How to choose and adjust the tracing parameters

Contents

I. CHOOSING TRACING PARAMETERS

The algorithm requires the specification of several parameters used during the tracing that also strongly impact the tracing result. These parameters have to be chosen sensibly in order to obtain a sufficiently accurate fiber tracing. Here, these parameters are referred to as σ_{conv}, r_{min}, r_{max}, σ_{smooth}, $steps$, $thresh$, r_{step} and min_loop_length. Their exact function is described in II B to II G, but here a general overview of how to find sensible values for them is given. Since this algorithm could be applied to a wide variety of fibrous systems and imaging techniques, it is hard to speculate on what values would work best in each case. So far the algorithm has only been applied to SEM images and the guidelines given here are based on the experience gathered on those images. In general it was possible to find one set of parameters that works well on one kind of image. Therefore the results of similar images are very comparable since they can be analyzed with the same set of parameters. Also one or more parameters could be adjusted within a considerable range while maintaining a decent tracing quality. It has also proven sometimes useful, to completely disregard the following guidelines to obtain a sufficient tracing quality. The algorithm has also been designed to be scale invariant. This means, that if a image is magnified or the resolution is increased, the tracing should be almost identical if the parameters are adjusted accordingly. Some parameters have to be scaled linearly while others can stay constant if the scale of the image to be traced changes. The parameters whose range is recommended relative to the fiber diameter have to be scaled linearly with the magnification of the image. All other parameters do not need to be changed.

σ_{conv} should be chosen at around half the fiber diameter. It determines the amount of smoothing before the computation of the hessian matrix. A function has been implemented to map each pixel tho the smaller eigenvalue of its Hessian, flip the sign and optionally color code the fiber direction (See figure 13 or 1). The resulting image only depends on σ_{conv} and serves to adjust it. If σ_{conv} is chosen to small, the edges of the fiber will be brighter than its center. If σ_{conv} is chosen to big, the fibers will seem washed out and blurry. If σ_{conv} is chosen correctly, some structure should be visible in the fibers and the center of them should be brightest part of the image. There also should not be any visible noise left. In the tracing, a σ_{conv} that is to small can be one cause for multiple tracings of one fiber. The tracing with a σ_{conv} that is chosen to big, will have some fibers not traced, or bundles of them only traced once.

r_{min} is typically chosen as 0, but sometimes a value around or less than half fiber diameter can improve the tracing. It determines the minimal distance between the node and the pixels it considers for the angle of the newly spawned node.

r_{max} is one of the most challenging parameters to adjust, since it does not strongly impact the amount of branches and there is no additional metric by which to judge its choice. Although the sensible range varies, a usable starting point seems to be twice the fiber diameter or slightly more than that. If there are many branches not being traced, r_{max} might be to small. On the other hand, if the tracing of a fiber seems to strongly influenced by the surrounding fibers, r_{max} is likely

to big. It determines the maximal distance between the node and the pixels it considers for the angle of the newly spawned node. Big values of r_{max} increase the computation time, since more pixels have to be considered.

σ_{smooth} directly impacts the amount of new nodes spawned. The lower σ_{smooth}, the more nodes will be spawned. 0.5 radians appears to be a sensible starting value independent of the fiber diameter. It should not be chosen much higher than 0.8 radians and not lower than 0.35 radians. σ_{smooth} is responsible for the smoothing of the angle dependent curvature. The local maxima of this smoothed function determine the directions new nodes are spawned in. If the smoothing is stronger with a bigger σ_{smooth}, close local maxima will unite to one single local maxima, and only one new node will be spawned instead of several. If to many nodes are spawned, and they lie angularly close to each other, σ_{smooth} will likely have to be increased. If not enough branches are being found, σ_{smooth} might be to big.

steps is not a very important variable. It is mostly a trade off between run time and angular resolution. It determines the amount of points for which the smoothed angle dependent function is evaluated. Therefore it determines the amount of possible angles in which new nodes can be spawned. If the value is not chosen so low, that not all local maxima are captured, only small decreases in tracing quality will be caused. Values often chosen for it here were 100 and 360.

thresh has a somewhat similar impact as σ_{smooth}. It to strongly impacts the amount of new nodes being spawned. *thresh* determines the threshold above which local maxima in the smoothed angle dependent curvature are used to spawn new nodes. If a lot of nodes are spawned into noisy regions of the image where no fibers are present, *thresh* likely has to be increased. It should be noted, that with lower σ_{smooth}, the local maxima will become higher and more likely reach above *thresh*. If no nodes are spawned at all, *thresh* is likely the cause and has to be reduced. *thresh* should never be chosen below zero. Zero is also a good starting point, from which *thresh* can carefully be increased. A upper limit for *thresh* can not really be generally given, since it varies a lot between different kinds of images. In gerneral, once a decent set of parameters has been found, it is a good idea to increase *thresh* while simultaneously decreasing σ_{smooth} or vice versa. This will cause the amount of spawned nodes to remain relatively constant, but change their positions.

r_{step} is, similar to *steps*, somewhat of a trade of between run time and resolution. It determines how far away form the spawning node, new nodes will be placed. A smaller r_{step} means, that more nodes have to be spawned for the same length of fiber. If it is chosen to big, the nodes can skip by branches of fibers, meaning that these branches will not be traced. It is however possible to choose r_{step} to small. Especially when the fiber thickness exceeds 5 pixels, a to small value of r_{step} might cause single fibers to be traced several times or even whole meshworks to be falsely traced on top of one single fiber. A sensible staring point is between half and full fiber thickness.

min_loop_length is the minimal length of loops expressed in the amount of nodes part of the loop. A good range is between 5 and 10, and 7 appears as a usable starting value.

Great care has to be taken since it can easily be used to manipulate results by allowing or not allowing smaller loop sizes in the tracing. A good idea is to find the extremes of its sensible range and choose the middle. If the parameter is chosen to small, a lot of unwanted connections will be generated, resulting in regions extreamly densely populated with connections. If this parameter is chosen to large, a lot of almost closed loops will be visible in the tracing.

II. HOW THE FIBER TRACING WORKS

A. Desired tracing result and datastructure

The aim is to trace all fibers on a grayscale image by populating the image with nodes that are connected to one another. The nodes should lie on fibers and the connections between them should run along fibers so that if we plot all the connections between the fibers, each fiber in the image has exactly one plotted line running along it and every plotted line is on a fiber. One node can be connected to an arbitrary amount of other nodes, allowing for branching of the traced network. The positions of nodes are not bound to pixel positions but can lie anywhere between the pixels of the image but not outside the image. The connections between nodes should also never cross. In the case of crossing fibers, a node connected with both fibers should be placed at the junction instead. These nodes are placed based on an algorithm described below that also takes several input parameters influencing the fiber tracing. These input parameters have to be adjusted based on the kind of image to be traced (See I).

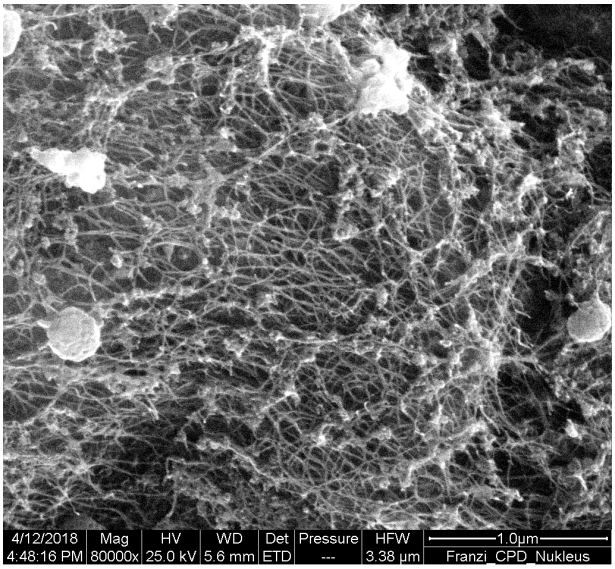
B. Mathematical basis

The algorithm works on the basis of the Hessian of the Gaussian transform to compute the smoothed second order derivatives of the image brightness profile (For more information see IV A 1 to IV A 3). The convolution with the three second order derivatives of the Gaussian yields the Hessian matrix in every pixel. Before this convolution, the image brightness is rescaled so the brightest pixel has a value of 255 and the darkest one a value of 0. Using the Hessian matrix, the directional second order derivative in any pixel and any direction can be easily computed. Generally negative second order derivatives with a big absolute value indicate the presence of a fiber. Also this fiber likely travels perpendicular to the direction of this directional second order derivative. Here the first input parameter of the algorithm is the standard deviation σ_{conv} of the Gaussian whose second order derivatives are used to compute the hessian.

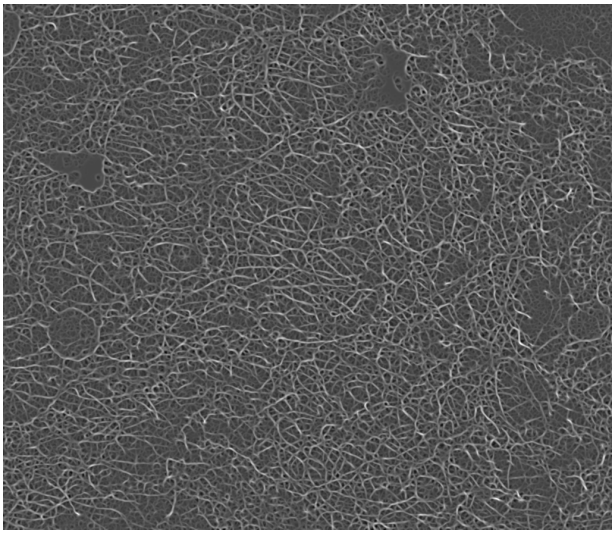
C. Positioning of starting nodes

The algorithm works by already existing nodes spawning new nodes. To kick start this process, at least one starting node has to be created. The starting node(s) should be placed on top of fibers or the tracing will likely terminate immediately. The placing of the starting node(s) can be done manually, but an automated function for this task has also been implemented. Its goal is to make sure, the placed nodes lie on fibers and are not clustered together to closely. It tries to achieve this by first computing the smaller eigenvalue of the hessian in each pixel and then flipping its sign. On top of a fiber, this smaller eigenvalue is negative and has a large absolute value, so when its sign is

flipped this results in a large positive value. The values obtained this way generate a second greyscale image (See figure 1b). On this second greyscale image the visibility of fibers should be enhanced and it should be lower noise. Subsequently the function places a starting node at the brightest pixel of this generated greyscale image. This brightest pixel will correspond to a negative hessian eigenvalue with a large absolute value and therefore likely be on top of a fiber (See IV A 3). Next a exclusion zone in a specifiable radius is created around the already placed node(s). Now the next node is placed again at the brightest pixel of the generated greyscale image that lies outside the exclusion zone(s) of the already placed starting node(s). This process can be repeated to create a arbitrary amount of starting nodes. The input parameters of this function are the amount of nodes to be placed, the radius of the exclusion zones and optionally a cutoff for the brightness of the brightest pixel, below which no new starting node will be placed. For our images, 100 starting nodes were placed with a exclusion radius of 60 pixels corresponding to about 15 times fiber thickness. A cutoff value of zero was chosen. These input parameters however are not considered to be part of the tracing process, since the starting nodes can also be placed manually and no fiber tracing is being done by this function. Also they do not seem to strongly impact the final tracing result.



(a) SEM image of actin fibers. Imaged by Daniel Flormann.



(b) -1 times the smaller eigenvalue of the Hessian matrix in each pixel of figure 1a is mapped to the brightness in the corresponding pixel.

Figure 1: Using the eigenvalues of the Hessian to remove noise and enhance fiber visibility. The Hessian is computed using the second order derivatives of a Gaussian filter. Then the smaller eigenvalue is computed and its sign is flipped. The value obtained this way is displayed in figure 1b.

D. Basics of the tracing procedure

The way the fiber network is now populated with nodes is, that the already existing nodes can spawn new nodes they are then connected to. Each node can try to spawn new nodes exactly once. These new nodes should be spawned in a direction fibers are travelling

in. The newly spawned nodes then have the opportunity to again spawn new nodes exactly once. This way the nodes spread along the fibers and thereby trace them. The algorithm terminates when in the final step no new nodes are spawned. At this point the network should be completely traced. Therefore the heart of the algorithm is the way, one node attempts to spawn new nodes.

E. Nodes spawning new nodes

The function described here is called exactly once for every node. Its goal is to find the directions along which fibers travel near the node and then spawn new nodes in those directions. Later this function is then called again on these newly spawned nodes, that again can spawn new nodes. The process terminates when in the final step no new nodes are spawned.

The starting point for this function is the position of the node it is called on. This position is expressed by two double precision floating point values, the nodes x and y coordinate, and lies somewhere on the image that is being traced. Now all pixels that lie in a specifiable distance interval to this position are determined. This means, that two radii r_{min} and r_{max} have to be specified. These radii r_{min} and r_{max} are input parameters for the algorithm. All pixels that lie outside the smaller and inside the bigger radius around the node position will be considered. The processing of one of these pixels will now be described, but all of the selected pixels will be processed this way.

Let x_{node} and y_{node} be the coordinates of the node position, x_{pixel} and y_{pixel} be the coordinates of the pixel position. Furthermore be $\Delta x = x_{pixel} - x_{node}$ and $\Delta y = y_{pixel} - y_{node}$. The connection vector $\vec{v}$ (See figure 2) that points from the node to the pixel is then given by

$$\vec{v} = \begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix}$$

A vector perpendicular to $\vec{v}$ shall be called $\vec{v_p}$ (See figure 2) and amounts to

$$\vec{v_p} = \begin{pmatrix} -\Delta y \\ \Delta x \end{pmatrix} \perp \begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix} = \vec{v}; \quad \vec{v} \cdot \vec{v_p} = 0$$

For each pixel that is being processed, two

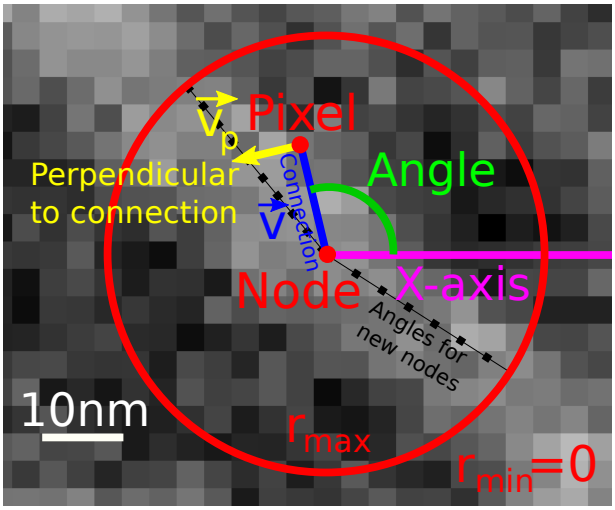


Figure 2: SEM image of an actin fiber imaged by Daniel Flormann and illustrations clarifying the function of the automated fiber tracing algorithm.

For each pixel between r_{min} and r_{max} the negative normalized second order derivative along $\vec{v_p}$ is computed along with the angle of $\vec{v}$ to the x-axis. These values are then used to determine in which angles new nodes are spawned.

values are computed. The angle of $\vec{v}$ relative to the x-axis (See figure 2) and the negative normalized directional second order derivative in the pixel along $\vec{v_p}$. Here the latter value is abbreviated to res. In the following,

the computation of res will be described. Be

$$H = \begin{pmatrix} a & b \\ b & c \end{pmatrix}$$

$$a = \frac{d^2}{dx^2}$$

$$b = \frac{d^2}{dx\,dy} = \frac{d^2}{dy\,dx}$$

$$c = \frac{d^2}{dy^2}$$

the Hessian matrix in the pixel and a, b and c its corresponding entries (See IV A 2). The value of res amounts to

$$res = -\frac{\vec{v_p}^{\,\intercal}\ H\ \vec{v_p}}{|\vec{v_p}|^2}$$

Here, $\vec{v_p}^{\,\intercal}\ H\ \vec{v_p}$ yields the second order derivative in the pixel along $\vec{v_p}$. The denominator of $|\vec{v_p}|^2$ compensates for the length of $\vec{v_p}$ and normalizes the derivative. The $-$ flips the sign of everything. Because we expect the presence of a fiber to yield a negative second order derivative with a big absolute (See IV A 3), flipping the sign allows us to search for a big positive value of res. To actually compute res, the values for the vector $\vec{v_p}$ and the matirx H need to be inserted:

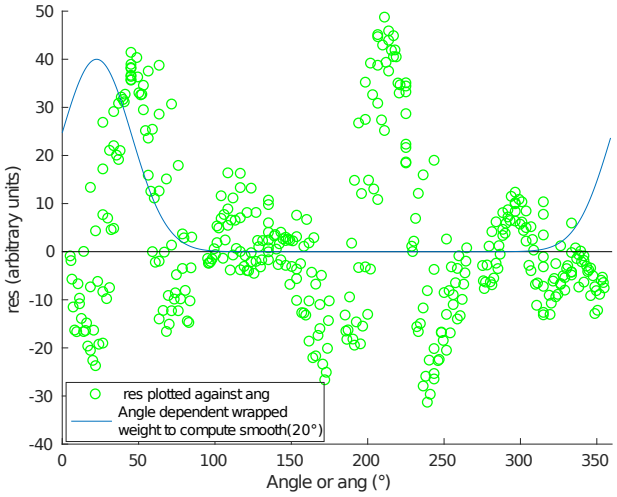
$$\begin{aligned} res &= -\frac{(-\Delta y\ ,\ \Delta x)\begin{pmatrix} a & b \\ b & c \end{pmatrix}\begin{pmatrix} -\Delta y \\ \Delta x \end{pmatrix}}{\Delta x^2 + \Delta y^2} \\ &= -\frac{\Delta y^2\ a - 2\ \Delta x\ \Delta y\ b + \Delta x^2\ c}{\Delta x^2 + \Delta y^2} \end{aligned}$$

Now for each of the selected pixels, two values have been computed. The angle of $\vec{v}$ to the x-axis and res. From now on, the angle of $\vec{v}$ to the x-axis is abbreviated to ang. This yields the ordered pair (ang, res) for every processed pixel. The goal of the function described here is to find angles along which fibers travel. These will be the angles, where the corresponding res values are predominantly big. Since res values are only known for discrete ang values, these ang values are not equidistant and the ang values of several pixels can even be identical, the individual res values are not well suited to determine any such angles. Subsequently a way to find angles, where the res values in the vicinity are generally large is required. Therefore a measure, that takes several res values with similar ang values into account is needed. To achieve this, a weighted average over the res values is used. This weight is only significant for the res values within a small angle interval. Specifically, the weight of the res values is given by a Gaussian of the ang values. This measure can be evaluated for any angle, by shifting the center of the Gaussian to that angle. The standard deviation σ_{smooth} of this Gaussian has to be specified and can be used to control the size of the angle interval with significant weight. This measure evaluated at a specific angle α is abbreviated to $smooth(\alpha)$, because the resulting function can be regarded as a smoothing of the angle dependent res values. It should be noted that this Gaussian wraps around, since the angle describes a closed circle (See figure 3a). Mathematically $smooth(\alpha)$ is computed by

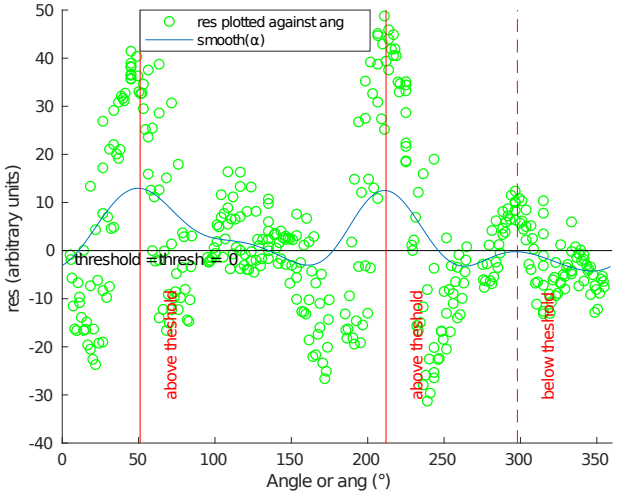
$$\begin{aligned} weight_i := \ & \exp\left(\frac{(ang_i - \alpha)^2}{2\ \sigma^2_{smooth}}\right) \\ & + \exp\left(\frac{(ang_i - \alpha + 2\pi)^2}{2\ \sigma^2_{smooth}}\right) \\ & + \exp\left(\frac{(ang_i - \alpha - 2\pi)^2}{2\ \sigma^2_{smooth}}\right) \end{aligned}$$

$$smooth(\alpha) = \frac{\sum_{i=1}^{n} res_i\ weight_i}{\sum_{i=1}^{n} weight_i}$$

where n is the total number of processed pixels, res_i the res value of the i-th computed



(a) Visualization of the angle dependent weight for the *res* values used to compute $smooth(\alpha)$.



(b) The local maxima of $smooth(\alpha)$ are the angles in which new nodes are spawned if they lie above the threshold *thresh* (Here $thresh = 0$)

Figure 3: Nodes spawn new nodes in the angles where $smooth(\alpha)$ has a local maximum above *thresh* (See figure 3b). $smooth(\alpha)$ is computed by the weighted mean of the *res* values. This angle dependent weights is shown in figure 3a. The *res* values can be thought of as a negative angle dependent second order derivative.

pixel and ang_i the *ang* value of the i-th computed pixel. The two additional summands to the weight of the i-th computed pixel $weight_i$ are to wrap the Gaussian around the circle once. Theoretically a infinite amount of Gaussians would have to be summed up, each shifted by 2π, but the contributions of the missing terms are negligible in the usable σ_{smooth} range (See I).

$smooth(\alpha)$ is then computed for a specifiable number of angles, that are evenly distributed between 0 and 2π radians (See figure 3). The number of points, $smooth(\alpha)$ is computed for, is a input parameter of the algorithm and called *steps*. Next the local maxima of the evaluated points of $smooth(\alpha)$ that lie above a specifiable threshold *thresh* are determined. The angle of those local maxima are then used to try and spawn new nodes (See figure 3b). The new nodes are spawned in those angles relative to the old node (See figure 2) in the specifiable distance r_{step}, when they lie inside the image and not to close to other already existing nodes.

F. Preventing the crossing of connections

To prevent connections between nodes from crossing each other, which would cause an oversampling, a minimal distance between the nodes was implemented. If a new node is to be placed closer to a existing node than $\frac{r_{step}}{\sqrt{2}}$, the node will not be placed (See figure 4).

G. Connections between existing nodes

The part of the algorithm that has been described so far only allows for branching in the network. But there has to be a way for the branches to reunite and form closed loops. To achieve this, two already existing nodes can

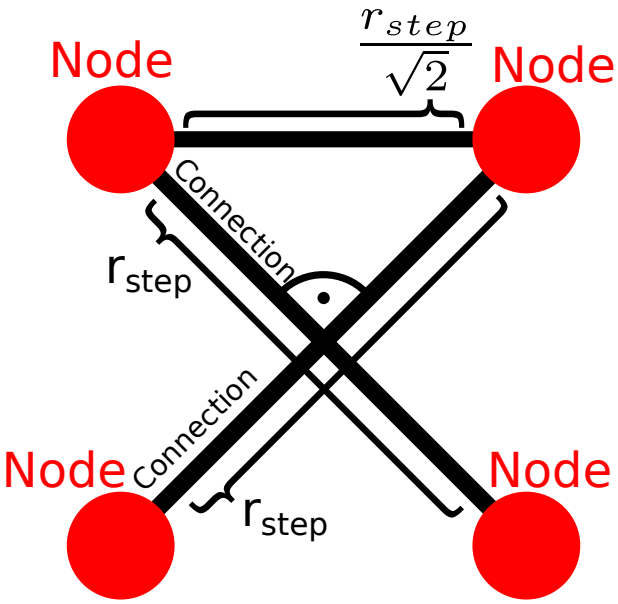


Figure 4: The maximal distance between 4 node whose connections are crossed is $\frac{r_{step}}{\sqrt{2}}$

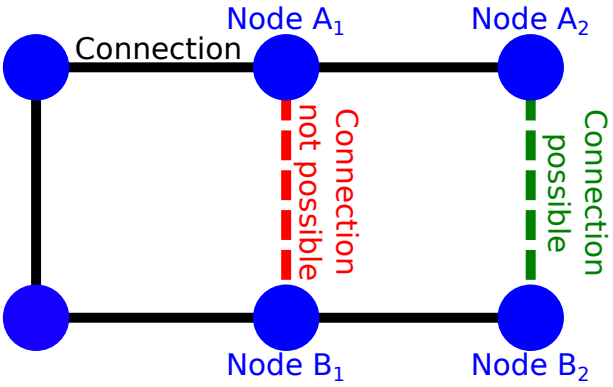


Figure 5: Example of the effects of *min_loop_length*. Here *min_loop_length* = 4 therefore the left connection between the nodes A_1 and B_1 not possible but the one between A_2 and B_2 is.

connect under certain conditions. This can happen, when a node A wants to spawn a new node that would be too close to an already existing node B. If this condition is met, it is checked whether these two nodes A and B are already connected in the network over a specifiable number of steps *min_loop_length*. More specifically, the check is if it is possible to jump only between connected nodes from node A within *min_loop_length* steps to node B. If this is not possible, the two nodes are connected (See figure 5). If *min_loop_length* is set to zero or this second check is not done, the algorithm will constantly build triangles and connect way too many nodes.

H. Extracted parameters

The raw tracing data does not help directly when trying to find differences in images. Therefor relevant parameters have to be extracted from the tracing data to allow for useful quantification of the image.

One of the most relevant parameters in the context of actin meshes is the mesh hole size since it is assumed to correlate with the physical properties of the cell [1]. In the tracing data mesh holes are easily defined as closed loops inside of which there are no other closed loops. Their size can now be quantified by any number of measures such as their area or their circumference. Therefore once the meshwork has been traced accurately, the distribution of its mesh hole size can be extracted and compared between different conditions.

The tracing data also allows for the easy recognition of fiber junctions or branches. These junctions can easily be recognized by the nodes that are connected to more than two other nodes. In a continuous fiber without junctions each node is connected to exactly two other nodes while its endpoints are connected to only one other node. Therefor any node that has more than two connections is regarded as a junction of fibers. Now the distances between those junctions can be easily computed and regarded as a measure for fiber length

or cross linking frequency of the network. This measure will, however, underestimate fiber length since not all fibers end at their junctions and therefore has to be used with caution.

III. VALIDATION OF THE TRACING ALGORITHM

A. Highest reliably traceable fiber density

A test was performed to find the highest reliably traceable fiber density. For this test, artificial images were created. These were 1000 by 1000 pixels in size and a grid of horizontal and vertical white lines five pixel in thickness was generated on a black background. As a measure for line density, the percentage of black pixels on the image was used since black pixels are not part of any line. When the amount of white lines increases, the percentage of black pixels on the image decreases. The images were traced with parameters that also worked well on SEM images of the actin cortex of adherent cells (See table I). As a measure for the quality of the fiber tracing, the mesh hole areas computed by the algorithm were compared to the known mesh hole areas of the generated image (See figure 6). This discrepancy between the exact and computed values was then expressed by the standard deviation of the computed areas from the exact areas. Be a_{exact} the exact mesh hole area of the generated image and be a_{comp_i} the i-th computed mesh hole area. Furthermore be n the number of computed areas. Then the standard deviation dev of the computed areas from the exact value is:

$$dev = \sqrt{\sum_{i=1}^{n} \frac{(a_{comp_i} - a_{exact})^2}{n}}$$

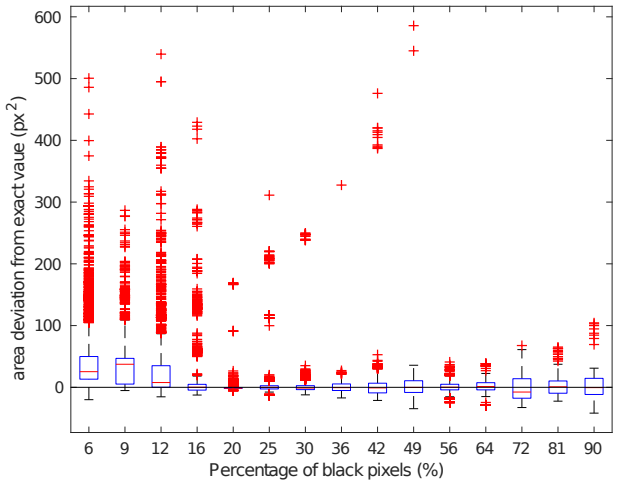
This standard deviation dev was expressed as a percentage of the exact value and mapped to the percentage of black pixels in figure 6b. In figure 6 a big increase in the discrepancy between the computed and exact values can be observed below the 20% black pixel mark. The image with the highest still reasonably well tractable fiber density is shown in figure 7. Therefore we can expect the algorithm to not be drastically affected by fiber density if no more than 80% of the pixels in a image are part of fibers. Also the parameters were not optimized for the generated images, so when the parameters of the tracing algorithm are adjusted, even higher line densities could likely be traced reliably.

Table I: Parameter set used to trace the generated images with varying fiber density.

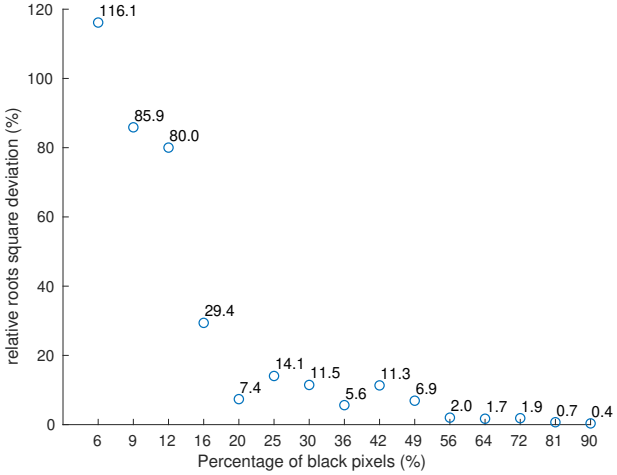
Parameter	Value
σ_{conv}	2.3 pixel
r_{min}	0 pixel
r_{max}	8 pixel
σ_{smooth}	0.55 radians
$steps$	360
$thresh$	0.5
r_{step}	3 pixel
min_loop_length	7

B. Validation of mesh hole areas on images with known ground truth

A already existing fiber quantification tool is DiameterJ [2]. It is a plugin for ImageJ [3]. For the validation and of this plugin, a



(a) The deviation of the computed areas from the exact value mapped to the percentage of black pixels. Ideally all data points should lie on the x-axis.



(b) Standard deviation of the computed areas from the exact value expressed as a percentage of the exact value mapped to the percentage of black pixels.

Figure 6: Test for the highest reliably traceable fiber density by comparing the known mesh hole areas of generated images to the mesh hole area distribution computed by the tracing. The images were generated as white lines on a black background. Therefore the percentage of black pixels decreases as line density increases.

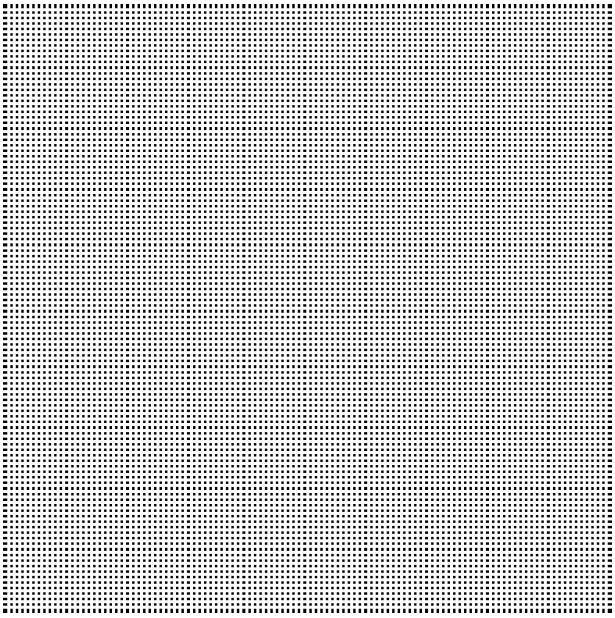
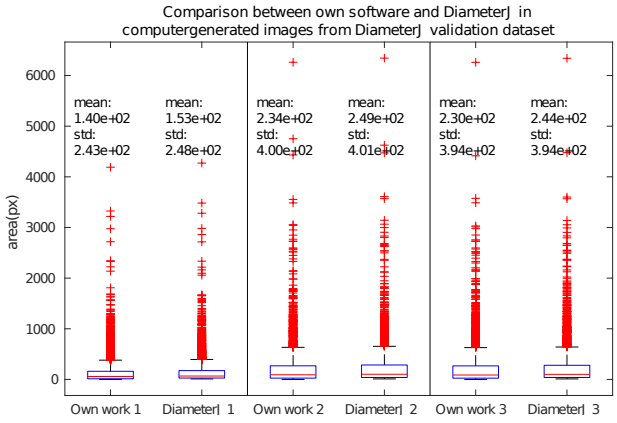


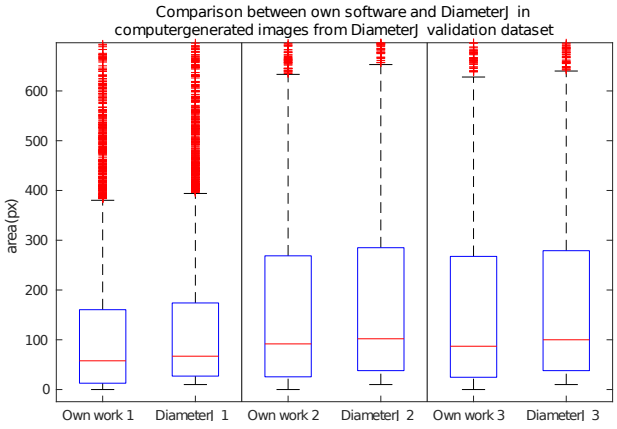
Figure 7: Generated image with the highest still reasonably well tractable fiber density.

dataset was published [4] from which 3 suitable images were selected. One of these images is shown in figure 9. They were selected, because they are already segmented into only black and white pixels and therefore the output of DiameterJ can be regarded as ground truth. Also the images did not consist of regular patterns, making the tracing more difficult and likely to fail. Additionally the white lines were 3 pixel thick, a value comparable to the thickness of actin fibers in our SEM images. DiameterJ considers the thickness of the fibers when computing the mesh hole areas. To account for this, the circumference times half the fiber diameter was subtracted from the area of every loop in the tracing data. The tracing was done using a parameterset that also works well on the SEM images of the actin fibers (See table II) our program was developed for. The mean values computed from the DiameterJ mesh hole area distributions were never more than 10% off from the values our tracing algorithm produced. Also the mean produced by DiameterJ is always between 6% and 10% more than our

result. Therefore all our results deviate in a similar way from the ground, causing the results to be very comparable (See figure 8).



(a) Mesh hole area distributions produced by DiameterJ and our fiber tracing algorithm.



(b) Figure 8a magnified

Figure 8: Evaluating the viability of our fiber tracing algorithm by comparing the produced mesh hole area distribution to the one generated by DiameterJ. The images analyzed for this test only consisted of black and white pixels (See figure 9) and therefore the output of DiameterJ can be regarded as ground truth.

Table II: Parameter set used to trace the images from the DiameterJ validation dataset [4].

Parameter	Value
σ_{conv}	1.8 pixel
r_{min}	0 pixel
r_{max}	8 pixel
σ_{smooth}	0.55 radians
$steps$	360
$thresh$	0.5
r_{step}	3 pixel
min_loop_length	7

C. Comparison to DiamerJ on our SEM images of the actin cortex

DiameterJ is a easily accessible fiber analysis software that can extract some of the parameters we are interested in, such as the mesh hole area distribution. When it is applied to our specific imaging results of the actin cortex, many fibers are not found (See figure 10). Our software shows superior fiber recognition on these specific images. While it can also be argued, that the analysis by DiameterJ can be used to objectively compare different conditions and the imperfections cancel out in the comparison, starting with a more representative analysis will ensure better comparability.

D. Advantages and limitations of our analysis

As is visible in figure 10a, most fibers are traced accurately. There are, however, some fibers that are not recognized and some traces that do not correspond to any fiber. Also the three-dimensional structure of the network is not taken into account at all. But it is unlikely, that any fiber recognition software

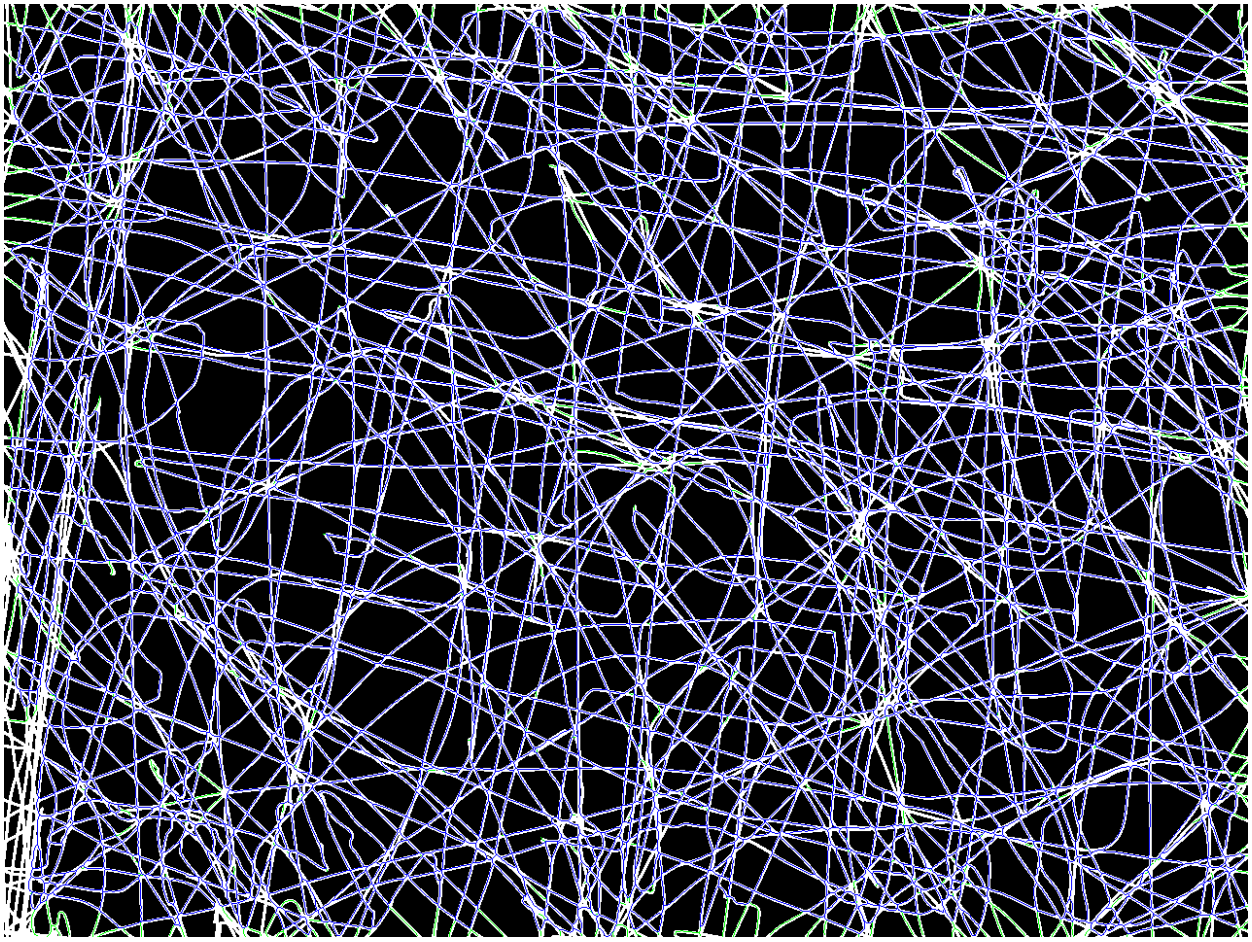
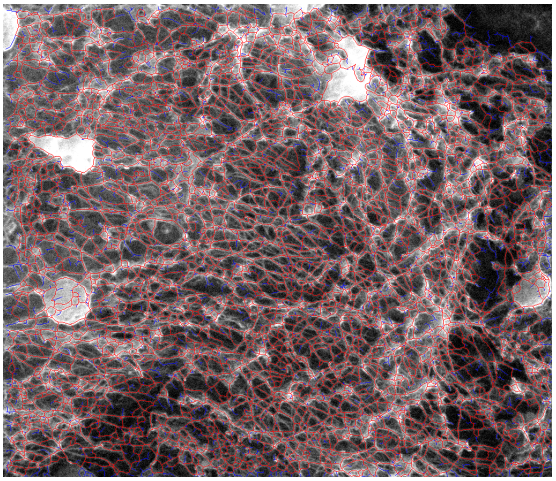
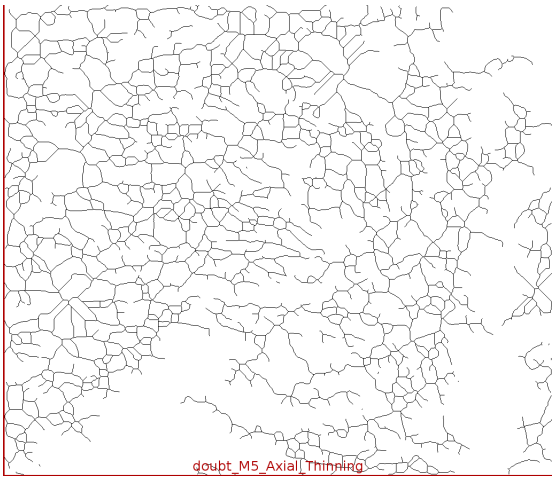


Figure 9: Image from the DiameterJ validation dataset [4] with visualized tracing data from our fiber tracing algorithm. The original image from the dataset is only black and white, all colored lines are part of the visualized tracing data. Blue lines are part of a closed loop and green lines are not and therefore green lines do not contribute to the mesh hole area distribution.

will ever yield perfect results on these images, due to the present noise, limited resolution and impurities on the sample. Additionally because the obtained images are only two-dimensional, we are not able to resolve the three-dimensional structure of the actin network. Also manual fiber tracing is unlikely to yield repeatable results for different operators because some structures can not be explicitly identified. The results of our software on the other hand are repeatable, and the imperfections in the tracing data are present in all analyzed images to the same degree. Therefore the parameters extracted from the tracing of different images can be compared and used to objectively quantify the differences between conditions. Furthermore the large amount of data generated by the tracing of each image ensures good statistics, comparability and causes random variations to be less significant. In each analyzed image, several hundred up to thousands of mesh holes are found.



(a) SEM image of actin cortex with visualized tracing data from our program. Blue lines are not part of any closed loop and do therefore not contribute to the mesh hole area distribution, in contrast to the red lines. Parameters used for tracing in table III



(b) Fibers found on image 10a by DiameterJ.

Figure 10: Comparison between the fibers found by our software and DiameterJ on a SEM images of the actin cortex. Imaging by Daniel Flormann.

Table III: Parameter set used to trace all SEM images that were analyzed for this work.

Parameter	Value
σ_{conv}	1.8 pixel
r_{min}	0 pixel
r_{max}	8 pixel
σ_{smooth}	0.5 radians
$steps$	360
$thresh$	2
r_{step}	3 pixel
min_loop_length	7

IV. MATHEMATICAL BACKGROUND

A. Mathematical operations on images

1. *Derivatives and convolutions of images*

Images can be understood as functions, that map a two-dimensional space to a one-dimensional space. Specifically, the pixel-position is mapped to the pixel brightness. Of course, this map is only defined at discrete points namely the pixels but can be approximated as a continuous map. Here it should be noted, that these maps that correspond to the images can be thought of as bounded and arbitrarily smooth. This is the case, because the values of images are bounded and are only defined at discrete points (pixels) with finitely small distances between them. Therefore the map corresponding to the image can be interpolated between the pixels in a arbitrarily smooth way. Therefore not only the image its self is bounded but all derivatives as well. This approximation of images as continuous maps allows us to apply mathematical operations to the image to enhance or get rid of certain features. Specifically, we can compute the

derivatives of the map corresponding to the image. If we compute the derivative directly on the original image, any noise will be strongly amplified by the derivative, making the output unusable. Therefore the image usually has to be smoothed for noise reduction first. This is especially true for SEM images since they are prone to noise. This noise reduction or smoothing can be achieved by convoluting the image with a Gaussian kernel. Mathematically the convolution of a map $f(\vec{r})$ with a kernel $k(\vec{r})$ is defined as:

$$(f * k)(\vec{r}) = \int f(\vec{r'})\ k(\vec{r'} + \vec{r})\ d\vec{r'}$$

The Gaussian kernel $g(\vec{r})$ is defined as:

$$g(\vec{r}) = \frac{1}{\sigma\sqrt{2\,\pi}}\ \exp -\frac{\vec{r}^{\,2}}{2\,\sigma}$$

With the standard deviation σ and the position vector $\vec{r}$. In practice on an image, the kernel also consists of discrete pixels and is usually smaller than the picture. The kernel is scanned over the image, while the kernel pixel values are multiplied with the corresponding image pixel values to compute the convoluted image.
This way the picture can be smoothed and the noise drastically reduced. But the derivatives still have to be computed. This is difficult on a discrete map because the derivative of a function $f(\vec{r})$ in the direction $\vec{u}$ is defined as $\frac{df(\vec{r})}{d\vec{u}} = \lim_{h\to 0} \frac{f(\vec{r}+h\,\vec{u})-f(\vec{r})}{h}$, but the pixels have discrete distances and therefore the $\lim_{h\to 0}$ can not be computed with arbitrary precision.
Instead of first smoothing the image and then computing the derivatives, both those steps can be combined into one. To illustrate this, let us consider the derivative of a convolution of sufficiently smooth function $f(\vec{r})$ and kernel $k(\vec{r})$:

$$\begin{aligned}
&\frac{d}{dr_i}(f * k)(\vec{r}) \\
&= \frac{d}{dr_i}\int f(\vec{r'})\ k(\vec{r'} + \vec{r})\ d\vec{r'} \\
&= \int f(\vec{r'})\ \frac{d}{dr_i}k(\vec{r'} + \vec{r})\ d\vec{r'} \\
&= \int f(\vec{r'})\ k_i(\vec{r'} + \vec{r})\ d\vec{r'}
\end{aligned}$$

Where r_i is the i-th coordinate and $k_i(\vec{r})$ is the derivative in the i- th coordinate of the kernel. So instead of first convoluting and then calculating the derivative, we can simply convolute with the derivative of the kernel $k(\vec{r})$. This is especially useful if the discrete kernel for picture convolution is generated by discrete evaluations of a analytical function like the Gaussian kernel. This way we can calculate the analytical derivative for maximal accuracy before the discrete evaluation of the kernel.
With this method we can compute any order derivative by simply deriving the kernel.

2. The Hessian matrix of images

Image brightness profile curvature can be used to enhance the visibility of fibers in an image (See IV A 3). The second-order derivative or curvature of a two dimensional map $f : \mathbb{R}^2 \to \mathbb{R};\ (x, y) \mapsto f(x, y)$ takes the form of a 2x2-Matrix called the Hessian matrix H:

$$H = \begin{pmatrix} \frac{d^2 f}{dx^2} & \frac{d^2 f}{dx\,dy} \\ \frac{d^2 f}{dy\,dx} & \frac{d^2 f}{dy^2} \end{pmatrix}$$

Here, $\frac{d^2}{dx\,dy} = \frac{d^2}{dy\,dx}$ holds for sufficiently smooth functions due to the theorem of

Schwarz. Therefore the Hessian matrix is symmetric and we obtain 3 different analytical kernels or derivatives of the Gaussian $g(x, y)$. To calculate the Hessian in every pixel of the smoothed image, we convolute the image with the 3 discrete kernels obtained by evaluating the analytical derivatives of the Gaussian $g(x, y) = \frac{1}{\sigma\sqrt{2\,\pi}} \exp -\frac{x^2+y^2}{2\,\sigma}$(See Figure 11).

$$\frac{d^2}{dx^2}\, g = \frac{\left(\left(\frac{x}{\sigma}\right)^2 - 1\right) \exp -\frac{x^2+y^2}{2\,\sigma^2}}{\sigma^3\sqrt{2\,\pi}}$$

$$\frac{d^2}{dx\,dy}\, g = \frac{d^2}{dy\,dx}\, g = \frac{x\,y\, \exp -\frac{x^2+y^2}{2\,\sigma^2}}{\sigma^5\sqrt{2\,\pi}}$$

$$\frac{d^2}{dx^2}\, g = \frac{\left(\left(\frac{y}{\sigma}\right)^2 - 1\right) \exp -\frac{x^2+y^2}{2\,\sigma^2}}{\sigma^3\sqrt{2\,\pi}}$$

If we choose to look at the curvature of a specific point or pixel with coordinates (x, y) the Hessian H takes the form:

$$H = \begin{pmatrix} a & b \\ b & c \end{pmatrix}$$

$$a = \frac{d^2 f(x,y)}{dx^2}$$

$$b = \frac{d^2 f(x,y)}{dx\,dy} = \frac{d^2 f(x,y)}{dy\,dx}$$

$$c = \frac{d^2 f(x,y)}{dy^2}$$

Using this Hessian matrix H any second-order directional derivatives in this point can easily be computed as:

$$\frac{d^2 f(\vec{r})}{d\vec{u}^2} = \vec{u}^\intercal H \vec{u} = a\,u_x^2 + 2\,b\,u_x\,u_y + c\,u_y^2$$

$$\vec{u} = \begin{pmatrix} u_x \\ u_y \end{pmatrix};\ \vec{r} = \begin{pmatrix} x \\ y \end{pmatrix}$$

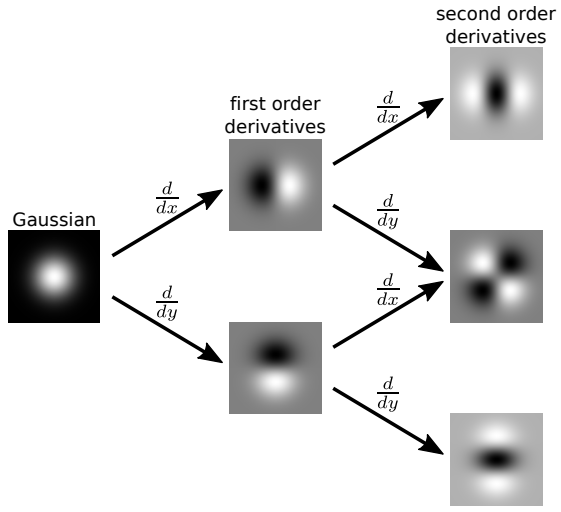


Figure 11: Visualization of the discrete kernels obtained by evaluation of the analytical derivatives of the Gaussian. Visualization generated by OpenCV [5]

Here it should be noted, that this second order directional derivative scales with $|\vec{u}|^2$. Therefore we implicitly assume, that $|\vec{u}| = 1$ when comparing directional derivatives from here on.
To better understand the information contained in the Hessian matrix, we can work out the eigenvalues:

$$\lambda_1 = \frac{a + c - \sqrt{(a-c)^2 + 4\,b^2}}{2}$$

$$\lambda_2 = \frac{a + c + \sqrt{(a-c)^2 + 4\,b^2}}{2}$$

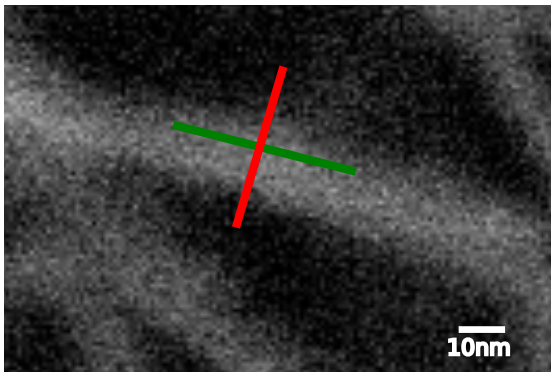
and the two perpendicular not normalized eigenvectors:

$$\vec{v_1} = \begin{pmatrix} a - c - \sqrt{(a-c)^2 + 4\,b^2} \\ 2\,b \end{pmatrix}$$
$$= 2 \begin{pmatrix} \lambda_1 - c \\ b \end{pmatrix}$$
$$\vec{v_2} = \begin{pmatrix} a - c + \sqrt{(a-c)^2 + 4\,b^2} \\ 2\,b \end{pmatrix}$$
$$= 2 \begin{pmatrix} \lambda_2 - c \\ b \end{pmatrix}$$

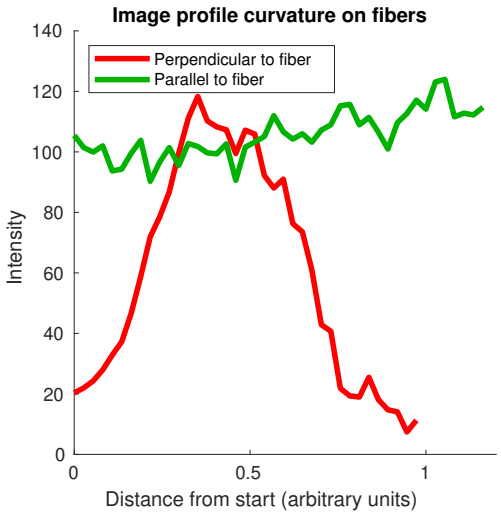
Here λ_1 is the smallest possible second-order directional derivative and $\vec{v_1}$ indicates the associated direction of smallest curvature. Conversely λ_2 is the biggest curvature value and $\vec{v_2}$ the associated direction. This is because $\sqrt{(a-c)^2 + 4\,b^2} > 0$ and the only difference between λ_1 and λ_2 is the sign before this summand.
Here it is important to note that this statement does not apply to the absolute values of λ_1 and λ_2 since they can both either be positive or negative. Therefore the function might be much more strongly curved in the direction of $\vec{v_1}$ than in the direction of $\vec{v_2}$ because $\vec{v_1}$ has a large absolute value and is negative.
If we do a coordinate transform and use the normalized versions of v_1 and v_2 as basis vectors, the Hessian matrix becomes the diagonal matrix:

$$H = \begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix}$$

meaning, that in this new coordinate system, the mixed derivative terms are not important ($\frac{d^2}{dx\,dy} = \frac{d^2}{dy\,dx} = 0$).



(a) SEM image of actin fiber. Imaged by Daniel Flormann.

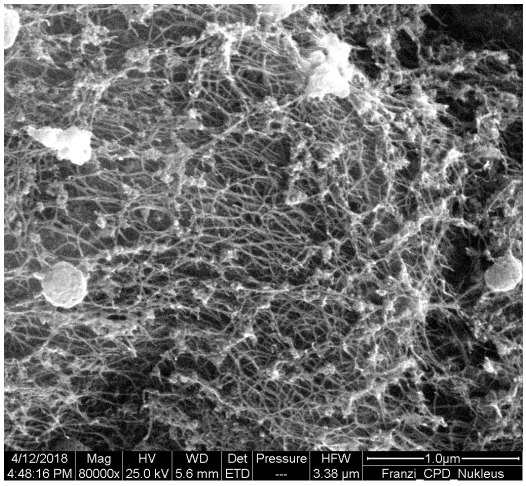


(b) image brightness profile of lines drawn in figure 12a.
Intensity profile generated by Fiji [6].

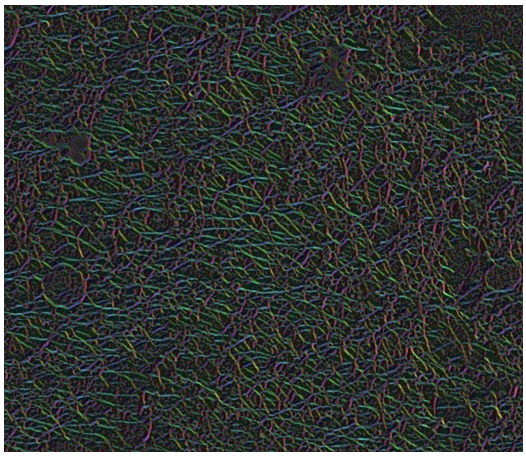
Figure 12: When travelling along a fiber, the image brightness profile is relatively constant and when travelling perpendicular to a fiber, the image brightness forms a local maximum in the middle of the fiber. This causes a large negative second-order derivative or curvature in the middle of fibers.

3. Usefulness of the Hessian matrix in fiber recognition

Fibers in images can be recognized as line like structures, whose brightness is different



(a) SEM image of actin fibers by Daniel Flormann.



(b) -1 times smaller eigenvalue of the Hessian matrix of figure 13a is mapped to the brightness. The color encodes the direction of the corresponding eigenvector. This causes fibers travelling in the same direction have the same color. (See IV A 1 and IV A 2)

Figure 13: Using the eigenvalues of the Hessian to remove noise and background and enhance fiber visibility. The Hessian is computed by calculating the second order derivatives of the Gaussian filter. Then the smaller eigenvalue is computed and its sign is flipped. The value obtained this way is displayed in figure 13b.

from the local background. Without loss of generality, here we assume them to be brighter than the local background.

Let us now consider a point in the middle of a fiber. The brightness of the image along the fiber is going to remain relatively constant. On the other hand, the brightness perpendicular to the fiber will decrease quickly, forward as well as backward. This means there is a local maximum of brightness in the middle of the fiber when looking at the image brightness on a line perpendicular to the fiber (See Figure 12). On a local maximum of a sufficiently smooth function, the second-order derivative or curvature is always negative.

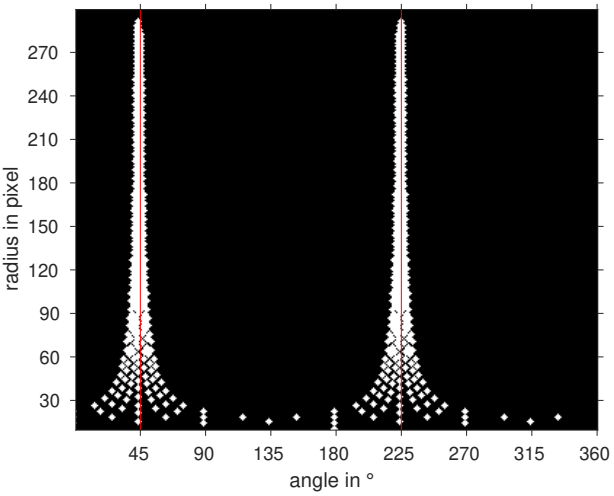
For this reason, we expect the centres of fibers to have negative second order derivatives. We can use this for example to enhance the visibility of fibers in images and filter their the background. This can be archived by creating a new image, whose pixel values consist of -1 times the smaller eigenvalue of the Hessian matrix in that pixel (See figure 1). The -1 has to be used to swap the sign because we expect the second-order derivative to be negative on top of fibers. Also the eigenvector corresponding to this will be perpendicular to the fiber direction. Therefore we know the direction the fiber is travelling in (See figure 13). This method is very similar to the one used by Kikinis et. al. and was inspired by them [7]. Another reason this second order derivative is useful for fiber recognition is, that the kernels used to compute the Hessian, can be thought of as a negative fiber template. They are dark in the middle, with two bright spots to the sides. Therefore, if such a kernel is placed on a fiber, that is bright with less brightness to its sides, it will yield a negative response with a large absolute value. If we now reverse the sign of

this response, we obtain a map with enhanced fibers (See Figure 11).

4. Using polar coordinates for fiber tracing

Usually, pixel positions of an image are expressed as Cartesian coordinates (x, y). The value x can be thought of as how far the pixel is to the right and y as how far up the pixel is positioned. Therefore this coordinate system expresses position relative to two predefined directions, namely the basis vectors. The polar coordinates, on the other hand, use the angle ϕ and the radius r to express position. Here $x = r\cos\phi$ and $y = r\sin\phi$. These coordinates have therefore no predefined directions or angles because the angle itself is a variable. For this reason, they are well suited for fiber tracking. If the origin of the polar coordinates is set on top of a fiber or line, bright pixels will accumulate at the angle, the line is travelling in (See figure 14). [8]

(a) Example image in Cartesian coordinates



(b) Figure 14a in polar coordinates. Origin of the polar coordinates is in the center of the image 14a

Figure 14: Difference in parameterization of an image. Figure 14a shows the image in Cartesian coordinates and figure 14b shows all white pixels of figure 14a mapped to polar coordinates. The origin of the polar coordinates is in the middle of the figure 14a. Using figure 14b the angle, the line in 14a is traveling in, can easily be determined as 45°. This is indicated by the red vertical lines.
Images generated using OpenCV [5]

B. Previous versions of the fiber tracing algorithm

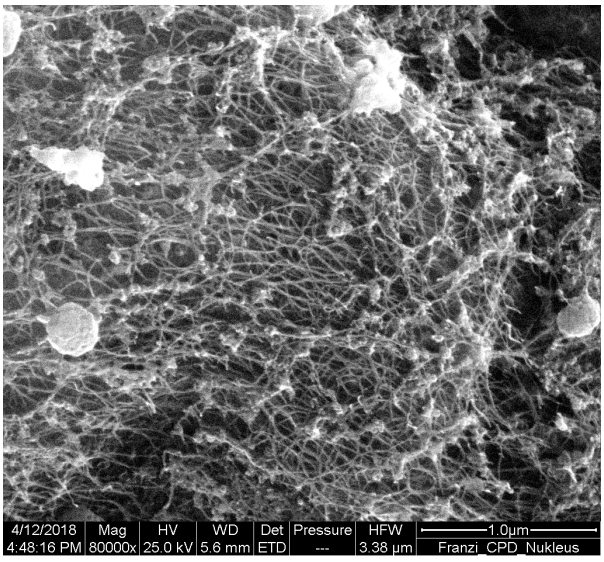
A developement history of the algorithm can be found at:

`https://github.com/SRaent/Actin`

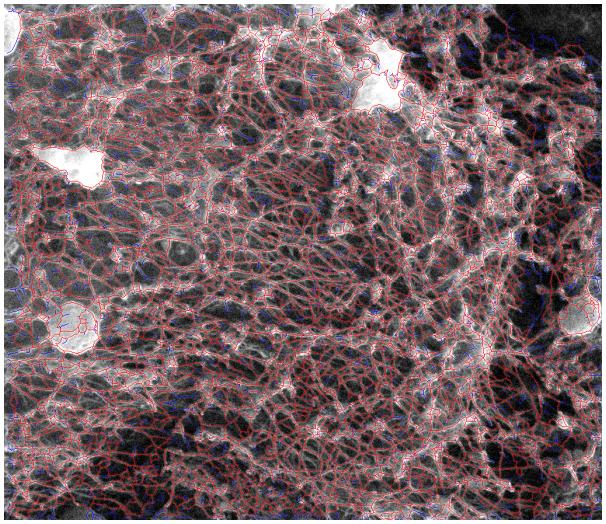
Earlier versions only differ in one important part, namely the way nodes spawn new nodes (See II E). The main difference here was, that instead of looking at the second order derivative of the image, the image brightness was used directly. Therefore the explanation given in **??** still applies for all other parts. The section II E is also still almost completely applicable, but instead of the *res* value that is computed for every processed pixel, the pixel brightness was used and there was no threshold applied to the local maxima of the $smooth(\alpha)$ function, instead all local maxima were used to spawn new nodes. In this development stage the tracing algorithm was only slightly modified and the effort was focused on preprocessing the image to increase fiber visibility before applying the tracing algorithm. The first idea that was pursued in the hopes of increasing fiber vis-

ibility was to Fourier transform the image, apply a mask to the result and then perform the inverse Fourier transform. In the Fourier transform of a image, the information about the features of the image can be thought of as being ordered by the feature size. Since all fiber in our SEM image have the same thickness, the hope was, that the information about the fibers was mostly stored in a small part of the Fourier transform. After some experimentation, the best working mask for the Fourier transform was determined to be a hollow oval outside of which all frequency components were set to zero. A oval was necessary to select the same frequency components in all directions of a non square image. In the case of a square image with a aspect ration of 1:1, the oval would become a circle. The center of the oval was chosen at a frequency of 0.5 per fiber diameter, corresponding to a period of twice the fiber diameter (fiber diameter is about 4 pixel). The thickness of this oval then selects the frequency range around this center, that was used to generate the processed image. The thickness of the oval in the frequency space was chosen around half its semi major axis. Subsequently a small gaussian filter with a standard deviation of 1 pixel was applied to the image to reduce the left over noise. Next a threshold was applied to the pixel brightness. The brightness of pixels lying below that threshold were raised to that threshold. In the final step of image preprocessing, the brightness of the pixels was rescaled, so that the brightest pixel had a value of 255 and the darkest one a value of 0. This in combination with the previous threshold, set every pixel below the threshold to zero, since the darkest pixels in the image were the ones set to the threshold. All these steps led to a resulting image with reduced the noise, also caused by setting the contributions of high frequencies to zero and lower frequency contribution were also removed. In total, fiber visibility was somewhat increased. One big problem was, that the centers of larger mesh holes became almost as bright as the centers of the fibers. The biggest problem during this phase of the development was however that the fiber tracing would very often connect two fibers that are close together and run parallel without being actually connected. This was very likely, because only the pixel brightness was considered which is not direction sensitive unlike the res value used in the current version. When two fibers run close and parallel to each other, and a node on one of them is processing pixels within r_{max} that are part of the second fiber, it will see the brightness of this second fiber and try to spawn a new node in this direction. This will cause a false connection between those two fibers. To try and circumvent this problem, a additional check was implemented. This check was executed every time a new node was supposed to be spawned. It looked at the image brightness profile along a line in the direction, the new node was to be spawned in. Here the thickness and the length of the considered line could be specified. If the smoothed brightness along this line fell below a threshold, the new node would not be spawned. A note written after the implementation of this additional check reads:"linefun seems to work pretty good, but did not bring the improvements I hoped for, but still some improvement." where "linefun" is the name of the check. The tracing results during this development stage are shown in Figure 15.

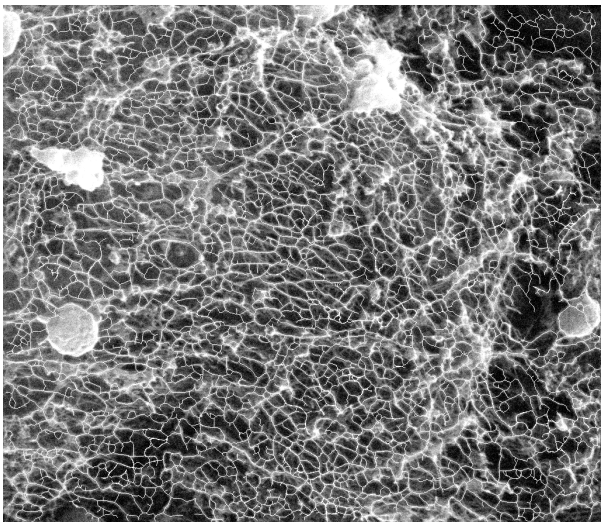
Subsequently the development efforts were again directed to enhancing the fiber visibility. This time a method inspired by and very similar to the one used by Kikinis et. al. [7] was implemented. This method maps every pixel to the smaller eigenvalue of the hessian



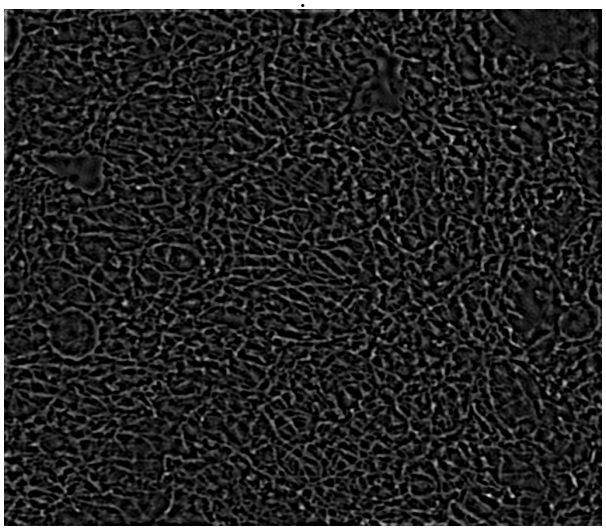
(a) SEM image of the actin cortex. Imaged by Daniel Flormann.



(b) Visualized tracing data from the final version of the tracing algorithm. Red lines are part of closed loops and blue lines are not. Parameters used are shown in table III



(c) Figure 15a with visualized tracing data from an older version of the tracing algorithm. Traced lines are drawn in white due to a older version of the visualization function.



(d) Image used in 15c for the actual tracing. It is the image from figure 15a after the preprocessing of the earlier versions of the tracing algorithm based on the Fourier transform.

Figure 15: Comparison between older and newer versions of the tracing algorithm.

matrix computed by the convolution with second order Gaussian derivatives (See figure 1). Subsequently the algorithm, still based on the pixel brightness instead of the res value was applied to the image produced this way. A considerable improvement was achieved by moving the thresholding from the image to the local maxima of the $smooth(\alpha)$ function.

Specifically, there was no longer any threshold applied during the preprocessing of the image, but instead only the local maxima of $smooth(\alpha)$ above a specifiable threshold were considered for new node positions, as is the case in the final version. From here on the step to using the hessian directly for tracing, as is done in the final version, was not very far.

A big difference between earlier and the final version not visible in Figure 15 is the amount of parameters and their usable range. In earlier versions a lot more parameters had to be chosen for the preprocessing of the image and other functions no longer in use. These parameters also had to be extremely fine tuned and deviations of 10% in one of them would be detrimental to the tracing quality. In the final version, some parameters can be changed almost by a factor of two and the tracing is usable in the whole range. This causes the search for usable parameters to be much faster and more pleasant. Also the same parameter set can be used for a wider range of images, making the results more comparable.

---

[1] V. Noireaux, R. M. Golsteyn, E. Friederich, J. Prost, C. Antony, D. Louvard, and C. Sykes. Growing an actin gel on spherical surfaces. *Biophys J*, 78(3):1643–1654, Mar 2000. ISSN 0006-3495. doi: 10.1016/S0006-3495(00)76716-6. URL `https://www.ncbi.nlm.nih.gov/pubmed/10692348`. 10692348[pmid].

[2] Nathan A. Hotaling, Kapil Bharti, Haydn Kriel, and Carl G. Simon. Diameterj: A validated open source nanofiber diameter measurement tool. *Biomaterials*, 61:327 – 338, 2015. ISSN 0142-9612. doi: https://doi.org/10.1016/j.biomaterials.2015.05.015. URL `http://www.sciencedirect.com/science/article/pii/S0142961215004652`.

[3] Curtis T. Rueden, Johannes Schindelin, Mark C. Hiner, Barry E. DeZonia, Alison E. Walter, Ellen T. Arena, and Kevin W. Eliceiri. Imagej2: Imagej for the next generation of scientific image data. *BMC Bioinformatics*, 18(1):529, Nov 2017. ISSN 1471-2105. doi: 10.1186/s12859-017-1934-z. URL `https://doi.org/10.1186/s12859-017-1934-z`.

[4] Nathan A. Hotaling, Kapil Bharti, Haydn Kriel, and Carl G. Jr Simon. Dataset for the validation and use of diameterj an open source nanofiber diameter measurement tool. *Data Brief*, 5:13–22, Jul 2015. ISSN 2352-3409. doi: 10.1016/j.dib.2015.07.012. URL `https://www.ncbi.nlm.nih.gov/pubmed/26380840`. 26380840[pmid].

[5] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[6] Johannes Schindelin, Ignacio Arganda-Carreras, Erwin Frise, Verena Kaynig, Mark Longair, Tobias Pietzsch, Stephan Preibisch, Curtis Rueden, Stephan Saalfeld, Benjamin Schmid, Jean-Yves Tinevez, Daniel James White, Volker Hartenstein, Kevin Eliceiri, Pavel Tomancak, and Albert Cardona. Fiji: an open-source platform for biological-image analysis. *Nature Methods*, 9:676 EP –, Jun 2012. URL `https://doi.org/10.1038/nmeth.2019`. Perspective.

[7] Yoshinobu Sato, Shin Nakajima, Nobuyuki Shiraga, Hideki Atsumi, Shigeyuki Yoshida, Thomas Koller, Guido Gerig, and Ron Kikinis. Three-dimensional multi-scale line filter for segmentation and visualization of curvilinear structures in medical images. *Medical Image Analysis*, 2(2):143–168, Jun

1998. ISSN 1361-8415. doi: 10.1016/S1361-8415(98)80009-1. URL https://doi.org/10.1016/S1361-8415(98)80009-1.

[8] Christoph Winkler, Marlene Vinzenz, J. Victor Small, and Christian Schmeiser. Actin filament tracking in electron tomograms of negatively stained lamellipodia using the localized radon transform. *Journal of Structural Biology*, 178(1):19 – 28, 2012. ISSN 1047-8477. doi: https://doi.org/10.1016/j.jsb.2012.02.011. URL http://www.sciencedirect.com/science/article/pii/S1047847712000640.