corresponding ground truth segmentations and tree hierarchy. Before graph extraction we perturb the surface of the binary volumes by iteratively selecting a random foreground or background surface voxel and flipping its value if this does not change the topology of the object. The surface noise level is defined as $\frac{\#\text{voxel value changes}}{\#\text{total number of surface voxels}}$. An example of a volume with applied surface noise is presented in Fig. 11.



**Fig. 10** A similarity comparison between graphs of different scales using the *edge match ratio* [41] and the graphs extracted from the lowest and highest scale volume as template graph

(a) Synthetic 1 (no noise)          (b) Synthetic 1 (noise level 0.2)

**Fig. 11** The foreground of one of the synthetic vessel datasets used for evaluation without (**a**) and with maximal noise (**b**) applied. In (**a**) a very small bump which is categorized as a separate branch according to the ground truth is highlighted



**Fig. 12** The edge match ratio [41] between the ground truth graph and the graph extracted from the corresponding volume [39] with and without iterative refinement. Prior to the graph extraction salt-and-pepper noise is added to the surface (*x*-axis). For each noise level, minimum, maximum and average of 10 datasets for 4 surface noise seeds are shown
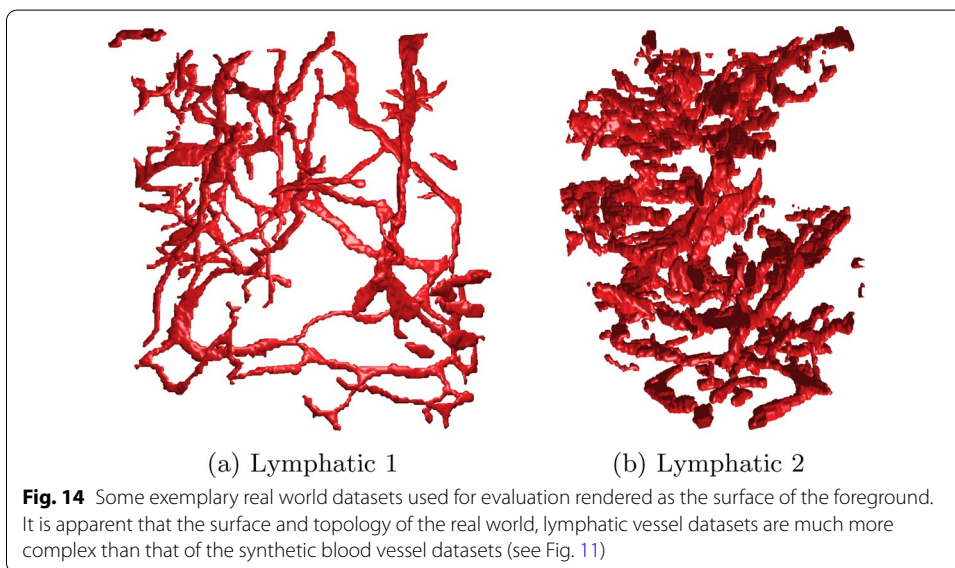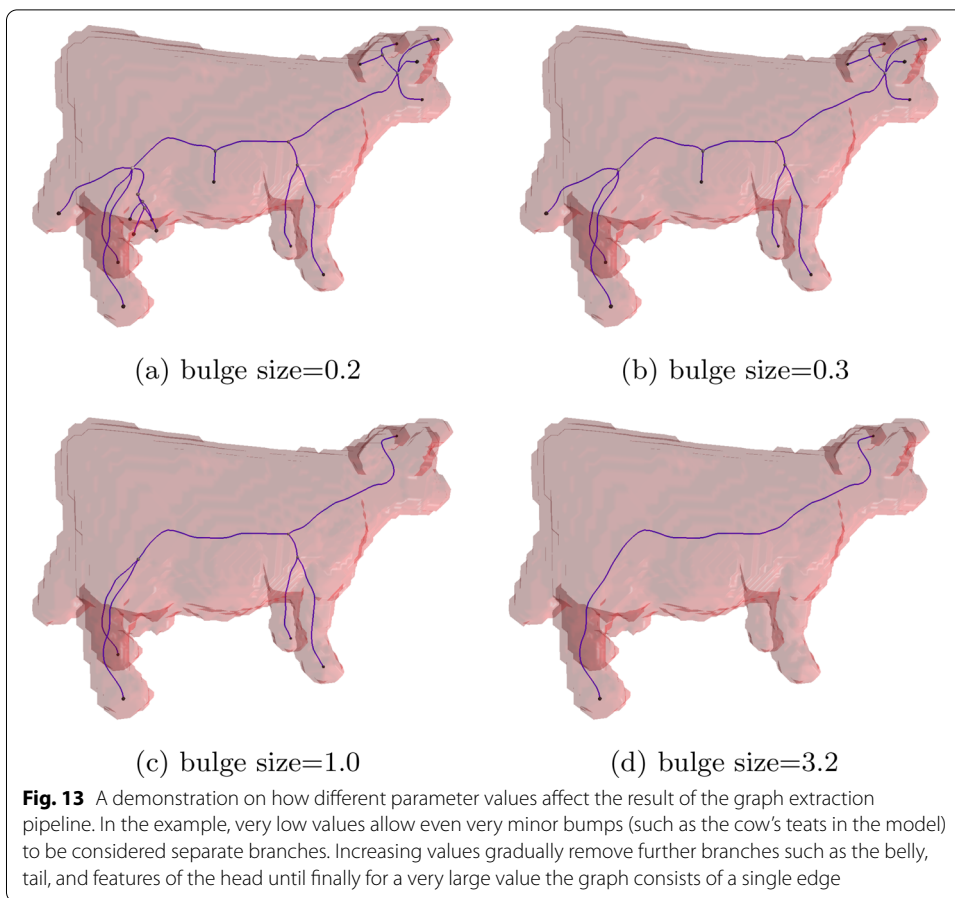
For each of the volumes, four variants with different random noise seeds were evaluated. As shown in Fig. 12, the edge match ratio [41] rapidly decreases when not using iterative refinement while the refinement approach presented in this paper manages to reproduce the expected graph even for high noise levels with only slight decreases in similarity.

Note that we chose a bulge size of 1.5 for the iterative refinement, which is unusually low for blood vasculature. However, Vascusynth generates some branches that are very short compared to the radius parent vessel, and in some cases are entirely enclosed, which are only visible as very shallow bumps (or not visible at all) in the generated volume (see highlighted region in Fig. 14(a) as an example). For real world applications this is likely negligible. Instead, higher robustness is likely preferable, necessitating higher bulge size, as discussed in the following section.

### Influence of the bulge size parameter

As explained in Refinement, the parameter of the proposed method can be selected a-priori based on the application and the desired *outcome* of the pipeline. The *quality* of

(a) bulge size=0.2                                (b) bulge size=0.3

(c) bulge size=1.0                                (d) bulge size=3.2

**Fig. 13** A demonstration on how different parameter values affect the result of the graph extraction pipeline. In the example, very low values allow even very minor bumps (such as the cow's teats in the model) to be considered separate branches. Increasing values gradually remove further branches such as the belly, tail, and features of the head until finally for a very large value the graph consists of a single edge



(a) Lymphatic 1                                (b) Lymphatic 2

**Fig. 14** Some exemplary real world datasets used for evaluation rendered as the surface of the foreground. It is apparent that the surface and topology of the real world, lymphatic vessel datasets are much more complex than that of the synthetic blood vessel datasets (see Fig. 11)

the result does therefore not directly depend on the parameter. Hence, we consider our method *unbiased* (S1). Nevertheless, in Fig. 13 we present an overview of the influence

of different `bulge size` values on the result of the pipeline using a test dataset with low-profile, but variable-depth bulges. As can be seen, a very low bulge size results in separate branches even for very small surface features. Increasing the parameter gradually reduces the complexity of the extracted topology. For real world applications, the bulge size should be set a-priori based on knowledge of the dataset. For example, lymphatic vessels (especially in the pathological case, illustrated in Fig. 14(b), but even in healthy individuals, see Fig. 14(a)) often have comparatively short branches, but also suffer from irregular surface features, often near branching points. We therefore suggest a bulge size of 1.5, where the edge case corresponds to nearly spherical, but slightly elongated bulges. On the other hand, (healthy) blood vasculature often exhibits a smooth surface, round diameter, and clearly defined branches, so that a bulge size of 3.0 or higher can be chosen confidently.
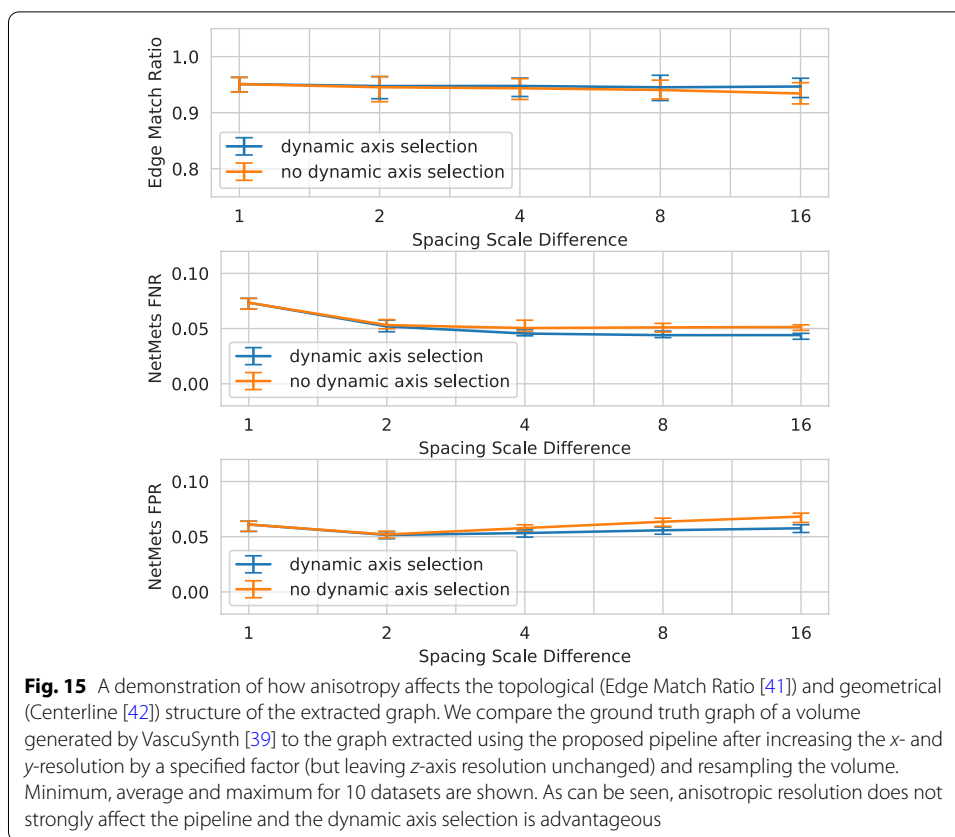
### Anisotropic resolution

The third secondary requirement states that a pipeline should be able to operate on volumes with anisotropic resolution (S3). In order to evaluate our proposed pipeline in terms of that property, we use a volume with known ground truth and isotropic voxel spacing, generated using Vascusynth [39]. We then artificially create datasets with different, anisotropic resolutions by resampling the dataset in specific dimensions. As an example, confocal microscopes due to their anisotropic point spread function generate volumes with a comparatively low z-resolution, which is why in this experiment we chose to leave the z-axis resolution intact and increase the resolution in x- and y-direction (akin to the *resample* strategy). In total, we evaluated 10 datasets with resolution scale differences of 1, 2, 4, 8, and 16. We compared the extracted topology using the graph matching method and edge match ratio similarity measure proposed in [41] and the quality of the extracted centerline geometry using the NetMets framework [42]. In the case of NetMets the average radius of all edges multiplied by 10 was chosen as the parameter $\sigma$. The results are shown in Fig. 15.

Although the ground truth is not matched perfectly and some drift between scales is noticeable (see discussion about ground truth quality above), in general no trend towards positive or negative impact on the evaluated score can be observed based on the level of anisotropy. Furthermore, it should be noted that the dynamic axis selection discussed in Skeletonization indeed improves the result for larger anisotropy, although for smaller levels there is next to no difference between both variants. We therefore conclude that the proposed pipeline is in fact able to process data of even highly anisotropic resolution.

### Robustness against shape deviations

For the evaluation of robustness of the pipeline, especially with regard to deviations from the cylindrical (S2) shape of analyzed vasculature, we use the robustness measure GERoMe introduced in [41]. Additionally, the FNR and FPR values (false negative/positive rate) of NetMets [42] were used to measure the geometric error using in the framework of [41]. The (real world) lymphatic and synthetic blood vessel datasets were rotated and resampled (with doubled resolution in each axis) in 10° steps around the z-axis, processed using the proposed pipeline, compared to the (non-rotated) template

**Fig. 15** A demonstration of how anisotropy affects the topological (Edge Match Ratio [41]) and geometrical (Centerline [42]) structure of the extracted graph. We compare the ground truth graph of a volume generated by VascuSynth [39] to the graph extracted using the proposed pipeline after increasing the *x*- and *y*-resolution by a specified factor (but leaving *z*-axis resolution unchanged) and resampling the volume. Minimum, average and maximum for 10 datasets are shown. As can be seen, anisotropic resolution does not strongly affect the pipeline and the dynamic axis selection is advantageous

graph. The minimum similarity of all steps denotes the robustness index. The similarity is defined as the product of the edge match ratio and one minus the average normalized difference between property values in the set of matched edges, and thus takes topological differences as well as changes in property value into account. For the synthetic datasets the ground truth graph provided by the tool was used as a template and thus are also measuring the accuracy of the method. For real world datasets, generation of an accurate annotated ground truth graph is virtually impossible [41] and thus only the robustness can be evaluated. We compare the results without refinement to the graphs extracted using a bulge size of 1.5 and iteration until reaching a fixed point.

The aggregated results in Table 2 show that for all properties the refinement procedure results in significantly higher robustness for all datasets, indicating an improvement compared to the method without refinement iteration, roughly corresponding to the state-of-the-art method by Drechsler and Laura [23]: In all cases, a higher GERoMe robustness score is achieved. While there is little difference between the versions with and without refinement for in terms of the geometric NetMets measure, for the (more complex) lymphatic datasets an improved robustness in terms of the is achieved as well.
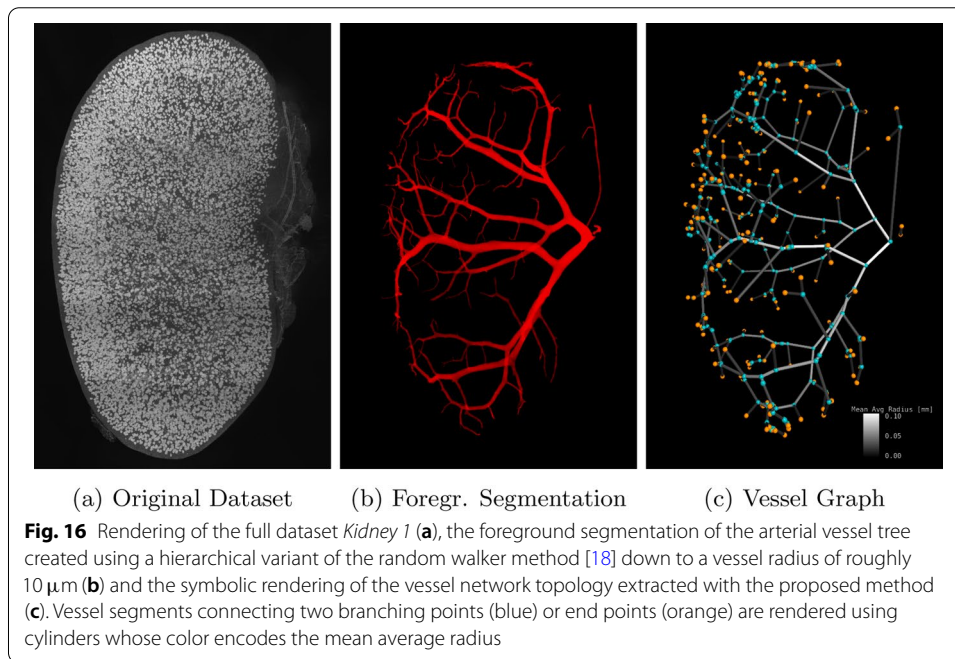
### Case study: large real datasets

Finally, we demonstrate the applicability of the proposed method on large real datasets. For this purpose we use the scan of an antibody-stained mouse kidney sample obtained via light sheet microscopy as part of the study [40]. In dataset *Kidney 1* (176GiB), which

**Table 2** Results of the robustness test [41] using 36 rotations around the z-axis for each respective dataset

| Dataset | Property Refinement | $\mathcal{G}_{length}$ | $\mathcal{G}_{roundMean}$ | $\mathcal{G}_{straight}$ | $\mathcal{N}_{FNR}$ | $\mathcal{N}_{FPR}$ |
|---|---|---|---|---|---|---|
| Lymphatic 1 | Iterative Refinement | **0.781** | **0.780** | **0.823** | **0.0345** | **0.0355** |
| | No Refinement | 0.450 | 0.500 | 0.578 | 0.0515 | 0.0505 |
| Lymphatic 2 | Iterative Refinement | **0.736** | **0.775** | **0.785** | **0.0328** | **0.0341** |
| | No Refinement | 0.486 | 0.536 | 0.614 | 0.0550 | 0.0539 |
| Lymphatic 3 | Iterative Refinement | **0.725** | **0.760** | **0.762** | **0.0431** | **0.0386** |
| | No Refinement | 0.487 | 0.516 | 0.612 | 0.0653 | 0.0632 |
| Synthetic 1 | Iterative Refinement | **0.862** | **0.600** | **0.916** | 0.0445 | 0.0477 |
| | No Refinement | 0.623 | 0.446 | 0.696 | 0.0408 | 0.0488 |
| Synthetic 2 | Iterative Refinement | **0.819** | **0.598** | **0.895** | 0.0425 | 0.0461 |
| | No Refinement | 0.603 | 0.452 | 0.690 | 0.0393 | 0.0479 |

$\mathcal{G}_{property}$ denotes the GERoMe index for the given *property*. $\mathcal{N}_{FNR}$ and $\mathcal{N}_{FPR}$ use the same perturbation procedure, but measure the geometric error of NetMets [42] of the calculated centerlines and therefore show the aggregated maximum value. Refined graphs were extracted using a bulge size of 1.5 and iteration until reaching a fixed point. Relative score differences above 10% are highlighted as bold text



(a) Original Dataset    (b) Foregr. Segmentation    (c) Vessel Graph

**Fig. 16** Rendering of the full dataset *Kidney 1* (**a**), the foreground segmentation of the arterial vessel tree created using a hierarchical variant of the random walker method [18] down to a vessel radius of roughly 10 μm (**b**) and the symbolic rendering of the vessel network topology extracted with the proposed method (**c**). Vessel segments connecting two branching points (blue) or end points (orange) are rendered using cylinders whose color encodes the mean average radius

shows the complete kidney, the arterial vessel tree was segmented semiautomatically using a hierarchical variant of the random walker method [18] down to a vessel diameter of roughly 5 μm. Dataset *Kidney 2* (28 GiB) is a subset of *Kidney 1* where arterial vessels were segmented down to the capillary level. Both segmentation datasets were postprocessed to remove cavities and noise resulting in occasional small loops on the surface. Then the proposed pipeline was applied with a bulge size of 3.

For *Kidney 1* the method finished after 7 iterations and roughly 1 day and 10 h (2059 min) resulting in a graph with 524 nodes and 566 edges. For processing dataset

*Kidney 2* roughly 7 h (413 min) of computation time was required. The resulting graph after reaching a fixed point in iteration 7 has contains 685 nodes and 673 edges.

Figure 16 illustrates how the algorithm captured the topology and properties of vessel segments of the arterial vessel tree in *Kidney 1*: The colors of the cylinders representing vessel segments between two branching or end points fade from white (indicating a high radius) to black (low radius) for segments farther away from the root of the tree. Figure 17 is a closeup of *Kidney 2* that demonstrates the effectiveness of the dimensionless pruning parameter: There are no spurious branches generated, e.g., small bump on the surface of a large vessel, and small branches are preserved as well. Here the bulge size was chosen so that small bulges where the foreground segmentation leaked into the start of otherwise not labeled capillary vessels are not included in the graph.

## Discussion

In Results, we have shown by quantitative evaluation that the proposed pipeline fulfills all primary Requirements for Broad Biomedical Application. In particular, it is scalable in terms of run time and memory (P1, P2), which was proven by applying the pipeline to datasets up to a size of more than 800GB on a consumer-grade computer. It was shown that the novel iterative refinement approach is crucial for image resolution invariance (P3) while the state of the art (graph extraction without refinement [23]) heavily suffers from surface noise. We were further able to show that the proposed iterative refinement also increased the robustness of the method, in particular for datasets with an irregular vessel shape (S2). The applicability of the proposed pipeline to large real datasets were also shown in a small case study.

The single parameter of our method, the bulge size, is independent of dataset scale, and can be determined a-priori depending on the desired properties and shape of the resulting graph. We therefore consider our method unbiased (secondary requirement S1). Further, we have shown that the dynamic axis selection approach in the thinning step increases accuracy for highly anisotropic resolution datasets (S3) compared to the standard approach removing the same number of voxels in each dimension. The fourth secondary requirement (the ability to process volumes of arbitrary topology) directly follows from how the topology is extracted from the binary skeletonized volume (see Topology Extraction) and, in particular, is not restricted to a tree topology, in contrast to



**Fig. 17** Closeup rendering of the foreground segmentation of the arterial vessel network in *Kidney 2* overlayed with the nodes and centerlines of the vessel network. As can be seen, there are no spurious branches, even at larger radius vessels, and small radius vessels are preserved thanks to the dimensionless pruning criterium (bulge size). For the start of even finer, capillary vessels that are not included, but are visible as small humps in the foreground segmentation, no branches in the vessel tree are generated since a relatively large bulge size of 3 was selected

other methods [23]. Furthermore, the pipeline is independent of the imaging conditions (S5) by operating on an existing foreground segmentation.

It should be noted that the proposed pipeline—like all topological thinning-based methods—by definition suffers from distortions in the binary input image that causes changes to the topology, i.e., cavities in foreground objects and loops on the boundary. While these features are not expected in the actual vessel structure, imaging artifacts, noise or problems with the segmentation procedure can still produce such artifacts in practice. However, careful post-processing of the segmentation can mitigate these effects: Cavities in foreground objects can reliably be removed using a modified variant of [43] operating on the *background* without labeling, but removing objects smaller than a specified size. The threshold can typically be chosen very liberally, e.g., as a percentage of the volume diagonal. Removal of loops on the surface is more difficult, but in our experience a (binary) median filter performs well. The filter size should be chosen to not disturb the smallest vessels in the image, but still be able to cover surface loops in the image.

Another problem in practice is that segmentation procedures sometimes only preserve the vessel wall, but not the lumen, i.e., they create (partially) hollow vessels. If this wall segmentation is of high quality and thus does not have holes, it can be converted to a full segmentation using the streaming variant of [43] as described above, filling the (elongated) cavity of the vessel lumen. If this is not the case, more sophisticated preprocessing (e.g., first filling holes in the vessel wall, then the lumen) is required to apply the proposed method in a meaningful way.

## Conclusion

Analyzing ultra large datasets by scalable algorithms is still a huge challenge. We have presented a pipeline for extracting the topology and various features of vessel networks from large three-dimensional images. We were able to show that our method fulfills previously identified Requirements for Broad Biomedical Application. Our main contribution, a novel iterative refinement approach and careful engineering of all pipeline stages, allowed us to demonstrate the scalability and thus applicability to very large datasets, e.g., generated by modern microscopes (primary requirements). At the same time, our pipeline can be applied to a wide range of problem domains due to its robustness, unbiased nature, and lack of assumptions about the topology and morphology of the analyzed vessel systems (secondary requirements).

We will now apply the proposed pipeline and continue previous work [5] using previously unachieveable level of detail, which hopefully leads to new biomedical discoveries.

In the future, we would like to explore how even more specific image features [34] can be integrated into the pipeline without compromising its scalability. Additionally, the current version of the pipeline is entirely single-threaded. While it is possible to more efficiently use the available resources of modern hardware by simultaneously processing multiple datasets in separate processes, it would be desirable to accelerate a single run using multiple processor cores or even GPUs (graphics processing units). However, this is a challenging task: While approaches to parallel skeletonization algorithms exist, they sometimes pose problems with regard to guarantees about the mediality of the centerline and reproducibility in general. It may be worthwhile to try to integrate more robust

thinning algorithms [24, 25] into the pipeline, which may reduce the number of required refinement iterations and thus the overall runtime. Furthermore, at least in the current formulation, the connected-component-analysis [43], which is used in variations in several places in this paper is inherently sequential. Offloading work to the GPU requires even more attention to detail with regard to memory management. Furthermore, we would like to explore automatic segmentation approaches for vessel structures in large volumetric datasets, with special focus on (irregular) lymphatic vessel systems, to facilitate the usability of the presented pipeline even further.

The presented pipeline is part of version 5.1 of Voreen [44] a widely-used [45, 46] open-source volume processing and rendering framework.

## Methods

In the following, the four pipeline stages and the iterative approach are described in more detail, with special attention to the requirements for broad biomedical application. Figure 18 provides an overview of the data flow between individual stages of the pipeline.

### Skeletonization

Like other vessel network analysis pipelines [19, 23], we use a modified version of the established topological thinning algorithm by Lee et al. [20] which has the advantage that any voxel can be evaluated in terms of these properties by only considering its 26-Neighborhood, which is advantageous when operating on very large volumes. Furthermore, we are not aware that other approaches solve the inherent problems of analysis of large datasets. In the original formulation, the algorithm iteratively removes voxels that lie on the surface and whose deletion does not change the topology of the object in 6 (static) subiterations until no more voxels can be removed.

For an efficient implementation, further considerations are necessary. We model the surface of the object explicitly as a sequence of voxel positions, which is initialized by scanning the volume before the first iteration step. In subsequent subiterations, we build the *active surface* (voxels that can potentially be deleted in the next iteration) by retaining voxels from the previous active surface that are not considered during the current subiteration and adding all foreground voxels in the 26-Neighborhood of a voxel after its deletion. If a voxel was considered, but not deleted during the current subiteration, it will be removed from the active surface even although it is still part of the surface of



**Fig. 18** A schematic overview of the data flow in the proposed pipeline. For skeletonization topological information is not available in the first iteration of the pipeline
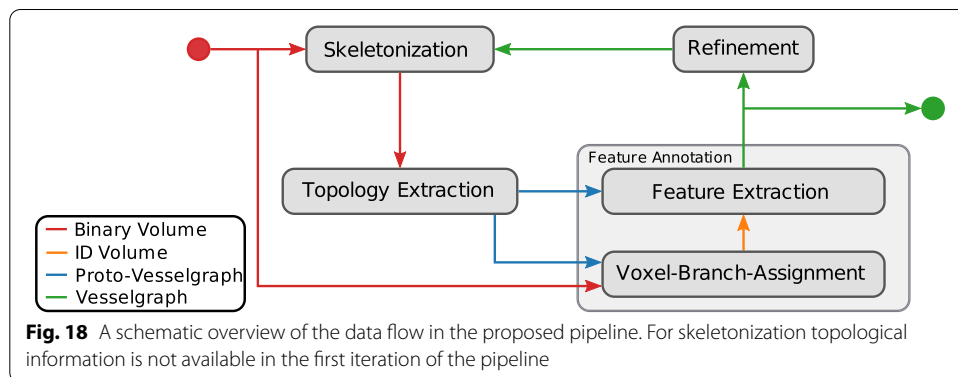
the object. If one of its neighbors is deleted, it will be re-added to the active surface and reconsidered for deletion in the following iteration. This implementation has a runtime of $\mathcal{O}(n)$ in the number of voxels in the volume (P1): In each subiteration, only voxels in the active surface are considered. As a voxel is either deleted entirely or removed from the active surface after each iteration (i.e., 6 subiterations) and only added again once one of its 26 neighbors is deleted, it will be considered for deletion a constant amount of times. In order to fulfill requirement P2, the volume is stored on disk and dynamically mapped to memory using operating system capabilities. Constant runtime improvements could be observed using a compressed (2 bits per voxel) representation and storing the volume as $32 \times 32 \times 32$ (8 KB) blocks in linear memory, thus reducing disk access by exploiting the spatial locality of surface voxels. Active surfaces are stored as on disk (linear) voxel positions in sorted sequences. During a subiteration the previous active surface is read from front to back. Simultaneously, the new active surface is built slice-by-slice in memory by collecting positions, and sorting and removing duplicates before writing to disk, requiring $\mathcal{O}(m^{\frac{2}{3}})$ main memory (P2).

To account for volumes with anisotropic resolution, we keep track of the real-world depth of voxel layers deleted for each direction and select the direction with the lowest total depth as the direction for the next subiteration. This way, the *speed* of voxel deletion is equalized on average for highly anisotropic volumes (S3). If the last subiterations of all 6 directions (which may have happened out of order) did not delete any voxels, the active surface is empty and the skeletonization is completed.

### Topology extraction

Previous work [19, 33] does not describe an efficient implementation of topology extraction for large input datasets. For example, Chen et al. [19] perform a tree search following skeleton voxels through the volume. This is unsuitable for large volumes as it either requires keeping the complete volume in memory or involves very frequent random access to hard disk with high latency. Additionally, their method breaks loops in the graph and always creates a tree topology.

In constrast, the topology extraction in the proposed pipeline extracts the complete centerline graph in a *single* pass over the volume using a modified version of the streaming connected component finding algorithm by Isenburg and Shewchuk [43]. Instead of all foreground components, we consider the three skeleton voxels classes (regular (2 neighbors), end ($< 2$ neighbors), and branch ($> 2$ neighbors) [19]) separately and extract components for each. Individual connected components of end or branch voxels make up nodes in the graph. Individual chains of regular voxels are distinct connected components in the volume and have exactly two end points (except for the separately handled case of closed loops). They form the initial vessel segment centerlines and edges in the graph. Unlike [19], we do not force the position of nodes to be defined by a single voxel and instead use the barycenter of the connected region of voxels. Using the two end points of a segment of regular voxels we can find the nodes that are connected by the path of regular voxels and thus construct an edge. To perform this node-edge-matching process efficiently for large graphs, for each node voxel (i.e., an end or branch voxel) we insert a reference to the original node into a (static, on-disk) k–d tree. Thus, for each run

of regular voxels we can efficiently query the position of node voxels that are 26-neighbors of its tips, and thus link node and edge data structures. Extracted nodes and edges (including centerlines) form the *Proto-Vesselgraph*, are each stored in files on the disk and are dynamically mapped to memory. Furthermore, the algorithm of [43] (and this modification) can be shown to only require $\mathcal{O}(m^{\frac{2}{3}})$ main memory (P2). Moreover, we extract the whole graph in a single pass over the volume in $\mathcal{O}(n \log n)$ time (P1) and do not make any assumptions about the topology of the extracted network (S4).

Since the individual branch voxels necessarily lie on voxel positions of the original volume, they resemble a ragged centerline. This both artificially increases the line length and produces a larger number of ambiguous cases in the Voxel-Branch-Assignment. We therefore smooth all centerlines of the Proto-Vesselgraph using local Bezier curves.

### Voxel-branch assignment

Drechsler and Laura [23] calculate the *volume* property of edges by assigning voxels of the foreground segmentation to the nearest centerline point. As illustrated in Fig. 19(a), this creates incorrect results in some cases. While these errors are potentially not as severe for the calculated *volume*, morphological properties based on the radius of the vessel can be heavily affected. In order to correctly calculate edge-associated properties for the previously created Proto-Vesselgraph structure, we therefore first create a mapping between voxels of the foreground segmentation and the edges of the Proto-Vesselgraph. This process is performed in 4 steps that are described below and illustrated in Fig. 19.

#### *Centerline Voronoi mapping*

As a first approximation of the edge ID map, we find the nearest centerline point for each voxel of the foreground segmentation. We accelerate the search using a static on-disk k–d tree of all centerline points of all edges of the Proto-Vesselgraph, each annotated with the corresponding edge ID. As we iterate over all $n$ foreground voxels and perform a $\log n$ search in the k–d tree for each, the total runtime is within $\mathcal{O}(n \log n)$, as is the construction of the k–d tree (P1). We iterate over the whole volume in $32 \times 32 \times 32$ blocks of voxels for spatial locality of lookups, thus reducing the need to randomly reload parts of the k–d tree from disk.



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

**Fig. 19** If the centerline of a small vessel (blue) is closer to the surface of a larger vessel (yellow) than its own centerline, some voxels of the larger vessel will be assigned to the smaller vessel (**a**). After the initial Voronoi Mapping, connected foreground components of the same ID are remapped using IDs from seed points of branches (**b**). Unlabeled regions are identified (**c**) and flood-filled from labeled voxels (**d**).

### Connected component remapping

As illustrated in Fig. 19(a), matching voxels via minimal euclidean distance does not yield a proper voxel wise vessel segment map in all cases. When two vessels with highly dissimilar radii lie close to each other, some voxels of the larger vessel will be ascribed to the smaller vessel. These cut off regions have to be identified and remapped. We perform a modified connected component analysis [43] on the generated edge ID volume in which we consider two voxels *mergable* if they have the same ID. Then we map the connected component IDs back to the edge IDs using a table constructed by sampling at one centerline point of each edge in the result of the connected component analysis.

### Cut-off region identification

Cut-off regions do not have an associated edge ID and can therefore be identified using another pass of a connected component analysis [43] where only voxels with undefined IDs are considered. In a single pass over the whole volume, we collect the axis aligned bounding boxes for each cut-off region (see Fig. 19(c)).

### Cut-off region flooding

Previously identified cut-off regions are relabeled by flooding all voxels without edge IDs starting from adjacent voxels with valid edge IDs. For the simple case of a region cut-off from a single vessel section, this fills the region with the section's edge ID. In a more complicated case of a cut-off region adjacent to multiple different vessel sections, the resulting labeling approximates the $L1$-Voronoi-cells spanned from the surface of adjacent regions. A side effect of this procedure is that foreground regions that do not resemble (complete) vessels at the boundary of the volume (and thus do not contain any centerline voxels), are *not* flooded with valid IDs in this step and ignored in subsequent steps.

For efficiency, all identified disconnected regions are processed separately as cuboidal subvolumes and collected during a single pass over the ID-volume. Currently valid bounding boxes of cut-off regions are organized in interval trees. For the z-axis, a single tree references all bounding boxes. For each slice, the active regions can be queried to construct an interval tree for the y-axis. For each line in the slice, using the y-axis tree an interval tree for the x-axis can be constructed. For each voxel in the line, all active regions can be queried from the x-axis tree so that the current voxel value can be written to the corresponding subvolumes. After this collection step, each subvolume is flooded separately. Similar to the skeletonization step, we use the concepts of an active surface, slice-wise processing and memory mapping of files on disk to guarantee linear runtime (P1) and $\mathcal{O}(m^{\frac{2}{3}})$ memory requirement (P2).

The results are written back to the ID volume in a process similar to the collection of the subvolumes. The number of disconnected regions is bounded by $n$. Consequently, the construction of the range trees and the querying for each voxel are within $\mathcal{O}(n \log n)$ (P1). All subvolumes and range trees can be stored on disk and mapped dynamically to memory and therefore do not require additional memory (P2).

### Feature extraction

Using the Proto-Vesselgraph and the generated edge ID volume, we can compute the same edge properties as Drechsler and Laura [23]. Some geometric features can be computed directly from the Proto-Vesselgraph. These include *length* (arc-length of the centerline), *distance* (euclidean distance of connected nodes) and *straightness* ($\frac{distance}{length}$). Drechsler and Laura [23] propose *curveness* ($\frac{length}{distance}$), but as the *distance* of any vessel segment is guaranteed to be smaller than its arc-*length*, we argue that straightness (with a guaranteed domain of [0, 1]) is superior to curveness ([1, ∞)).

For other features we collect information from the edge ID volume and the Proto-Vesselgraph. For each foreground voxel in the edge ID volume, we find the corresponding edge and query the closest of its centerline point(s). For efficient lookups, the centerline points are organized in a (static, on-disk) k–d tree for each edge. We equally distribute and store the volume occupied by the foreground voxels alongside the corresponding centerline point(s). Surface voxels are assigned to the closest centerline point, for which thus the minimum, maximum and average surface distance is computed.

Values collected for individual centerline points can be aggregated to compute more per-edge features: For each of the surface distance-based values (*minimum, maximum, average, roundness*:$=\frac{minimum}{maximum}$) we compute the mean and standard deviation, totalling in 8 *additional* morphological features not discussed in [23]. Compared to [34] this provides a relatively coarse understanding of the local vessel morphology, but is a good approximation that can be computed efficiently. Finally, the per-voxel volume can be accumulated to compute the total *volume* occupied by each vessel segment in the vessel graph. The average cross-section of a vessel is obtained by dividing the volume by the length [23].

In total, the number of centerline points is obviously bounded by *n* so that the cost of building and querying the k–d trees is within $\mathcal{O}(n \log n)$. Additionally, the label volume is accessed in $32 \times 32 \times 32$ chunks improving memory locality of searches in the k–d trees (P1). All components of the Proto-Vesselgraph, the k–d trees and the created graph are stored on disk so that main memory requirements are met (P2).

### Refinement

Skeletonization algorithms tend to produce a number of small spurious branches, especially for vessels with irregular surfaces. We therefore propose to refine the generated vessel graph by pruning spurious branches. A simple, but scale-dependent way of defining *deletability* is to set a global minimum length [28]. However, this is problematic if vessels of different scales are present in a single dataset. Instead, we propose *bulge size* as a scale-independent dimensionless measure (S1). Intuitively, the bulge size measures how far a bump, bulge or branch has to extend from a parent vessel in order to be considered a separate vessel. This size is expressed relative to the radius of its parent vessel and itself, making it scale-independent. More formally, the *bulge size* is an edge feature, that is computed during the Feature Extraction, and is only defined for *bulging edges*, i.e., edges that connect a leaf node (degree 1) and a branching point (degree > 2).

For all centerline points of the edge, we decide whether they are within a branching region (*inner points*) or not (*outer points*) during the feature extraction. If a point has

associated surface points, neither of which are adjacent to surface voxels of another edge, it is condidered an outer point. An inner point is characterized by either not having any associated surface voxels or by having a surface voxel that is a 26-neighbor of a point that belongs to another edge. The inner length of a node is the arc length defined by a run of centerline points starting from that node and ending at the last inner voxel. This corresponds to the length of the piece of centerline that lies within a branching area. The *tip radius* is defined as the minimum distance to the surface measured from the centerline point closest to the node. Without loss of generality, let $n_b$ and $n_e$ be the branching point node and leaf node, respectively.

$$bulge\_size(e = (n_b, n_e)) = \frac{length(e) - inner\_length(n_b) + tip\_radius(n_e)}{avgRadiusMean(e)}$$

This definition provides a dimensionless measure of shape that performs well even for corner cases and can be computed efficiently during the feature extraction stage of the pipeline. The calculation and applicability is illustrated in Fig. 20.

The pruning of spurious branches is a node-oriented, iterative approach outlined in Algorithm 1. A pruning step iterates over all nodes in the graph and collects deletable branches. All deletable edges and nodes that were only connected to those edges are then removed from the graph. Additionally, two edges that share a common node of degree 2 are merged by connecting their centerlines and recomputing properties from the attributes of points in the combined centerlines. The pruning step is repeated until a fixed point is reached.

---

**Algorithm 1** Graph refinement algorithm

---
1: **function** REFINE$(G, t)$
2:     **repeat**
3:         $D = \{\}$
4:         **for** node $n \in G$ **do**
5:             $E = $ edges connected to $n$
6:             $P = \{\}$
7:             **for** $e \in E$ **do**
8:                 **if** $bulging(e) \wedge bulge\_size(e) < t$ **then**
9:                     $P = P \cup \{e\}$
10:                **end if**
11:             **end for**
12:             **while** $|E| - |P| < 2$ **do**                    ▷ Retain two edges
13:                 $P = P - \{\underset{e \in P}{\text{argmax}}\, bulge\_size(e)\}$
14:             **end while**
15:             $D = D \cup P$
16:         **end for**
17:         Delete edges in $D$ and remove orphaned nodes
18:         Remove nodes with degree 2 and merge edges
19:     **until** $D = \{\}$                              ▷ Iterate until reaching fixed point
20:     **return** G
21: **end function**

---

While the refined *graph* does not retain any signs of deleted edges, the simultaneously computed centerlines and edge properties are not preserved by the refinement procedure. This happens because the volume regions associated with now deleted branches do not contribute to the properties of the remaining branches until the ID map and features are recomputed. While this affects the bulge size, the pruning scheme will not
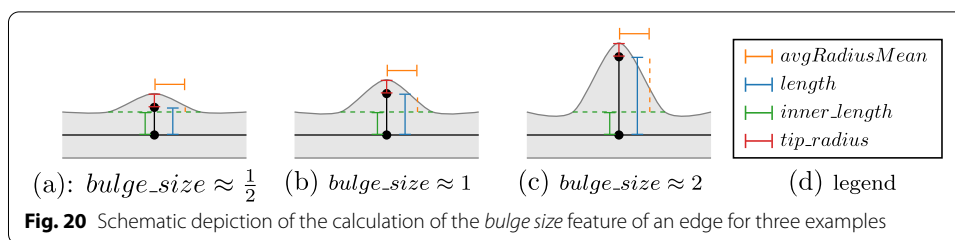
**Fig. 20** Schematic depiction of the calculation of the *bulge size* feature of an edge for three examples

erroneously delete branches, since the missing regions (lowered radii) cause the bulge size to be *overrestimated*.

### Iterative scheme

In order to ensure that centerlines and edge properties match the refined graph, we employ an iterative scheme where the previously extracted and refined graph is fed back into the first stage of the pipeline to improve the generated results.

We modify the skeletonization algorithm to force it to generate a voxel skeleton with a topology that matches the refined graph of the previous iteration. All voxels of nodes with degree 1 in the graph are set to be *fixed* foreground during the skeletonization of the input volume. This means that they are *never* considered for deletion. These voxels mark the beginning/end points of centerlines to be extracted during the skeletonization. The skeletonization algorithm is modified to not preserve voxels at the end of voxel lines (non-*line voxels* [20]). The resulting voxel skeleton connects all previously extracted endvoxels with medially positioned lines.

The iteration can be stopped if it reaches a fixed point. This is the case if two consecutive iterations generate the same graph. As the number of edges never increases, the check can be reduced to a simple integer comparison. Optionally, an upper limit for the number of iterations can be specified. As shown in the evaluation (section Results), the number of deleted edges in each iteration rapidly decreases, so that the intermediate result after *k* iterations in many cases can be considered a very good approximation of the graph produced by fixed point iteration even for a very small *k*.

### Notes on the implementation

The presented pipeline was implemented and all experiments were performed in the volume processing and visualization framework Voreen [44]. The algorithm is available both in the graphical application as well as the command line application (*voreentool*) which is suitable for offline, headless processing.

All volume formats supported by Voreen can be used as input for the graph extraction. Of those, currently volumes in dicom, ome.tiff, hdf5 and nifti format are supported for out-of-core processing.

Currently, the extracted topology can either be saved as a .csv file with one row per edge in the graph, which includes all numerical properties discussed in the *Feature*

*Extraction* section of the paper, or as a compressed json file which includes the full extracted information, including centerines. Release 5.2.1 of Voreen will additionally include an option to export the centerline geometry as a Wavefront OBJ file.

### Abbreviations
PC: Personal computer; RAM: Random access memory; RSS: Resident set size (a measure of memory consumption of a process); FPR: False positive rate; FNR: False negative rate; GPU: Graphics processing unit.

### Authors' contributions
DD developed and implemented the algorithms, performed the experiments and wrote the manuscript. AS gave guidance in the development of the method and experiments, and contributed to the manuscript. RH and FK gave guidance in terms of requirements for practical application and contributed to the manuscript. XJ supervised the project and contributed to the manuscript. All authors have read and approved the manuscript.

### Availability of data and materials
The artificial test data was generated using the tool Vascusynth [39], which is freely available online http://vascusynth.cs.sfu.ca. The datasets *Lymphatic 1/2/3* and *Kidney 1/2* used for evaluation are available upon request. An implementation of the presented pipeline is part of version 5.1 of the volume processing and rendering framework Voreen [44], which is available at https://www.uni-muenster.de/Voreen/. All scripts used in the evaluation are publicly available online to facilitate reproducing the presented results: https://zivgitlab.uni-muenster.de/d_dree02/graph_extraction_evaluation.

## Declarations

### Ethics approval and consent to participate
The datasets Lymphatic 1/2/3 were obtained for and analyzed as part a preceding study. Ethics approval had been obtained as part of this preceding study and is documented in the resulting publication [5]. South West London Research Ethics Committee, London, United Kingdom (REC ref: 05/Q0803/257) and ethics committee of the medical faculty of the University of Münster (ref: 2008-319-f-S). All procedures involving animals (i.e., procedures performed to obtain datasets Kidney 1/2 [40]) were carried out in strict accordance with the German animal protection legislation (Tierschutzgesetz und Tierschutzversuchstierverordnung). The protocol was approved (84-02.04.2016.A218) by the Committee on the Ethics of Animal Experiments of the Landesamt für Natur, Umwelt und Verbraucherschutz (LANUV)"

### Competing interests
The authors declare that they have no competing interests. The work was not performed within the scope of Aaron Scherzinger's business capacity at Ubisoft.

### Author details
[1]Faculty of Mathematics and Computer Science, University of Münster, Münster, Germany. [2]Ubisoft, Mainz, Germany. [3]European Institute for Molecular Imaging, Münster, Germany. [4]Charité – Universitätsmedizin, Berlin, Germany.

### References
1. El-Baz A, Jiang X, Suri JS, editors. Biomedical image segmentation: advances and trends. Boca Raton, FL: CRC Press; 2017.
2. Metzen JH, Kröger T, Schenk A, Zidowitz S, Peitgen H, Jiang X. Matching of anatomical tree structures for registration of medical images. Image Vis Comput. 2009;27(7):923–33.
3. Hägerling R, Pollmann C, Andreas M, Schmidt C, Nurmi H, Adams RH, Alitalo K, Andresen V, Schulte-Merker S, Kiefer F. A novel multistep mechanism for initial lymphangiogenesis in mouse embryos based on ultramicroscopy. EMBO J. 2013;32(5):629–44.
4. Blinder P, Tsai PS, Kaufhold JP, Knutsen PM, Suhl H, Kleinfeld D. The cortical angiome: an interconnected vascular network with noncolumnar patterns of blood flow. Nat Neurosci. 2013;16(7):889.
5. Hägerling R, Drees D, Scherzinger A, Dierkes C, Martin-Almedina S, Butz S, Gordon K, Schäfers M, Hinrichs K, Ostergaard P, Vestweber D, Goerge T, Mansour S, Jiang X, Mortimer PS, Kiefer F. VIPAR, a quantitative approach to 3D histopathology applied to lymphatic malformations. JCI Insight. 2017;2(16):e93424.
6. Nazir A, Cheema MN, Sheng B, Li H, Li P, Yang P, Jung Y, Qin J, Kim J, Feng DD. OFF-eNET: an optimally fused fully end-to-end network for automatic dense volumetric 3D intracranial blood vessels segmentation. IEEE Trans Image Process. 2020;29:7192–202.

7.  Ma Y, Hao H, Xie J, Fu H, Zhang J, Yang J, Wang Z, Liu J, Zheng Y, Zhao Y. ROSE: a retinal OCT-angiography vessel segmentation dataset and new model. IEEE Trans Med Imaging. 2021;40(3):928–39.

8.  Campinho P, Lamperti P, Boselli F, Vermot J. Three-dimensional microscopy and image analysis methodology for mapping and quantification of nuclear positions in tissues with approximate cylindrical geometry. Philos Trans R Soc B. 2018;373:20170332.

9.  Hamilton N. Quantification and its applications in fluorescent microscopy imaging. Traffic. 2009;10(8):951–61.

10. Zhao Y, Xie J, Zhang H, Zheng Y, Zhao Y, Qi H, Zhao Y, Su P, Liu J, Liu Y. Retinal vascular network topology reconstruction and artery/vein classification via dominant set clustering. IEEE Trans Med Imaging. 2020;39(2):341–56.

11. Lesage D, Angelini ED, Bloch I, Funka-Lea G. A review of 3D vessel lumen segmentation techniques: models, features and extraction schemes. Med Image Anal. 2009;13(6):819–45.

12. Moccia S, Momi ED, Hadji SE, Mattos LS. Blood vessel segmentation algorithms: review of methods, datasets and evaluation metrics. Comput Methods Prog Biomed. 2018;158:71–91.

13. Machado S, Mercier V, Chiaruttini N. Limeseg: a coarse-grained lipid membrane simulation for 3D image segmentation. BMC Bioinform. 2019;20(1):2–1212.

14. Li C, Fan Y, Cai X. Pyconvu-net: a lightweight and multiscale network for biomedical image segmentation. BMC Bioinform. 2021;22(1):14.

15. Waibel DJE, Boushehri SS, Marr C. Instantdl: an easy-to-use deep learning pipeline for image segmentation and classification. BMC Bioinform. 2021;22(1):103.

16. Dong R, Yuan G. Giniclust3: a fast and memory-efficient tool for rare cell type identification. BMC Bioinform. 2020;21(1):158.

17. Lösel PD, van de Kamp T, Jayme A, Ershov A, Faragó T, Pichler O, Jerome NT, Aadepu N, Bremer S, Chilingaryan SA, Heethoff M, Kopmann A, Odar J, Schmelzle S, Zuber M, Wittbrodt J, Baumbach T, Heuveline V. Introducing biomedisa as an open-source onlineplatform for biomedical image segmentation. Nat Commun. 2020;11:5577.

18. Drees D, Jiang X. Hierarchical random walker segmentation for large volumetric biomedical data. CoRR arXiv:2103.09564 (2021).

19. Chen Y, Laura CO, Drechsler K. Generation of a graph representation from three-dimensional skeletons of the liver vasculature. In: International conference on biomedical engineering and informatics, 2009; p. 1–5.

20. Lee T, Kashyap RL, Chu C. Building skeleton models via 3-D medial surface/axis thinning algorithms. Graph Model Image Process. 1994;56(6):462–78.

21. Choi W, Lam K, Siu W. Extraction of the Euclidean skeleton based on a connectivity criterion. Pattern Recognit. 2003;36(3):721–9.

22. Ilg M, Ogniewicz R. The application of Voronoi skeletons to perceptual grouping in line images. In: Conference on Speech and Signal Analysis, 1992; p. 382–385.

23. Drechsler K, Laura C O. Hierarchical decomposition of vessel skeletons for graph creation and feature extraction. In: Conference on Bioinformatics and Biomedicine, 2010; p. 456–461.

24. Palágyi K. Parallel 3D 12-subiteration thinning algorithms based on isthmuses. In: Proceedings of advances in visual computing, ISVC, 2013; vol. 8033, p. 87–98.

25. Palágyi K. A sequential 3D curve-thinning algorithm based on isthmuses. In: Proceedings of advances in visual computing, ISVC, 2014; vol. 8888, p. 406–415.

26. Bertrand G, Couprie M. Transformations topologiques discretes. In: Géométrie Discrète et Images Numériques, 2007; p. 187–209.

27. Chen Y, Drechsler K, Zhao W, Laura CO. A thinning-based liver vessel skeletonization method. In: International Conference on Internet Computing and Information Services, 2011; p. 152–155.

28. Chothani P, Mehta V, Stepanyants A. Automated tracing of neurites from light microscopy stacks of images. Neuroinformatics. 2011;9(2–3):263–78.

29. Zhou Y, Toga AW. Efficient skeletonization of volumetric objects. IEEE Trans Vis CG. 1999;5(3):196–209.

30. Almasi S, Xu X, Ben-Zvi A, Lacoste B, Gu C, Miller EL. A novel method for identifying a graph-based representation of 3-D microvascular networks from fluorescence microscopy image stacks. Med Image Anal. 2015;20(1):208–23.

31. Cheng X, Föhst S, Redenbach C, Schladitz K. Detecting branching nodes of multiply connected 3D structures. In: International symposium on mathematical morphology and its applications to signal and image processing, 2019; p. 441–455.

32. Couprie M, Coeurjolly D, Zrour R. Discrete bisector function and euclidean skeleton in 2D and 3D. Image Vis Comput. 2007;25(10):1543–56.

33. Babin D, Pizurica A, Velicki LU, Matic V, Galic I, Leventic H, Zlokolica V, Philips W. Skeletonization method for vessel delineation of arteriovenous malformation. Comput Biol Med. 2018;93:93–105.

34. Rodriguez A, Ehlenberger DB, Hof PR, Wearne SL. Rayburst sampling, an algorithm for automated three-dimensional shape analysis from laser scanning microscopy images. Nat Protoc. 2006;1(4):2152.

35. Bates R, Irving B, Markelc B, Kaeppler J, Muschel RJ, Grau V, Schnabel JA. Extracting 3D vascular structures from microscopy images using convolutional recurrent networks. CoRR arXiv:1705.09597 (2017).

36. Grochowski E. Emerging trends in data storage on magnetic hard disk drives. Datatech. 1998; 11–16.

37. McCallum J. Historical cost of Computer Memory and Storage. https://jcmit.net/mem2015.htm. Accessed: 2021.

38. Grady L. Random walks for image segmentation. IEEE Trans Pattern Anal Mach Intell. 2006;28(11):1768–83.

39. Hamarneh G, Jassi P. Vascusynth: simulating vascular trees for generating volumetric image data with ground truth segmentation and tree analysis. Comput Med Imaging Graph. 2010;34(8):605–16.

40. Kirschnick N, Drees D, Redder E, Erapaneedi R, da Graca AP, Schäfers M, Jiang X, Kiefer F. Rapid methods for the evaluation of fluorescent reporters in tissue clearing and the segmentation of large vascular structures. iScience (2021, accepted).

41. Drees D, Scherzinger A, Jiang X. GERoMe: a method for evaluating stability of graph extraction algorithms without ground truth. IEEE Access. 2019;7:21744–55.

42. Mayerich D, Björnsson C, Taylor J, Roysam B. NetMets: software for quantifying and visualizing errors in biological network segmentation. BMC Bioinform. 2012;13(8):1–19.

43. Isenburg M, Shewchuk J. Streaming connected component computation for trillion voxel images. In: Workshop on Massive Data Algorithmics;2009.

44. Meyer-Spradow J, Ropinski T, Mensmann J, Hinrichs KH. Voreen: a rapid-prototyping environment for ray-casting-based volume visualizations. IEEE Comput Graph Appl. 2009;29(6):6–13.

45. Landell R, Kanan LF, Buzzatti D, Vicharapu B, De A, Clarke T. Material flow during friction hydro-pillar processing. Sci Technol Weld Join. 2019; p.1–7.

46. Benz F, Wichitnaowarat V, Lehmann M, Germano RF, Mihova D, Macas J, Adams RH, Taketo MM, Plate K-H, Guérit S, et al. Low wnt/β-catenin signaling determines leaky vessels in the subfornical organ and affects water homeostasis in mice. eLife. 2019;8:43818.

47. Hu C, Hui H, Wang S, Dong D, Liu X, Yang X, Tian J. Cerebral vessels segmentation for light-sheet microscopy image using convolutional neural networks. In: Medical imaging: biomedical applications in molecular, structural, and functional imaging, 2017; p. 101370.

## Publisher's Note