PIMA REPORT :

Enhancing filters for medical application

Charline Curaut[1] 3810013,
Supervisor : Garance Martin

February to May 2023

[1]A link to the project's github

I. Introduction

This project is a subpart of a larger project in collaboration between the LIP6 laboratory, ISIR and Saint-Antoine Hospital. Its main goal is to improve a medical intervention called ERCP (standing for Endoscopic Retrograde Cholangio-Pancreatography). This intervention aims to drain the bile ducts of a patient. It is an endoscopic procedure meaning that instruments are inserted in the body via natural ways. Thus, it is done with an endoscope that the endoscopist controls and guides through the patient's body. He/She can see what he/she is doing via 2D X-rays radiography images that are created by injecting a contrast agent in the patient's bile ducts that is detected by casting X-rays on it. The resulting images are called cholangiograms. That is what we are going to use for this project. Sixteen of those images will be used and stored in our database to compare the efficiency of the algorithms further developed in this project.

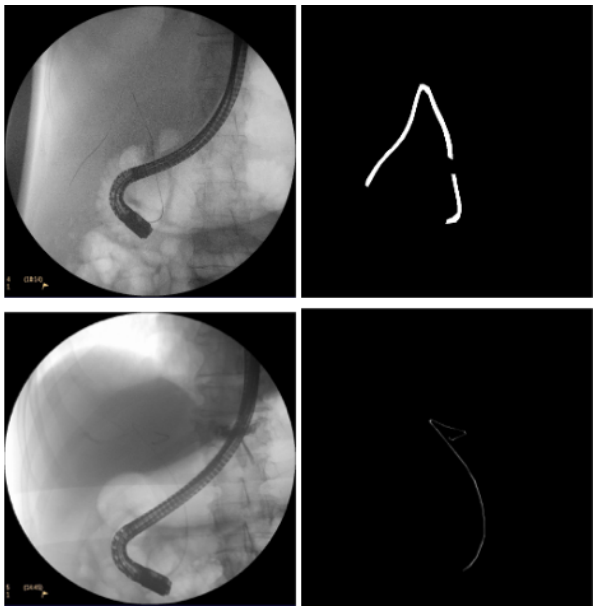

Figure 1: Two examples of cholangiograms (on the left) and their respective annotation of the guidewire (on the right)

Unfortunately, in cholangiograms, the guidewire is really thin and low contrasted. It makes it very hard to be detectable on the obtained images.

In order to better see this object, we need to enhance it on the images. For that, we could cast more intense X-rays on the patients but too much of it could be dangerous for them. Instead, there is another way to do it that will not be of any danger for the patient. This method is the use of vesselness filters. They are special filters capable of enhancing long and thin elements on an image and reducing the global noise of the image at the same time. Rather used as a pre-processing step in the segmentation of vessels to enhance contrast in angiography images, this project suggests applying these filters on our data images. There are several approaches on vesselness filters but the most populars rely on differential information, in particular on second derivatives as they are called Hessian based methods. Seven of them are presented and compared in the article[LMK+21] attached to this project. The main difficulty for us is that

they are all applied to 3D images of liver. For each one of them the principal information is given, especially the vesselness functions whose values have to be computed for each pixel of the image. These values depend directly or indirectly on the eigenvalues of the Hessian matrix. But in the given article, they are all written for 3D filters.

So our goal is to choose two of those filters and to implement them for 2D images.

After that, we will evaluate them individually to decide if they are efficient or if they are fast to compute. Then we will compare them between one another to see if one is faster, gives better or different results or both. After analyzing the results of the article[LMK+21], we chose to implement the Frangi and the Jerman filters. The first one is described as one of the two pioneers among vesselness filters and it is using the eigenvalues in a scale-space framework. The second one is a more advanced version than the Frangi filter and it includes a volume dimension. It was created to overcome the non-uniform response that we had for different diameter vessels. It adds a volume ratio of tubular structures in the vesselness function. In their article they use the MCC (Matthews Correlation Coefficient) and Dice metrics computed with the results of a confusion matrix to measure the efficiency and to compare the filters between one another. The results of these two metrics are respectively between -1 and 1 and between 0 and 1. The closer to 1, the better the results. They use a database different from ours but both of them seem to have great results even though we can also cite Zhang filter and RORPO which is a particular filter among the seven because it does not rely on differential information and is not linear. For that reason we decided not to choose this filter in order to compare two Hessian-based filters and especially a basic one such as Frangi that we could compare with a more advanced one to see if the changes are really that useful. In the paper, they use a third metric to compare the filters. It is the ROC curve which depicts the true positive rate vs the false positive rate. In their results, the Zhang filter seems to have a true positive rate more important than the others for a same false positive rate. So we chose the Jerman filter as it is neither better nor worse than the others.

II. State-of-the-art

The first article[LMK+21] explained that the most popular approach in vessel filtering is the use of second derivatives especially via the Hessian matrix. The first ones to try it were Sato et al.[SNS+98] and Frangi et al.[FNVV00]. The Sato filter uses directly the eigenvalues of the matrix in his function while the Frangi filter uses them to compute some measures for the blobness, the tubular structures and the norm of the matrix to compute the vesselness function. Then others tried to improve their technique such as Meijering et al.[MJS+04] or Jerman et al.[JPL6] that developed the method respectively for the special field of fluorescence microscopy because the function detects even more elongated structures and to generalize the response for any dimension of the vessels with a volume ratio. Zhang et al.[ZZL+18] tried to improve that of Jerman et al.[JPL6] for the vesselness function to have a better affinity to blurred images for Computed Tomography (CT) data in the context of hepatic vascular networks.

But Hessian-based filters are not the only technique for vessel filtering. Law and

Chung[LC08] proposed the Optimally Oriented Flux (OOF) technique. It is an optimization framework based on image gradient flux. From that, they created another matrix than the Hessian and the associated eigenvalues could still be used in the previous vesselness functions. And the last filter that the article detailed is Ranking the Orientation Responses of Path Operators (RORPO) proposed by Merveille et al.[MTNP18] who didn't use any differential information. Unlike the others this filter is semi-global and non-linear. It uses the mathematical notion of path opening.

Other non Hessian-based filters are mentioned. For instance, there is that of Sazak et al. who used openings like RORPO but with top-hat tensors and structuring elements. Or there are filters based on phase tensors or related to wavelets.

In our project, all the non differential-based filters are too complicated to implement and they are not even detailed in the paper, except for the RORPO filter. Among the rest, the OOF filter is the only one using a different matrix than Hessian to compute the differential information. But if the base is different, the rest of the method is the same as the others. So in order to compare the filters, it would be best to first compare their vesselness function using the same base i.e. the Hessian matrix. Even though it would be interesting after that to use the vesselness functions tested with this other differential matrix to compare the result and then see if this base is better or not.

III. Materials and methods

For this project, we used a total of sixteen images from Saint-Antoine Hospital to test the filters. They are called cholangiograms which is a 2D X-ray radiograph of a little part of the intestine, around a zone called duodenum. That's where the intervention is done inside the patient. The resulting images of the radiography are here to help the endoscopist aim for the right place. But X-ray radiographs are only made of shades of gray and they are usually low-contrasted. It can be a bit difficult for human eyes to distinguish anything as the edges are not always sharp.

In our case, we focus on a little wire in the image that is an important element of the operation. It is so thin that it is hardly seeable on certain cholangiograms. Fortunately for us, the Hospital provides us with sixteen other images that are the references of the first ones. These images are totally black except for a white line drawn by hand to localize the wire. With those two colors images, we can easily compute the percentage of guidewire in the images by counting the number of white pixels on each reference and dividing it by the total number of pixels of the corresponding image. On average, the wire represents about 0.09% of the images with a standard deviation of 0.03%.

All the sixteen images are not exactly of the same size but it is globally around 1000*1000 pixels. But each reference has the same size as the associated image. They were all compressed to a jpeg format which means that there was a loss of information, invisible or almost for the human eyes, especially on the edges.

In order to increase the contrast of the wire in the sixteen images of our given database, we will implement and use two specific vesselness filters : the Frangi filter and the Jerman filter. Both of them are based on differential information stored in the Hessian matrix of each image. By definition, the Hessian matrix of a function $f(x1, x2)$

is defined, at each point, as :

$$H(f) = \begin{bmatrix} h_{11} & h_{12} \\ h_{21} & h_{22} \end{bmatrix} = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} \end{bmatrix} \tag{1}$$

But actually the f function requires to be continuous and twice differentiable. Thus, our images being digital ones, they are neither one nor the other. That is why, to compute their Hessian matrix, we first have to define f from each image I by applying a Gaussian kernel of standard deviation σ to it. So our Hessian matrix is actually defined, at each pixel, as :

$$H_I = g_\sigma \star \begin{pmatrix} \frac{\partial^2 I}{\partial x^2} & \frac{\partial^2 I}{\partial x \partial y} \\ \frac{\partial^2 I}{\partial y \partial x} & \frac{\partial^2 I}{\partial y^2} \end{pmatrix} = g_\sigma \star \begin{pmatrix} I_{xx} & I_{xy} \\ I_{xy} & I_{yy} \end{pmatrix} \tag{2}$$

Plus, by doing this operation, we also set up a multiscale framework that is especially useful for vessels images because then the scale of the vessels of interest depends on σ[LMK+21]. But the wire that interests us has only one dimension so it won't be necessary to compute multiple scales for this research. Instead it will be more about finding the right σ to correctly enhance the wire without enhancing too much background noise with it.

Both of our filters use the eigenvalues of this matrix noted λ_1 and λ_2 such that $|\lambda_1| < |\lambda_2|$. If everything goes well, a pixel of a vessel of interest (the wire in our case) is signaled by λ_1 being small (ideally zero), and λ_2 of a large magnitude. The sign of λ_2 is supposed to be an indicator of brightness/darkness. Because their values can represent the different structures of an image, they allow us to compute other measures and to treat the different cases of each filter's response[LMK+21].

The Frangi filter[FNVV00] is making the hypothesis that only relying directly on these two eigenvalues can be discriminant. Instead the eigenvalues are actually used to compute two other measures. The first one is a geometric ratio that accounts for the deviation from a blob-like structures in 2D images :

$$R_b = \lambda_1 / \lambda_2 \tag{3}$$

This ratio attains its maximum for blob-like structure.

The second measure is computed based on the interesting fact that background pixels have a small magnitude of their derivative and so their eigenvalues. Thus, this measure is the norm of the Hessian matrix :

$$S = \|H\|_F = \sqrt{\sum_{j \leqslant D} \lambda_j^2} \tag{4}$$

Where D is the dimension of the image. In our project, we only treat 2D images so we will take $D = 2$. This measure is supposed to be low in the background where the eigenvalues are not detecting any structures and so are small. And because this measure increases whenever one of the λ is getting higher, it will become larger for any region of contrast. The vesselness function of the filter is then a combination of the computed measures :

$$V_\rho(s) = \begin{cases} 0 & \text{if } \lambda_2 < 0 \\ exp(-\frac{R_b^2}{2\beta^2})(1 - exp(-\frac{S^2}{2\gamma^2})) & \text{otherwise.} \end{cases} \quad (5)$$

The conditions of this function are specifically specified for dark curvilinear structures which is precisely the wire nature in our images. For bright objects on a dark background, the condition should be reversed. The two parameters β and γ are thresholds which control the impact of each measure on the filter function.

The Jerman filter[JPL6] does not change the conditions of its vesselness function depending on the image's colors (bright structures on dark background or the contrary) and directly redefine each eigenvalue λ_i; $i = 1, 2$ instead :

$$\lambda_i := \begin{cases} -\lambda_i & \text{bright structures on dark background} \\ \lambda_i & \text{dark structures on bright background.} \end{cases} \quad (6)$$

This filter was suggested to be more robust to bifurcations. To make it less susceptible to noise in image regions of uniform intensity, a third λ is introduced and set to λ_2. Initially, the vesselness filters were created for 3D images and thus manipulated three eigenvalues. In our project, we can't compute the third one and most of the measures and vesselness function can be remodeled for 2D images. Unfortunately, one of the forces of the Jerman filter is to use the differences between the two λ_2 and λ_3. To make the filter less susceptible to noise in image regions of uniform intensity, the third eigenvalue was supposed to be redefined. We will simply do the same thing with the introduced variable $\lambda_3 = \lambda_2$ which is not a big assumption because their values are supposed to be close to one another in the 3D approach. The regularized eigenvalue λ_p is so defined, at each scale s, as :

$$\lambda_p(s) = \begin{cases} \lambda_3 & \text{if } \lambda_3 > \tau \max_x \lambda_3(x, s) \\ \tau \max_x \lambda_3(x, s) & \text{if } 0 < \lambda_3 \leqslant \tau \max_x \lambda_3(x, s) \\ 0 & \text{otherwise.} \end{cases} \quad (7)$$

where τ is a cutoff threshold between zero and one.

With that regularized eigenvalue, we obtain the following vesselness function for the Jerman filter applied on 2D images :

$$F = \begin{cases} 0 & \lambda_2 \leqslant 0 \text{ or } \lambda_p \leqslant 0 \\ 1 & \lambda_2 \geqslant \lambda_p/2 > 0 \\ \lambda_2^2(\lambda_p - \lambda_2)(\frac{3}{\lambda_2 + \lambda_p})^3 & \text{otherwise.} \end{cases} \quad (8)$$

In order to compare both filters, I used two similarity metrics introduced in the article related to my project [1] : the Dice and the Matthews Correlation Coefficient (MCC). They are based on the confusion matrix computed between the thresholded results of our filters and the given annotation of each image. To recall, this matrix gives the numbers of true positives (tp), true negatives (tn), false positive (fp) and false negatives (fn). The Dice and the MCC coefficients are respectively defined as :

$$Dice = \frac{2tp}{fp + fn + 2tp} \quad (9)$$

$$MCC = \frac{tp \cdot tn - fp \cdot fn}{\sqrt{(tp + fp)(tp + fn)(tn + fp)(tn + fn)}} \tag{10}$$

Dice score lies in [0,1] whereas MCC lies in [-1,1]; the closer to 1 the better the results.

IV. Experimental part and results

A. The Hessian matrix

Because both filters are based on the Hessian matrix, it was important to make sure it was well computed. I first tried to apply a Gaussian kernel to the image and then I convolved the result with derivative filters. So I wrote a function that creates a Gaussian kernel with the sigma s parameter I give it, using the formula of the filter :

$$g_\sigma(i, j) = \frac{1}{2\pi\sigma^2} e^{-\frac{i^2+j^2}{2\sigma^2}} \tag{11}$$

Then I used the Sobel kernels to compute the second derivatives :

X-Direction Kernel

-1	0	1
-2	0	2
-1	0	1

Y-Direction Kernel

-1	-2	-1
0	0	0
1	2	1

I tried using sigma $s = 3$, but there were some uniform parts in the image where the Hessian matrix gave higher results than expected. For instance, the images are all circled by a black zone where the Hessian matrix should be zero since there are no edges. Therefore, the beginning and the ending of this matrix should be around zero. That was the case for the beginning but not for the ending that had values that could go between -30 to 15. I found it odd so I tried to compute the Hessian matrix with the same value of sigma and another derivative kernel, the simplest one :

X-Direction Kernel

0	0	0
-1	0	1
0	0	0

Y-Direction Kernel

0	-1	0
0	0	0
0	1	0

This time, the beginning of the matrix was still zero and the ending was between -2 and 1. That was better but still I couldn't understand why it was not zero. After some research, I tried a last solution where I didn't directly use derivative kernels. Instead, I computed the second derivatives of the function of the Gaussian kernel :

$$\frac{\partial g_\sigma}{\partial x \partial x} = (\frac{1}{2\pi\sigma^4})(\frac{x^2}{\sigma^2} - 1)exp(-\frac{x^2 + y^2}{2\sigma^2}) \tag{12}$$

$$\frac{\partial g_\sigma}{\partial y \partial y} = (\frac{1}{2\pi\sigma^4})(\frac{y^2}{\sigma^2} - 1)exp(-\frac{x^2 + y^2}{2\sigma^2}) \tag{13}$$

$$\frac{\partial g_\sigma}{\partial x \partial y} = (\frac{xy}{2\pi\sigma^6})exp(-\frac{x^2+y^2}{2\sigma^2}) \tag{14}$$

before convolving it with the image. With still the same sigma value, the results were better as the ending of the matrix was approximately between -0,1 and 0.

A short study was done to compare the three Hessian matrices to decide which one to use for the vesselness filters. From now on, we will name them $H1$, $H2$, and $H3$ respectively to their order of creation. Their range of values were really different between one another. That's why I normalized them between 0,0 and 1,0, so I could compare them more easily. After that, I searched for the one that had the greatest results for the edges that were then supposed to be represented by both the lowest and the highest values in each matrix : 0,0 and 1,0.

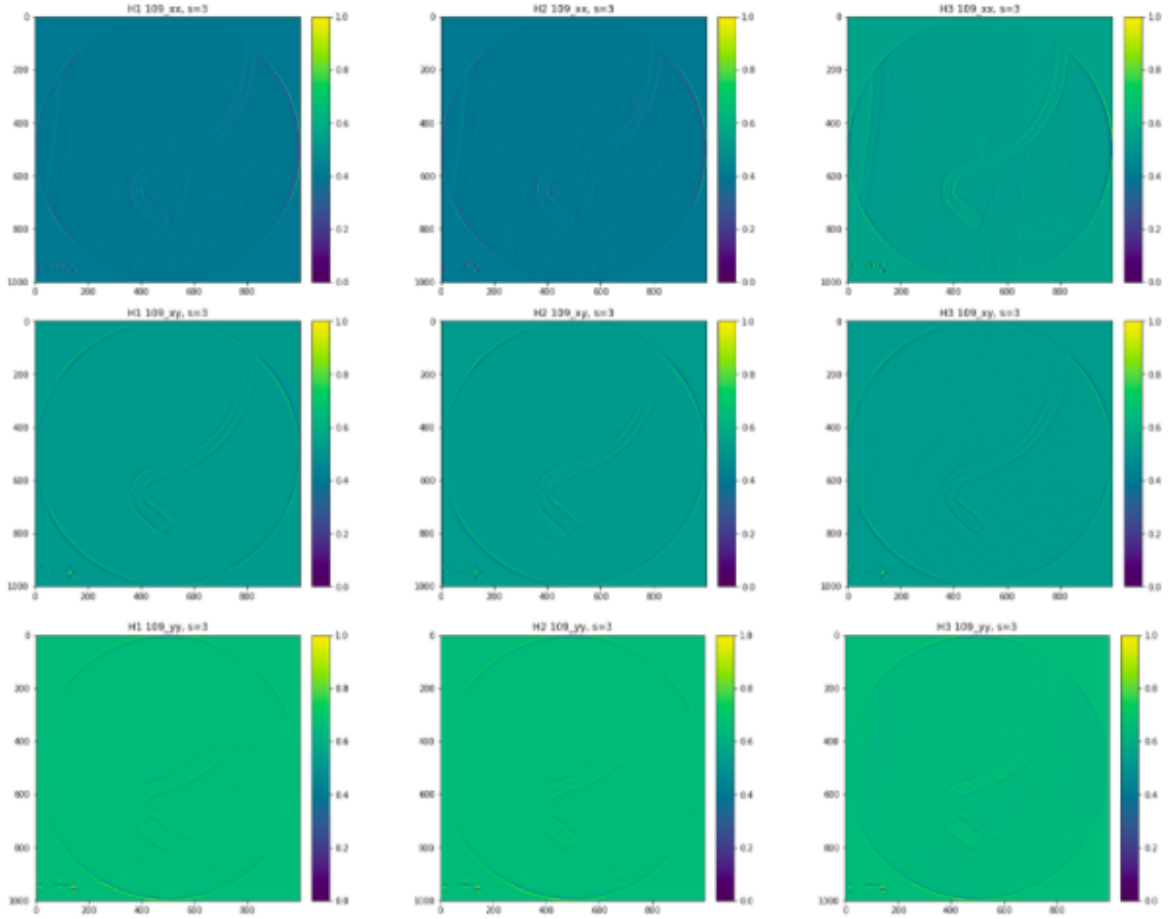

Figure 2: The three second derivatives of the three normalized Hessian matrices

From those results and judging by the scale and the images colors, the three matrices were then binarized to put everything between 0,32 and 0,68 to zero and keeping the matrices values for the rest.

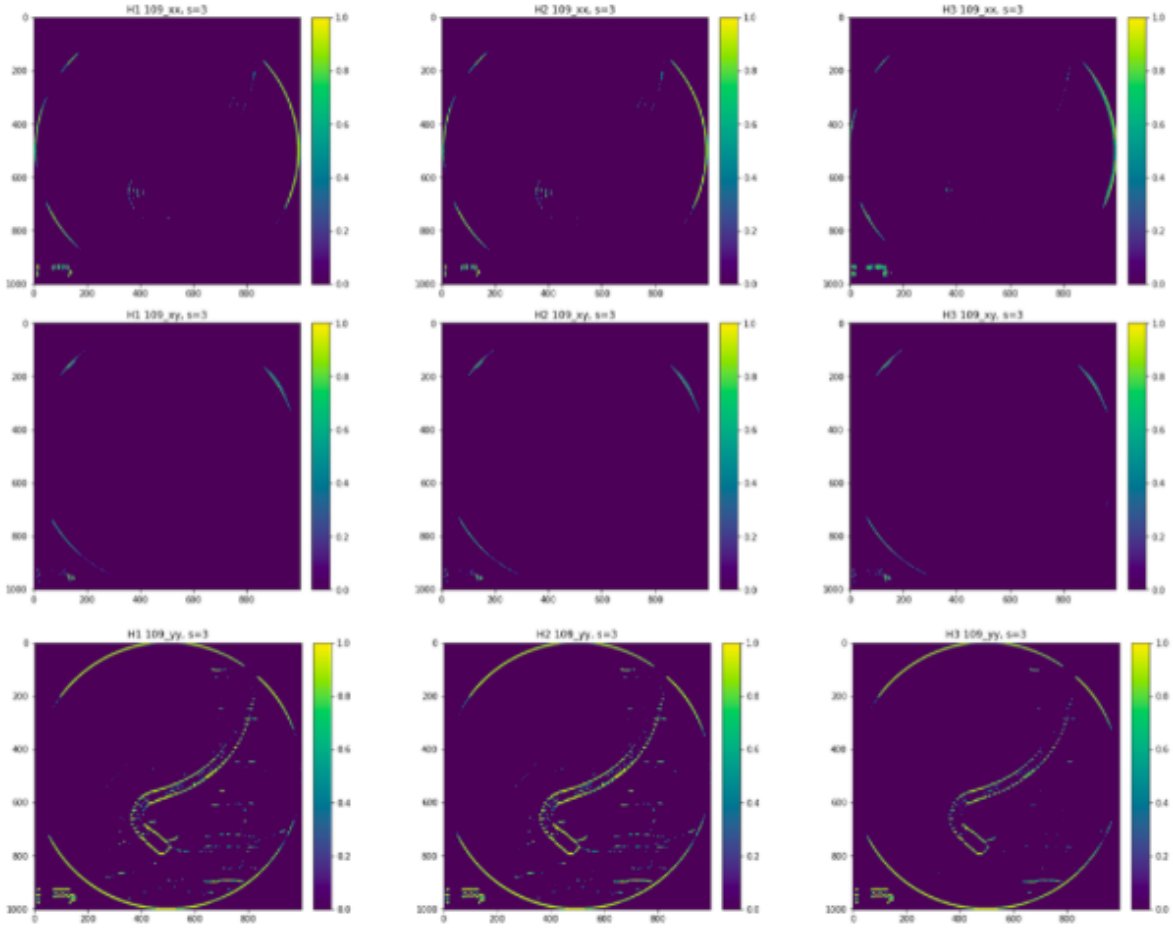

Figure 3: The three second derivatives of the three normalized and binarized Hessian matrices

Finally, it was more interesting to compute the differences between those results, it was also easier to analyze them. So I compared the filters two by two by doing a simple subtraction. I subtracted the normalized and binarized $H1$ from the normalized and binarized $H2$ (same thing for $H1$ and $H3$, and for $H2$ and $H3$).

When we observe the results, we must keep in mind that the scale goes from -1.0 to 1.0 which corresponds to a color scale from blue dark to yellow. Where the image is green, the difference between the two filters is next to zero : that means that both filters are equal. Where it is yellow, the first filter (the first way of computing the Hessian matrix) is better than the one we subtracted from it (the second way). And when it is blue dark, it is the opposite : the second filter is better.

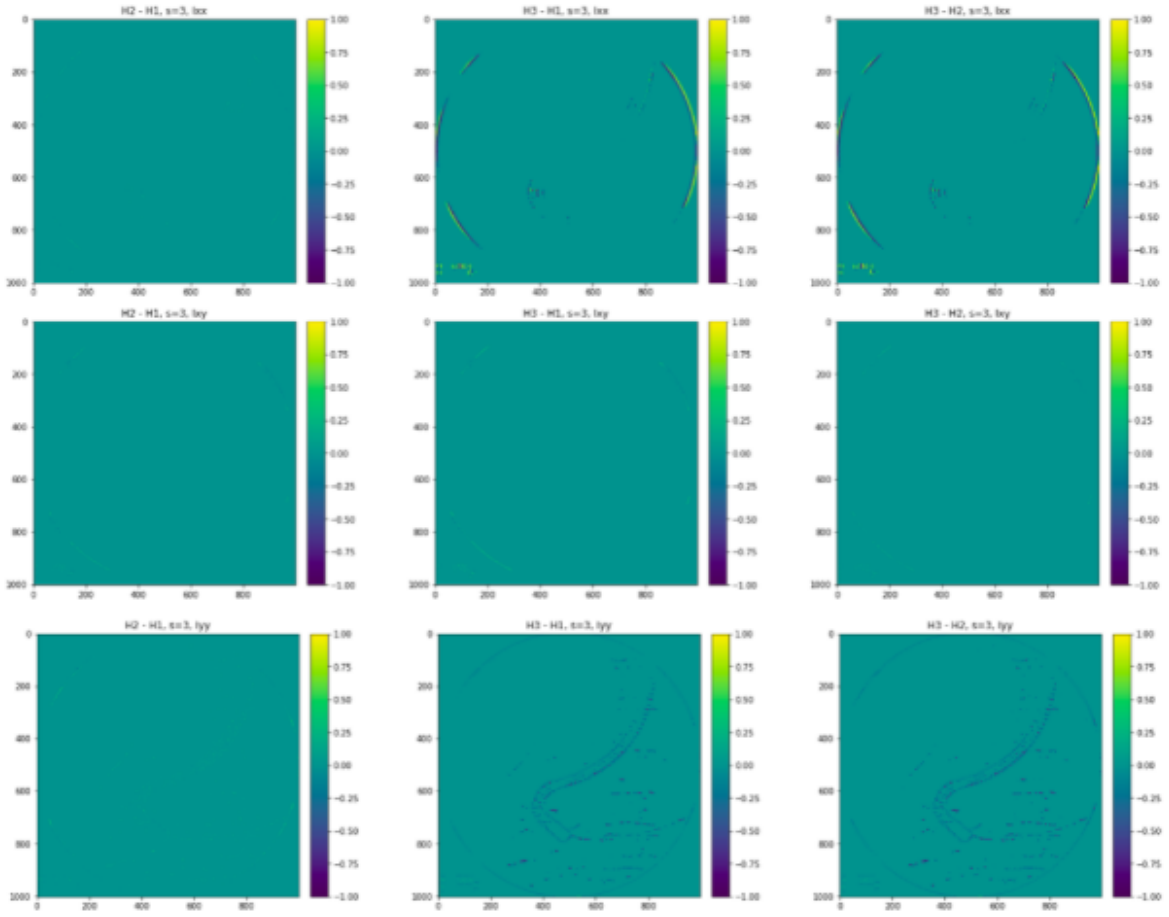

Figure 4: The three possible differences between the three normalized and binarized Hessian matrices for the three second derivatives

Here, we can see that there is not much difference between $H1$ and $H2$. Their subtractions are all green for the three possible second derivatives. However, we can see some yellow points here and there on these subtractions so we could say that between these two, $H2$ is slightly more efficient than $H1$.

When we observe both subtractions from $H3$, we can see that, for the xy-derivative, there are some edges that $H3$ detects more efficiently than the two others. And for the xx-derivative, we can also notice that, for the same edges, there are some values more detected by $H3$ and some more detected by the two others : they are all equivalent for that derivative. Most importantly, there is a difference for the yy-derivative. There is a lot of blue dark on the image : part of it is located near the guidewire but the rest is here because of the noise. That means that both $H1$ and $H2$ are more effective to detect edges than $H3$ but they are also more affected by the noise.

In this project, there is a lot of noise in the images and the final version $H3$ of the Hessian matrix is less affected by it while still capable of detecting the wire. For that reason, l chose to use this version for the rest of the project.

B. The eigenvalues

The first thing to do when building the vesselness filters is to compute their eigenvalues. They both come from the same Hessian matrix but the Jerman filter needs to redefine them according to the images it analyzes. By definition, an eigenvalue λ of a matrix H is defined by the equation : $HX = \lambda X$, where X is an eigenvector and $X \neq 0$. Thus we have : $HX - X = 0$, that becomes : $(H - \lambda I)X = 0$. From this equation and because we set $X \neq 0$, the matrix $(H - \lambda I)$ is not invertible. Therefore, we have :

$$\begin{aligned}
\Delta(H - \lambda I) = 0 &\Leftrightarrow \Delta \begin{bmatrix} I_{xx} - \lambda & I_{xy} \\ I_{xy} & I_{yy} - \lambda \end{bmatrix} = 0 \\
&\Leftrightarrow (I_{xx} - \lambda)(I_{yy} - \lambda) - I_{xy}^2 = 0 \\
&\Leftrightarrow I_{xx}I_{yy} - I_{xx}\lambda - I_{yy}\lambda + \lambda^2 - I_{xy}^2 = 0 \\
&\Leftrightarrow \lambda^2 - (I_{xx} + I_{yy})\lambda + I_{xx}I_{yy} - I_{xy}^2 = 0 \\
&\Leftrightarrow \lambda^2 - trace(H)\lambda + \Delta(H) = 0
\end{aligned}$$

We obtain the characteristic polynomial of H that we can resolve using the determinant : $\Delta(\lambda^2 - trace(H)\lambda + \Delta(H)) = trace(H)^2 - 4 * 1 * \Delta(H)$. Then the solutions are : $\lambda_1 = \frac{trace(H)+\sqrt{\Delta}}{2}$ and $\lambda_2 = \frac{trace(H)-\sqrt{\Delta}}{2}$. This determinant is supposed to always be superior to zero but for some instances, it could be under zero. That was due to the float approximation (I checked the minimum value obtained and it was really close to zero). That is why I put the λs to zero every time the determinant was negative. The results were put in a 3D matrix with the same dimension of the image and a depth of 2, one for each eigenvalue.

C. The Frangi filter

With the eigenvalues, I computed the two measures R_b and S needed for the Frangi vesselness function and stored them in a matrix the same way I did to store the eigenvalues. Then I computed the value of the vesselness function for each pixel of the image. I added a parameter to tell if the searched vessels are bright or dark on a contrasted background (the default value was set according to our database). Another parameter was added to let the user choose or not the value of the function's parameter γ. By default, the value of γ is computed inside the function as specified in the article[FNVV00] : $\gamma = \max(S)/2$. This value is supposed to work in most cases. For the β parameter, I set the default value to 0.5, as it was done for the existing Frangi filter in the Python library skimage.filters .

To see how well my version of the filter worked, I used the image 109.png in the database and applied both filters on it for the same scale $s = 3$ arbitrarily chosen.

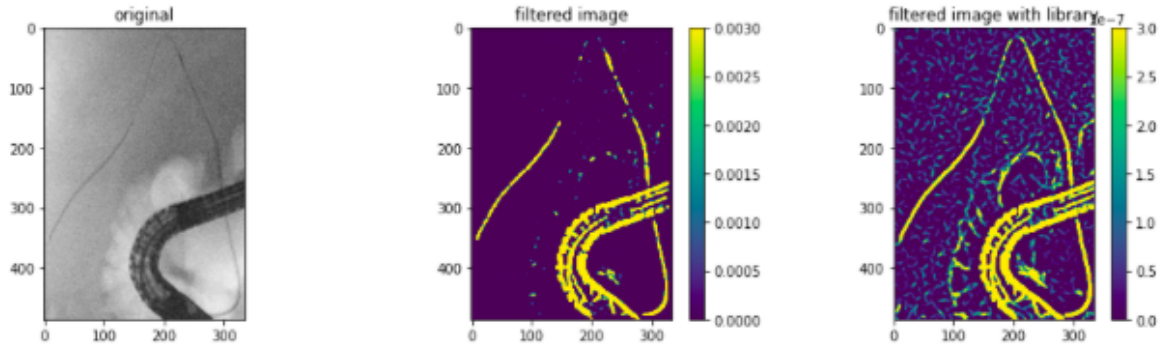

Figure 5: Comparison between the result of our version of the filter and the existing one of the Python library

Modification on the scale color was necessary to better see the results because the values of the catheter's pixels are too high and it hides those of the wire. But the results are pretty much alike except for a 10e-4 factor between them. The existing Frangi filter gives a smoother result so we can see all the wire even if there is a lot of noise. Our implementation of the filter is way less affected by the noise but we can't see the full wire.

As explained before, in our case, we don't need the multiscale function, but it is important to find the best scale for our results. I tested a range of scales from 1 to 4 with a step of 0.5 on the same image to have an idea of the ideal value.

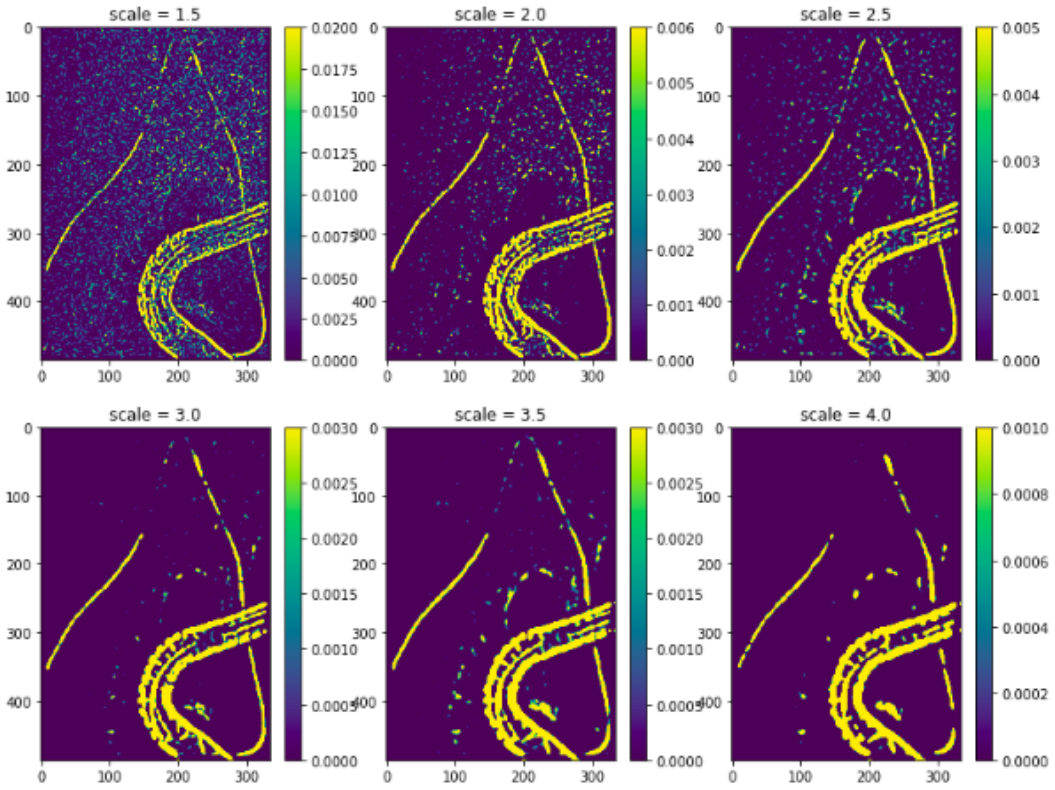

Figure 6: Results of the Frangi filter for different scales on the same image 109.png

It is obvious that the scale $s = 1.5$ is too thin. The wire is lost with the noise surrounding it. For the scales $s = 3.5$ and $s = 4$, the wire is more incomplete than with the others and the noise has higher value than with the other : it will be more difficult to threshold the image. Finally both scales $s = 2.0$ and $s = 2.5$ are more affected by noise than $s = 3.0$. But $s = 2.5$ gives a slightly better result than $s = 2.0$ (the wire is more continuous). If we correctly threshold the filtered images we can compare $s = 2.5$ to $s = 3.0$. A better way to compare the results for each scale would be to use the MCC coefficient as done in the article linked to the project[LMK+21]. I used two ways of thresholding. The first one is a simple threshold where every pixel's value of the image below a specific parameter is put to zero and the rest is kept as it is. The second one is a mean-C threshold detailed in the article[DVK+21] that applies a simple threshold on the image convolved with a mean kernel.

I iterated the scale values through an array from 2.3 and 3.3 and for each scale I iterated the threshold values through an array. For each value, I calculated the MCC to find the best threshold value for a given scale by keeping the best MCC score.

scale	(MCC for simple threshold, threshold value)	(MCC for mean-C threshold, threshold value)
2.3	(0.12057468826019557, 0.004)	(0.157945079170754, 0.02)
2.4	(0.13226435995628366, 0.008)	(0.1638282726381286, 0.04)
2.5	(0.13219892030891559, 0.005)	(0.16615822347167372, 0.03)
2.6	(0.1277939194715094, 0.003)	(0.16649093673178547, 0.01)
2.7	(0.1399526300578841, 0.006)	(0.1693552782487215, 0.03)
2.8	(0.13859760972608928, 0.004)	(0.17067279309896502, 0.02)
2.9	(0.13482604485739635, 0.002)	(0.1726790656221511, 0.01)
3.0	(0.11937292905182098, 0.001)	(0.16769043694498892, 0.0)
3.1	(0.14444988150998245, 0.002)	(0.1791434388327074, 0.01)
3.2	(0.13927750972856628, 0.001)	(0.17084092053558877, 0.01)
3.3	(0.12185215766494488, 0.001)	(0.17695707749056755, 0.0)

Figure 7: MCC scores for the best threshold found for each tested scale on the image 109.png

First, we can notice that the MCC score is always higher using the mean-C threshold rather than the simple one. Another thing to underline is the fact that the threshold values are really low for both techniques. When comparing both MCC values for the 3.0 scale, the threshold values are both zero (or almost) but the MCC is higher with the second technique. That is because of the mean kernel. Thresholding is not the only way to improve a filtered image. Here, the best MCC score is about 0.180 for a scale of 3.1 (and no threshold). It is also for this scale that the MCC has the best value with the simple threshold. But we can't use one image to choose the scale, so I tried this test on another image (108.png).

scale	(MCC for simple threshold, threshold value)	(MCC for mean-C threshold, threshold value)
2.3	(0.12013728199939752, 0.013)	(0.1568871510436948, 0.2)
2.4	(0.1332264147614699, 0.012)	(0.1607768204331488, 0.05)
2.5	(0.13173808779043972, 0.013)	(0.1594487026586633, 0.15)
2.6	(0.12882357808153255, 0.014)	(0.1564960792060317, 0.15)
2.7	(0.14342714853340668, 0.009)	(0.16814690399759918, 0.05)
2.8	(0.13945985987974707, 0.011)	(0.16499275059416824, 0.05)
2.9	(0.136417445915871, 0.013)	(0.15961129018106013, 0.05)
3.0	(0.13313256694022363, 0.014)	(0.15712573546771832, 0.15)
3.1	(0.1451836676066374, 0.006)	(0.17376165212362538, 0.05)
3.2	(0.13975567007623438, 0.014)	(0.1652931373138184, 0.05)
3.3	(0.13505764629716668, 0.013)	(0.15615698658905264, 0.05)

Figure 8: MCC scores for the best threshold found for each tested scale on the image 108.png

We can notice the same thing as before about the comparison between the two ways of thresholding. This time, the best MCC value is about 0.174 for a scale value of 3.1. From this point on, I chose to compute the Frangi filter with a scale $s = 3.1$. I tested it for every other image. Here are interesting results I obtained for some of the sixteen images :

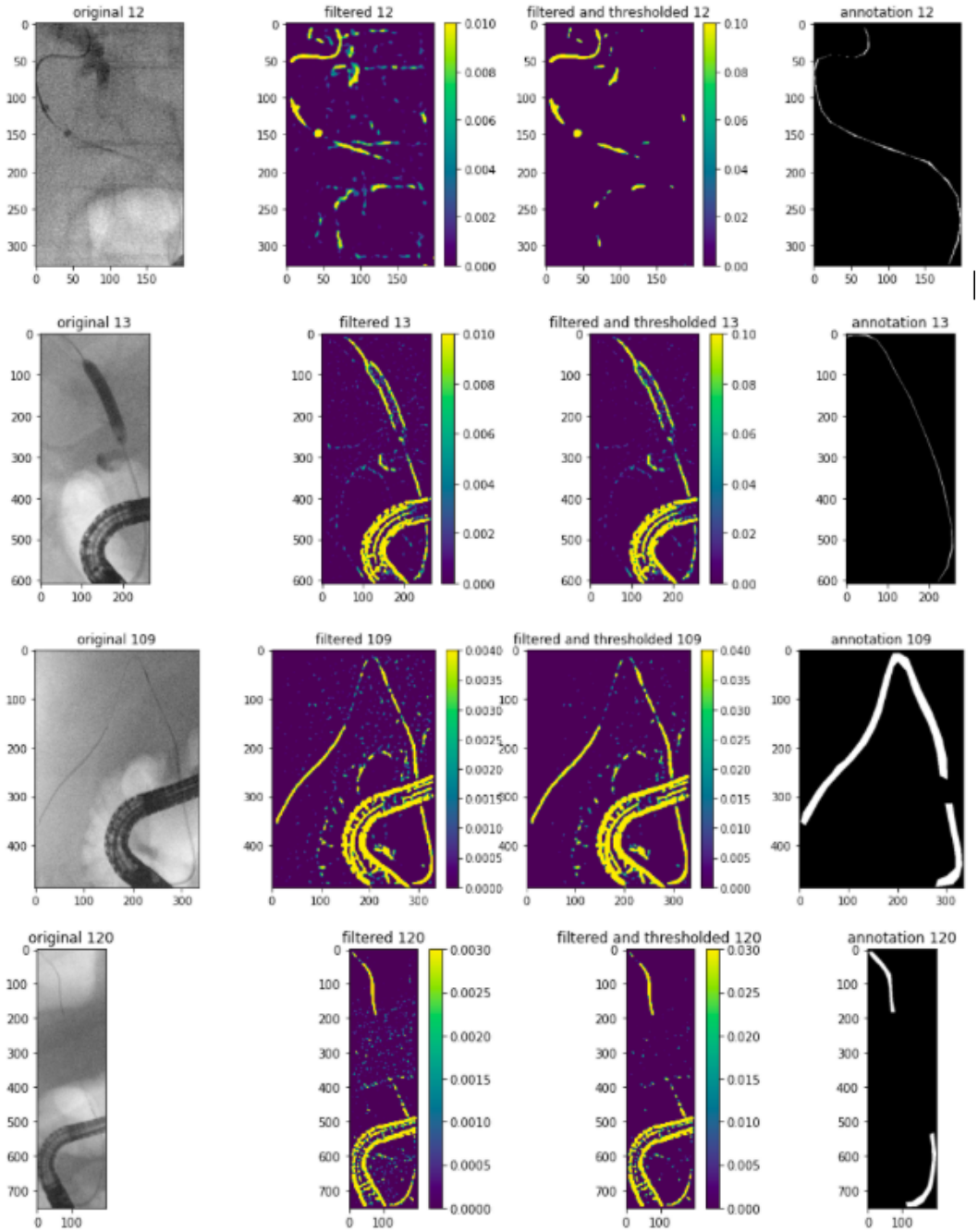

Figure 9: The original image of the database on the left, the Frangi filtered image on the middle left, the thresholded version of the filtered image on the middle right, and the annotation given on the right

I restrained the image to the zone where the wire is because the black circle surrounding the image is too sharp and the values of the vesselness function are too high for those edges. The first inconvenience is that this is not the wire we are looking for and the second is that the values are so high in comparison of those for the wire, they hide it on the result. Restraining the zone of action of the Frangi filter allows us to get less noise in the result. From that, it is actually quite easy to reconstruct an image with the same dimension as the original by putting every other pixel to zero (they are not in the wire zone so they are not interesting for this experiment).

About the results, in comparison to the annotations, the filter works pretty well. There are some images where the wire thickness is equal to that of another structure in the background such as in the image 12.png where the filter can only detect the end of the wire. The noise has almost completely disappeared but in real life, especially in our medical context, this image would be a hard to use. For the image 13.png, the filter has well detected the wire and also detected the bile ducts edges where the wire was lost in it. The two else images are here to show how the effectiveness of the thresholding step. The image 109.png has lost a lot of noise but the less-contrasted part of the wire is still a bit seeable on the image. About the image 120, it is quite the contrary : the less-contrated part is completely lost with the noise but the result is actually what we were expecting (because the annotation doesn't count that part either).

D. The Jerman filter

Just like I did for the Frangi filter, I used the eigenvalues to compute the $_p$ measure I needed to introduce. Then I computed the value of the Jerman vesselness function for each pixel of the images. This time, I couldn't find a default value for the intrinsic parameter τ of this filter, neither in any article nor in the Python library skimage as this filter has not been implemented yet. A special experiment must be conducted to optimize it. In the article attached to the project[LMK$^+$21], it is explained that the authors first tried to set the scale-space parameters before trying to optimize the intrinsic parameters of any filters. We will follow their example by trying to find the best scale for the Jerman filter with a default value of $\tau = 1$ arbitrarily chosen. As explained for the Frangi filter, the multiscale thing is again not really useful in our case. Just like I did for the Frangi filter, I tried to analyze the filtered images restrained to the wire zone with a range of scales from 1.5 to 4.0. Here are my results.

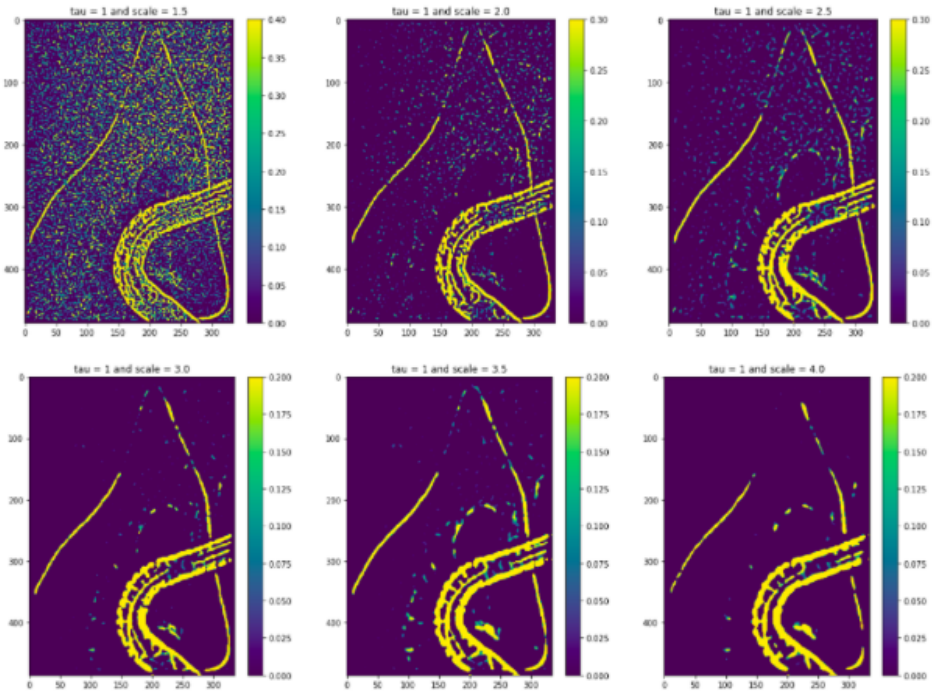

Figure 10: Results of the Jerman filter for different scales on the same image 109.png

Unlike before, it is hard for a human to decide which one of these results is the best. The scale 1.5 gives obviously too much noise. The difference between the scales 2.0 and 2.5 is thin but we can see that the wire seems thicker on the second image than on the first one, and it has less noise. When we compare the last third ones, it is clear that the third one has less noise but also less wire. We can't detect the less contrasted part of the wire which is a problem because some of the images have a wire that is even lower-contrasted than the one tested here. That is why it was necessary to use the MCC coefficient to find the best scale for this image. I iterated through an array of scales between 2.5 and 3.5 and for each scale, I tried a range of thresholds to obtain the best MCC score possible for an image for a given scale. Once again, I used two ways of thresholding, the simple and the mean-C ones. Another threshold, an hysteresis threshold, was also tested for this image but it only took much more time to compute without producing a better result. The MCC values were still higher using the mean-C threshold rather than the simple threshold.

scale	(MCC for simple threshold, threshold value)	(MCC for mean-C threshold, threshold value)
2.3	(0.12377071948136999, 0.2)	(0.15349674911730868, 0.8)
2.4	(0.13397314112591624, 0.24)	(0.15282887989728516, 1.35)
2.5	(0.13554784971141662, 0.22)	(0.15761874738481663, 1.0)
2.6	(0.13331916640334726, 0.15)	(0.1633675246623953, 0.5)
2.7	(0.14284128367051874, 0.17)	(0.16195470746255322, 1.1)
2.8	(0.1434533994852059, 0.14)	(0.16725567701218266, 0.8)
2.9	(0.14095411114594739, 0.1)	(0.1762816888550408, 0.25)
3.0	(0.12221750911967234, 0.11)	(0.17791731698078309, 0.1)
3.1	(0.14842168439348521, 0.1)	(0.17507419989856265, 0.5)
3.2	(0.14192342222880994, 0.09)	(0.18589414045946556, 0.1)
3.3	(0.1251370601978605, 0.09)	(0.18212693725165174, 0.1)

Figure 11: MCC scores for the best threshold found for each tested scale on the image 109.png

The values with the mean-C threshold are still higher than those with the simple threshold. Once again, we chose the scale from those values (but the values have to also be ones of the highest with the simple threshold). According to this coefficient, the best scale for our database seems to be $s = 3.2$. The same test is done with another image of the database (108.png) and the result is approximetaly the same : the highest MCC value is obtained with a scale $s = 3.1$. I did a third test to make sure the intrinsic parameter doesn't change the result of the experiment, so I tried the same thing on the image 109.png but with a different value of $\tau = 0.5$ and the results are consistent : the scale $s = 3.2$ is still the best one for this image. From now on, the scale used to compute the results of each image with the Jerman filter will be the mean of the two values we found : $s = 3.15$. One interesting thing to notice is that this value is really close, almost the same as the one we chose for the Frangi filter.

Now that we found the best scale to execute the Jerman filter, it is about finding the right tau parameter. First, I computed a bunch of images with different values of tau to visually see the impact of this parameter. But the metrics still represent a more secure way to optimize the parameters of the filters.

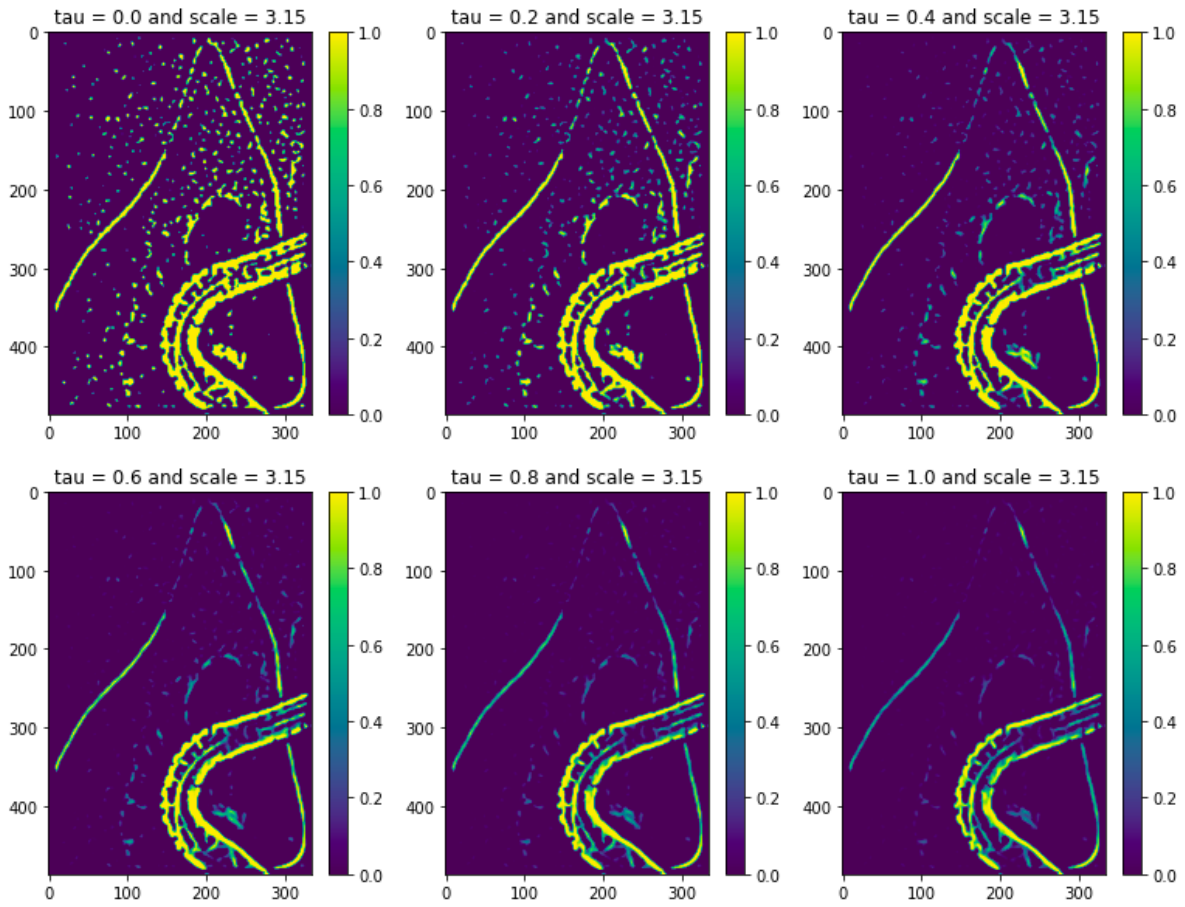

Figure 12: Image filtered using the Jerman filter for different values of τ

Using the same method as before to find the best value of tau for a given scale, I iterated through a range of tau from 0.0 to 1.0 with a 0.1 step and computed the given MCC value with the right threshold. This time, I also computed the Dice value.

tau	MCC with mean-C threshold	Dice with mean-C threshold
0.0	(0.1506695904113459, 0.0)	(0.2528632470294422, 0.0)
0.1	(0.1613567249369416, 0.95)	(0.2608425242557426, 0.95)
0.2	(0.16816007134558947, 0.75)	(0.26577725459420576, 0.75)
0.3	(0.17212538909771344, 1.45)	(0.2675472965409157, 0.55)
0.4	(0.1750091464779987, 1.0)	(0.26931773152140975, 0.5)
0.5	(0.1762581097709602, 1.05)	(0.27051239512726927, 0.4)
0.6	(0.1773136223941172, 0.35)	(0.27130148110440455, 0.35)
0.7	(0.17761268532423202, 0.25)	(0.2718129665634526, 0.25)
0.8	(0.17833962355787605, 0.3)	(0.2723056590171813, 0.2)
0.9	(0.17853200971446637, 0.25)	(0.27215219976218785, 0.15)
1.0	(0.17909635824138118, 0.15)	(0.2726962789791545, 0.15)

Figure 13: MCC and Dice scores for the best threshold found for each tested τ on the image 109.png

For that experiment, the MCC and the Dice coefficients both show the value 1.0 as the best one for the τ parameter. The results are the highest for it with the same threshold value. We can also notice that the Dice coefficient gives higher values than the MCC coefficient. Finally I tested this filter on the sixteen images of the database and here are interesting results for some of them.

With this filter the results 14 are not to be compared in the same way we did with the Frangi filter 9. For some images like 13.png or 14.png, the thresholding step has worked and the results are quite good compared to the annotation. We can still make the same comment about the detection of the bile duct edges. But for some other images like 12.png or 30.png, the thresholding step seems to be of no real need. The results before and after this step are alike : this filter gives almost directly the right result but because the values are all close to eachother (between 0 and 1), it is harder to threshold the filtered image without losing important parts of the wire.

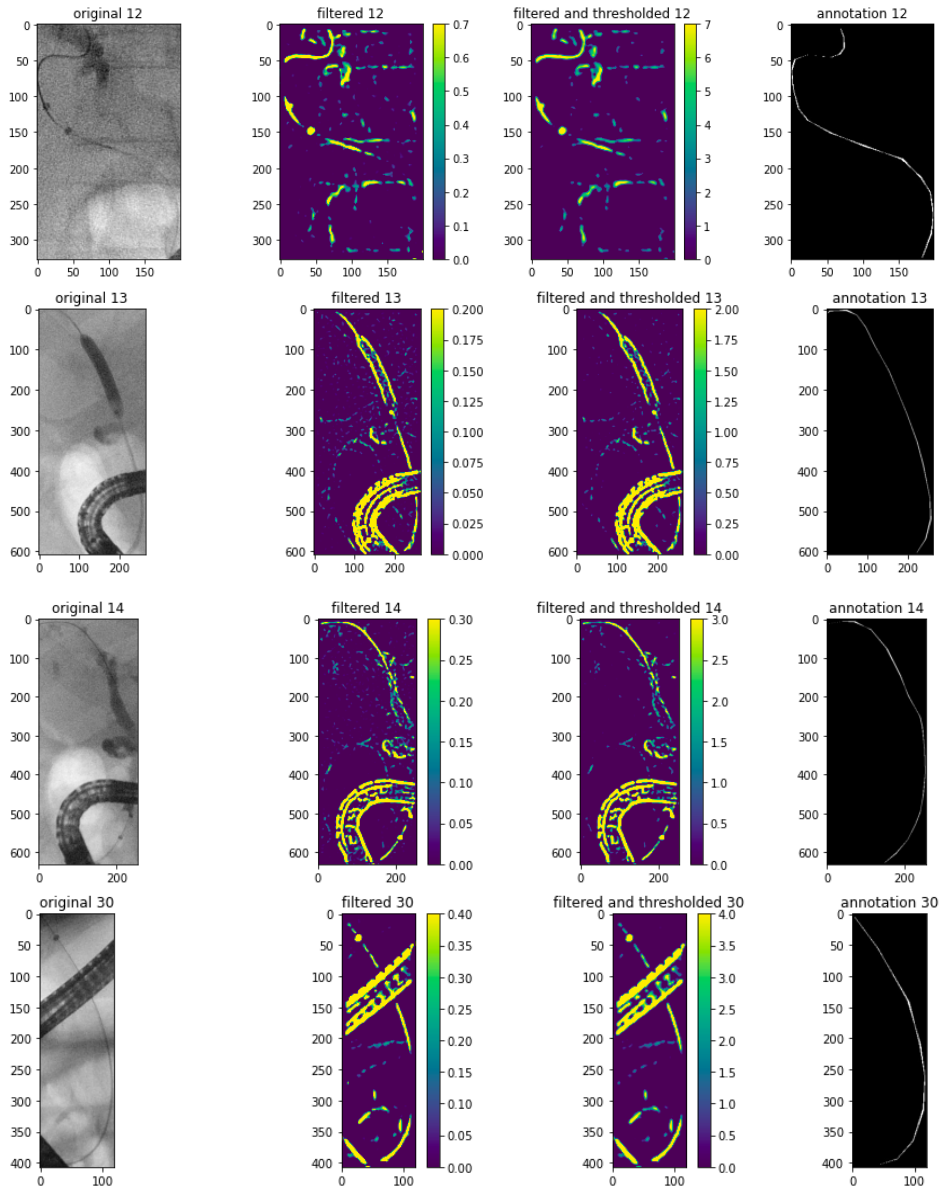

Figure 14: The original image of the database on the left, the Jerman filtered image on the middle left, the mean-C thresholded version of the filtered image on the middle right, and the annotation given on the right

With this simple threshold, the results are different but they are surprisingly better for a human eye, despite the lower values of the MCC or Dice coefficients. There are some good and bad results 15. Starting with a bad one, the image 12.png is still not usable by the doctors after application of the filter : the noise has disappeared but the wire is not enough continuous. The result for the image 13.png is better and complete with the same comment about the detection of the bile duct edges. The resulted filtered image for 108.png is really good and the noise has been greatly removed by the thresholding step. The last example is here to show how the filter can detect a wire that is really low-contrasted from the background : it is difficult to see it on the image but the most part of the wire is detected by the filter and that is exactly what we were

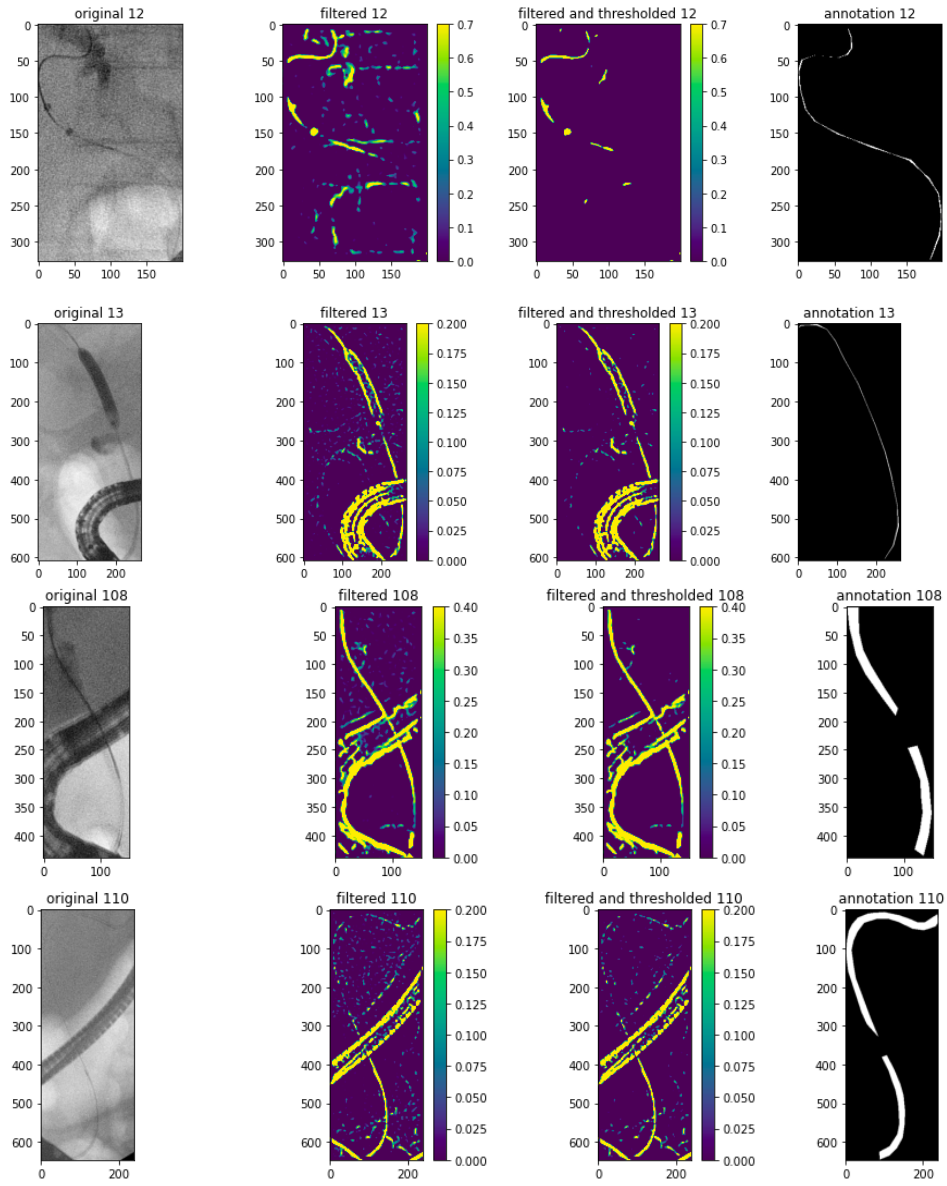

Figure 15: The original image of the database on the left, the Jerman filtered image on the middle left, the simple threshold version of the filtered image on the middle right, and the annotation given on the right

looking for.

E. Comparison of the filters

After implementing those two filters, we can finally compare them. From the previous results, I computed both filter on every image in the database with the optimize parameters I found, and applying a mean-C threshold on it to maximize the MCC value. I then calculated the Dice value.

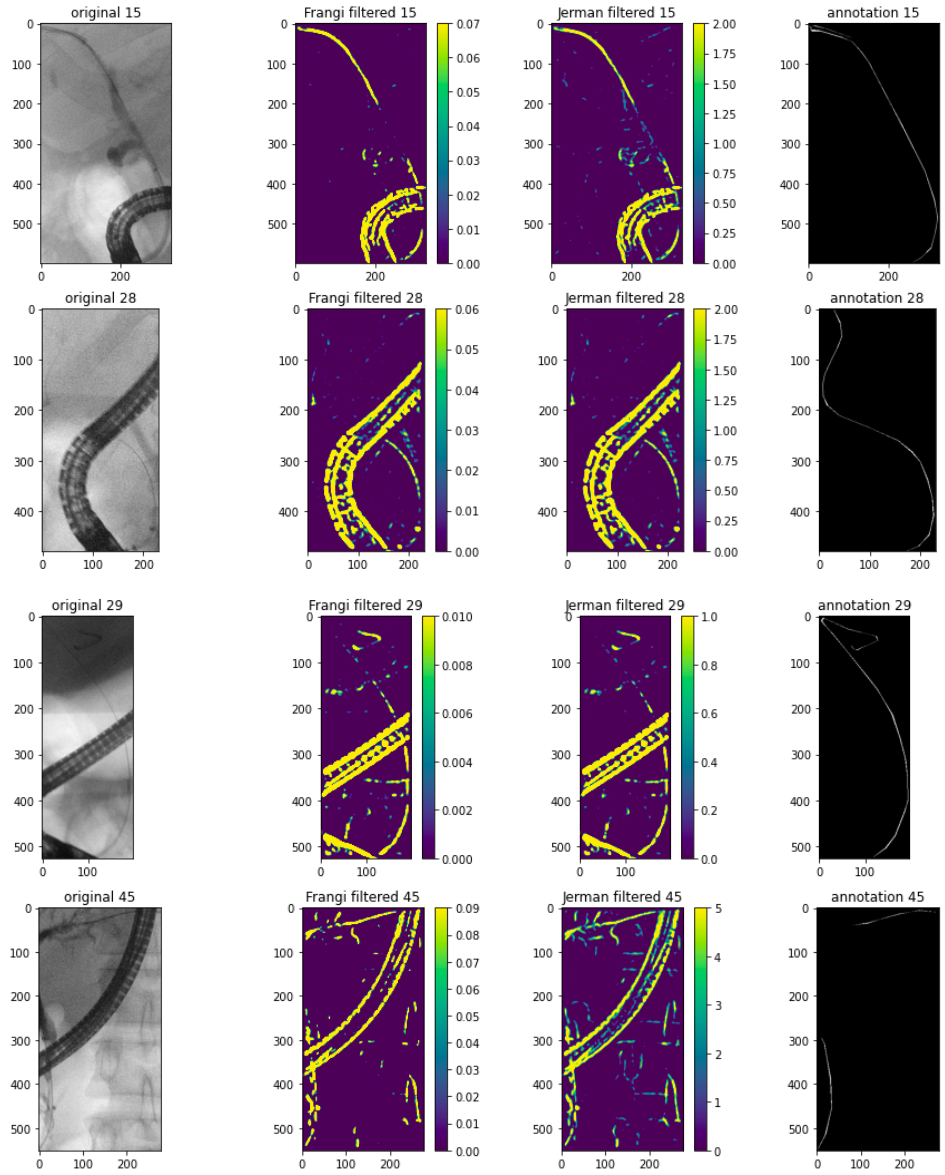

Figure 16: The original image of the database on the left, the Frangi filtered and thresholded image on the middle left, the Jerman filtered and thresholded image on the middle right, and the annotation given on the right

Here 16 there are several things to say. In some case (images 28.png and 29.png), we can see that the filters give the approximately the same visual results. Sometimes, like with the image 15.png, the Jerman filter seems more intelligent as it keeps some noise to keep the part of the wire (in the middle) that is lost by the Frangi filter. And there is a third case, such as image 45.png, where the Frangi filter is much more efficient because the Jerman filter keeps a lot of noise from the background that the Frangi filter can get rid off without affecting the wire detection (on the side here).

Analysing the coefficients values 17, and taking the standard deviation into account, the Frangi and the Jerman filters give equal results. Regarding the execution time, the mean is telling us that the Jerman filter is twice faster than the Frangi filter but the standard deviation is completing this gap.

```
MCC
Frangi : MCC = 0.15113366019819702  +/-  0.0765080037319915
Jerman : MCC = 0.16140696934032167  +/-  0.09074902393854431

Dice
Frangi : Dice = 0.1840371117226429  +/-  0.05609232519796267
Jerman : Dice = 0.19737498397876524  +/-  0.09392491041856506

execution time
Frangi : time = 0.9521060585975647  +/-  0.4144899891896518
Jerman : time = 0.5059463381767273  +/-  0.22570796431438145
```

Figure 17: The mean and standard deviation of MCC, Dice and time of execution for the two filters Frangi and Jerman

I tried to do the same experiment by maximizing the Dice metric this time (and not the MCC). The visual and numerical results are alike the first comparison.

V. Conclusion

Even if the numbers are telling us the Jerman filter is more efficient than the Frangi filter, we can't forget the initial goal of our project : helping the endoscopists to see the wire in the image. In this case, we may conclude the Frangi filter will be more useful for them. The Jerman filter was created after but it is supposed to be more robust to bifurcation which is not something we specially need in this situation. Opting for the "old version" may actually be smarter for this case because the changes made in the Jerman version are not compatible with our images.

Bibliography

[CXZ19] Hengfei Cui, Yong Xia, and Yanning Zhang. 2d and 3d vascular structures enhancement via improved vesselness filter and vessel enhancing diffusion. *IEEE Access*, 7:123969–123980, 2019.

[DVK+21] Sonali Dash, Sahil Verma, Kavita, Md. Sameeruddin Khan, Marcin Wozniak, Jana Shafi, and Muhammad Fazal Ijaz. A hybrid method to enhance thick and thin vessels for blood vessel segmentation. *Diagnostics*, 11(11):2017, Oct 2021.

[FNVV00] Ro Frangi, W.J. Niessen, Koen Vincken, and Max Viergever. Multiscale vessel enhancement filtering. *Med. Image Comput. Comput. Assist. Interv.*, 1496, 02 2000.

[13] K. Kuttler. A first course in linear algebra. `https://math.libretexts.org/Bookshelves/Linear_Algebra/A_First_Course_in_Linear_Algebra_(Kuttler)/07%3A_Spectral_Theory/7.01%3A_Eigenvalues_and_Eigenvectors_of_a_Matrix`, 2013. A Spectral Theory, Eigenvalues and Eigenvectors of a Matrix, chap. 7.1.

[17] R. Boomgaard. Gaussian smoothing and gaussian derivatives. `https://staff.fnwi.uva.nl/r.vandenboomgaard/IPCV20172018/index.html`, 2017. Lecture Notes, Images Processing, chap. 6.1.

[JPL6] Tim Jerman, Franjo Pernuš, Boštjan Likar, and Žiga Špiclin. Enhancement of vascular structures in 3d and 2d angiographic images. *IEEE Transactions on Medical Imaging*, 35(9):2107–2118, 2016.

[LC08] Max Law and Albert Chung. Three dimensional curvilinear structure detection using optimally oriented flux. pages 368–382, 10 2008.

[LMK+21] Jonas Lamy, Odyssée Merveille, Bertrand Kerautret, Nicolas Passat, and Antoine Vacavant. Vesselness filters: A survey with benchmarks applied to liver imaging. In *2020 25th International Conference on Pattern Recognition (ICPR)*, pages 3528–3535, 2021.

[LMN+20] Antonia Longo, Stefan Morscher, Jaber Malekzadeh Najafababdi, Dominik Jüstel, Christian Zakian, and Vasilis Ntziachristos. Assessment of hessian-based frangi vesselness filter in optoacoustic imaging. *Photoacoustics*, 20:100200, 2020.

[MJS+04] Erik H. W. Meijering, Mathews Jacob, J.-C. Floyd Sarria, Pascal Steiner, Harald Hirling, and Michael A. Unser. Neurite tracing in fluorescence microscopy images using ridge filtering and graph searching: principles and validation. *2004 2nd IEEE International Symposium on Biomedical Imaging: Nano to Macro (IEEE Cat No. 04EX821)*, pages 1219–1222 Vol. 2, 2004.

[MTNP18] Odyssée Merveille, Hugues Talbot, Laurent Najman, and Nicolas Passat. Curvilinear structure analysis by ranking the orientation responses of path operators. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(2):304–317, 2018.

[SNS+98] 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, 1998.

[Tan10] Olena Tankyevych. Filtering of thin objects : applications to vascular image analysis. 10 2010.

[TTESS21] Khaddouj Taifi, Naima Taifi, Azougaghe Es-Said, and Safi Said. An automatic detection by classification of cracked pixels or noncracked pixels in road surface. *Mathematical Problems in Engineering*, 2021:1–10, 12 2021.

[ZZL+18] Rui Zhang, Zhuhuang Zhou, Chung-Chih Lin, Po-Hsiang Tsui, and Shuicai Wu. An improved fuzzy connectedness method for automatic three-dimensional liver vessel segmentation in ct images. *Journal of Healthcare Engineering*, 2018:2376317, 10 2018.