Artificial vascular image generation using blood vessel texture maps

Adriano dos Reis Carvalho,[1] Matheus Viana da Silva,[1] and Cesar H. Comin[1,*]

[1] *Department of Computer Science, Federal University of São Carlos, São Carlos, SP, Brazil*

Background: Current methods for identifying blood vessels in digital images typically involve training neural networks on pixel-wise annotated data. However, manually outlining whole vessel trees in images tends to be very costly. One approach for reducing the amount of manual annotation is to pre-train networks on artificially generated vessel images. Recent pre-training approaches focus on generating proper artificial geometries for the vessels, while the appearance of the vessels is defined using general statistics of the real samples or generative networks requiring an additional training procedure to be defined. In contrast, we propose a methodology for generating blood vessels with realistic textures extracted directly from manually annotated vessel segments from real samples. The method allows the generation of artificial images having blood vessels with similar geometry and texture to the real samples using only a handful of manually annotated vessels. **Methods**: The first step of the method is the manual annotation of the borders of a small vessel segment, which takes only a few seconds. The annotation is then used for creating a reference image containing the texture of the vessel, called a texture map. A procedure is then defined to allow texture maps to be placed on top of any smooth curve using a piecewise linear transformation. Artificial images are then created by generating a set of vessel geometries using Bézier curves and assigning vessel texture maps to the curves. **Results**: The method is validated on a fluorescence microscopy (CORTEX) and a fundus photography (DRIVE) dataset. We show that manually annotating only 0.03% of the vessels in the CORTEX dataset allows pre-training a network to reach, on average, a Dice score of 0.87±0.02, which is close to the baseline score of 0.92 obtained when all vessels of the training split of the dataset are annotated. For the DRIVE dataset, on average, a Dice score of 0.74±0.02 is obtained by annotating only 0.29% of the vessels, which is also close to the baseline Dice score of 0.81 obtained when all vessels are annotated. **Conclusion**: The proposed method can be used for disentangling the geometry and texture of blood vessels, which allows a significant improvement of network pre-training performance when compared to other pre-training methods commonly used in the literature.

Keywords: Blood vessel model, Texture generation, Artificial image, Segmentation

I. INTRODUCTION

Generating accurate segmentations of blood vessels from digital images is an important task for diagnosis [1–6] as well as for developing precise measurements to support novel studies regarding the role of blood vessels on disease evolution [7–10]. Current state-of-the-art approaches for segmentation involve training neural networks on manually annotated samples [1]. The samples used for training must represent well the distribution of the whole dataset so that the neural network can generalize to unseen samples. An important difficulty with such an approach is that the pixel-wise annotation of blood vessels is very costly.

One common approach to reduce the amount of samples that need to be manually annotated is to pre-train the network on artificially generated blood vessels [11–14]. Given that blood vessels tend to have a tubular structure, a simple methodology for creating artificial vasculature is to randomly generate tubes with varying thickness and to blur the tubes with a smoothing filter [15, 16]. More advanced techniques involve simulating the growth of vascular trees [17] to generate images with accurate distribution of vessel geometries. Common to many approaches that have been developed is the focus on the *geometry* of the vessels, and not on the *appearance* of the vessels. One line of work that does focuses on appearance consists of different approaches for style transfer and image generation [18–20], but these techniques do not allow precise control of the generated vessels regarding properties such as vessel density, caliber, tortuosity, and local intensity changes. Furthermore, they require the development of an additional neural network training procedure for the generative model, which in practice requires hyperparameter tuning and can have poor performance for samples at the tail of the data distribution.

In this work, we develop a methodology for extracting the texture of manually annotated blood vessels from real samples and generating artificial images using the extracted textures. The method only requires the annotation of a few vessel segments of a dataset. Given a dataset containing blood vessel images with different appearances (example images are shown in Figure 1), a common practice would be to annotate all blood vessels in some of the samples to train a segmentation algorithm. The developed method involves the annotation of one or a few vessel segments from some of the images, shown in green in Figure 1. The annotations are used for generating artificial images containing blood vessels with the same texture as the annotated segments, which can then be used for pre-training neural networks. This allows the network to be adjusted to as many artificial images as desired, each image containing a rich set of blood vessel geometries and appearances. Notably, the annotations can be done in seconds, or a few minutes if a more diverse dataset is desired. The neural network may then be refined on a few fully annotated real samples.

* Corresponding author: comin@ufscar.br

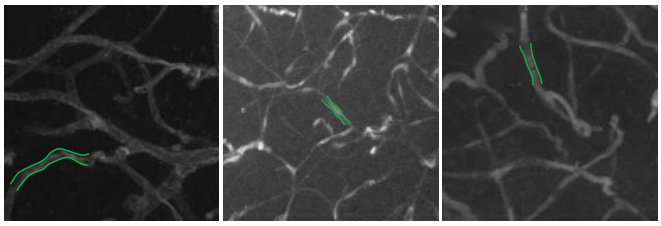

FIG. 1. Three fluorescence microscopy images containing blood vessels with different appearances. Green lines show example annotations of blood vessel segments.

The methodology involves creating standardized representations of the appearance of the vessels by mapping the annotated segments into a rectilinear grid. The result is called a *texture map* since it shares some similarities with texture maps in computer graphics [21]. A procedure is then developed for associating a given texture map to a randomly generated Bézier curve. This is done by defining two corresponding sets of points, one for the texture map and one for the Bézier curve. A Delaunay triangulation is generated for each set of points, and linear transformations are estimated between pairs of corresponding triangles of the two sets of points. This procedure allows the generation of artificial vessels with varying geometries and realistic textures that can be systematically controlled.

We show that artificial images generated using the methodology are effective for pre-training neural networks. On a fluorescence microscopy dataset, images generated from only 5 annotated vessel segments, which corresponds to annotating 0.03% of the vessels in the dataset, lead to an average Dice score of 0.87±0.02, which is close to the baseline value of 0.92 obtained when training using the fully annotated dataset. For a fundus photography dataset, an average Dice score of 0.74±0.02 is achieved by annotating 10 maps (0.29% of the vessels), which is also close to the baseline value of 0.81 obtained when all vessels are annotated. While the manual annotation of the entire datasets can take weeks or months, 10 vessel segments can be annotated in under 5 minutes.

The code of the methodology and the obtained results are publicly available online[1].

II. RELATED WORKS

In [22] the authors proposed a methodology that extended the traditional L-systems grammar [23], which was used for synthesizing angiographic images simulating computed tomography and magnetic resonance images. The generation of synthetic vessels was aimed at assisting in validating segmentation algorithms and to be used in virtual surgeries.

Vessel tree simulators were used to compare artificially created data with real vascular trees [24]. The authors approached the comparison using the texture properties of the trees' geometry (not the vessels' surface). They showed that the models had important correlations with different types of confocal brain tissue images. Vascular tree parameter changes were also observed between healthy and tumorous tissue.

A few recent works [11, 12, 14] used biologically simulated vascular trees for training neural networks using the method of Schneider et al. [17]. The method consists of an iterative generation of arterial trees through two main processes: constructive (growth) and destructive (degeneration). Each vessel segment is modeled as a cylindrical tube and tissue metabolism demands drive the angiogenic process. For instance, in [15] the model was used to simulate magnetic resonance angiography images, which were then used to train different neural network architectures. It was shown that the synthetic data greatly contributed to network pre-training.

The aforementioned works focused mostly on generating plausible geometries for the vessels. Usually, the texture is added using simple procedures such as blurring and Gaussian or Poisson noise. Another line of work considers the generation of vasculature with plausible geometry and appearance using generative adversarial networks [18–20, 25, 26]. But the generated vessels cannot be tightly controlled, for instance, one cannot easily generate samples containing blood vessels with large tortuosity or with low-contrast segments (i.e., discontinuities). In addition, a new training procedure needs to be applied when there is a distribution shift in the data, such as when changing microscope parameters. Our model allows the precise control of the geometry of the vessels as well as the selection of specific texture patterns for the generated images.

III. METHODOLOGY

The method can be divided into five main steps. They are: (a) manual vessel delineation; (b) creation of vessel models; (c) generation of vessel texture maps, (d) transformation of the maps to follow randomly generated curves, and (e) insertion of the transformed maps into an artificially generated background. In the following sections, we describe each of the steps[2].

[1] `https://github.com/AdrianoCarvalho/texture_codes.git`

[2] Regarding the mathematical notation used in the text, vectors and points are represented in bold. Sets of vectors and points are represented by uppercase letters. Matrices are represented by uppercase letters and sets of matrices are represented by letters using a calligraphic font.

A. Manual vessel delineation

The first step consists of manually annotating a few blood vessel segments. Segments are annotated by delineating their borders, as shown in Figure 1. There is no need to annotate the whole segment, only a portion of the segment is enough to obtain its texture pattern. A simple graphical user interface was created for this task, which stores the pixel coordinates of the two delineated borders as a .json file, but the segments can be annotated on any software, provided a file containing the coordinates is generated. The sets of points from the annotated borders are represented as $\tilde{P}_{b1}$ and $\tilde{P}_{b2}$.

Vessels annotated from different images generate a dataset of vessel segments. The method can work with a single annotated segment, but additional segments can increase the diversity of the textures of the vessels on the artificial images. Thus, it is useful to annotate vessels from different images and having distinct appearances. In our experiments, five segments usually led to a good segmentation performance.

B. Generation of vessel models

The manual delineations $\tilde{P}_{b1}$ and $\tilde{P}_{b2}$ are used for obtaining additional information about the vessels such as their medial lines and normal vectors at each border position. The set of all such information for a given vessel segment is henceforth called the *vessel model* of the segment. An example of a vessel model is shown in Figure 2. The procedure used to generate a vessel model is explained next.

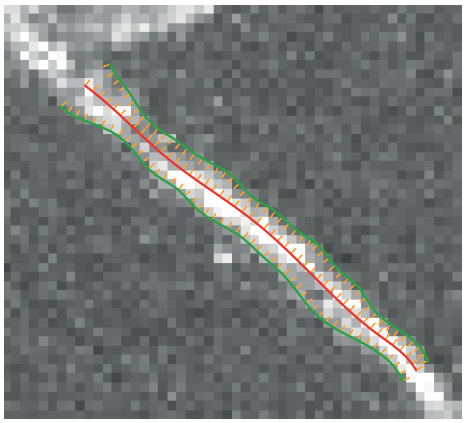

FIG. 2. Illustration of a vessel model. A precise delineation of the vessel border is represented in green, the vessel's medial axis is represented in red, and normal vectors to each curve are represented as orange lines.

Depending on the annotation tool used, the points in $\tilde{P}_{b1}$ and $\tilde{P}_{b2}$ might not be equally spaced, and the annotations might be noisy due to the precision required to trace vessel borders. Thus, for creating the vessel model, the first step is to interpolate the manual delineation of the vessel borders, which is done using a two-stage procedure. A linear interpolation of the points is first performed to uniformly sample the manual annotations. Then, a cubic spline interpolation is performed on the result of the first interpolation. This two-stage interpolation is useful to make sure that the interpolated curves pass close to all manually marked points. In our implementation, the functions *splprep* and *splev* of the SciPy Python package were used.

Both interpolations have a parameter δ setting the distance between interpolated points, which equivalently sets the total number of points used in the interpolated curves. A value of $\delta = 1$ pixel was used in all experiments since smaller values lead to redundant information due to the finite size of the image grid. The result of the interpolation process are two sets of points P_{b1} and P_{b2} representing high-resolution smooth curves describing the borders of the vessel segment.

For each point in P_{b1} and P_{b2}, a normal vector to the curve at the position of the point is calculated using the spline representation of the curve. The two respective sets of normal vectors are represented as N_{b1} and N_{b2}. Figure 2 shows in green the interpolated curves and in orange the normal vectors at each point.

Next, P_{b1} and P_{b2} are used to calculate the medial axis of the blood vessel, which is obtained using a Voronoi tessellation of the border points [27]. In our implementation, the Voronoi tessellation is calculated using the *Voronoi* class from the SciPy Python package. Figure 3 shows an example of Voronoi tessellation of a vessel segment. The tessellation provides the medial axis with sub-pixel accuracy, which makes the following procedures more accurate. However, the points defining the medial axis are not equally spaced. Thus, the same two-stage interpolation applied to the manual delineations is applied to the medial axis using the same value of δ. Respective normal vectors are also similarly calculated. The resulting sets of medial axis points and respective normal vectors are represented, respectively, as P_m and N_m. They are illustrated in Figure 2.

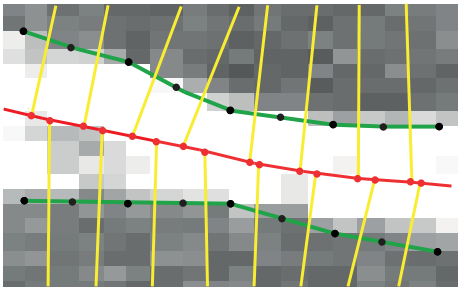

FIG. 3. Example of Voronoi tessellation of vessel borders. The points used for the calculation are shown in black, they were obtained using $\delta = 2$ in the interpolation for a better visualization of the tessellation. Yellow lines represent Voronoi cells. Red lines and points represent the calculated medial axis.

The three generated curves (two borders and one medial axis) represented by P_{b1}, P_{b2} and P_m as well as the normal vectors N_{b1}, N_{b2} and N_m are stored and define the vessel model of the segment.

C. Vessel texture map creation

Vessel models are used for generating vessel texture maps, which are images containing only the texture of the vessels. To do so, a coordinate system following the model geometry is defined. The medial axis defines one coordinate, while perpendicular lines to the medial axis define the second coordinate. The intensities of the vessel are then mapped into a new rectilinear grid. Figure 4 shows an illustration of the procedure. However, some important details need to be taken into account when generating the coordinates.

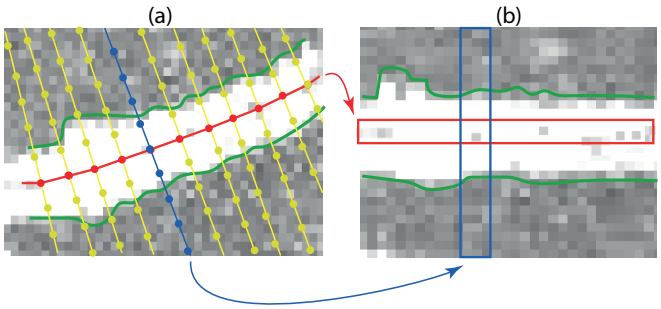

FIG. 4. Creation of a texture map using the coordinate system of the vessel. (a) The medial axis (red) defines one coordinate, and perpendicular lines to the medial axis define a second coordinate (yellow and blue). (b) The intensities of the vessel are mapped to a rectilinear grid. The central row of the resulting image always corresponds to the medial axis. Image columns correspond to the perpendicular lines in (a). In this illustration, a coarse grid of points was used for visual clarity. The distances between the points used in the actual implementation are all equal to one pixel.

First, a set of sampling lines perpendicular to the medial axis are created. Figure 4(a) shows in yellow and blue examples of perpendicular lines. The normal vectors of the medial axis are used for defining a first guess for the direction of the lines. However, the normal of an axis point and the normals of corresponding border points might not be aligned. For instance, on high-curvature segments or when the border at one side significantly changes, the normal vector at the medial axis might lead to very distinct points at the opposing borders of the vessel.

To obtain a precise cross-section of the vessel at a medial axis point $\mathbf{p}_m$, we find the line that best aligns with the normal vector $\mathbf{n}_m$ at the point as well as the normals of the corresponding points at the border of the segment. The procedure used for finding the best-fitting line is described in Algorithm 1 and is illustrated in Figure 5. The algorithm receives as input the medial axis point $\mathbf{p}_m$ with respective normal $\mathbf{n}_m$ and the complete sets of border points P_{b1} and P_{b2} with their respective normals N_{b1} and N_{b2}. A set of angles Θ is defined in the range $[-45°, 45°]$ with steps of $\Delta\theta = 3°$ (line 2 of Algorithm 1).

Algorithm 1 Calculation of the optimal cross-section angle

Input $\mathbf{p}_m, \mathbf{n}_m, P_{b1}, P_{b2}, N_{b1}, N_{b2}$
Output θ_o # Optimal angle

1: $\Delta\theta \leftarrow 3°$
2: $\Theta \leftarrow [-45°, -45° + \Delta\theta, \ldots, 45° - \Delta\theta, 45°]$
3: $A \leftarrow$ empty
4: **For each** θ **in** Θ **do**
5: $\quad \tilde{\mathbf{n}}_m \leftarrow \text{rotate}(\mathbf{n}_m, \theta)$
6: $\quad \mathbf{p}_{b1} \leftarrow \text{get\_closest}(\mathbf{p}_m, \tilde{\mathbf{n}}_m, P_{b1})$
7: $\quad \mathbf{p}_{b2} \leftarrow \text{get\_closest}(\mathbf{p}_m, \tilde{\mathbf{n}}_m, P_{b2})$
8: $\quad \mathbf{n}_{b1} \leftarrow \text{get\_normal}(\mathbf{p}_{b1}, N_{b1})$
9: $\quad \mathbf{n}_{b2} \leftarrow \text{get\_normal}(\mathbf{p}_{b2}, N_{b2})$
10: $\quad e \leftarrow \text{alignment}(\tilde{\mathbf{n}}_m, \mathbf{n}_m, \mathbf{n}_{b1}, \mathbf{n}_{b2})$
11: $\quad$ append e to A
12: **end for**
13: $\theta_o \leftarrow \Theta[\text{argmax}(A)]$
14: **return** θ_o

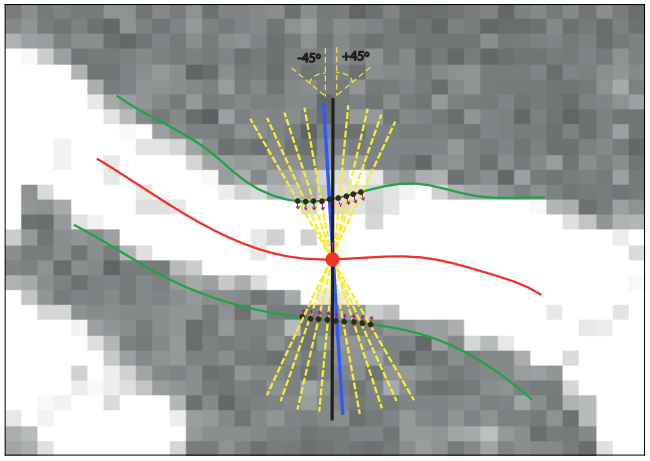

FIG. 5. Example of cross-section angle identification for the medial axis point indicated in red. Candidate directions are shown as dashed lines. The blue line leads to the best alignment between the medial axis normal (black line) and the normals at border points (purple arrows), and thus is chosen as the cross-section direction.

The medial point $\mathbf{p}_m$ and respective vector $\mathbf{n}_m$ define an initial candidate line for a cross-section of the vessel at point $\mathbf{p}_m$ (black line in Figure 5). For each angle θ in Θ, the vector $\mathbf{n}_m$ is rotated by θ (line 5 of Algorithm 1), thus defining a new candidate line for the cross-section of the vessel. Candidate lines are shown as dashed lines in Figure 5. The function *get_closest*($\mathbf{p}_m$, $\tilde{\mathbf{n}}_m$, P_{b1}) (line 6 of Algorithm 1) returns the point $\mathbf{p}_{b1}$ in P_{b1} with the smallest point-to-line distance to the line defined by $\mathbf{p}_m$

and $\tilde{\mathbf{n}}_m$. The same calculation is done for the other border, defining point $\mathbf{p}_{b2}$. The respective normals are also obtained (lines 8 and 9 of Algorithm 1). Points $\mathbf{p}_{b1}$ and $\mathbf{p}_{b2}$ of each candidate line are shown as small brown dots in Figure 5 and respective normals are shown as purple arrows.

The alignment between vectors $\tilde{\mathbf{n}}_m$, $\mathbf{n}_m$, $\mathbf{n}_{b1}$ and $\mathbf{n}_{b2}$ (line 10 of Algorithm 1) is then calculated as

$$e = 2\mathbf{n}_m \cdot \tilde{\mathbf{n}}_m + \mathbf{n}_{b1} \cdot \tilde{\mathbf{n}}_m + \mathbf{n}_{b2} \cdot \tilde{\mathbf{n}}_m, \quad (1)$$

where $\mathbf{x} \cdot \mathbf{y}$ represents the dot product between $\mathbf{x}$ and $\mathbf{y}$. The equation represents the degree of alignment between the candidate line with direction represented by $\tilde{\mathbf{n}}_m$ and the normals of the medial and border points. A weight of 2 is used on $\mathbf{n}_m \cdot \tilde{\mathbf{n}}_m$ to slightly favor the alignment with the medial axis. The angle θ_o with the largest value of e (line 13 of Algorithm 1) defines the direction of the cross-section at point $\mathbf{p}_m$. This direction is illustrated as a blue line in Figure 5.

The identified cross-sectional lines together with the medial axis define a 2D coordinate system based on the vessel's geometry. The medial axis represents a coordinate along the blood vessel, while the perpendicular lines represent a coordinate along cross-sections of the vessel. Sampling points are then created along each perpendicular line. Given a parameter r setting the range of the perpendicular lines, that is, the distance between each endpoint of a line and the medial axis, the sampling points $S = [-r, -r + \delta, \ldots, r - \delta, r]$ are created. The parameter δ sets the distance between points and is the same used when interpolating the border and medial axis points. The sampling points are then oriented according to the perpendicular lines found using Algorithm 1. Examples of sampling points are shown as yellow dots in Figure 4(a).

Each cross-sectional line can be mapped to a column of an output image. The number of sampling points in S defines the number of rows of the image. An illustration is shown in Figure 4(b). Thus, for each pixel of the output image, the corresponding intensity of the original annotated segment can be calculated using bilinear interpolation. In our implementation, this is done using the *map_coordinates* function of the SciPy Python package. The output image defines the vessel texture map M.

The main parameter of the map creation procedure is the range r used for generating the cross-section sampling points. If r is too small, the cross-section lines might not capture the whole vessel. The transformation also needs to capture some portion of the background of the image for the artificial image generation, but if r is too large, other vessels might be included in the texture map, which will lead to artifacts. A good compromise is to use the largest diameter of the vessel segment together with an additional range of around 10 pixels to capture the background.

The sets of border points P_{b1} and P_{b2} are also projected onto the map. They are used for generating a segmentation mask for identifying pixels belonging to the vessel and to the background. Notice that it is not necessary to project the medial axis coordinates since it will always correspond to the middle row of the map.

D. Association of vessel textures to arbitrary curves

The texture maps can be used for generating textures for arbitrary curves. In our methodology we focused on Bézier curves, which are simple to implement and can faithfully represent the curvilinear structure of a vessel. A Bézier curve is defined by the initial and final points, $\mathbf{p}_i$ and $\mathbf{p}_f$, as well as a set P_c of q intermediate control points that control the geometry and the smoothness of the curve. The curve is created using Algorithm 2. q linearly spaced values are created in the range $[0.2, 0.8]$ (line 1 of Algorithm 2). These values are used for defining q respective points along a straight line between $\mathbf{p}_i$ and $\mathbf{p}_f$ (line 5 of Algorithm 2). Each point is then displaced along a normal vector $\mathbf{n}_l$ to the straight line (line 7 of Algorithm 2). The normal vector is a unit vector having a dot product with vector $\mathbf{p}_f - \mathbf{p}_i$ equal to zero. The translation amount is randomly drawn with uniform probability in the range $[-d_m, d_m]$, where d_m is a parameter of the method. Lower values of d_m lead to more straight curves.

Algorithm 2 Bézier curve generation

Input $\mathbf{p}_i, \mathbf{p}_f, q, d_m, n_p$
Output P_r # Points sampled along Bézier curve

1: $ts \leftarrow \text{linspace}(0.2, 0.8, q)$
2: $\mathbf{n}_l \leftarrow \text{normal}(\mathbf{p}_i, \mathbf{p}_f)$
3: $P_c \leftarrow$ empty
4: **For each** t **in** ts **do**
5: $\quad \mathbf{p}_l \leftarrow t(\mathbf{p}_f - \mathbf{p}_i)$
6: $\quad d_c \leftarrow \text{uniform}(-d_m, d_m)$
7: $\quad \mathbf{p}_c \leftarrow \mathbf{p}_i + \mathbf{p}_l + d_c\mathbf{n}_l$
8: $\quad$ append $\mathbf{p}_c$ to P_c
9: **end for**
10: $P_r \leftarrow \text{bezier}(\mathbf{p}_i, \mathbf{p}_f, P_c, n_p)$
11: **return** P_r

A Bézier curve with order $q + 1$ can then be defined using control points $\mathbf{p}_i$, $\mathbf{p}_f$ and P_c. The application of the curve for transforming the maps requires a set of sampling points along the curve. Thus, for any given curve, $n_p = 100$ uniformly spaced points between the initial and final points are created along the curve. The generated points of the curve are represented as P_r.

Given a vessel map M and the points P_r, a procedure is defined to transform the map to follow the geometry of the curve. To do so, we first create two new curves by shifting points P_r to the left-hand and right-hand sides of the original curve. Figure 6 shows an example of the resulting geometry. The curves are created by finding all points with a point-to-line distance of d_b to the original

Bézier curve, where d_b is given by half the number of rows of the vessel map. Then, all points having the endpoints of the Bézier curve as the closest points to them are removed. This defines two curves, which are then linearly interpolated to have the same number of points n_p as P_r. The set of the three generated curves is called the *vessel geometry*. The points defining the three curves, and respective vessel geometry, are all represented by the set P_g.

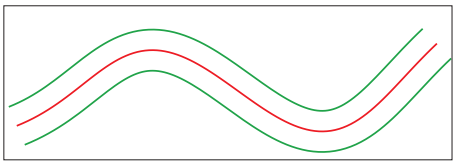

FIG. 6. Example of a generated vessel geometry. The central curve is generated using Algorithm 2. The green curves are translated versions of the central curve.

Having obtained points P_g, a respective set of points P_v needs to be defined for the vessel map with a one-to-one correspondence with points in P_g so that the intensities can be transformed from the map to the curve. Before defining P_v, it is important to observe that the length of the Bézier curve, measured as the path length along the curve, and the length of the vessel map, measured as the number of columns of the map, are usually distinct. Thus, the vessel map is expanded to have the same length as the Bézier curve. This is done by repeating the map column-wise the number of times required for the lengths to be compatible.

Representing as c the number of columns of the vessel map, n_p sampling points are defined at row 0 of the map and at the column positions $[0, \Delta_s, 2\Delta_s, \ldots, (n_p-1)\Delta_s]$, where $\Delta_s = (c-1)/(n_p-1)$. That is, n_p points are created along the first row of the map with equal spacing between them from column positions 0 to $c-1$ (the last column of the map). The same procedure is repeated for the middle and last row of the map. Set P_v is composed of the created points. Figure 7(a) shows in red, yellow, and blue the points in P_v.

Having points P_g and P_v, with a one-to-one correspondence between them, a Delaunay triangulation of each set of points is calculated. This defines two sets of triangles, also with a one-to-one correspondence. For each pair of triangles (one from the map and the other from the curve), an affine transform is estimated and the intensities are mapped using bilinear interpolation. In our implementation, this procedure is applied using the class *PiecewiseAffineTransform* and the *warp* function of the scikit-image Python package. Figure 7 shows an example triangulation of the original map and the Bézier curve together with the transformed image.

The Delaunay triangulation might present some artifacts because points that are on different parts of the border might generate triangles that are outside the curve. These triangles can be seen on the right side of Figure 7(b).

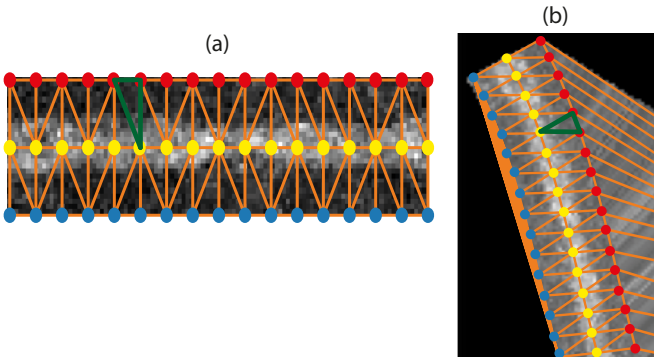

FIG. 7. Example of map transformation. (a) Original map, with points P_v represented by circles. (b) Transformed map, with points P_g also represented by circles. Orange lines represent the Delaunay triangulation of the points. The triangles highlighted in green in both images are a transformed version of each other.

These artifacts were removed by zeroing all pixels outside the vessel geometry.

Besides the vessel texture, it is also necessary to map the location of vessel pixels so that a corresponding binary image can be created and used as a target for segmentation algorithms. This is done by creating a binary vessel image from the map image. This is simple since we have the position of the vessel borders on the map. The binary image is then transformed using the same transformation applied to the grayscale image, but using a nearest neighbor interpolation. Figure 8 shows an example of a transformed map and transformed binary vessel.

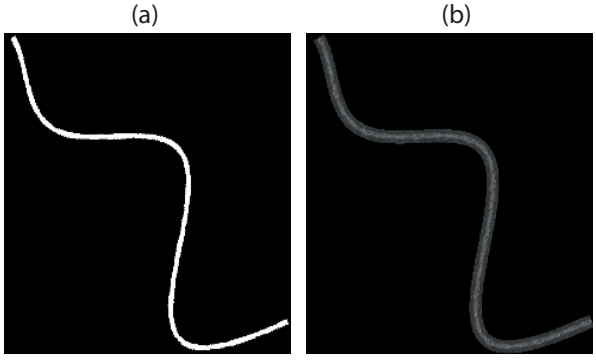

FIG. 8. Example of a transformed map and binary vessel mask. Notice that the transformed map includes the vessel and, by design, a portion of the background of the original image which the map was extracted from.

E. Generation of artificial images using the transformed vessel maps

Having a collection of transformed maps, it is possible to insert them into an image to generate a realistic vasculature tissue image. One possibility is to generate a

vascular tree and place the maps on each vessel segment of the tree. However, blood vessel identification is usually implemented as a semantic segmentation task. This is because digital samples usually contain only a portion of the vasculature of a tissue. In addition, for most biological tissues, the vasculature does not have a standard geometry at the typical scales captured by most imaging modalities. Therefore, we consider that the global geometry of the generated artificial images is not particularly relevant for vessel segmentation methods. Local properties such as the curvature, caliber, border definition, and texture are likely more relevant. This is particularly true for Convolutional Neural Networks, which are known to be more biased towards local features [28, 29]. Thus, we generate artificial images by randomly placing texture maps into an image, without following a global vascular structure.

The only remaining step is to generate an artificial background image B to insert the maps into. This can be done in several ways. For instance, it is possible to generate a random image with values drawn from a normal distribution. One can also annotate a few background regions from a sample and replicate these regions multiple times to generate a full image.

Note that since the transformed maps include some portion of the background of the samples they were extracted from, the artificial background image does not need to be realistic. The main challenge of the segmentation algorithm will be to segment the vessel from the texture map background, and not from the artificial background. Thus, we do not consider the artificial background generation as part of our methodology. In the experiments, we used a simple methodology to create B, consisting of dividing a real sample into small windows and replacing windows containing blood vessels with windows that did not contain blood vessels. It is important to notice that the size of B defines the size of the final artificial blood vessel image.

To generate images without a sharp change in intensity between the artificial background and the map background, before insertion onto the background the maps are normalized as

$$\tilde{M} = M - \mu_m + \mu_b, \tag{2}$$

where μ_m and μ_b are the average intensity of, respectively, the map's background and the artificial background image. A normalization using the standard deviation of the intensities may also be used in case the intensities of the vessel maps have a larger standard deviation than the artificial background.

The procedure used for generating an artificial image is summarized in Algorithm 3. The algorithm receives as input a database of texture maps $\mathcal{M}$, the number of maps m to use, an initial background image B, the number of vessels n_v to insert, the minimum length s_l of the artificial vessels and the q and d_m parameters for the Bézier curve generation. First, a subset $\tilde{\mathcal{M}}$ of m texture maps is randomly drawn from $\mathcal{M}$ (line 1 of Algorithm 3). A value s_m setting the maximum length of the vessel geometries is then calculated (line 3 of Algorithm 3). It is given by the number of rows or columns of the image, whichever is larger.

Algorithm 3 Artificial image generation

Input $\mathcal{M}, m, B, n_v, s_l, q, d_m$
Output I # Generated artificial image

1: $\tilde{\mathcal{M}} \leftarrow \text{draw}(\mathcal{M}, m)$
2: $I \leftarrow \text{copy}(B)$
3: $s_m \leftarrow \max(\text{size}(I))$
4: **for** $i \leftarrow 1, n_v$ **do**
5: $\quad d_e \leftarrow \text{uniform}(s_l, s_m)$
6: $\quad \mathbf{p}_i, \mathbf{p}_f \leftarrow \text{get\_endpoints}(d_e)$
7: $\quad P_g \leftarrow \text{generate\_geometry}(\mathbf{p}_i, \mathbf{p}_f, q, d_m)$
8: $\quad M \leftarrow \text{draw}(\tilde{\mathcal{M}}, 1)$
9: $\quad M_t \leftarrow \text{transform}(M, P_g)$
10: $\quad I \leftarrow \text{insert}(M_t, I)$
11: **end for**
12: **return** I

For each vessel to be inserted into the image, the length d_e of the vessel is drawn with uniform probability in the range $[s_l, s_m]$. The function *get_endpoints* (line 6 of Algorithm 3) randomly generates two points $\mathbf{p}_i$ and $\mathbf{p}_f$ with distance d_e between them. Next, the geometry of the vessel P_g is generated by the function *generate_geometry* using the methodology presented in Section III D. One texture map is then randomly drawn from the set of available maps. The map is transformed into the artificial geometry by the *transform* function using the method presented in Section III D and inserted into the image I.

The result is an artificial image containing the inserted blood vessel textures. Parameters n_v, s_l, q, and d_m can be set according to the density and typical curvatures of the vessels from the real dataset. Parameter m should be as large as possible since the diversity of the textures is increased. However, larger m requires more annotated vessel segments.

IV. RESULTS AND DISCUSSION

In this section, we present the results of the vessel model and texture map creation methods. We also show that the generated artificial images can be used to efficiently train neural networks. The experiments were conducted on an AMD Ryzen 5 5600G 3.90 GHz CPU with 16.0GiB RAM, except for the training and validation of neural networks, which were executed on an Intel I9 12900K 3.20 GHz CPU with an RTX 3090 GPU. Versions 1.10 and 0.21 of, respectively the SciPy and scikit-image Python packages were used.

A. Datasets

Three datasets are used to show the potential of the methodology. The first dataset is composed of 50 confocal microscopy images of the mouse cortex. The images have a size of 1376 × 1104 pixels, with each pixel representing 0.908 × 0.908 μm. Details about the image acquisition procedure can be found in [30]. This dataset is challenging because the blood vessels do not have well-defined borders and some samples have low contrast. Figure 9 contains examples of images from the dataset. This dataset is henceforth called *CORTEX*.

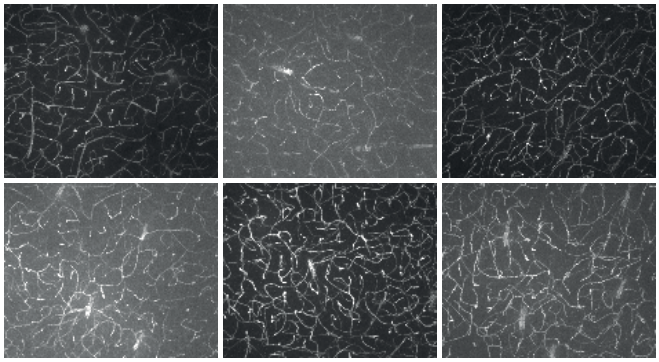

FIG. 9. Examples of images from the CORTEX dataset.

We also use the DRIVE dataset [31], one of the most popular datasets for evaluating blood vessel segmentation algorithms. The dataset contains 20 training and 20 test color fundus photographs, each image having a size of 584 × 565 pixels. Following common practice in the literature, only the green channel of the images was used. The fundus images from the DRIVE dataset are markedly distinct from the microscopy images from the CORTEX dataset. Thick vessels tend to have better contrast with the background when compared to the CORTEX dataset, but the DRIVE images contain a larger number of very thin vessels, which tend to be difficult to segment.

As discussed above, one of the motivations for our method is to improve upon the common pre-training procedure of generating artificial vessels with trivial textures. Thus, we also compare our method with a technique used by recent works to simulate artificial vessels [11, 12, 14], which we refer to as *TREE*. The technique consists of using the method of Schneider et al. [17] to generate realistic artificial vascular trees. A Gaussian blur followed by Gaussian noise is then applied to the images. In [15] a dataset containing 136 artificial samples was generated using the artificial tree method and made available online. We use this dataset in the experiments.

The CORTEX dataset is used in the following sections to provide examples of artificial images generated using the proposed methodology. All three datasets are used in Section IV F to quantify the potential of the method for pre-training neural networks.

B. Vessel models

To measure the typical time spent on manual annotation of vessel segments, six annotators were asked to delineate 3 vessel segments in a given image from the CORTEX dataset, and we recorded the time taken for the annotation of the vessels. The results are shown in Table I. Overall, the annotation of a vessel segment takes an average of 27 seconds, with the time varying according to the characteristics of the vessel. Vessels with larger dimensions or that are very tortuous tend to take more time to be annotated.

Annotation time in seconds				
	Vessel one	Vessel two	Vessel three	Average
Annotator one	11.86	11.22	10.8	11.29
Annotator two	30	34	33	32.33
Annotator three	51.16	28.55	31	36.90
Annotator four	27.69	27.75	27.65	27.70
Annotator five	21.48	22.49	32.28	25.42
Annotator six	24.42	17.98	38.47	26.96

TABLE I. Vessel segment annotation time for six annotators and three vessel segments.

For the following experiments, a total of 343 vessel segments were annotated from the 50 images in the CORTEX dataset by a single annotator. On average, between 5 and 8 vessels were annotated for each image. The minimum and maximum caliber of the vessels were, respectively 4.1 and 25.59 pixels, with an overall average caliber of 8.65 pixels.

A vessel model was created for each annotated segment. Figure 10 shows examples of created models. It is important to have a rich set of vessel models. Thus, we annotated vessels with large variations of intensity, contrast, tortuosity, caliber, and noise amount.

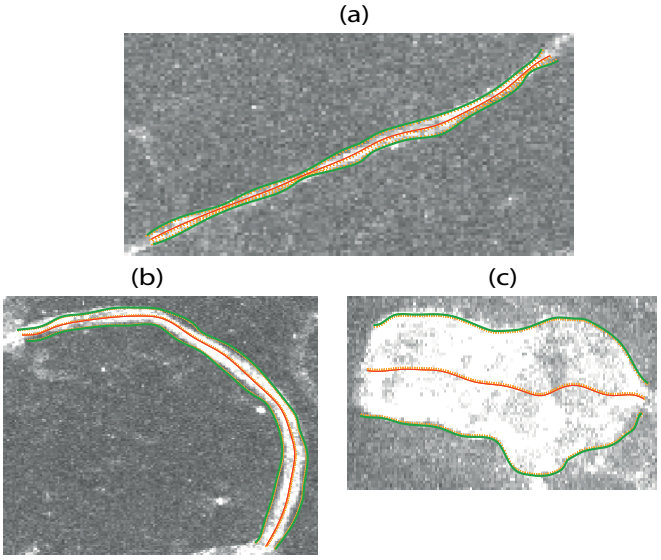

FIG. 10. Vessel models generated from manual annotations.

C. Vessel texture maps

The vessel models were used for generating respective vessel texture maps. The database of 343 vessel texture maps obtained is represented as $\mathcal{M}$. Figure 11 shows examples of vessel maps from $\mathcal{M}$. Each vessel map contains the intensities of the vessel segment and, by design, the background of the sample which the vessel was extracted from. The manual annotation indicating the borders of the vessel is also stored together with the texture map. Again, it is important to have maps containing vessels with varying characteristics since this allows the generation of a diverse set of artificial images.

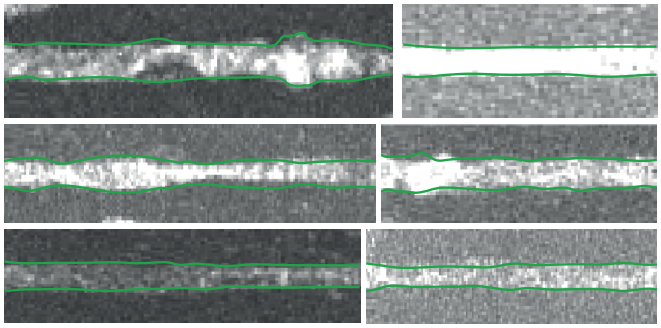

FIG. 11. Examples of vessel texture maps.

D. Vessel geometries

Vessel geometries were generated by randomly drawing the initial and final points $\mathbf{p}_i$ and $\mathbf{p}_f$ of the Bézier curves such that the Euclidean distance d_e between the points was in the range $[500, 1376]$ pixels. Figure 12 shows examples of geometries that can be generated. $q = 30$ control points were used for the curve in Figure 12(a), while $q = 2$ was used for the curve in Figure 12(b). The number of control points has a large impact on tortuosity. Figures 12(c), 12(d), and 12(e) show curves with $q = 6$ and different values of d_m, which controls the maximum distance between control points and a straight line between the initial and final points. Lower d_m values lead to straighter vessels.

For the remainder of the experiments, we used $q = 6$ and $d_m = 500$ for generating the geometries.

E. Artificial image generation

Several artificial images with realistic textures were generated using Algorithm 3. We considered different situations according to the number of annotations m (and respective vessel maps) required for generating the images. We considered situations where only a single annotation is used ($m = 1$), as well as situations where $m = 5$, $m = 10$, and $m = 160$ annotations are used. The annotations used

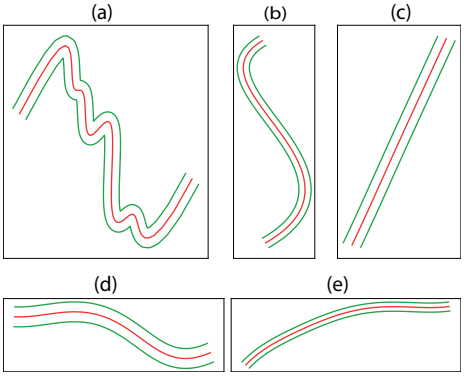

FIG. 12. Examples of generated vessel geometries. Curves (a) and (b) have, respectively $q = 30$ and $q = 2$ control points. Curves (c), (d), and (e) were generated with $q = 6$ and, respectively, maximum control point displacements of $d_m = 1$, $d_m = 100$, and $d_m = 250$.

were randomly drawn from the $\mathcal{M}$ set of 343 annotations. The number of vessels n_v inserted into each image was randomly drawn with uniform probability in the range $[20, 50]$ and the minimum vessel length used was $s_l = 500$.

The results are images containing n_v artificial vessels. Figure 13 shows examples of images generated from a single annotated vessel segment ($m = 1$). In this case, all vessels in the image have the same texture but different geometries. Interestingly, these images can be used for investigating the performance of segmentation algorithms when only the geometry of the blood vessels changes throughout the images.

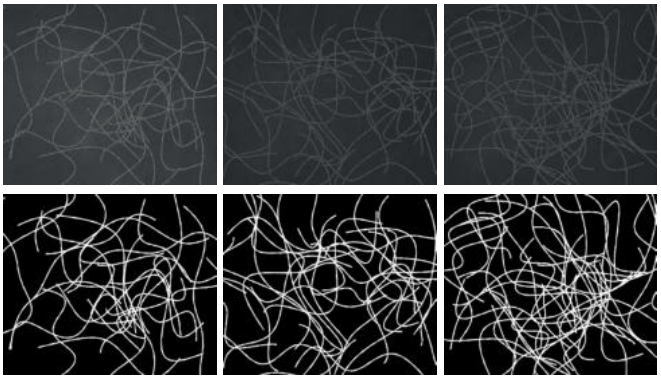

FIG. 13. Examples of images generated from a single annotated vessel segment.

Figure 14 shows images generated from different numbers of annotated vessels. Figures 14(a), 14(b), 14(c) and 14(d) were generated from, respectively, $m = 1$, $m = 5$, $m = 10$ and $m = 160$ vessel annotations. The images show a large diversity of geometries and vessel textures, as well as different degrees of vessel density.

It is possible to use additional data augmentation transforms to further diversify the images. For instance, the images can have the size, brightness, and contrast ran-

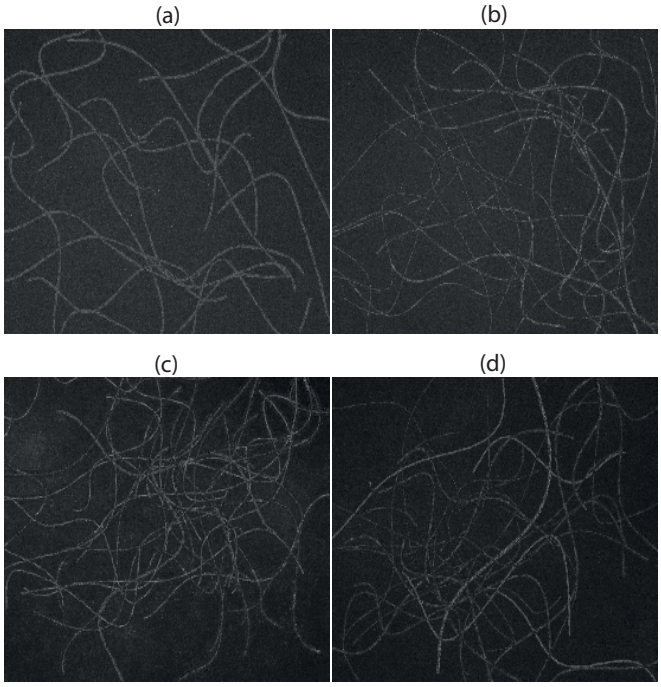

FIG. 14. Examples of artificial images generated from (a) $m = 1$, (b) $m = 5$, (c) $m = 10$, and (d) $m = 160$ vessel texture maps.

domly changed. Gaussian noise can also be added to the images (an example is shown in Figure 15). Since there are many distinct approaches for data augmentation, our method is defined independently of any specific data augmentation pipeline.

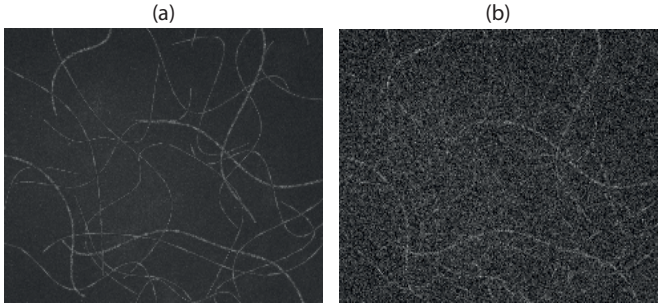

FIG. 15. Example of Gaussian noise insertion in an artificial image. (a) Original image. (b) Image after noise inclusion.

F. Validation using neural networks

The artificial images were used to verify if the proposed methodology can assist in training neural networks. The neural network used is based on the U-net architecture [32] using residual blocks. Figure 16 illustrates the architecture. The network was trained using a learning rate of 0.01 with a polynomial learning rate scheduler with an exponent of 0.9 and a batch size of 8. The performance was quantified using the Dice and centerlineDice (clDice) [33] metrics.

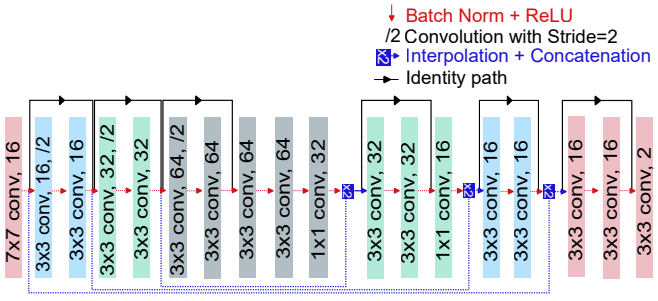

FIG. 16. Neural network architecture used.

The first experiment concerns the CORTEX dataset. The dataset was randomly split into 40 training images and 10 validation images. A set of vessel texture maps, $\mathcal{M}$, was extracted from the 40 training images. A respective set $\mathcal{I}$ of 100 artificial images was generated using a fixed subset of m texture maps from $\mathcal{M}$. To make sure that the results are not due to the specific maps used, n_d different sets were generated. The specific procedure used to generate the artificial datasets is described in Algorithm 4. The results presented in this section are always averaged over the n_d datasets.

We considered cases where $m = 1$, $m = 5$, $m = 10$, and $m = 160$ vessels were annotated. For $m = 1$, $m = 5$ and $m = 10$, $n_d = 10$ datasets were generated. Only one dataset of 100 images was generated for $m = 160$ maps since this set already has a large number of maps and random fluctuations due to the specific maps used are not expected.

Algorithm 4 Data generation for the neural network experiments

Input $\mathcal{M}, m, n_d$
Output null # The data is written to the disk

for $i \leftarrow 1, n_d$ **do**
  $\tilde{\mathcal{M}} \leftarrow \text{draw}(\mathcal{M}, m)$
  $\mathcal{I} \leftarrow$ empty
  for $i \leftarrow 1, 100$ **do**
    $I \leftarrow \text{generate\_image}(\tilde{\mathcal{M}})$
    append I to $\mathcal{I}$
  end for
  write $\mathcal{I}$ to disk
end for

For comparison, a baseline model was trained on the 40 original training images using all available vessels. This model represents the best result that can be obtained with the training parameters considered since all pixels are annotated and used for training. As an additional comparison, the network was also trained on 100 randomly selected images generated by the TREE method,

and the training was repeated 10 times on different sets of randomly selected images to asses the performance variability when using different images. This dataset is used as a baseline comparison with artificial images containing realistic vessel geometries without real textures.

In all experiments the networks were trained for 300 epochs. The performance of all trained models was measured on the validation set consisting of 10 real images. The models with the lowest validation loss found during training were used. The only data augmentation used was the inclusion of Gaussian noise and the darkening of the image according to the distance from the center of the sample to simulate microscope illumination.

The results are shown in Table II. In the table, average and standard deviation values among the 10 sets of 100 images are shown for $m = 1$, $m = 5$, and $m = 10$ annotated maps. No statistics were calculated for the 160 maps and baseline images because they have a single set of images. The results show that the performance difference between 5 and 160 maps is small. Thus, 5 maps seem to be enough to reach the best accuracy for this dataset when using artificial images. Interestingly, while we observe a 0.05 difference in Dice score between the artificial images using 5 maps and the baseline, the clDice difference is only around 0.03. Thus, the centerlines of the vessels are being segmented with an accuracy that is very close to the baseline. Training on the TREE images results in a Dice score of 0.79. Thus, our method achieves a 10.1% Dice score improvement over the usual pre-training strategy using artificial tree generation.

Experiment	Dice	clDice
Baseline	0.92	0.94
TREE	0.79±0.02	0.85±0.04
Using 1 map	0.85±0.03	0.89±0.05
Using 5 maps	0.87±0.02	0.91±0.02
Using 10 maps	0.88±0.01	0.91±0.01
Using 160 maps	0.88	0.91

TABLE II. Average and standard deviation of the Dice and clDice metrics obtained for neural networks trained on artificial vessel images and evaluated on the CORTEX dataset.

Example segmentations of a network trained with artificial images generated from 1 annotation are shown in Figure 17. The images indicate the reason for the difference observed between the Dice and clDice metrics. The network trained on the artificial images tends to generate segmented vessels with the same caliber. Thus, while the centerline of the vessels is correctly identified, the caliber is not always correct. Adding more annotations with high-caliber vessels may mitigate this problem.

To verify if the method is general enough to be applied to different datasets, the experiments done for the CORTEX dataset were repeated on the DRIVE dataset. The artificial images were created using exactly the same procedure used for the CORTEX dataset. In short, a set of vessel texture maps was extracted from the 20 training images of the original train/test split of the DRIVE

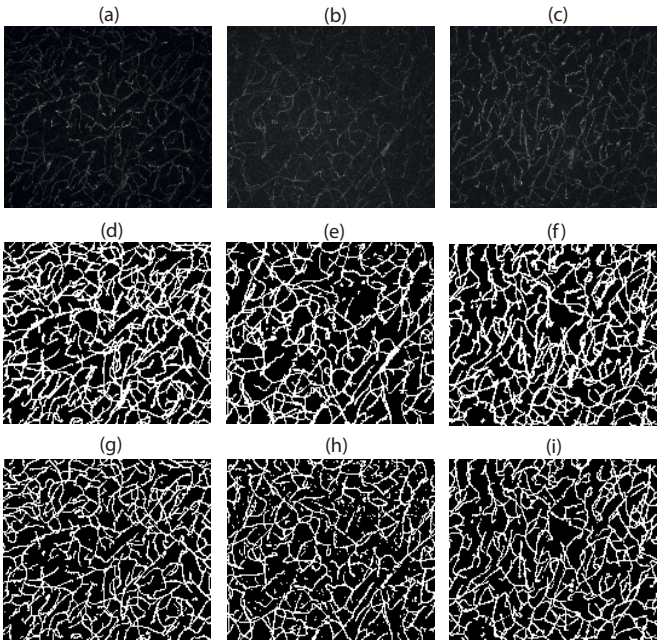

FIG. 17. Example segmentation of a network trained on artificial images. Three images of the validation set (real images) are shown in (a), (b), and (c). The corresponding ground truth annotations are shown in (d), (e), and (f). The results of the network are shown in (g), (h), and (i).

dataset. Algorithm 4 was then applied to create artificial images using $m = 1$, $m = 5$, $m = 10$, and $m = 160$ maps. $n_d = 10$ sets of 100 images were created for the $m = 1$, $m = 5$, and $m = 10$ cases, and one set of 100 images was created for $m = 160$. A baseline model was trained on the 20 training images of the dataset for comparison. An additional model was trained 10 times on different sets of 100 randomly selected images generated by the TREE method. The performance of all trained models was evaluated on the original test split of the dataset.

There were two changes in network training compared to the CORTEX dataset. The networks were trained for 1000 epochs, which was necessary for convergence. A lightweight data augmentation consisting of random flipping, random rotation, and random resized crops was necessary to avoid overfitting[3].

The results of the experiments are shown in Table III. For the DRIVE dataset, 10 maps resulted in a Dice score of 0.74, which is close to the baseline score of 0.81. The TREE images led to poor results since the method used to generate the images was originally developed for microscopy images. In contrast, our method can naturally adapt to different imaging modalities.

The amount of manual annotation required by our methodology was measured for the CORTEX and DRIVE

[3] The transformation RandomAffine(angle=45, scale=(0.95, 1.20)) from the *torchvision* Python package was used for the rotation and random resized crop.

Experiment	Dice	clDice
Baseline	0.81	0.80
TREE	0.21±0.01	0.23±0.06
Using 1 map	0.68±0.05	0.67±0.06
Using 5 maps	0.68±0.05	0.68±0.06
Using 10 maps	0.74±0.02	0.73±0.02
Using 160 maps	0.75	0.75

TABLE III. Average and standard deviation of the Dice and clDice metrics obtained for neural networks trained on artificial vessel images and evaluated on the DRIVE dataset.

datasets. For each map used in the experiments, the length of the vessel represented by the map was measured. This is given by the length of the curve represented by the set of medial axis points P_m used to create the map. For the CORTEX dataset, 5 maps required, on average, annotating a total vessel length of 513 pixels. For comparison, the vasculature of the 40 training images has a total length of 1555520 pixels. Thus, the Dice score of 0.87 is reached using 0.03% of the annotation effort used on the full dataset.

For the DRIVE dataset, the creation of 10 maps required on average annotating a total vessel length of 510 pixels. The number is similar to annotating 5 maps on the CORTEX dataset because the annotations on the DRIVE dataset for each map were usually shorter since vessel segments on the DRIVE dataset tend to have shorter lengths. The 20 training images have a total vessel length of 174080 pixels. Thus, the Dice score of 0.74 is reached using 0.29% of the vessels annotated on the full dataset. Note that the method is used only for pre-training the network. After pre-training, the network can be refined on the real samples.

V. CONCLUSION

We presented a methodology for generating artificial blood vessel images with realistic textures. The method is built upon the fact that vessel segments can be used as a reference for a coordinate system, which allows the definition of texture maps. The method is fast, being able to generate hundreds of images from a single vessel segment annotation, and can significantly decrease the manual annotation effort when segmenting novel datasets.

We showed that by annotating only 5 vessel segments of the CORTEX dataset, corresponding to only 0.03% of the vessels, the generated artificial images can be used for training a neural network with a Dice score of 0.87±0.02 and a clDice score of 0.91±0.02, which are very close to the baseline values of, respectively, 0.92 and 0.94 obtained when annotating the whole dataset. The annotation can be done in under 2 minutes by a trained annotator. For the DRIVE dataset, annotating only around 0.29% of the vessels, or 10 segments, led to a Dice score of 0.74±0.02 and a clDice score of 0.73±0.02, which are also close to the baseline values of, respectively, 0.81 and 0.8.

The results were compared with those obtained by a common procedure used for pre-training neural networks in recent works [11, 12, 14, 15], which we called the TREE method. For the CORTEX dataset, the TREE method resulted in a Dice score of 0.79±0.02 and a clDice score of 0.85±0.04, which are lower than the values obtained by our method. For the DRIVE dataset, a Dice score of 0.21±0.01 and a clDice score of 0.23±0.06 were obtained using the TREE method. The reason for the weak performance is that the method was developed with a focus on microscopy images. The advantage of our method is that it can adapt to different imaging modalities since the appearance of the vessels is extracted directly from small segments of the real images. Thus, the method worked well for both fluorescence microscopy and fundus photography images.

The developed methodology also allows to easily add new annotations to the dataset. For instance, one can annotate a specific blood vessel that was not segmented correctly by the network and generate a new set of artificial images containing blood vessels with the same appearance but different geometries. The network can then be refined on the images.

The analysis was done with lightweight pre-processing and augmentation procedures to avoid biasing the results. That is, it is important to verify that the performance of the method does not come from a complex augmentation pipeline, but from the representation capability of the textures captured from the real images. For future analyses, more elaborate pre-processing steps, such as histogram equalization [34, 35], and augmentation procedures may be used to improve the results of the method.

The generation of vessel texture maps also opens new possibilities for disentangling the influence of shape and texture on neural network training. It allows identifying the importance of each aspect and to apply data augmentation and style transfer techniques separately for them. This is a promising research direction with some recent interesting results regarding the texture bias of CNNs [29].

FUNDING

C. H. Comin thanks FAPESP (grant no. 21/12354-8) for financial support. M. V. da Silva thanks FAPESP (grant no. 23/03975-4) and the Google PhD Fellowship Program for financial support.

CRediT AUTHOR STATEMENT

Adriano dos Reis Carvalho: Methodology, Software, Validation, Investigation, Visualization, Data Curation, Writing - Original Draft. **Matheus Viana da Silva**: Software, Methodology, Writing – review and editing. **Cesar H. Comin**: Conceptualization, Investigation,

Methodology, Software, Supervision, Project Administration, Writing - Original Draft, Writing – review and editing.

[1] Muthu Rama Krishnan Mookiah, Stephen Hogg, Tom J MacGillivray, Vijayaraghavan Prathiba, Rajendra Pradeepa, Viswanathan Mohan, Ranjit Mohan Anjana, Alexander S Doney, Colin NA Palmer, and Emanuele Trucco, "A review of machine learning methods for retinal blood vessel segmentation and artery/vein classification," Medical Image Analysis **68**, 101905 (2021).

[2] Nabila Eladawi, Mohammed Elmogy, Fahmi Khalifa, Mohammed Ghazal, Nicola Ghazi, Ahmed Aboelfetouh, Alaa Riad, Harpal Sandhu, Shlomit Schaal, and Ayman El-Baz, "Early diabetic retinopathy diagnosis based on local retinal blood vessel analysis in optical coherence tomography angiography (octa) images," Medical physics **45**, 4582–4599 (2018).

[3] SN Sangeethaa and P Uma Maheswari, "An intelligent model for blood vessel segmentation in diagnosing DR using cnn," Journal of medical systems **42**, 175 (2018).

[4] Zhenwei Li, Mengli Jia, Xiaoli Yang, and Mengying Xu, "Blood vessel segmentation of retinal image based on dense-u-net network," Micromachines **12**, 1478 (2021).

[5] Jasem Almotiri, Khaled Elleithy, and Abdelrahman Elleithy, "A multi-anatomical retinal structure segmentation system for automatic eye screening using morphological adaptive fuzzy thresholding," IEEE Journal of Translational Engineering in Health and Medicine **6**, 1–23 (2018).

[6] Arun T Nair and K Muthuvel, "Blood vessel segmentation and diabetic retinopathy recognition: an intelligent approach," Computer Methods in Biomechanics and Biomedical Engineering: Imaging & Visualization **8**, 169–181 (2020).

[7] Niccolò Roda, Giada Blandano, and Pier Giuseppe Pelicci, "Blood vessels and peripheral nerves as key players in cancer progression and therapy resistance," Cancers **13**, 4471 (2021).

[8] Julie Ouellette, Xavier Toussay, Cesar H Comin, Luciano da F Costa, Mirabelle Ho, María Lacalle-Aurioles, Moises Freitas-Andrade, Qing Yan Liu, Sonia Leclerc, Youlian Pan, *et al.*, "Vascular contributions to 16p11. 2 deletion autism syndrome modeled in mice," Nature Neuroscience **23**, 1090–1101 (2020).

[9] Sau May Wong, Jacobus FA Jansen, C Eleana Zhang, Erik I Hoff, Julie Staals, Robert J van Oostenbrugge, and Walter H Backes, "Blood-brain barrier impairment and hypoperfusion are linked in cerebral small vessel disease," Neurology **92**, e1669–e1677 (2019).

[10] Parviz Dolati, Alexandra Golby, Daniel Eichberg, Mohamad Abolfotoh, Ian F Dunn, Srinivasan Mukundan, Mohamed M Hulou, and Ossama Al-Mefty, "Pre-operative image-based segmentation of the cranial nerves and blood vessels in microvascular decompression: can we prevent unnecessary explorations?" Clinical neurology and neurosurgery **139**, 159–165 (2015).

[11] Lei Mou, Yitian Zhao, Huazhu Fu, Yonghuai Liu, Jun Cheng, Yalin Zheng, Pan Su, Jianlong Yang, Li Chen, Alejandro F Frangi, *et al.*, "Cs2-net: Deep learning segmentation of curvilinear structures in medical imaging," Medical image analysis **67**, 101874 (2021).

[12] Mihail Ivilinov Todorov, Johannes Christian Paetzold, Oliver Schoppe, Giles Tetteh, Suprosanna Shit, Velizar Efremov, Katalin Todorov-Völgyi, Marco Düring, Martin Dichgans, Marie Piraud, *et al.*, "Machine learning analysis of whole mouse brain vasculature," Nature methods **17**, 442–449 (2020).

[13] Cheng Chen, Kangneng Zhou, Zhiliang Wang, Qian Zhang, and Ruoxiu Xiao, "All answers are in the images: A review of deep learning for cerebrovascular segmentation," Computerized Medical Imaging and Graphics , 102229 (2023).

[14] Navodini Wijethilake, Mithunjha Anandakumar, Cheng Zheng, Peter TC So, Murat Yildirim, and Dushan N Wadduwage, "Deep-squared: deep learning powered descattering with excitation patterning," Light: Science & Applications **12**, 228 (2023).

[15] Giles Tetteh, Velizar Efremov, Nils D Forkert, Matthias Schneider, Jan Kirschke, Bruno Weber, Claus Zimmer, Marie Piraud, and Björn H Menze, "Deepvesselnet: Vessel segmentation, centerline prediction, and bifurcation detection in 3-d angiographic volumes," Frontiers in Neuroscience **14**, 592352 (2020).

[16] Olubunmi Omobola Sule, Serestina Viriri, and Abdultaofeek Abayomi, "Effects of image enhancement techniques on cnns based algorithms for segmentation of blood vessels: A review," in *2020 International Conference on Artificial Intelligence, Big Data, Computing and Data Communication Systems (icABCD)* (IEEE, 2020) pp. 1–6.

[17] Matthias Schneider, Johannes Reichold, Bruno Weber, Gábor Székely, and Sven Hirsch, "Tissue metabolism driven arterial tree generation," Medical image analysis **16**, 1397–1414 (2012).

[18] Oleksandra Tmenova, Rémi Martin, and Luc Duong, "Cyclegan for style transfer in x-ray angiography," International journal of computer assisted radiology and surgery **14**, 1785–1794 (2019).

[19] Chunwei Ma, Zhanghexuan Ji, and Mingchen Gao, "Neural style transfer improves 3d cardiovascular mr image segmentation on inconsistent data," in *Medical Image Computing and Computer Assisted Intervention–MICCAI 2019: 22nd International Conference, Shenzhen, China, October 13–17, 2019, Proceedings, Part II 22* (Springer, 2019) pp. 128–136.

[20] Chulin Wu, Heye Zhang, Jiaqi Chen, Zhifan Gao, Pengfei Zhang, Khan Muhammad, and Javier Del Ser, "Vessel-gan: Angiographic reconstructions from myocardial ct perfusion with explainable generative adversarial networks," Future Generation Computer Systems **130**, 128–139 (2022).

[21] John F Hughes, Andries van Dam, Morgan McGuire, David F Sklar, James D Foley, Steven K Feiner, and Kurt Akeley, "Computer graphics: Principles and practice," (2013).

[22] Miguel A Galarreta-Valverde, Maysa MG Macedo, Choukri Mekkaoui, and Marcel P Jackowski, "Three-dimensional synthetic blood vessel generation using

stochastic l-systems," in *Medical Imaging 2013: Image Processing*, Vol. 8669 (International Society for Optics and Photonics, 2013) p. 86691I.

[23] Aristid Lindenmayer, "Mathematical models for cellular interactions in development i. filaments with one-sided inputs," Journal of theoretical biology **18**, 280–299 (1968).

[24] Marek Kociński, Artur Klepaczko, Andrzej Materka, Martha Chekenya, and Arvid Lundervold, "3d image texture analysis of simulated and real-world vascular trees," Computer methods and programs in biomedicine **107**, 140–154 (2012).

[25] Dan Popescu, Mihaela Deaconu, Loretta Ichim, and Grigore Stamatescu, "Retinal blood vessel segmentation using pix2pix gan," in *2021 29th Mediterranean Conference on Control and Automation (MED)* (IEEE, 2021) pp. 1173–1178.

[26] He Zhao, Huiqi Li, Sebastian Maurer-Stroh, and Li Cheng, "Synthesizing retinal and neuronal images with generative adversarial nets," Medical image analysis **49**, 14–26 (2018).

[27] Andrea Tagliasacchi, Thomas Delame, Michela Spagnuolo, Nina Amenta, and Alexandru Telea, "3d skeletons: A state-of-the-art report," in *Computer Graphics Forum*, Vol. 35 (Wiley Online Library, 2016) pp. 573–597.

[28] Md Amirul Islam, Matthew Kowal, Patrick Esser, Sen Jia, Bjorn Ommer, Konstantinos G Derpanis, and Neil Bruce, "Shape or texture: Understanding discriminative features in cnns," arXiv preprint arXiv:2101.11604 (2021).

[29] Robert Geirhos, Patricia Rubisch, Claudio Michaelis, Matthias Bethge, Felix A Wichmann, and Wieland Brendel, "Imagenet-trained cnns are biased towards texture; increasing shape bias improves accuracy and robustness," arXiv preprint arXiv:1811.12231 (2018).

[30] Moises Freitas-Andrade, Cesar H Comin, Matheus Viana da Silva, Luciano da F Costa, and Baptiste Lacoste, "Unbiased analysis of mouse brain endothelial networks from two-or three-dimensional fluorescence images," Neurophotonics **9**, 031916 (2022).

[31] J. Staal, M.D. Abramoff, M. Niemeijer, M.A. Viergever, and B. van Ginneken, "Ridge-based vessel segmentation in color images of the retina," IEEE Transactions on Medical Imaging **23**, 501–509 (2004).

[32] Olaf Ronneberger, Philipp Fischer, and Thomas Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18* (Springer, 2015) pp. 234–241.

[33] Johannes C Paetzold, Suprosanna Shit, Ivan Ezhov, Giles Tetteh, Ali Ertürk, Helmholtz Zentrum Munich, and Bjoern Menze, "cldice—a novel connectivity-preserving loss function for vessel segmentation," in *Medical Imaging Meets NeurIPS 2019 Workshop* (2019).

[34] Dhurairajan Vijayalakshmi and Malaya Kumar Nath, "A strategic approach towards contrast enhancement by two-dimensional histogram equalization based on total variational decomposition," Multimedia Tools and Applications **82**, 19247–19274 (2023).

[35] Dhurairajan Vijayalakshmi and Malaya Kumar Nath, "A novel multilevel framework based contrast enhancement for uniform and non-uniform background images using a suitable histogram equalization," Digital Signal Processing **127**, 103532 (2022).