UNIVERSITÄT PASSAU

Hervorhebung von tubulären Strukturen mithilfe des Frangi-Filters

Bachelorarbeit von

Severin Primbs
89808

Universität Passau
Fakultät für Mathematik und Informatik

Lehrstuhl für Mathematik mit Schwerpunkt digitale Bildverarbeitung

Betreuer: Prof. Dr. Tomas Sauer

16. August 2021

Inhaltsverzeichnis

1 Einleitung

1.1 Motivation

Die digitale Bildverarbeitung nimmt immer mehr Einfluss auf die moderne Technik. Sei es in der Robotik, bei der sich Maschinen mithilfe von Bildern orientieren und zum Teil sogar bewegen können, oder sei es nur das Scannen von QR-Codes, die Informationen in quadratischen Mustern speichern können. Sowohl in diesen beiden als auch in vielen weiteren Einsatzgebieten müssen Daten aus den Bilder verwendet und verarbeitet werden. Daher ist es nicht verwunderlich, dass auch die Medizintechnik von der digitalen Bildverarbeitung profitiert. Speziell in der Kardiologie sind präzise Diagnosen von großer Bedeutung.

Dazu muss man wissen: Der Blutkreislauf ist eines der wichtigsten Systeme des menschlichen Körpers und hat die Aufgabe alle Zellen und Organe mit Sauerstoff und Nährstoffen zu versorgen. Um alle Körperzellen zu erreichen, verzweigt sich die sogenannte Hauptschlagader zu kleineren abspreizenden Ästen, die sich wiederum in kleinere verzweigen. Es ähnelt einer Baum- oder Wurzelstruktur. Im Laufe dieser Arbeit ist häufig von einer Tubularstruktur oder tubulären Struktur die Rede, welches als Synonym für solche gefäßartigen oder verwurzelten Strukturen steht. Das Netz aus kleinsten Verzweigungen - den sogenannten Blutgefäßen - nennt man Kapillarnetz. Doch wie nahezu jeder Körperbestandteil können auch die Blutgefäße erkranken. Im Jahr 2019 starben laut statistischem Bundesamt 331 211 Menschen an Kreislauferkrankungen. Das ist ein Gesamtanteil von mehr als 35% der Todesfälle in Deutschland in diesem Jahr (Anzahl aller Todesfälle: 939 520).

Ein Arzt muss die kleinsten Veränderungen in den teilweise winzigen Blutgefäßen erkennen können. Für Diagnosen ist es notwendig die Blutgefäßstrukturen in optimaler Art und Weise darzustellen. Mithilfe einer Angiografie können die Blutgefäße des menschlichen Körpers abgebildet werden. Dabei wird dem Patienten zunächst ein Kontrastmittel injiziert. Bei der Angiografie gibt es verschiedene Verfahren. Zum einen kann man auf das klassische Röntgen zurückgreifen. Hier erhält man üblicherweise 2D-Aufnahmen der zu untersuchenden Stelle. Meist wird dieses Verfahren zur Erkennung von Gefäßverengungen eingesetzt. Eine weitere Methode ist die Abbildung mithilfe von Computertomografen. Dort kann man auf 3D-Aufnahmen zurückgreifen (vgl. [1]).

1.2 Problemstellung

Sowohl bei 2D-Aufnahmen als auch im Dreidimensionalen kommt man schnell zu der Erkenntnis, dass die Blutgefäße nicht einwandfrei abgelichtet werden. Dieser Thematik hat sich Alejandro F. Frangi in seinem insgesamt vierköpfigen Team im Jahre 1998 gewidmet. In ihrem gemeinsamen Dokument „Multiscale vessel enhancement filtering“, auf welchem diese Arbeit überwiegend basiert, stellen sie fest, dass die Hauptnachteile dieser Projektionen, das Überlappen mit nicht gefäßartigen Strukturen ist. Außerdem sind die kleinen Gefäße mit geringem Kontrast schwer erkennbar.

Dieses Problem löst der Frangi-Filter. Die Gefäßstrukturen sollen durch eine Ge-

fäßsegmentierung verbessert werden. Dabei verstärken sie die kleinen Gefäße und reduzieren die Überlappung von Organen in den Bildern. Die Segmentierung des Gefäßbaums soll die Darstellung dann erleichtern. Außerdem können nun Messungen auf den Gefäßprojektionen durchgeführt werden (vgl. [2]).

1.3 Zielsetzung

Das Ziel dieser Forschung ist es, einen Einblick zu gewinnen, wie der Filter, den Frangi entwickelt hat, im Detail funktioniert. Dafür werden mathematische Konzepte eingeführt, die man zur Anwendung benötigt. Die Vor- und Nachteile in der Ausführung des Filters sollen anhand von anschaulichen Beispielen analysiert und verstanden werden. Des Weiteren soll erläutert werden, welche Möglichkeiten und Ansätze existieren, den Filter zu optimieren.

Am Ende dieser Bachelorarbeit sollten die vier Schritte des Frangi-Filters in aller Ausführlichkeit verstanden werden. Um dieses Resultat zu erreichen, ist das Verstehen von mathematischen Grundkonzepten – wie das Prinzip von Eigenwerten, Normen und Bildverarbeitungswerkzeugen – von Nöten. Außerdem soll gezeigt sein, auf welchen Objekten der Filter gut und auf welchen er weniger gut funktioniert. Mithilfe einer vollständigen Implementierung des Konzeptes von Frangi soll die Umsetzung in ein funktionierendes System verdeutlicht werden.

1.4 Aufbau der Arbeit

Um diese Aspekte zu erreichen, ist diese Arbeit wie folgt aufgebaut: In Kapitel 2 wird über die Grundlagen des Frangi-Filters gesprochen. Es werden einige Konstrukte definiert, Sätze und Lemmata vorgestellt und manche davon bewiesen. Dieses Kapitel ist in zwei Unterabschnitte gegliedert. Im ersten Teil (Kapitel 2.1) werden die mathematischen Grundlagen, wie z.B. Eigenschaften von Matrizen, Eigenwerte und Normen, behandelt. Im zweiten Teil 2.2 werden bildverarbeitungsspezifische Definitionen, wie z.B. die Faltung, angeschnitten. Das nächste Kapitel 3 ist das Wichtigste dieser Arbeit. Dort wird die Funktionsweise des Frangi-Filters unter Miteinbezug des vorangegangenen Abschnitts ausführlich erläutert. Nach einem kurzen Einführungsabschnitt 3.1, zielt der folgende Teil 3.2 auf die Bestimmung der Hesse-Matrix eines Bildes/Volumens ab. Im Anschluss (Kap. 3.3) werden die Eigenwerte der Hesse-Matrix analysiert. Unterabschnitt 3.4 verwendet diese Eigenwerte anschließend und fasst diese zur *Vesselness-Funktion* von Frangi zusammen. Da sich die Berechnung dieser Werte im Dreidimensionalen vom Zweidimensionalen unterscheidet, wird in diesem Kapitel auf beide Möglichkeiten eingegangen. Im letzten Unterkapitel dieses Teils 3.5 wird der neue Pixel- bzw. Voxelwert berechnet. Die darauffolgende Sektion 4 soll dann Verbesserungsmöglichkeiten und Alternativen zum Frangi-Filter darstellen. Neben dem Frangi-Net (Kap. 4.1) soll auch der Ansatz zur Unterscheidung von Atemwegs- und Gefäßstrukturen (Kap. 4.2) erläutert werden. Anschließend wird in Kapitel 5 meine Implementierung in Ausschnitten und anhand von Beispielen erläutert. Dieses gliedert sich wieder in die 2D- und 3D-Variante auf. Der vollständige Quelltext befindet sich im Anhang 8. Kapitel 6 analysiert nun die Implementierung mit verschiedenen Eingabeparametern (Abschnitt 6.1) und führt Experimente mit unterschiedlichen Mustern (Abschnitt 6.2) durch, um die Stärken und Schwächen

des Filters zu erforschen. Letztendlich werden in Abschnitt 7 die wesentlichsten Erkenntnisse auf den Punkt gebracht.

2 Grundlagen

2.1 Mathematische Grundlagen

In den folgenden Abschnitten werden einige Definitionen, Lemmata und Sätze vorgestellt. Vorab ist zu erwähnen, dass viele davon aus Komplexitätsgründen im Vergleich zu den Originaldefinitionen leicht angepasst wurden. Um mathematische Grundgerüste wie zum Beispiel einen *Körper* nicht definieren zu müssen, wurden alle mathematischen Hilfsmittel auf den Körper der reellen Zahlen spezifiziert, obwohl diese im Wesentlichen viel allgemeiner gelten würden. Für diese Arbeit wird lediglich diese Spezifikation benötigt.

2.1.1 Matrizen und Eigenwerte

Ziel dieses Abschnittes ist die Bestimmung der Eigenwerte von quadratischen Matrizen. Dafür müssen zunächst einige mathematische Grundkonzepte definiert werden, unter anderem der Begriff des $\mathbb{R}$-Vektorraums.

Definition 2.1 ($\mathbb{R}$-Vektorraum):
Eine Menge V zusammen mit einer inneren Verknüpfung

$$+ : V \times V \to V, (v, w) \mapsto v + w \tag{1}$$

und einer äußeren Verknüpfung

$$\cdot : \mathbb{R} \times V \to V, (\lambda, w) \mapsto \lambda \cdot w \tag{2}$$

heißt $\mathbb{R}$-Vektorraum, wenn $\forall v, w, u \in V$ und $\lambda, \mu \in \mathbb{R}$ gilt:

Kommutativität:	$v + w = w + v$	(3a)
Assoziativität bzgl. +:	$v + (w + u) = (v + w) + u$	(3b)
Existenz des neutr. Elements bzgl. +:	$\exists 0 \in V$ mit $v + 0 = v$	(3c)
Existenz des negativen Inversen:	$\exists -v \in V$ mit $v + (-v) = 0$	(3d)
Distributivität I:	$(\lambda + \mu) \cdot v = (\lambda \cdot v) + (\mu \cdot v)$	(3e)
Distributivität II:	$\lambda \cdot (v + w) = (\lambda \cdot v) + (\lambda \cdot w)$	(3f)
Assoziativität bzgl. ·:	$(\lambda\mu) \cdot v = \lambda \cdot (\mu \cdot v)$	(3g)
Existenz des neutr. Elements bzgl. ·:	$1 \cdot v = v$	(3h)

Im Folgenden sei V stets ein $\mathbb{R}$-Vektorraum. Ein weiterer Begriff, der im Laufe der Arbeit noch mehrmals benötigt wird, ist die Matrix.

Definition 2.2 (Matrix):
Eine Tabelle mit Zahlen aus m Zeilen und n Spalten nennt man eine Matrix vom

Typ $m \times n$. Sei $A \in \mathbb{R}^{m \times n}$, dann gilt:

$$A := (a_{ij})_{1 \leq i \leq m, 1 \leq j \leq n} = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \dots & a_{2n} \end{pmatrix} \tag{4}$$

Diese Definition reicht für unsere Zwecke. In der Realität hat eine Matrix weitaus mehr Semantiken, die wir aber an dieser Stelle nicht benötigen und somit ignorieren. Ein wichtiges Beispiel für eine Matrix ist die Einheitsmatrix. Diese ist so definiert:

Beispiel 2.3 (Einheitsmatrix):
Sei $I_n \in \mathbb{R}^{n \times n}$. Dann heißt I_n *Einheitsmatrix*, wenn gilt

$$I_n := \begin{pmatrix} 1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 1 \end{pmatrix}$$

Eine weitere Definition, die wir im späteren Verlauf dieser Arbeit noch mehrmals benötigen, ist eine transponierte Matrix.

Definition 2.4 (transponierte Matrix):
Sei $A \in \mathbb{R}^{m \times n}$ eine reelle Matrix. Wir bezeichnen $A^\top \in \mathbb{R}^{n \times m}$ als die *transponierte Matrix* zu A, wenn gilt: $(a_{ij})_{1 \leq i \leq m, 1 \leq j \leq n} = (a_{ji})_{1 \leq i \leq m, 1 \leq j \leq n}$

Beispiel 2.5:
Die Transponierte Matrix von $A = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$ ist $A^\top = \begin{pmatrix} 1 & 3 \\ 2 & 4 \end{pmatrix}$.

Darüber hinaus wird im weiteren Verlauf der Arbeit die Definition einer normalen Matrix benötigt, welche im Reellen wie folgt bezeichnet werden kann.

Definition 2.6 (normale Matrix in $\mathbb{R}$):
Eine reelle Matrix $A \in \mathbb{R}^{n \times n}$ heißt *normal*, wenn gilt $A^\top \cdot A = A \cdot A^\top$

Definition 2.7 (Spur einer Matrix):
Die Spur einer Matrix $A \in \mathbb{R}^{n \times n}$ ist die Summe aller Diagonaleinträge.

$$\operatorname{spur} A = \sum_{i=1}^{n} a_{ii}$$

Um uns der Berechnung von Eigenwerten einer Matrix zu nähern, müssen wir uns zunächst mit der Determinante einer Matrix beschäftigen.

Definition 2.8 (Determinante):
Sei $A \in \mathbb{R}^{n \times n}$ und sei $A'_{ij} \in \mathbb{R}^{(n-1) \times (n-1)}$ diejenige Matrix, die durch Streichen der j-ten Spalte und i-ten Zeile aus A hervorgeht. Die Determinante ist dann so definiert:

$$\det A := \begin{cases} a_{11} & n = 1 \\ \sum_{i=1}^{n} (-1)^{i+1} \cdot a_{ij} \cdot \det A'_{ij} & n \geq 2 \end{cases} \tag{5}$$

Auch hier gilt: Die Determinante einer Matrix hat weitaus mehr Semantiken und auch Berechnungsmöglichkeiten als hier vorgestellt. Obige Definition reicht allerdings für unsere Zwecke. Um die Berechnung dieser rekursiven Definition zu verstehen, sollten wir uns die Determinanten von 2×2- und 3×3- Matrizen im Allgemeinen ansehen, da diese überwiegend im Laufe dieser Arbeit vorkommen.

Beispiel 2.9:

1. $n = 2$:
   Sei $A = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$. Dann gilt für $\det A$:

$$\det A = \det \begin{pmatrix} a & b \\ c & d \end{pmatrix} = a \cdot \det d - b \cdot \det c = ad - bc$$

2. $n = 3$:
   Sei $B = \begin{pmatrix} a & b & c \\ d & e & f \\ g & h & i \end{pmatrix}$. Dann gilt für $\det B$:

$$\begin{aligned} \det B &= \det \begin{pmatrix} a & b & c \\ d & e & f \\ g & h & i \end{pmatrix} \\ &= a \cdot \det \begin{pmatrix} e & f \\ h & i \end{pmatrix} - d \cdot \det \begin{pmatrix} b & c \\ h & i \end{pmatrix} + g \cdot \det \begin{pmatrix} b & c \\ e & f \end{pmatrix} \\ &= a \cdot (ei - hf) - d \cdot (bi - hc) + g \cdot (bf - ec) \\ &= aei + bfg + cdh - ceg - bdi - afh \end{aligned}$$

Langsam nähern wir uns dem Begriff des charakteristischen Polynoms, das wesentlich für die Bestimmung der Eigenwerte einer Matrix ist. Vorerst definieren wir aber das Polynom.

Definition 2.10 (Polynome):
Sei $n \in \mathbb{N}$ und seien $a_i \in \mathbb{R}$ für $0 \leq i \leq n$. Ein *Polynom* ist ein Term der Form:

$$\sum_{i=0}^{n} a_i x^i$$

Dabei heißt n der *Grad* des Polynoms.

Da nun geklärt ist, was ein Polynom ist, kann man nun auch das charakteristische Polynom definieren.

Definition 2.11 (charakteristisches Polynom):
Sei $A \in \mathbb{R}^{n\times n}$. Das charakteristische Polynom χ_A ist definiert als:

$$\chi_A(\lambda) = \det(\lambda \cdot I_n - A)$$

Dabei ist I_n die Einheitsmatrix, die aus Beispiel 2.3 bekannt ist.

Des Öfteren wurden bereits Eigenwerte erwähnt. Jetzt ist es an der Zeit diese zu definieren.

Definition 2.12 (Eigenwerte und Eigenvektoren):
Sei $A \in \mathbb{R}^{n\times n}$ und V ein $\mathbb{R}$-Vektorraum. $\lambda \in \mathbb{R}$ heißt *Eigenwert*, wenn ein $0 \neq v \in V$ existiert, sodass

$$A \cdot v = \lambda \cdot v$$

gilt. Jedes vom Nullvektor verschiedene $v \in V$ heißt *Eigenvektor* von A.

Aber wie bestimmt man Eigenwerte? Dafür gibt es mehrere Möglichkeiten. Eine einfache Möglichkeit wird mit dem folgenden ersten Satz dieses Kapitels erläutert.

Satz 2.13:
Sei $A \in \mathbb{R}^{n\times n}$. Die Nullstellen des charakteristischen Polynoms χ_A sind die Eigenwerte von A.

Beweis. Ein ausführlicher Beweis findet sich in [3]. □

Definition 2.14:
Sei $A \in \mathbb{R}^{n\times n}$. Dann ist A eine *obere Dreiecksmatrix*, wenn sie von folgender Form ist:

$$A = \begin{pmatrix} a_{11} & \cdots & \cdots & a_{1n} \\ 0 & a_{22} & \cdots & a_{2n} \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & a_{nn} \end{pmatrix}$$

A heißt *strikt*, wenn gilt: $a_{11}, \cdots, a_{nn} = 0$.

Bemerkung 2.15:
Sei $A \in \mathbb{R}^{n\times n}$ eine obere Dreiecksmatrix, dann sind die Eigenwerte die Einträge der Hauptdiagonalen.

Beispiel 2.16:
Sei $A = \begin{pmatrix} 1 & 2 \\ 4 & 3 \end{pmatrix}$. Dann gilt:

$$\begin{aligned} \chi_A(\lambda) &= \det(\lambda \cdot I_2 - A) \\ &= (\lambda - 1)(\lambda - 3) - (-4)(-2) = \lambda^2 - 4\lambda + 3 - 8 \\ &= \lambda^2 - 4\lambda - 5 = (\lambda + 1)(\lambda - 5) \end{aligned}$$

Somit sind $\lambda_1 = -1$ und $\lambda_2 = 5$ die Eigenwerte von A.

Im nächsten Unterabschnitt 2.1.2 beschäftigen wir uns mit der Bestimmung der Hessematrix. Dafür müssen wir wissen, was eine symmetrische Matrix ist.

Definition 2.17 (symmetrische Matrix):
Eine Matrix $A \in \mathbb{R}^{n\times n}$ heißt *symmetrisch*, wenn gilt:

$$A = A^\top$$

Ein wichtiger Satz, der für symmetrische reelle Matrizen gilt und den wir später noch brauchen werden, folgt hier.

Satz 2.18:
Symmetrische reelle Matrizen haben reelle Eigenwerte.

Beweis. In [4] wird dieser Satz (dort als Lemma) ausführlich bewiesen. □

Diese Definitionen, Beispiele und Sätze benötigen wir nun zur Bestimmung der Hesse-Matrix einer Funktion. Diese Matrix ist von entscheidender Bedeutung bei der Implementierung/Anwendung des Frangi-Filters. Deshalb wurden obige Definitionen eingeführt.

2.1.2 Hesse-Matrix

In diesem Unterabschnitt werden Definitionen wie zum Beispiel Ableitungsregeln, Stetigkeit und Differenzierbarkeit als bekannt vorausgesetzt. Da beim Frangi-Filter die Hesse-Matrix eines jeden Pixels/Voxels bestimmt werden muss, ist es zunächst von großer Bedeutung die Definition der Hesse-Matrix zu kennen.
Vorstufe der Hesse-Matrix ist der Gradient.

Definition 2.19 (Gradient):
Sei $f : \mathbb{R}^n \to \mathbb{R}$ eine partiell differenzierbare Funktion. Dann ist der Vektor

$$\nabla f = \left(\frac{\partial f}{\partial x_1}, ..., \frac{\partial f}{\partial x_n} \right)$$

der Gradient der Funktion f.

Der Gradient ist also die erste Ableitung einer mehrdimensionalen Funktion. Aber wie sieht es mit der zweiten Ableitung aus? Dafür gibt es die Hesse-Matrix.

Definition 2.20 (Hesse-Matrix):
Sei $f : \mathbb{R}^n \to \mathbb{R}$ eine zweimal stetig differenzierbare Funktion. Dann ist die Matrix

$$\mathcal{H}f = \begin{pmatrix} \frac{\partial^2 f}{\partial^2 x_1^2} & \cdots & \frac{\partial^2 f}{\partial^2 x_n x_1} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial^2 x_1 x_n} & \cdots & \frac{\partial^2 f}{\partial^2 x_n^2} \end{pmatrix}$$

die sogenannte *Hesse-Matrix*.

Eine wichtige Eigenschaft der Hesse-Matrix ist in dieser Bemerkung formuliert.

Bemerkung 2.21:
Die Hesse-Matrix ist symmetrisch.

Beispiel 2.22 (mehrdimensionale Gaußverteilung):
Die *mehrdimensionale Gaußverteilung* oder auch *multivariate Normalverteilung* ist für ein Skalar s mit Dimension D so definiert:

$$G_D(x, s) = \frac{1}{\sqrt{2\pi s^2}^D} \cdot \exp\left(-\frac{\|x\|_2^2}{2s^2} \right)$$

Betrachten wir zunächst die 2-dimensionale Gaußverteilung. Diese ist in Abbildung 1 graphisch für den Parameter $s = 1$ abgebildet. Der Gradient dieser Funktion lautet

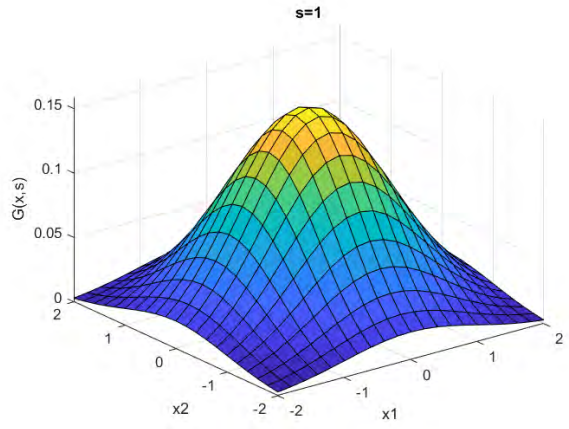


Abbildung 1: Graph der 2-Dimensionalen Gaußverteilung

nun so:

$$\nabla G_2(x,s) = \begin{pmatrix} -\frac{x_1}{2\pi s^4} \exp\left(-\frac{x_1^2+x_2^2}{2s^2}\right) \\ -\frac{x_2}{2\pi s^4} \exp\left(-\frac{x_1^2+x_2^2}{2s^2}\right) \end{pmatrix} = -\frac{1}{2\pi s^4} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \exp\left(-\frac{x_1^2+x_2^2}{2s^2}\right)$$

Analog kann man zur Bestimmung der Hesse-Matrix vorgehen. Die Hesse-Matrix lautet dann so:

$$\mathcal{H}G_2(x,s) = \frac{1}{2\pi s^6} \begin{bmatrix} x_1^2 - s^2 & x_1x_2 \\ x_1x_2 & x_2^2 - s^2 \end{bmatrix} \exp\left(-\frac{x_1^2+x_2^2}{2s^2}\right)$$

Bei der 3-Dimensionalen Gaußverteilung gilt gleiches Vorgehen. Das Bestimmen des Gradienten kann übersprungen werden. Die Hesse-Matrix sieht so aus:

$$\mathcal{H}G_3(x,s) = \frac{1}{\sqrt{8 \cdot \pi^3 s^7}} \begin{bmatrix} x_1^2 - s^2 & x_1x_2 & x_1x_3 \\ x_1x_2 & x_2^2 - s^2 & x_2x_3 \\ x_1x_3 & x_2x_3 & x_3^2 - s^2 \end{bmatrix} \exp\left(-\frac{x_1^2+x_2^2+x_3^2}{2s^2}\right)$$

Diese Erkenntnis aus dem Beispiel benötigen wir im weiteren Verlauf der Arbeit, wenn wir die Ableitung eines Bildpunktes nach Frangi bestimmen.
Eine weitere mathematische Grundkenntnis, die wir benötigen um den Frangi-Filter zu erklären, ist das Konzept der Normen.

2.1.3 Normen

Im letzten Anwendungsschritt des Frangi-Filters wird die Frobenius-Norm der Hesse-Matrix benötigt. Diese ist nach Frangi [2] gleich der euklidischen Norm der Eigenwerte. Dies soll im folgenden Abschnitt bewiesen werden. Da der Begriff Norm oft mit der Definition des Skalarprodukts einhergeht, kommt diese hier zuvor.

Definition 2.23 (Skalarprodukt):
Sei V ein $\mathbb{R}$-Vektorraum. Ein Skalarprodukt ist eine Abbildung $V \times V \to \mathbb{R}$, dass gilt:

1. Bilinearität: $\forall x, y, z \in V, r, s \in \mathbb{R}$ gilt:

$$\langle rx + sy, z\rangle = r\langle x, z\rangle + s\langle y, z\rangle \text{ und } \langle x, ry + sz\rangle = r\langle x, y\rangle + s\langle xz\rangle$$

2. Symmetrie: $\forall x, y \in V$ gilt: $\langle x, y\rangle = \langle y, x\rangle$

3. Positive Definitheit: $\forall x \in V$ gilt:

$$\langle x, x\rangle \geq 0 \text{ und } \langle x, x\rangle = 0 \Leftrightarrow x = 0$$

Definition 2.24 (Orthogonalität und Orthonormalität):

1. Zwei Vektoren $v, w \in V$ stehen orthogonal zueinander, wenn gilt: $\langle v, w\rangle = 0$

2. Eine Matrix $A \in \mathbb{R}^{n\times n}$ heißt orthogonal, wenn sowohl die n Spaltenvektoren, als auch die n Zeilenvektoren alle orthogonal zueinander stehen.
   Insbesondere gilt:

$$A^\top \cdot A = I_n \text{ und } A \cdot A^\top = I_n$$

3. Ist eine Matrix $A \in \mathbb{R}^{n\times n}$ orthogonal und es gilt für alle n Spalten-, bzw. Zeilenvektoren $\|x\| = 1$, dann heißt A orthonormal oder unitär.

Nachdem nun das Skalarprodukt und die Orthogonalität erläutert worden sind, kann man nun definieren, was man unter einer Norm verstehen darf.

Definition 2.25 (Norm):
Sei V ein Vektorraum. Eine Norm auf V ist eine Abbildung $\|\cdot\| : V \to \mathbb{R}_0^+$, so dass für alle $x, y \in V$ und alle $\lambda \in \mathbb{R}$ gilt (vgl. [5]):

1. $\|x\| \geq 0$ und $\|x\| = 0 \Leftrightarrow x = 0$

2. $\|\lambda x\| = |\lambda| \cdot \|x\|$

3. $\|x + y\| \leq \|x\| + \|y\|$

Bemerkung 2.26:
Es gelten folgende Zusammenhänge:

(a) $\|x\| = \sqrt{\langle x, x\rangle}$

(b) $|\langle x, y\rangle| \leq \|x\| \cdot \|y\|$ (Cauchy-Schwarz, Beweis findet sich in [6])

Eine wichtige Norm, die wir brauchen werden, ist die 2-Norm.

Definition 2.27 (2-Norm/Euklidische Norm):
Die 2-Norm oder auch Euklidische Norm ist definiert durch:

$$\|\cdot\|_2 : \mathbb{R}^n \to \mathbb{R} \text{ mit } \|x\|_2 := \sqrt{\sum_{i=1}^{n} x_i^2}$$

Lemma 2.28:
Die 2-Norm ist eine Norm.

Beweis. Es genügt die Normeigenschaften zu überprüfen. Sei dazu $x, y \in V$ und $\lambda \in \mathbb{R}$.

1. $\|x\|_2 \geq 0$ und $\|x\|_2 = 0 \Leftrightarrow x = 0$

$$\|x\|_2 = \sqrt{\sum_{i=1}^{n} x_i^2} = \sqrt{\underbrace{x_1^2}_{\geq 0} + \cdots + \underbrace{x_n^2}_{\geq 0}} \geq 0$$

und

$$x = 0 \Leftrightarrow x_1, x_2, ..., x_n = 0 \Leftrightarrow x_1^2 + x_2^2 + \cdots + x_n^2 = 0$$
$$\Leftrightarrow \sqrt{x_1^2 + x_2^2 + \cdots + x_n^2} = 0 \Leftrightarrow \|x\|_2 = 0$$

2. $\|\lambda x\|_2 = |\lambda| \cdot \|x\|_2$

$$\|\lambda x\|_2 = \sqrt{(\lambda x_1)^2 + \cdots + (\lambda x_n)^2} = \sqrt{\lambda^2 (x_1^2 + \cdots + x_n^2)}$$
$$= |\lambda| \sqrt{x_1^2 + \cdots + x_n^2} = |\lambda| \, \|x\|_2$$

3. $\|x + y\|_2 \leq \|x\|_2 + \|y\|_2$

$$\|x + y\|_2^2 = \sum_{i=1}^{n} (x_i + y_i)^2$$
$$= \sum_{i=1}^{n} x_i^2 + 2 \cdot \sum_{i=1}^{n} x_i y_i + \sum_{i=1}^{n} y_i^2$$
$$\underset{\text{2.26 (b)}}{\leq} \|x\|_2^2 + 2 \cdot \|x\|_2 \, \|y\|_2 + \|y\|_2^2$$
$$= (\|x\|_2 + \|y\|_2)^2$$

$$\Rightarrow \|x + y\|_2 \leq \|x\|_2 + \|y\|_2$$

□

Nachdem wir nun wissen, dass die 2-Norm eine Norm ist, können wir selbes Vorgehen bei der Frobenius-Norm anwenden. Aber zunächst zur Definition:

Definition 2.29 (Frobenius-Norm):
Sei A eine $n \times m$-Matrix. Dann ist die Frobenius-Norm definiert als:

$$\|\cdot\|_F : \mathbb{R}^{n \times m} \to \mathbb{R} \text{ mit } \|A\|_F := \sqrt{\sum_{i=1}^{n} \sum_{j=1}^{m} a_{ij}^2}$$

Bemerkung 2.30:
Es lassen sich folgende Beziehungen erkennen:

(a) Sei $A \in \mathbb{R}^{m\times n}$. Dann besitzt A m verschiedene Spaltenvektoren $a_1, ..., a_m$ mit $a_i = \begin{pmatrix} a_{i1} \\ \vdots \\ a_{in} \end{pmatrix}$, $i \in \{1, ..., m\}$.
Dann gilt:

$$\|A\|_F = \|(\|a_1\|_2, ..., \|a_m\|_2)\|_2$$

(b) Sei $A \in \mathbb{R}^{n\times n}$. Dann gilt: $\|A\|_F = \sqrt{\text{spur}(A^\top A)}$ (Leicht nachzurechnen)

Lemma 2.31:
Die Frobenius-Norm ist eine Norm.

Beweis. Der Beweis geht analog zu Lemma 2.28. □

Zunächst müssen wir an dieser Stelle noch eine kleine Bemerkung anführen, die in [6] als Korollar zu finden ist.

Bemerkung 2.32:
Ist $A \in \mathbb{R}^{n\times n}$ eine symmetrische Matrix, so gibt es eine orthogonale Matrix S, sodass für $\lambda_1, ..., \lambda_n \in \mathbb{R}$ gilt

$$S^\top AS = \begin{pmatrix} \lambda_1 & \cdots & 0 \\ 0 & \ddots & 0 \\ 0 & \cdots & \lambda_n \end{pmatrix}$$

Beweis. Ein Beweis findet sich in [6]. □

Nachdem wir die 2-Norm und Frobenius-Norm behandelt haben, gelangen wir nun zum entscheidenden Satz dieses Abschnittes.

Satz 2.33:
Sei $A \in \mathbb{R}^{n\times n}$ normal mit den Eigenwerten $\lambda_1, ..., \lambda_n$ und sei $\lambda = (\lambda_1, ..., \lambda_n)$ der Vektor der Eigenwerte von A. Dann gilt:

$$\|A\|_F = \|\lambda\|_2 \tag{6}$$

Beweis. Nach Bemerkung 2.30 (b) gilt:

$$\|A\|_F = \sqrt{\text{spur}\,(A^\top A)}$$

Sei nun S eine unitäre Matrix und T eine obere Dreieckmatrix, sodass $A = S^\top \cdot T \cdot S$ gilt, dann folgt:

$$\begin{aligned}\sqrt{\text{spur}\,(A^\top A)} &= \sqrt{\text{spur}\,((S \cdot T \cdot S^\top)^\top \cdot (S \cdot T \cdot S^\top))} \\ &= \sqrt{\text{spur}\,(S \cdot T^\top \cdot S^\top \cdot S \cdot T \cdot S^\top)} = \sqrt{\text{spur}\,(S \cdot T^\top \cdot T \cdot S^\top)} \\ &= \sqrt{\text{spur}\,(T^\top \cdot T)} = \|T\|_F\end{aligned}$$

Sei N die strikt obere Dreiecksmatrix von T und Λ die Einträge der Hauptdiagonale, dann gilt:

$$\|T\|_F = \sqrt{\|\Lambda\|_F^2 + \|N\|_F^2}$$

Da A normal war, gilt, dass T nur aus Einträgen der Hauptdiagonale besteht, wodurch folgt: N besteht nur aus Nullen. Nach Bemerkung 2.15 gilt, dass bei oberen Dreicksmatrizen, die Eigenwerte auf der Hauptdiagonalen stehen. Schlussendlich gilt dann:

$$\|T\|_F = \sqrt{\|\Lambda\|_F^2 + \|N\|_F^2} = \sqrt{\|\Lambda\|_F^2} = \|\Lambda\|_F = \|\lambda\|_2$$

□

Kurz wird in Frangi's Dokument [2] auch auf die geometrische Bedeutung der Gleichungen eingegangen.

2.1.4 Ellipsoid

Dafür definieren wir uns einen Ellipsoiden.

Definition 2.34 (Ellipsoid):
Ein *Ellipsoid* ist der Rotationskörper einer Ellipse um ihre Hauptachse.

Bemerkung 2.35 (Eigenschaften von Ellipsen und Ellipsoiden):
Seien $x, y, z \in \mathbb{R}^+$.

- Die Fläche einer Ellipse, die von x und y aufgespannt wird, lässt sich mit der Formel $A = \pi \cdot x \cdot y$ berechnen.
- Das Volumen eines Ellipsoiden, der von x, y und z aufgespannt wird, lässt sich mit der Formel $V = \frac{4}{3}\pi \cdot x \cdot y \cdot z$ berechnen.
- Gelte $|x| \geq |y| \geq |z|$. Spannen x, y und z einen Ellipsoiden auf, so ist die größte Querschnittsfläche eine Ellipse mit den Achsen x und y.

Nachdem die wichtigsten mathematischen Grundlagen behandelt wurden, müssen wir uns auch noch bildverarbeitungsspezifischen Grundlagen beschäftigen.

2.2 Bildverarbeitung

Im weiteren Verlauf dieser Arbeit werden wir den Frangi-Filter auf zwei- und dreidimensionalen Bildern anwenden. Dafür sollten wir zunächst klären, was man unter einem Bild versteht.

Definition 2.36 (2-Dimensionales Bild):
Ein 2-Dimensionales Bild ist ein Array aus Pixelelementen, das mit diskreten Farbwerten gefüllt ist. Bei Farbbildern bestehen diese Pixel aus einem 3-Tupel, welche die RGB-Farbwerte beinhalten. In Graustufenbildern reicht ein einziger Farbwert.

Die Definition eines 3-Dimensionalen Bildes lautet ähnlich.

Definition 2.37 (3-Dimensionales Bild):
Ein 3-Dimensionales Bild ist ein Array aus Voxelelementen (Voxel sind 3-dimensionale Pixel), das ebenfalls mit diskreten Farbwerten belegt ist. Ein Synonym, das im Laufe dieser Arbeit verwendet wird, ist das *Volumen*.

Viele Definitionen für diesen Unterabschnitt benötigen wir nicht. Einzig sollten wir den Begriff der Faltung aus ein-, zwei- und dreidimensionaler Sichtweise definieren.

Definition 2.38 (Faltung):

- Sei $f, g \in \ell(\mathbb{R})$, dann ist die *Faltung* so definiert:

$$* : \ell(\mathbb{R}) \times \ell(\mathbb{R}) \to \ell(\mathbb{R})$$
$$(f * g)(x) = \sum_{k=-\infty}^{\infty} f(x-k)g(k)$$

- Sei $f \in \ell(\mathbb{R}^2)$ und $f, g \in \ell(\mathbb{R}^2)$, dann ist die *2-dimensionale Faltung* so definiert:

$$* : \ell(\mathbb{R}^2) \times \ell(\mathbb{R}^2) \to \ell(\mathbb{R}^2)$$
$$(f * g)(x, y) = \sum_{k=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} f(x-k, y-j)g(k, j)$$

- Sei $f \in \ell(\mathbb{R}^3)$ und $g \in \ell(\mathbb{R}^3)$, dann ist die *3-dimensionale Faltung* so definiert:

$$* : \ell(\mathbb{R}^3) \times \ell(\mathbb{R}^3) \to \ell(\mathbb{R}^3)$$
$$(f * g)(x, y, z) = \sum_{k=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} \sum_{i=-\infty}^{\infty} f(x-k, y-j, z-i)g(k, j, i)$$

Beispiel 2.39:
Wir betrachten die 2-Dimensionale Faltung. Seien dazu

$$f, g \in \ell(\mathbb{R}^2) \text{ mit } f = \begin{pmatrix} 1 & 2 \\ 0 & 3 \end{pmatrix}, g = \begin{pmatrix} 4 & 5 \\ 6 & 7 \end{pmatrix}$$

Dann gilt:

$$\begin{aligned}
(f * g)(1,1) &= 1 \cdot 7 + 2 \cdot 6 + 0 \cdot 5 + 3 \cdot 4 && = 31 \\
(f * g)(1,2) &= 2 \cdot 7 + 3 \cdot 5 && = 29 \\
(f * g)(2,1) &= 0 \cdot 7 + 3 \cdot 6 && = 18 \\
(f * g)(2,2) &= 3 \cdot 7 && = 21
\end{aligned}$$

Somit gilt:

$$f * g = \begin{pmatrix} 31 & 29 \\ 18 & 21 \end{pmatrix}$$

Nachdem wir alle Grundlagen behandelt haben, können wir uns nun mit der eigentlichen Funktionsweise des Frangi-Filters beschäftigen.

3 Funktionsweise des Frangi-Filters

Im Folgenden wird zunächst auf die Funktionsweise des Frangi-Filters eingegangen. [2] Nach einem kurzen Einführungsabschnitt, werden die einzelnen Schritte des Filters ausführlich erläutert.

3.1 Einführung

Für die Erklärung des Filters werden zunächst einige Variablen definiert. Im Folgenden ist stets

- D die Dimension des zu untersuchenden Objekts. In unserem Fall gilt $D \in \{2, 3\}$.
- L_D ein D-Dimensionales Bild.
- x ein D-Dimensionales Tupel mit $x = (x_1, ..., x_D)$.
- $L_D(x)$ der Pixel-/Voxelwert am Punkt x des Bildes/Volumens L_D.
- $s \in \mathbb{N}$ ein beliebiges positives Skalar.
- γ ein Parameter für normalisierte Ableitungen eines Bildes (mehr dazu in Unterabschnitt 3.5).

Nachdem einige Parameter nun definiert wurden, können wir uns mit der Funktionsabfolge des Filters beschäftigen. Die Anwendung des Frangi-Filters kann man grundsätzlich in vier Schritte einteilen.

- **Schritt 1:** Zweifache Ableitung des Bildes
- **Schritt 2:** Eigenwertanalyse
- **Schritt 3:** Anwendung der Vesselness-Funktion von Frangi
- **Schritt 4:** Berechnung des neuen Pixel-/Voxelwertes

In den folgenden Abschnitten werden nun diese Schritte sukzessive aufgegriffen und ausführlichst erläutert.

3.2 Bestimmung der Hesse-Matrix eines Bildes/Volumens

Das Hauptziel des Frangi-Filters ist das Suchen von geometrischen Strukturen in verzweigten Gefäßbildern, die als Röhrenform erkannt werden. Da in diesen Bildern die Gefäße unterschiedliche Größen haben, führt Frangi ein Skalar s ein, das innerhalb eines bestimmten Intervalls variiert. Die genaue Bedeutung wird in Abschnitt 3.5 erklärt. Um diese geometrischen Strukturen zu erkennen, schlägt er vor die lokalen Änderungen an allen Punkten zweiter Ordnung zu betrachten. Dabei definiert er die partielle Ableitung eines Bildes unter Betrachtung des Skalars s wie folgt:

$$\frac{\partial}{\partial x} L_D(x, s) := s^\gamma \cdot L_D(x) * \frac{\partial}{\partial x} G_D(x, s) \tag{7}$$

$2D$		$3D$			
λ_1	λ_2	λ_1	λ_2	λ_3	**Geometr. Struktur**
R	R	R	R	R	Rauschen/undefiniert
		N	N	H^-	helle Ebenenstruktur
		N	N	H^+	dunkle Ebenenstruktur
N	H^-	N	H^-	H^-	helle Tubularstruktur
N	H^+	N	H^+	H^+	dunkle Tubularstruktur
H^-	H^-	H^-	H^-	H^-	helle Kugelstruktur
H^+	H^+	H^+	H^+	H^+	dunkle Kugelstruktur

Tabelle 1: Auflistung möglicher Eigenwertzusammensetzungen der 2- und 3-Dimensionalen Hesse-Matrix und deren semantische Bedeutung. (vgl. [2])
(H=hoch, N = niedrig, R = Rauschen, +/- = Vorzeichen des Eigenwertes.)

Dies impliziert, was auch [7], [8] und [9] belegen, dass die zweite partielle Ableitung gleichermaßen definiert ist:

$$\frac{\partial}{\partial x^2} L_D(x,s) := s^\gamma \cdot L_D(x) * \frac{\partial}{\partial x^2} G(x,s) \tag{8}$$

(vgl. [2]). Dabei ist $G_D(x,s)$ die D-dimensionale Gaußverteilung, bekannt aus Beispiel 2.22. Konkret sind die Formeln dann für die zwei- und dreidimensionale Gaußverteilung in 9a und 9b gegeben.

$$G_2(x,s) := \frac{1}{\sqrt{2\pi s^2}^2} \cdot \exp\left(-\frac{\|x\|_2^2}{2s^2}\right) = \frac{1}{2\pi s^2} \cdot \exp\left(-\frac{x_1^2+x_2^2}{2s^2}\right) \tag{9a}$$

$$G_3(x,s) := \frac{1}{\sqrt{2\pi s^2}^3} \cdot \exp\left(-\frac{\|x\|_2^2}{2s^2}\right) = \frac{1}{\sqrt{2\pi s^2}^3} \cdot \exp\left(-\frac{x_1^2+x_2^2+x_3^2}{2s^2}\right) \tag{9b}$$

Zur Bestimmung der für den Filter nötigen Hesse-Matrix benötigt man also die Hesse-Matrix der zwei- bzw. dreidimensionalen Gaußverteilung. Diese sieht nach Beispiel 2.22 folgendermaßen aus:

$$H_{G_2}(x,s) = \frac{1}{2\pi s^6} \begin{bmatrix} x_1^2 - s^2 & x_1 x_2 \\ x_1 x_2 & x_2^2 - s^2 \end{bmatrix} \exp\left(-\tfrac{x_1^2+x_2^2}{2s^2}\right) \tag{10a}$$

$$H_{G_3}(x,s) = \frac{1}{\sqrt{8\cdot\pi^3 s^7}} \begin{bmatrix} x_1^2 - s^2 & x_1 x_2 & x_1 x_3 \\ x_1 x_2 & x_2^2 - s^2 & x_2 x_3 \\ x_1 x_3 & x_2 x_3 & x_3^2 - s^2 \end{bmatrix} \exp\left(-\tfrac{x_1^2+x_2^2+x_3^2}{2s^2}\right) \tag{10b}$$

Um die endgültige zweite Ableitung des Bildes L_D zu berechnen, ist nun eine Faltung (Definition 2.38) von Nöten. Dazu nimmt man jeden Matrixeintrag von $H_{G_D}(x,s)$ und faltet diesen mit $L_D(x)$. Zu guter Letzt multipliziert man das Resultat der Faltung mit s^γ. Dieses Ergebnis ist nun die finale Hesse-Matrix am Punkt x.

3.3 Eigenwertanalyse der Hesse-Matrix

Sei im Folgenden $H_L(x,s)$ die berechnete zweite Ableitung des Bildes L. Die Idee hinter der Analyse der zweiten Ableitungsmatrix ist die Hauptrichtungen zu extrahieren. Dort soll die lokale Struktur zweiter Ordnung des Bildes im Intervall $(-s,s)$

zerlegt werden. Mithilfe der Eigenwerte von $H_L(x, s)$ werden die grundlegenden Richtungen, in die das Bild zersetzt werden kann, extrahiert. Dies hat den Vorteil, dass nicht mehrere verschiedene Filter für unterschiedliche Richtungen angewandt werden müssen. (vgl. [2])

Die folgenden Angaben beziehen sich zunächst auf das Dreidimensionale. Im Anschluss wird auf das 2-Dimensionale eingegangen. Da die 3D-Hessematrix eine 3×3-Matrix ist, besitzt diese somit drei Eigenwerte. Sei dazu folgende Relation dieser Eigenwerte gegeben:

$$|\lambda_1| \leq |\lambda_2| \leq |\lambda_3| \tag{11}$$

Über die Eigenwerte von $H_L(x, s)$ lassen sich einige semantische Zusammenhänge herauslesen. Tabelle 1 zeigt eine Auflistung möglicher Eigenwertverhältnisse und derer Bedeutungen.

Unter Berücksichtigung von Gleichung 11 gilt grundsätzlich: Sind die Eigenwerte ≤ 0, so handelt es sich um eine hellere Struktur. Analog gilt bei Eigenwerten ≥ 0, dass es sich um eine dunklere Struktur handelt.

Haben alle drei Eigenwerte die gleichen Vorzeichen und einen hohen Absolutbetrag, so handelt es sich um eine kugelförmige Struktur. Sind die ersten beiden niedrig und einer im Absolutbetrag hoch, dann handelt es sich um eine plattenartige Struktur. Für den letzten Fall, bei dem ein Wert nahe null liegt und die anderen beiden sich annähernd im Vorzeichen und Wert gleich sind, handelt es sich nach Frangi um eine ideale Gefäßstruktur. Dabei gelten mathematisch betrachtet folgende Relationen:

$$|\lambda_1| \approx 0 \tag{12}$$

$$|\lambda_1| \ll |\lambda_2| \tag{13}$$

$$\lambda_2 \approx \lambda_3 \tag{14}$$

Es existieren auch Kombinationen von Eigenwerten, die zu keinen der zuvor behandelten Mustern passt. Diese sind dann in der ersten Zeile der Tabelle einzuordnen, welche als „Rauschen“ bezeichnet werden und für spätere Berechnungen von geringer Bedeutung sind.

Unter Verwendung von Gleichungen 12, 13 und 14 werden nun im Folgenden Verhältnisse definiert, die in der Definition der Vesselness-Funktion gebraucht werden.

Dafür betrachten wir die Situation geometrisch. In dieser Sichtweise spannen im Folgenden $|\lambda_1|$, $|\lambda_2|$ und $|\lambda_3|$ einen Ellipsoiden auf. Sei V das Volumen des Ellipsoiden, A dessen größte Querschnittsfläche und L die längste Halbachse. Nach Abschnitt 2.1.4 gelten folgende Voraussetzungen:

$$V = \frac{4\pi}{3}|\lambda_1\lambda_2\lambda_3| \tag{15}$$

$$A = |\lambda_2\lambda_3|\pi \tag{16}$$

Aufgrund von Gleichung 11 werden für Gleichung 16 λ_2 und λ_3 verwendet, da nach Bemerkung 2.35 die größte mögliche Querschnittsfläche diejenige Ellipse ist, die von λ_2 und λ_3 aufgespannt wird.

Frangi setzt nun V, A und L in Relationen und setzt dabei die geometrische Betrachtung gleich der berechneten Eigenwerte.

$$\mathcal{R}_{\mathcal{B}} := \frac{\frac{3 \cdot V}{4\pi}}{\sqrt[3]{\frac{A^2}{\pi^2}}} = \frac{|\lambda_1|}{\sqrt{|\lambda_2 \lambda_3|}} \tag{17}$$

In der nächsten Rechnung wird die Gleichheit der geometrischen Definition und der Definition mithilfe der Eigenwerte gezeigt.

$$\begin{aligned}\mathcal{R}_{\mathcal{B}} &= \frac{\frac{3 \cdot V}{4\pi}}{\sqrt[3]{\frac{A^2}{\pi^2}}} = \frac{\frac{3 \cdot \frac{4\pi}{3}}{4\pi} |\lambda_1 \lambda_2 \lambda_3|}{\sqrt[3]{\left(\frac{|\lambda_2 \lambda_3| \pi}{\pi}\right)^2}} = \frac{|\lambda_1 \lambda_2 \lambda_3|}{\sqrt[3]{|\lambda_2 \lambda_3|^2}} \\ &= \frac{|\lambda_1| \cdot |\lambda_2 \lambda_3|}{\sqrt{|\lambda_2 \lambda_3|} \cdot |\lambda_2 \lambda_3|} = \frac{|\lambda_1|}{\sqrt{|\lambda_2 \lambda_3|}}\end{aligned}$$

Diese Gleichung dient zur Unterscheidung zwischen einer kugelartigen 3D- und einer platten 2D-Struktur (vgl. [2]). Wie Tabelle 1 zeigt, gilt für eine ideale kugelartige Struktur $\lambda_1 = \lambda_2 = \lambda_3 > 0$. Berechnet man den Wert von $\mathcal{R}_{\mathcal{B}}$ für beide Mustertypen, so stellt sich heraus, dass im Falle einer perfekten Kugelstruktur gilt:

$$\mathcal{R}_{\mathcal{B}} = \frac{|\lambda_1|}{\sqrt{|\lambda_2 \lambda_3|}} = \frac{|\lambda_1|}{\sqrt{|\lambda_1 \lambda_1|}} = \frac{|\lambda_1|}{\sqrt{\lambda_1^2}} = \frac{|\lambda_1|}{|\lambda_1|} = 1$$

Analog zeigt Tabelle 1 auch, dass bei einer platten- oder linienartige 2D-Struktur die Eigenschaft $\lambda_1 = 0, \lambda_2 \in [0, \lambda_3], \lambda_3 > 0$ gilt. Nun folgt für die Relation $\mathcal{R}_{\mathcal{B}}$:

$$\mathcal{R}_{\mathcal{B}} = \frac{|\lambda_1|}{\sqrt{|\lambda_2 \lambda_3|}} = \frac{0}{\sqrt{|\lambda_2 \lambda_3|}} = 0$$

Kritisch ist hierbei der Fall $\lambda_2 = 0$ gilt. Dann ist rein mathematisch gesehen das Ergebnis dieser Formel nicht definiert. Im Hinblick auf die Implementierung kann aber in diesem Fall $\frac{0}{0} := 0$ angenommen werden. Dort wird im Falle von $\lambda_2 = 0$ $\lambda_2 = \text{eps}$ gesetzt, um Division durch 0 zu vermeiden. Frangi selbst ignoriert diesen Fall in seinem Dokument [2] mehr oder weniger. Nach ihm gilt immer $\mathcal{R}_{\mathcal{B}} \in [0, 1]$ wegen Gleichung 11 und den Absolutbeträgen in der Definition von $\mathcal{R}_{\mathcal{B}}$.
Da Relation 17 nur zur Unterscheidung von einer Kugel- und Platten-/Linienform dient, benötigt man ein zweites Verhältnis, das eine plattenartige und tubuläre Struktur unterscheiden kann. Dieses definiert Frangi wie folgt:

$$\mathcal{R}_{\mathcal{A}} := \frac{\frac{A}{\pi}}{L^2} = \frac{|\lambda_2|}{|\lambda_3|} \tag{18}$$

Auch hier soll die Gleichheit des geometrischen Zusammenhangs und des Eigenwertverhältnisses gezeigt werden.

$$\mathcal{R}_{\mathcal{A}} = \frac{\frac{A}{\pi}}{L^2} = \frac{\frac{|\lambda_2 \lambda_3| \pi}{\pi}}{|\lambda_3|^2} = \frac{|\lambda_2| \cdot |\lambda_3|}{|\lambda_3| \cdot |\lambda_3|} = \frac{|\lambda_2|}{|\lambda_3|}$$

Nun wollen wir auch hier das Ganze beispielhaft durchrechnen. Für eine ideale Plattenstruktur gilt: $\lambda_1 = 0, \lambda_2 = \lambda_3 > 0$. Rechnerisch sieht das dann wie folgt aus:

$$\mathcal{R}_\mathcal{A} = \frac{|\lambda_2|}{|\lambda_3|} = \frac{|\lambda_3|}{|\lambda_3|} = 1$$

Analog ist bei einer idealen Tubularstruktur $\lambda_1 = \lambda_2 = 0, \lambda_3 > 0$, bei denen dann der folgende Zustand eintritt:

$$\mathcal{R}_\mathcal{A} = \frac{|\lambda_2|}{|\lambda_3|} = \frac{0}{|\lambda_3|} = 0$$

Auch hier gilt wieder $\mathcal{R}_\mathcal{A} \in [0, 1]$ wegen Gleichung 11. Für den Fall, dass $|\lambda_1| = |\lambda_2| = |\lambda_3| = 0$ ist, nehmen wir wie oben an, dass $\frac{0}{0} := 0$ gilt. Tabelle 2 zeigt in Anlehnung an Tabelle 1 eine Übersicht der verschiedenen $|\lambda_i|$-Wertkombinationen und deren entsprechender $\mathcal{R}_\mathcal{B}$, $\mathcal{R}_\mathcal{A}$ im Grenzwert.

Die folgenden Zeilen beschreiben nun den Unterschied der 3D- zur 2D-Variante. Dabei passen wir die Relation aus 11 leicht an. Da die 2-Dimensionale Hessematrix eine 2×2-Matrix ist, hat diese somit nur noch zwei Eigenwerte, für die im Folgenden diese Ordnung gilt:

$$|\lambda_1| \leq |\lambda_2| \tag{19}$$

Auch an dieser Stelle kann man sich Tabelle 1 wieder anschauen. Im Zweidimensionalen verringert sich die Komplexität der Parameter. Dabei liegt der Hauptfokus lediglich noch in der Unterscheidung einer Kugel/Kreis- und Tubularstruktur. Die Platten- bzw. Ebenenunterscheidung fällt weg. Auch hier gilt: Ist λ_2 negativ, dann liegt eine helle Struktur vor. Im andere Fall logischerweise eine dunkle. Die Auseinandersortierung von 3D und 2D-Elementen fällt an dieser Stelle trivialerweise weg. Die Relationen 12 und 13 gelten auch hier. Beziehung 14 fällt weg.

Frangi definiert nun $\mathcal{R}_\mathcal{B}$ neu:

$$\mathcal{R}_\mathcal{B} := \frac{|\lambda_1|}{|\lambda_2|} \tag{20}$$

Die ganzen Ausführungen unterscheiden sich wenig zu denen für $\mathcal{R}_\mathcal{A}$ der 3D-Variante. Wie in Tabelle 2 festgehalten, gilt $\mathcal{R}_\mathcal{B} \to 1$ im Falle einer Kreis- und $\mathcal{R}_\mathcal{B} \to 0$ bei einer Tubularstruktur. Durch die Relationen 17 und 18, bzw. im Zweidimensionalen 20 werden auch Hintergrundpixel mit geringen Eigenwerten vom Betrag hervorgehoben. Um sich dieser Herausforderung zu stellen, führt Frangi einen weiteren Parameter $\mathcal{S}$ ein, der die Pixel mit beträglich großen Eigenwerten hervorhebt. Dieser ist definiert als die Frobenius Norm.

$$\mathcal{S} := \|\mathcal{H}\|_F = \sqrt{\sum_{j=1}^{D} \lambda_j^2} \tag{21}$$

Wie im mathematischen Grundlagenabschnitt 2.1 in Satz 2.33 gezeigt, ist die Frobenius-Norm einer normalen Matrix die 2-Norm ihrer Eigenwerte. So kann man Tabelle 2 noch um den Parameter $\mathcal{S}$ erweitern.

Im Folgenden Kapitel wird nun anhand dieser Parameter die Vesselness-Funktion von Frangi definiert und erläutert.

$\lvert\lambda_1\rvert$	$\lvert\lambda_2\rvert$	$\mathcal{R}_\mathcal{B}$	$\mathcal{S}$
R	R	?	N
N	H	0	H
H	H	1	H

(a) Auflistung möglicher Eigenwertverhältnisse und deren Werte von $\mathcal{R}_\mathcal{B}, \mathcal{S}$ im 2-Dimensionalen.

$\lvert\lambda_1\rvert$	$\lvert\lambda_2\rvert$	$\lvert\lambda_3\rvert$	$\mathcal{R}_\mathcal{B}$	$\mathcal{R}_\mathcal{A}$	$\mathcal{S}$
R	R	R	?	?	N
N	N	H	0	0	H
N	H	H	0	1	H
H	H	H	1	1	H

(b) Auflistung möglicher Eigenwertverhältnisse und deren Werte von $\mathcal{R}_\mathcal{B}, \mathcal{R}_\mathcal{A}, \mathcal{S}$ im 3-Dimensionalen.

Tabelle 2: *(H=hoch, N = niedrig, R = Eigenwerte sind keinen Schema zuordbar)*

3.4 Frangis Vesselness-Funktion

3.4.1 3-Dimensionale Vesselness Funktion

Im vorherigen Abschnitt 3.3 hat Frangi fürs 3-Dimensionale drei Relationen definiert. Hier sind diese zur Übersicht noch einmal angegeben.

$$\mathcal{R}_\mathcal{A} := \frac{|\lambda_2|}{|\lambda_3|},\ \mathcal{R}_\mathcal{B} := \frac{|\lambda_1|}{\sqrt{|\lambda_2\lambda_3|}},\ \mathcal{S} := \sqrt{\sum_{j=1}^{D} \lambda_j^2}$$

Um nun den Wert des neuen Voxels zu berechnen, definiert Frangi die sogenannte „Vesselnessfunktion“ $\mathcal{V}_s$. In dieser werden $\mathcal{R}_\mathcal{A}$, $\mathcal{R}_\mathcal{B}$ und $\mathcal{S}$ für ein bestimmtes Skalar s verarbeitet. Primär wird dabei in zwei Fälle unterschieden.

1. Fall: $\lambda_2 > 0$ *oder* $\lambda_3 > 0$
   Wie in Abschnitt 3.3 behandelt und wie auch Tabelle 1 zeigt bedeuten positive λ_2 und λ_3 eine dunkle und negative λ_2 und λ_3 eine helle Grundstruktur. An dieser Stelle muss/kann man sich entscheiden, um welche Art es sich handelt. Möchte man helle Strukturen hervorheben, so muss man die dunklen entfernen. Dies geschieht mit der Bedingung $\lambda_2 > 0$ oder $\lambda_3 > 0$. Möchte man andererseits dunkle Strukturen betonen, dann gehören die hellen entfernt. Dabei muss obige Bedingung leicht abgeändert werden zu $\lambda_2 < 0$ oder $\lambda_3 < 0$. Die Strukturen, die nicht zu der gewünschten gehören, sollen dann nicht berücksichtigt werden. Frangi möchte diese Pixel schwarz haben. Somit setzt er diesen Wert von $\mathcal{V}_s = 0$. In unserem Fall möchten wir die hellen Strukturen hervorheben und die Dunklen entfernen. In Folge dessen gilt stets obige Bedingung $\lambda_2 > 0$ oder $\lambda_3 > 0$.

2. Fall: $\lambda_2 \leq 0$ *und* $\lambda_3 \leq 0$
   Um nun perfekte helle tubuläre Struktur hervorheben zu können, rufe man sich noch einmal die Eigenschaften der Eigenwerte *(Gleichungen (12)-(14))*, von $\mathcal{R}_\mathcal{A}$, $\mathcal{R}_\mathcal{B}$ und $\mathcal{S}$ ins Gedächtnis.

   $$|\lambda_1| \approx 0,\ |\lambda_1| \ll |\lambda_2|,\ \lambda_2 \approx \lambda_3$$

   $$\Rightarrow \mathcal{R}_\mathcal{A} = 1, \mathcal{R}_\mathcal{B} = 0, \mathcal{S} \text{ ist hoch}$$

   Beschäftigen wir uns zunächst mit $\mathcal{R}_\mathcal{B}$. Dafür müssen wir zwei Grenzwerte betrachten. Wenn $\mathcal{R}_\mathcal{B} \to 1$ gilt, dann interessiert uns die Struktur im Wesentlichen nicht mehr, da es sich wie in Tabelle 2 festgehalten um eine kugelartige

Form handelt. Dieses Pixel ist somit keine Besonderheit und muss auch nicht hervorgehoben werden. Frangis Vesselnessfunktion soll den Wert 0 annehmen. Gilt andererseits $\mathcal{R}_\mathcal{B} \to 0$, dann ist dieses Pixel weiterhin interessant, da im Weiteren nur noch zwischen Platten- und Tubularform unterschieden werden muss. Frangi definiert anhand dieses Wissens einen Faktor:

$$\exp\left(-\frac{\mathcal{R}_\mathcal{B}^2}{2\beta^2}\right) =: \mathcal{V}_\mathcal{B}(\mathcal{R}_\mathcal{B}) \tag{22}$$

Dabei ist β ein Grenzwert, der die Sensitivität kontrollieren soll. Frangi selbst nimmt bei seinen Analysen den Wert $\beta = 0,5$ (vgl. [2]). Außerdem gilt

$$\lim_{\mathcal{R}_\mathcal{B}\to 0} \mathcal{V}_\mathcal{B}(\mathcal{R}_\mathcal{B}) = 1 \text{ und } \lim_{\mathcal{R}_\mathcal{B}\to 1} \mathcal{V}_\mathcal{B}(\mathcal{R}_\mathcal{B}) = \exp\left(-\frac{1}{2\beta^2}\right) \in (0,1]$$

Selbe Vorgehensweise können wir nun bei $\mathcal{R}_\mathcal{A}$ anwenden. Hat das Muster eine plattenartige Form, so gilt $\mathcal{R}_\mathcal{A} \to 0$. Diese Struktur ist für das Hervorheben von Tubularvoxeln unwichtig. Der andere Fall interessiert uns wieder. Wenn $\mathcal{R}_\mathcal{A} \to 1$, dann ist dieses Voxel Teil einer Tubularstruktur. Dieser gehört hervorgehoben. Analog zu $\mathcal{R}_\mathcal{B}$ bestimmt Frangi wiederum einen Faktor:

$$1 - \exp\left(-\frac{\mathcal{R}_\mathcal{A}^2}{2\alpha^2}\right) =: \mathcal{V}_\mathcal{A}(\mathcal{R}_\mathcal{A}) \tag{23}$$

α ist in diesem Faktor wieder ein Grenzwert zur Kontrolle der Sensitivität. Analog zu oben nimmt Frangi auch hier den Wert $\alpha = 0,5$. Auch hier kann man die Grenzwerte betrachten:

$$\lim_{\mathcal{R}_\mathcal{A}\to 0} \mathcal{V}_\mathcal{A}(\mathcal{R}_\mathcal{A}) = 0 \text{ und } \lim_{\mathcal{R}_\mathcal{A}\to 1} \mathcal{V}_\mathcal{A}(\mathcal{R}_\mathcal{A}) = 1 - \exp\left(-\frac{1}{2\alpha^2}\right) \in (0,1]$$

Der letzte Faktor, den Frangi hinzufügt, ist zum Unterdrücken von Voxeln mit geringen Eigenwerten, die eher in die Kategorie „Rauschen“ gehören. Dieser Faktor ist ähnlich wie $\mathcal{V}_\mathcal{A}$ aufgebaut.

$$1 - \exp\left(-\frac{\mathcal{S}^2}{2c^2}\right) =: \mathcal{V}_\mathcal{S}(\mathcal{S}) \tag{24}$$

Wenn die Frobenius-Norm gering ist, dann gilt $\lim_{\mathcal{S}\to 0} \mathcal{V}_\mathcal{S}(\mathcal{S}) = 0$ und ein Voxel wird im Wesentlichen nicht gewertet. Im anderen Fall, wenn diese hoch ist, dann wird dieses Voxel entsprechend gewichtet.

Führt man diese beiden Fälle inklusive aller Faktoren $\mathcal{V}_\mathcal{A}$, $\mathcal{V}_\mathcal{B}$ und $\mathcal{V}_\mathcal{S}$ zusammen, so entsteht die **Vesselness-Funktion von Frangi**:

$$\mathcal{V}_0(s) := \begin{cases} 0 & \lambda_2 > 0 \text{ oder } \lambda_3 > 0 \\ \underbrace{\left(1 - \exp\left(-\frac{\mathcal{R}_\mathcal{A}^2}{2\alpha^2}\right)\right)}_{\mathcal{V}_\mathcal{A}} \underbrace{\exp\left(-\frac{\mathcal{R}_\mathcal{B}^2}{2\beta^2}\right)}_{\mathcal{V}_\mathcal{B}} \underbrace{\left(1 - \exp\left(-\frac{\mathcal{S}^2}{2c^2}\right)\right)}_{\mathcal{V}_\mathcal{S}} & \text{sonst} \end{cases} \tag{25}$$

Zusammenfassend gilt nun: Wurde eine tubuläre Struktur für ein bestimmtes Skalar s erkannt, so gilt $\mathcal{V}_0 \to 1$, im anderen Fall $\mathcal{V}_0 \to 0$.

3.4.2 2-Dimensionale Vesselness Funktion

Analog ist nun das Vorgehen zur 2-Dimensionalen Vesselness Funktion. Hier noch einmal die Verhältnisse, die dafür benötigt werden

$$\mathcal{R}_{\mathcal{B}} := \frac{|\lambda_1|}{|\lambda_2|},\ \mathcal{S} := \sqrt{\lambda_1^2 + \lambda_2^2}$$

Das Vorgehen ist dabei exakt dasselbe wie in Abschnitt 3.4.1. Zunächst muss wieder in zwei Fälle unterschieden werden:

1. Fall: $\lambda_2 > 0$
   Auch an dieser Stelle soll noch einmal auf Tabelle 1 verwiesen werden. Positive Eigenwerte bedeuten eine dunkle Grundstruktur, negative Eigenwerte bedeuten eine helle Grundstruktur. In unserem Fall wird nun die helle Struktur hervorgehoben, wodurch wir alle $\lambda_2 > 0$ in Frangis Vesselness-Funktion $\mathcal{V}_0 = 0$ setzen. In der anderen Situation wird die Bedingung auf $\lambda_2 < 0$ invertiert. In den nächsten Zeilen gilt stets $\lambda_2 > 0$ als Standardbedingung.

2. Fall: $\lambda_2 \leq 0$
   Mit der aktualisierten Defintion von $\mathcal{R}_{\mathcal{B}}$ gilt nun auch Gleichung 22 mit $\mathcal{V}_{\mathcal{B}}$, 24 mit $\mathcal{V}_{\mathcal{S}}$ und deren Grenzwerte wieder. Letztendlich führen diese Erkenntnisse zur leicht modifizierten Vesselness-Funktion:

$$\mathcal{V}_0(s) := \begin{cases} 0 & \lambda_2 > 0 \\ \underbrace{\exp\left(-\frac{\mathcal{R}_{\mathcal{B}}^2}{2\beta^2}\right)}_{\mathcal{V}_{\mathcal{B}}} \underbrace{\left(1 - \exp\left(-\frac{\mathcal{S}^2}{2c^2}\right)\right)}_{\mathcal{V}_{\mathcal{S}}} & \text{sonst} \end{cases} \tag{26}$$

Jetzt ist die Vesselness-Funktion sowohl im 2- als auch im 3-Dimensionalen definiert. Schlussendlich ist nur noch die Konstruktion des finalen Voxel-/Pixelwertes nötig, was im Folgenden geschieht.

3.5 Bestimmung des neuen Pixel-/Voxelwertes

Die folgende Ausführung ist sowohl auf Pixel-, als auch auf Voxelebene äquivalent.

In klassischen Gefäßaufnahmen sind die Blutgefäße nie exakt gleich dick oder dünn. Deshalb ist es auch von Nöten den Frangi-Filter an diese Gegebenheit anzupassen. Bisher haben wir das Skalar s als eine feste Konstante in allen Berechnungen angesehen. Dieses Skalar wird nun wichtig. Semantisch betrachtet ist $2 \cdot s$ der maximale Durchmesser eines Gefäßes in Pixeln, der betrachtet werden soll. Der Anwender gibt nun dem Filter eine Menge Σ mit verschiedenen Skalaren.($\Sigma \subseteq \mathbb{N}$) Da nun $\mathcal{V}_0$ für kleine $s \in \Sigma$ in dicken Gefäßstrukturen einen geringen Wert annehmen wird, werden diese so gut wie nicht erkannt. Andersherum gesehen werden auch große $s \in \Sigma$ in dünnen Strukturen ebenfalls schlecht oder gar nicht erkannt (vgl. [2]). Um sich dieser Herausforderung zu stellen schlägt Frangi vor, die Vesselness-Funktion für jedes Skalar $s \in \Sigma$ anzuwenden. Da die Gefäße in den Aufnahmen unterschiedliche Stärken haben, existiert ein s (insofern es Element von Σ ist), das zu einer Gefäßstärke

passt. $\mathcal{V}_0$ wird dann an dieser Stelle den größten Wert haben. Mit diesem Wissen lässt Frangi die Voxel des Ausgabebildes nun so berechnen:

$$\mathcal{V}(\gamma) = \max_{s \in \Sigma} \mathcal{V}_0(s, \gamma) \tag{27}$$

An dieser Stelle sollte noch kurz die Bedeutung von γ erläutert werden. γ ist ein Parameter den Tony Lindenberg in seinem Paper „Edge Detection and Redge Detection with Automatic Scale Selection“ eingeführt hat. Er definiert eine Familie von normalisierten Ableitungen eines Bildes um einen fairen Vergleich bei Betrachtung von unterschiedlichsten Skalaren zu erhalten. In der in Abschnitt 5 vorgestellten Implementierung wird γ per Default auf 2 gesetzt. [2] definiert den Lindenberg-Parameter gleich direkt als 2.

4 Optimierungsmöglichkeiten und Alternativen des Frangi-Filters

Nun wissen wir, wie der Frangi-Filter im Detail funktioniert. In diesem Abschnitt, der auf Kapitel 3 aufbaut, werden zwei Ansätze gezeigt, die zur Optimierung des Filters möglich sind. Vorab soll gesagt sein, dass die folgenden Unterabschnitte im Vergleich zum vorangegangenen Kapitel lediglich dem Überblick dienen sollen. Sehr detaillierte Erläuterungen einzelner Abläufe werden hier nicht stattfinden. Der Sinn und Zweck dieses Abschnittes ist das Verständnis des groben Funktionsprinzips.
Wir werden uns zunächst mit dem sogenannten „Frangi-Net“ beschäftigen. Im Anschluss werden wir uns einem Filter basierend auf Frangi, der die Strukturen durch Anwendung einer „Penalty-Funktion“ hervorhebt, widmen.

4.1 Das Frangi-Net

Beginnen wir mit dem Frangi-Net. Die Angaben dieses Unterabschnittes beziehen sich größtenteils auf das Paper „Frangi-Net: A Neural Network Approach to Vessel Segmentation“ [10].
Was ist das Prinzip dieses Ansatzes? Der aus dem vorangegangenen Kapitel vorgestellte 2D-Frangi-Filter wird nun in ein neuronales Netzwerk umgewandelt. Die Autoren versprechen, dass eine Anwendung durch das Frangi-Net äquivalent zum Original-Filter ist. Um die Funktionsweise dieses Prinzips zu verstehen, ist es zunächst von Bedeutung ein (künstliches) neuronales Netzwerk (KNN) zu erklären:

Grundsätzlich kann man sagen, dass ein künstliches neuronales Netz sich an der Funktionsweise des menschlichen Gehirns orientiert. Dabei werden die im Gehirn untereinander verbundenen Nervenzellen durch sogenannte „Neuronen“ in einem Model abstrahiert. Ein neuronales Netz ist unter anderem auch wichtiger Bestandteil der Forschung im Bereich künstlicher Intelligenz. Basis dieses Netzes ist eine 3-Schichten Architektur. Wir beginnen mit der Input-Layer, die z.B. Signale aufnimmt. Die Daten werden an die Hidden-Layer weitergereicht, in der sämtliche Berechnungen ausgeführt werden. In der Output-Layer werden die neu gewonnenen Informationen z.B. in Form von Signalen wieder ausgegeben. (vgl. [11])

Baut man den 2D-Frangi-Filter nun in ein neuronales Netz um, so sollte man nach

[11] so vorgehen: Zunächst müssen wir die Skalare bestimmen, mit denen wir den Filter anwenden wollen. Dafür schlägt das Paper beispielhaft die Werte 3, 6 und 12 vor. Allerdings wird bei Anwendung des Frangi-Nets anders vorgegangen, als beim normalen Filter. Bei der ursprünglichen Variante wird für jeden Skalarwert einzeln das Gaußkernel berechnet, das für die Faltung vorgesehen ist. Das Bild wird beim Frangi-Net nun herunterskaliert. Bei der Verwendung eines 6-er Skalars wird das Originalbild auf die Hälfte und bei einem 12-er Skalar auf ein Viertel verkleinert. Anschließend wird auf den skalierten Bildern lediglich das 3-er Gaußsche Kernel angewandt. Das hat den Vorteil, dass lediglich das Gaußkernel für das Skalar drei berechnet werden muss.

Der Aufbau und Ablauf einer Netzstruktur mit einem Skalar ist sehr ähnlich zu

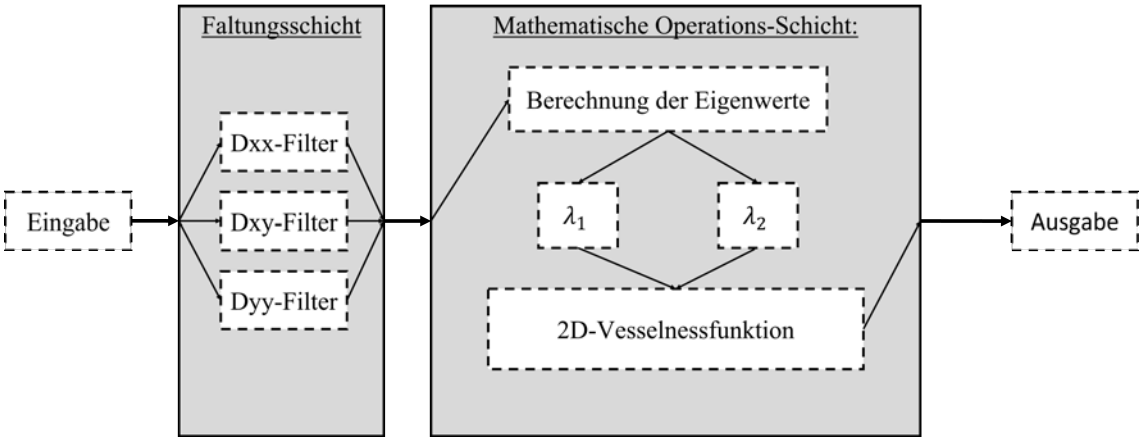


Abbildung 2: Aufbau der Frangi-Net Struktur mit einem Skalar.
(Orientiert an Paper [10] Fig.1b)

den Schritten 1 bis 3 aus Kapitel 3. Zunächst kommen die Eingabedaten in die Faltungsschicht, in der sie in die entsprechenden Richtungen der Hesse-Matrix abgeleitet und mit dem Originalbild gefalten werden. Anschließend wird für diese in der Mathematische Operationsschicht der neue Pixelwert berechnet und danach ausgegeben. Eine genaue Darstellung kann man in Abbildung 2 sehen. In Abb. 3 ist dann

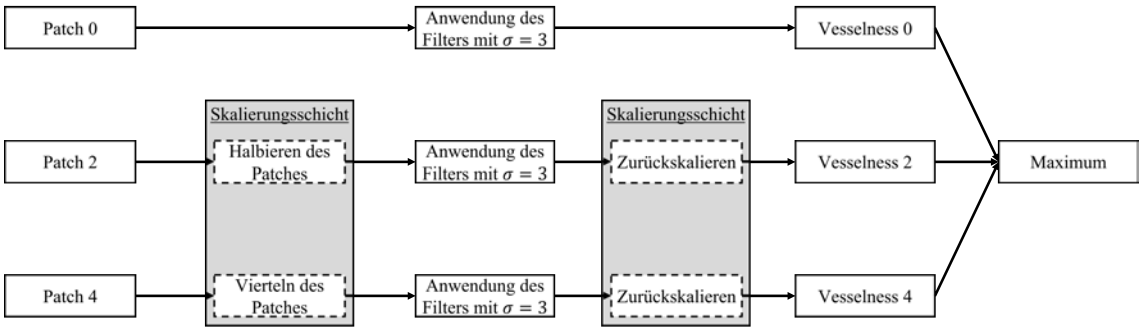


Abbildung 3: Grundstruktur des Frangi-Nets
(Orientiert an Paper [10] Fig.1c)

die Gesamtstruktur noch einmal dargestellt. Der größte Unterschied zum direkten Frangi-Filter ist das Hoch- und Herunterskalieren des Originalbildes. Die Berechnung der einzelnen Vesselbilder können nun parallel und unabhängig voneinander

berechnet werden. Für jeden Skalarwert werden nun die Werte der Vesselnessresultate analysiert und in einer „Sigma-Schicht“, in der die Pixel eindeutig klassifiziert werden, neu verteilt. Das Frangi-Net hat den Vorteil, dass es im Gegensatz zum ursprünglichen Filter trainierbar ist.[1] Nach [10] bringt das Trainieren des Netzes eine Verbesserung des F1-Scores um 17%. Der F1-Score gibt im Wesentlichen die Genauigkeit der Klassifizierung wieder.

4.2 Unterscheidung von Gefäß- und Atemwegsstrukturen

Nachdem wir uns einen neuronalen Ansatz angesehen haben, kommen wir nun zu einer Methode, die vor allem im speziell medizinischen Nutzen ihre Anwendung findet. Denn eine Herausforderung, die das Verwenden des Frangi-Filters bei direkten Gefäßstrukturen mit sich bringt, ist die Tatsache, dass fehlerhafte Berechnungen an Stellen von Atemwegswänden auftreten können. Deshalb haben die Autoren von [8] eine Methode entwickelt, bei der diese Problematik behoben werden kann. Es ist notwendig, dass man Informationen über den Bronchialbaum gewinnt. Der Vorschlag aus diesem Paper basiert auf einer gleichzeitigen Verstärkung der Gefäß- und Atemwegsbilder. Das Hauptziel sei allerdings ein eine präzisere und effizientere Gefäßverstärkung. Dazu sollen die Atemwegswände gedämpft werden. Die Funktionsweise kann man in drei Schritte einteilen:

1. Erkennung von Bronchialstrukturen
2. Schätzung der Atemwegswände
3. Anwendung einer „Penalty“-Funktion

Beginnen wir mit der *Erkennung von Bronchialstrukturen*. Wir müssen die Atemwegs- und Gefäßstrukturen unterscheiden. Das Problem: Beide sind Tubularstrukturen. Deshalb wenden die Autoren aus [8] dasselbe Frangi-Framework an. Die Vesselness-Funktion wird zweimal berechnet (angenommen die Skalare sind von annähernd gleicher Größe). Der einzige Unterschied ist, dass in Frangis Vesselness-Funktion zur Extraktion von Kapillargefäßen für die Parameter $\lambda_2 > 0$ oder $\lambda_3 > 0$ gelten muss und bei der Hervorhebung von Atemwegsstrukturen die umgekehrte Bedingung ($\lambda_2 < 0$ oder $\lambda_3 < 0$) gilt. Die Berechnung des endgültigen Vesselnesswertes unterscheidet sich nicht von der ursprünglichen Berechnung des Pixel-/Voxelwertes. Des Weiteren müssen die Mittellinien der Bronchialgefäße extrahiert werden. Dafür verwenden sie einen Skelettierungsalgorithmus für 3D-Graustufenbilder, um eine genauere Einschätzung der Mittellinien zu erhalten.

Im nächsten Schritt müssen dann die Wände der Atemwege geschätzt werden. Dafür nutzen wir das Skalar und den ersten Eigenvektor der zuvor berechneten Hesse-Matrix des maximalen Filterwertes (also das Skalar, bei dem der maximale Wert übernommen wurde). Das geschieht an jedem Punkt der zuvor berechneten Atemwegsmittellinie. Mithilfe dieser Parameter können wir die Atemwegsinnenwände schätzen. Die Außenwand kann dann mithilfe medizinischer Modelle berechnet werden. Der erste Eigenvektor kann nun als Normalenvektor einer Ebene betrachtet

[1] Für genauere Informationen zur Trainierbarkeit des Frangi-Net ist das Paper [10] empfehlenswert

werden. Schneidet man diese Ebene mit der Hohlkugel mit dem Skalar als Innenradius und dem Skalar addiert mit der Wandstärke als Außenradius um den zum Eigenvektor gehörenden Bildpunkt, so erhält man die Wandkandidatenvoxel.

Im letzten Schritt wird nun durch eine „Penalty“-Funktion an den Wandkandidaten angewandt. Dabei werden diese gedämpft. Die Funktion lautet nun wie folgt:

$$\mathcal{V}_{F_{\text{gefäße}}}(x \in \text{Wandkandidaten}(y)) = \max\left(0, \mathcal{V}_{F_{\text{gefäße}}}(x) - \mathcal{V}_{F_{atemweg}}(y)\right) \tag{28}$$

Die zugrundeliegende Idee ist, wie bereits erwähnt, das Abschwächen der Atemwegswände. Diese sollen nicht extrahiert werden. Dafür wurde in Gleichung 28 eine Formel definiert. Sei y ein Punkt auf der Mittellinie einer Tubularstruktur. Wir bestimmen die Wandkandidaten von y. Für diese Punkte berechnen wir den usprünglich berechneten Vesselwert (also für $\lambda_2 > 0$ oder $\lambda_3 > 0$) neu. Dieser ist nun das Maximum aus 0 und der Differenz aus dem zuvor bestimmten Vesselwert und dem Wert der Atemwegsvesselfunktion an Punkt y.

5 Implementierung

Nachdem wir uns nun mit Optimierungsmöglichkeiten beschäftigt haben, können wir uns der Implementierung widmen. In den nächsten Seiten wird der Code Zeile für Zeile anhand von Beispielen erläutert. Zu Beginn meiner Recherchen hat mir die Implementierung von [12] zum besseren Verständnis der Funktionsweise des Filters sehr geholfen, welche sich allerdings sehr von meinem im Folgenden vorgestellten Programmcode unterscheidet. Der vollständige Originalquelltext befindet sich im Anhang (Kapitel 8) inklusive ausführlicher Dokumentation. Die verwendete Programmiersprache ist MATLAB.

5.1 2D-Frangi-Filter

Wir beginnen zunächst mit den verwendeten Hilfsfunktionen im Zweidimensionalen.

5.1.1 convolution2D.m

Dazu müssen wir die Faltung im 2-Dimensionalen implementieren. In Abschnitt 2.2 ist die Funktionsweise der Faltung beispielhaft vorgerechnet. An dieser Stelle wird bereits darauf hingewiesen, dass in der Originalimplementierung diese Funktion aus Laufzeitgründen durch die in Matlab vordefinierte Funktion

```
imfilter (A,B,'conv');
```

ersetzt wird.
Das Prinzip der Funktion ist nicht schwer. Rufen wir uns Beispiel 2.39 aus Abschnitt 2.2 ins Gedächtnis. Dort waren:

$$A = \begin{pmatrix} 1 & 2 \\ 0 & 3 \end{pmatrix} \text{ und } B = \begin{pmatrix} 4 & 5 \\ 6 & 7 \end{pmatrix}$$

Die Funktion wird nun so aufgerufen:

```
C = convolution(A,B);
```

Im ersten Schritt der Faltung wird nun B um 180° rotiert. B ist dann:

$$B = \begin{pmatrix} 7 & 6 \\ 5 & 4 \end{pmatrix}$$

Dannach werden die Variablen *sizeAy*, *sizeAx*, *sizeBy*, *sizeBx* definiert, welche die Größen der Matrizen A und B angeben. *middleByRight* und *middleBxRight* sind die Größen, die die Anzahl der Einträge rechts der Mitte der Dimensionen von B angeben. *middleByLeft* und *middleBxLeft* geben dementsprechend die Anzahl der Einträge links der Mitte von B an. In unserem Beispiel gilt also speziell: *sizeAy*=*sizeAx*=*sizeBy*=*sizeBx*=2, *middleByRight*=*middleBxRight*=1, *middleByLeft*=*middleBxLeft*=0. Im Anschluss wird eine Matrix T erzeugt, die die Größen $(sizeAy + 2 \cdot sizeBy) \times (sizeAx + 2 \cdot sizeBx)$ hat. In der Mitte befinden sich die Werte von A, außen ist diese mit Nullen aufgefüllt. T sieht nun so aus:

$$T = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 2 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

startTx,*startTy*, *endTx* und *endTy* beschreiben dabei die Elemente von T, bei denen die Faltung mit B ausgeführt werden sollte. Die Nullen außen existieren lediglich zur Vermeidung von Randfällen. Nachdem nun alle Vorarbeiten abgeschlossen sind, kommen wir zur Ausführung der Faltung.

```
C=zeros(size(T));
for  y = startTy:endTy
     for  x = startTx:endTx
          tempA = T(y−middleByLeft:y+middleByRight,x−middleBxLeft:x+
               middleBxRight);
          prod = tempA.∗B;
          C(y,x)=sum(sum(prod));
     end
end
C=C(startTy:endTy,startTx:endTx);
```

Dieser Ausschnitt iteriert nun über alle relevanten Einträge von T, erstellt dafür die Variable *tempA*, die im Umkreis des Elementes $T(x, y)$ alle Elemente der Größe von A beinhaltet. Dabei sieht *tempA* im Schleifendurchlauf i jeweils so aus:

$$\underbrace{tempA = \begin{pmatrix} 1 & 2 \\ 0 & 3 \end{pmatrix}}_{i=1}, \underbrace{tempA = \begin{pmatrix} 2 & 0 \\ 3 & 0 \end{pmatrix}}_{i=2}, \underbrace{tempA = \begin{pmatrix} 0 & 3 \\ 0 & 0 \end{pmatrix}}_{i=3}, \underbrace{tempA = \begin{pmatrix} 3 & 0 \\ 0 & 0 \end{pmatrix}}_{i=4}$$

In jedem Iterationsschritt werden dabei die Elemente von *tempA* punktweise mit B multipliziert und in *prod* gesetzt.

$$\underbrace{prod = \begin{pmatrix} 7 & 12 \\ 0 & 12 \end{pmatrix}}_{i=1}, \underbrace{prod = \begin{pmatrix} 14 & 0 \\ 15 & 0 \end{pmatrix}}_{i=2}, \underbrace{prod = \begin{pmatrix} 0 & 18 \\ 0 & 0 \end{pmatrix}}_{i=3}, \underbrace{prod = \begin{pmatrix} 21 & 0 \\ 0 & 0 \end{pmatrix}}_{i=4}$$

Zuletzt werden alle Einträge des Ergebnisses aufeinander addiert. Dieser Wert wird dann in eine weitere Matrix C, die anfangs von selber Größe wie T ist, gespeichert.

$$C = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 31 & 29 & 0 & 0 \\ 0 & 0 & 18 & 21 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Die anfangs hinzugefügten Nullen werden wieder entfernt und wir erhalten folgendes Ergebnis von C:

$$C = \begin{pmatrix} 31 & 29 \\ 18 & 21 \end{pmatrix}$$

Wie bereits anfangs erwähnt wird diese Implementierung später aus Laufzeitgründen nicht berücksichtigt und durch *imfilter* ersetzt. Sie dient lediglich der Vollständigkeit und dem Verständnis. Eine Funktion, die sicher verwendet wird, ist die Berechnung der Hesse-Matrix eines Bildes, die im kommenden Absatz behandelt wird.

5.1.2 Hesse-Matrix2D.m

Die Funktion *hesseMatrix2D* hat drei Eingabeparameter:

- *Image*: Zweidimensionales Bildarray, bestehend aus Graustufen.
- s: Das derzeit angewandte Skalar.
- *gaussfactor*: Faktor, der von mir eingeführt wurde um die Faltung genauer ausführen zu können. (Default: 3)

Ein Beispielaufruf für ein bestimmtes Bild I kann so aussehen.

```
[XX, XY, YY] = hesseMatrix2D(I,2,3);
```

Die Rückgabewerte XX, XY und YY sind jeweils ein zweidimensionales Array mit den jeweiligen Ableitungen. XX ist die zweifache Ableitung in die erste Richtung, XY die Ableitung in die erste Richtung und anschließend in die zweite Richtung und YY die zweifache Ableitung in die zweite Richtung. Aber zunächst müssen wir die 2-Dimensionale Gauß'sche Ableitung implementieren. Rufen wir uns diese in der 2D-Version noch einmal ins Gedächtnis:

$$\mathcal{H}G_2(x,s) = \frac{1}{2\pi s^6} \begin{bmatrix} x_1^2 - s^2 & x_1 x_2 \\ x_1 x_2 & x_2^2 - s^2 \end{bmatrix} \exp\left(-\frac{x_1^2 + x_2^2}{2s^2}\right)$$

Zur Vermeidung von doppelten Code fassen wir den Vorfaktor $\frac{1}{2\pi s^6}$ und den Exponentialteil $\exp\left(-\frac{x_1^2+x_2^2}{2s^2}\right)$ zu einer Restfunktion zusammen, die sich in folgendem Codeausschnitt in der ersten Zeile befindet.

```
rest  = @(x,y) exp(-(x.^2+y.^2)/(2*s.^2))/(2*pi*s.^6);
dxx = @(x,y) (x.^2-s.^2).*rest(x,y);
dxy = @(x,y) x.*y.*rest(x,y);
dyy = @(x,y) (y.^2-s^2).*rest(x,y);
```

Die jeweiligen Unterschiede sind dann in den restlichen Zeilen implementiert.

Für die weitere Funktionsweise erstellen wir uns an dieser Stelle wieder ein Beispiel. Sei

$$I = \begin{pmatrix} 126 & 126 & 127 & 128 \\ 113 & 113 & 114 & 115 \\ 97 & 97 & 97 & 98 \\ 77 & 77 & 77 & 77 \end{pmatrix}, s = 1, gaussfactor = 2$$

Rufen wir nun die Funktion *hesseMatrix2D(I,s,gaussfactor)* auf, so stoßen wir nach der Deklaration der Funktionen auf diese Zeile:

```
[X, Y] = meshgrid(-gaussfactor*s:gaussfactor*s, -gaussfactor*s:gaussfactor*s);
hesFilXX = dxx(X,Y);
hesFilXY = dxy(X,Y);
hesFilYY = dyy(X,Y);
```

In unserem Fall werden dann zwei Matrizen erzeugt, die von folgender Form sind:

$$X = \begin{pmatrix} -2 & -1 & 0 & 1 & 2 \\ -2 & -1 & 0 & 1 & 2 \\ -2 & -1 & 0 & 1 & 2 \\ -2 & -1 & 0 & 1 & 2 \\ -2 & -1 & 0 & 1 & 2 \end{pmatrix}, Y = \begin{pmatrix} -2 & -2 & -2 & -2 & -2 \\ -1 & -1 & -1 & -1 & -1 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 2 & 2 & 2 & 2 & 2 \end{pmatrix}$$

Diese Werte werden nun in den Zeilen punktweise in die Gauß'schen Ableitungsfunktionen eingesetzt. Wir erhalten folgende Ergebnisse:

$$hessFilXX = \begin{pmatrix} 0.0087 & 0 & -0.0215 & 0 & 0.0087 \\ 0.0392 & 0 & -0.0965 & 0 & 0.0392 \\ 0.0646 & 0 & -0.1592 & 0 & 0.0646 \\ 0.0392 & 0 & -0.0965 & 0 & 0.0392 \\ 0.0087 & 0 & -0.0215 & 0 & 0.0087 \end{pmatrix},$$

$$hessFilXY = \begin{pmatrix} 0.0117 & 0.0261 & 0 & -0.0261 & -0.0117 \\ 0.0261 & 0.0585 & 0 & -0.0585 & -0.0261 \\ 0 & 0 & 0 & 0 & 0 \\ -0.0261 & -0.0585 & 0 & 0.0585 & 0.0261 \\ -0.0117 & -0.0261 & 0 & 0.0261 & 0.0117 \end{pmatrix},$$

$$hessFilYY = \begin{pmatrix} 0.0087 & 0.0392 & 0.0646 & 0.0392 & 0.0087 \\ 0 & 0 & 0 & 0 & 0 \\ -0.0215 & -0.0965 & -0.1592 & -0.0965 & -0.0215 \\ 0 & 0 & 0 & 0 & 0 \\ 0.0087 & 0.0392 & 0.0646 & 0.0392 & 0.0087 \end{pmatrix}$$

Die letzten drei Codezeilen sehen nun wie folgt aus:

```
hesXX = imfilter(double(Image), hesFilXX, 'conv');
hesXY = imfilter(double(Image), hesFilXY, 'conv');
hesYY = imfilter(double(Image), hesFilYY, 'conv');
```

An dieser Stelle werden die eben berechneten Gauß'schen Matrizen mit dem Originalbild I gefalten. Eine Implementierung der Faltung wurde in 5.1.1 vorgestellt. Die Hesse-Matrizen sehen nun endgültig so aus (auf zwei Nachkommastellen gerundet):

$$XX = \begin{pmatrix} -19.53 & -19.42 & -19.89 & -20.17 \\ -24.35 & -24.20 & -24.71 & -25.07 \\ -21.63 & -21.52 & -21.80 & -22.07 \\ -14.28 & -14.23 & -14.39 & -14.43 \end{pmatrix}, XY = \begin{pmatrix} 13.26 & 4.21 & -3.94 & -13.30 \\ 0.43 & 0.06 & -0.20 & -0.40 \\ -7.85 & -2.58 & 2.24 & 7.89 \\ -12.50 & -3.92 & 3.74 & 12.51 \end{pmatrix},$$

$$YY = \begin{pmatrix} -24.03 & -32.50 & -32.69 & -24.36 \\ -22.68 & -30.69 & -30.90 & -23.07 \\ -12.70 & -17.10 & -17.09 & -12.70 \\ -8.62 & -11.58 & -11.49 & -8.46 \end{pmatrix}$$

Nachdem wir die Hesse-Matrix berechnet haben, wenden wir uns der Berechnung von Eigenwerten der Hesse-Matrix zu.

5.1.3 getEigenValues2D.m

Wie in Satz 2.13 bestimmt, müssen wir, um die Eigenwerte einer Matrix zu berechen, die Nullstellen des charaktistischen Polynoms (Definition 2.11) bestimmen. Da wir das charakteristische Polynom vom Grad 2 ist und die Hesse-Matrix symmetrisch ist, können wir zur Berechnung der Nullstellen die allgemeine Lösungsformel für quadratische Gleichungen verwenden. Sei

$$\mathcal{H} := \begin{pmatrix} dxx & dxy \\ dxy & dyy \end{pmatrix} \text{ mit } dxx, dxy, dyy \in \mathbb{R}.$$

Dann gilt für das charakteristische Polynom $\chi_{\mathcal{H}}$:

$$\begin{aligned} \chi_{\mathcal{H}} &= \det \begin{pmatrix} \lambda - dxx & -dxy \\ -dxy & \lambda - dyy \end{pmatrix} = (\lambda - dxx) \cdot (\lambda - dyy) - dxy^2 \\ &= \lambda^2 - (dxx + dyy) \cdot \lambda + (dxx \cdot dyy - dxy^2) \end{aligned}$$

Berechnen wir nun die Diskriminante d der Lösungsformel:

$$d = \sqrt{(dxx + dyy)^2 - 4 \cdot (dxx \cdot dyy - dxy^2)} = \sqrt{(dxx - dyy)^2 + 4 \cdot dxy^2}$$

Dann gilt für die Eigenwerte

$$\lambda_{1,2} = \frac{dxx + dyy \pm d}{2}$$

Die folgenden Matlab-Code Zeilen implementieren diese Erkenntnis:

```
discriminant = sqrt((Dxx - Dyy).^2 + 4*Dxy.^2);
lam1 = (Dxx + Dyy + discriminant)/2;
lam2 = (Dxx + Dyy - discriminant)/2;
```

Die letzten vier Zeilen sortieren lediglich noch die Eigenwerte nach ihrem Absolutbetrag, sodass für weitere Berechnungen gilt: $|\lambda_1| \leq |\lambda_2|$

```
flip =abs(lam1)>abs(lam2);
temp = lam1;
lam1(flip)=lam2(flip);
lam2(flip)=temp(flip);
```

Nachdem wir uns nun den zwei bzw. drei verwendeten Hilfsfunktionen gewidmet haben, können wir uns nun der Hauptfunktion zuwenden.

5.1.4 frangi2D.m

Die Funktion *frangi2D* hat sieben Eingabeparameter:

- *Image*: Das 2-Dimensionale Eingabebild.
- *sigmas*: Ein Array gefüllt mit Skalaren, auf denen der Frangi-Filter angewandt werden sollte.
- *beta*, *c*: Das sind die beiden von Frangi definierten Grenzwerte (s. Abschnitt 3.3).
- *lindeberg*: Das ist der Lindeberg-Parameter.
- *brightness*: Dieser Parameter gibt an, ob nach hellen oder dunklen tubulären Strukturen gesucht werden sollte. Bei 0 soll nach dunklen Strukturen gesucht werden, bei 1 nach hellen.
- *gaussfaktor*: Dieser Parameter wurde von mir hinzugefügt, um den Grad der Faltung zu bestimmen. (3 per Default ist ein guter Wert.)

Nach einer kurzen Prüfung auf Validität der Eingabeparameter geht es dann mit der eigentlichen Implementierung des Frangi-Filters los. Erinnert man sich an das Einführungskapitel 3.1 der Funktionsweise dieses Filters zurück, so ist die Ausführung in vier Schritte unterteilt. In den folgenden Ausführungen werden die einzelnen Anwendungsschritte anhand des Originalbild eines Baumes in Abbildung 4 erläutert. Dabei wird die Funktion mit den Parametern *sigmas ={3,6,9,12,15}, β=0.5,*

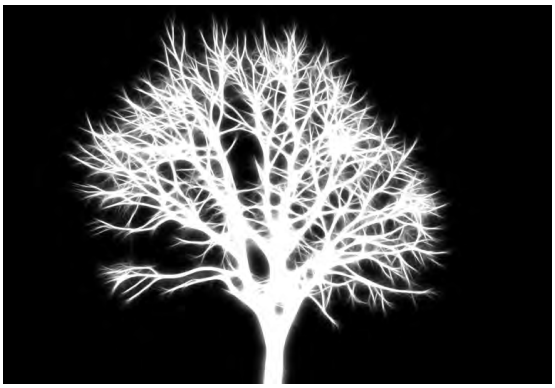


Abbildung 4: Bild eines Baumes (aus [13], invertiert und leicht verunschärft)

c=15, lindeberg=2 und gaussfaktor = 3 aufgerufen. Wir wollen die hellen tubulären Strukturen darstellen, also setzen wir den Parameter *brightness* auf 0.

- **Schritt 1:** Zweifache Ableitung des Bildes
  Um die zweite Ableitung eines Bildes zu berechnen, haben wir bereits eine Funktion implementiert, die in Unterkapitel 5.1.2 erläutert ist. Diese verwenden wir jetzt:

```
[Dxx,Dxy,Dyy] = hesseMatrix2D(Image,sigmas(i),gaussfaktor);
Dxx = (sigmas(i)^lindeberg)*Dxx;
Dxy = (sigmas(i)^lindeberg)*Dxy;
Dyy = (sigmas(i)^lindeberg)*Dyy;
```

  Wir rufen die Funktion *hesseMatrix2D* mit unseren Eingabebild *Image* und einem Skalar *sigmas(1)*, das in diesem Fall 3 ist, auf. Die drei erhaltenen zweifachenen Ableitungen in die jeweiligen Richtungen versehen wir nun noch mit den Faktor s^γ, der - wie in Kapitel 3.5 beschrieben - eine Familie von normalisierten Ableitungen eines Bildes definiert. Dies geschieht in den restlichen drei Codezeilen. Die Ableitungen visualisiert kann man in Abbildung 5 sehen. Schön zu beobachten sind die Unterschiede der einzelnen Ableitungen. Während in Abbildung 5a alle Stellen, bei denen in horizontaler Richtung große Änderungen auftreten, und in Abbildung 5c alle Stellen mit großer vertikaler Änderungsrate dick weiß umrandet sind, stellt man bei Abbildung 5b fest, dass überwiegend Stellen in Diagonalrichtung hervorgehoben sind. Das ist genau Sinn und Zweck der Ableitung.

(a) Dxx **(b)** Dxy **(c)** Dyy

Abbildung 5: Visuelle Darstellung der Ableitungsbilder mit Skalar $s = 3$

- **Schritt 2:** Eigenwertanalyse
  Nachdem nun die zweite Ableitung des Bildes bestimmt worden ist, können wir mit dem Analysieren der Eigenwerte fortfahren. Im folgenden Codeausschnitt werden zunächst die Eigenwerte mithilfe der Funktion, bekannt aus Abschnitt 5.1.3, ihren Absolutbeträgen nach sortiert in den Matrizen *lam1* und *lam2* gespeichert.

```
[lam1,lam2]=getEigenValues2D(Dxx,Dxy,Dyy);
lam2(lam2==0) = eps;
Rb = lam1./lam2;
S = sqrt(lam1.^2+lam2.^2);
```

  Im Anschluss werden alle Eigenwerte aus *lam2*, falls sie gleich Null sind, auf die kleinste Recheneinheit *eps* gesetzt, um eine Division durch Null zu vermeiden. In Abschnitt 3.3 wurde dies bereits erläutert. Die letzten beiden Zeilen sind die Konstanten $\mathcal{R}_\mathcal{B}$ und $\mathcal{S}$.

- **Schritt 3:** Anwendung der Vesselness-Funktion von Frangi
  Zum Schluss des ersten Iterationschrittes wird die 2D-Vesselness-Funktion von Frangi implementiert:

```
V0 = exp(-Rb.^2/(2*beta.^2)).*(1-exp(-S.^2/(2*c.^2)));
if brightness == 0
    V0(lam2>0)=0;
else
    V0(lam2<0)=0;
end
```

  Dabei ist die erste Zeile nichts anderes als der zweite Fall in $\mathcal{V}_0$. Der Rest unterscheidet lediglich noch, ob helle oder dunkle Strukturen hervorgehoben werden sollen.
  Zurück zu unserem Beispielbild. Der aktuelle Stand ist in Abbildung 6 abgedruckt. Man kann gut erkennen, dass Konturen bereits hervorgehoben sind,

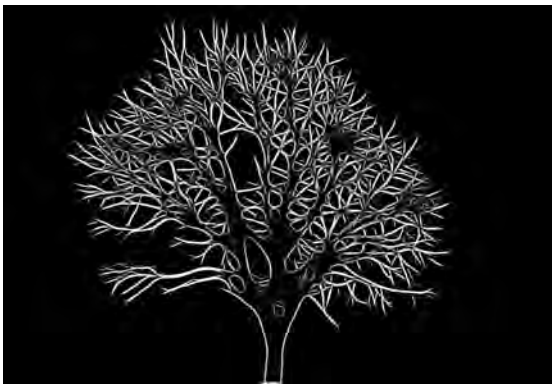


Abbildung 6: $\mathcal{V}_0$ nach dem ersten Iterationsschritt.

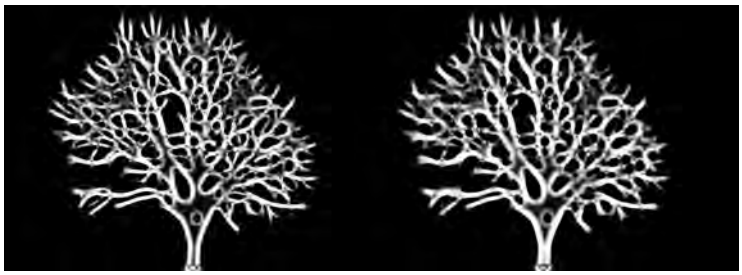
  bzw. dünne Äste sind bereits komplett weiß.

- **Schritt 4:** Berechnung des neuen Pixel/-Voxelwertes
  Als letztes müssen wir nun das Ausgabebild des Algorithmus berechnen. Dafür wenden wir die Schritte 1 bis 3 für jedes Skalar aus unserer *sigmas*-Menge an und setzen jedes Pixel auf das Maximum. Der folgende, letzte Codeausschnitt in diesem Kapitel zeigt (mehr in Pseudocode) den letzten Teil der Implementierung des zweidimensionalen Frangi-Filters.

```
MaxPicture = -Inf*ones(size(Image));
for i = 1:length(sigmas)
% Schritt 1;
% Schritt 2;
% Schritt 3;
MaxPicture = max(MaxPicture, V0);
end
```

  Die Teilabbildungen 7a bis 7e zeigen den Unterschied der einzelnen Skalare. Während bei einem geringen σ (s. Abb. 7a) die dünnsten Zweige hervorgehoben werden, werden bei einem hohen σ (s.Abb. 7e) die dicken Stämme hervorgehoben.

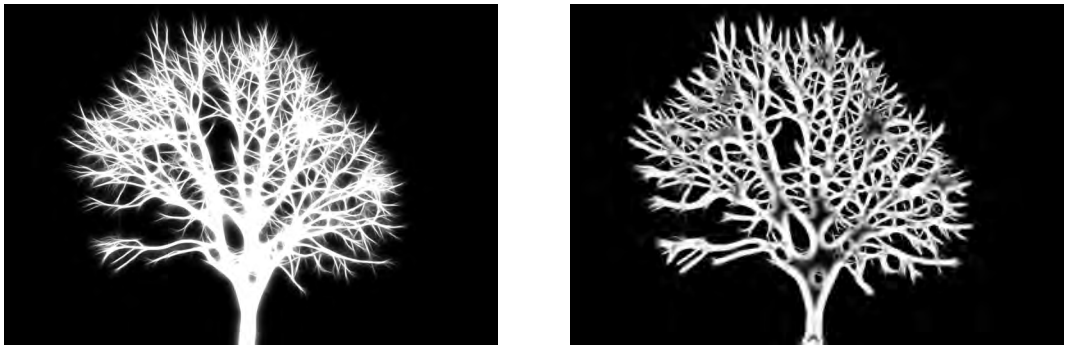
(a) $\sigma = 3$ **(b)** $\sigma = 6$ **(c)** $\sigma = 9$



(d) $\sigma = 12$ **(e)** $\sigma = 15$

Abbildung 7: Visuelle Darstellung der einzelnen Skalare $\sigma \in \{3, 6, 9, 12, 15\}$

Wie sieht nun das Endergebnis aus? Dieses ist in Abbildung 8 dargestellt. Man sieht, dass sowohl die dünnen Äste und Verzweigungen weiß hervorgehoben als auch die dicken Stämme sind.
Dies war die Implementierung des 2D-Frangi-Filters. Ähnliches Vorgehen gilt es nun beim Dreidimensionalen anzuwenden.



(a) Original **(b)** Nach Anwendung des Frangi-Filters

Abbildung 8: Gegenüberstellung des ursprünglichen Bildes und der mit dem Frangi-Filter bearbeiteten Bildes

5.2 3D-Frangi-Filter

Wie im Laufe dieser Arbeit schon mehrmals erwähnt, verläuft die 3-Dimensionale Herangehensweise ähnlich. Aus diesem Grund werden ähnliche Stellen zum 2-Dimensionalen weniger ausführlich behandelt. Auf neue oder abgeänderte Passagen wird ebenso genau wie im vorhergehenden Abschnitt eingegangen. Die erste Funktion, die wir besprechen müssen, ist wieder die zweite Ableitung, also in diesem Fall die *hesse-Matrix3D*.

5.2.1 hesseMatrix3D.m

Die Berechnung der dreidimensionalen Hesse-Matrix eines Volumens unterscheidet sich sehr wenig zur Berechnung der zweidimensionalen Hesse-Matrix eines Bildes. Deshalb wird in diesem Abschnitt, wie bereits erwähnt nur auf die Wesentlichen Unterschiede eingegangen. Sie besitzt als Eingabeparameter ein 3D-Volumen *Volume*, ein Skalar s und den Faktor *gaussfactor*, der bei der Faltung benötigt wird (*gaussfactor*). Wie im mathematischen Grundlagenteil 2.1 bereits behandelt, ist die zweite Ableitung der Gaußschen Normalverteilung so definiert:

$$\mathcal{H}G_3(x,s) = \frac{1}{\sqrt{8\cdot\pi^3 s^7}} \begin{bmatrix} x_1^2 - s^2 & x_1x_2 & x_1x_3 \\ x_1x_2 & x_2^2 - s^2 & x_2x_3 \\ x_1x_3 & x_2x_3 & x_3^2 - s^2 \end{bmatrix} \exp\left(-\frac{x_1^2+x_2^2+x_3^2}{2s^2}\right)$$

Die folgenden Zeilen Matlab-Code implementieren diese Ableitungen:

```
rest = @(x,y,z) exp(-(x.^2+y.^2+z.^2)/(2*s.^2))/(sqrt(8*pi^3)*s^7);
dxx = @(x,y,z) (x.^2-s.^2).*rest(x,y,z);
dxy = @(x,y,z) (x.*y).*rest(x,y,z);
dxz = @(x,y,z) (x.*z).*rest(x,y,z);
dyy = @(x,y,z) (y.^2-s.^2).*rest(x,y,z);
dyz = @(x,y,z) (y.*z).*rest(x,y,z);
dzz = @(x,y,z) (z.^2-s.^2).*rest(x,y,z);
```

Der Rest ist ähnlich zu dem Vorgehen in Kapitel 5.1.2. Es wird ein 3-Dimensionales *meshgrid* erstellt, anschließend die zweite Ableitung der Gaußfunktion für diese Werte berechnet und zu guter Letzt die Faltung mit der eben definierten Gauß-Hesse-Matrix berechnet. Wir erhalten sechs Ausgabeparameter. Für jeden Eintrag der Hesse-Matrix des oberen Dreiecks einen.
Um nun die Eigenwerte dieser Hesse-Matrix zu berechnen, betrachten wir im nächsten Abschnitt vorerst eine Hilfsfunktion.

5.2.2 get3DZeroPoints.m

Da wir bei 3×3-Matrizen ein charakteristisches Polynom dritten Grades haben, müssen wir, um die Eigenwerte zu berechnen, neue Wege gehen. Bei 2×2-Matrizen gibt es dafür die allgemeine Lösungsformel für quadratische Gleichungen. Für Polynome dritten Grades gibt es etwas ähnliches: Die Cardanische Formeln. Auf die mathematischen Hintergründe wird im weiteren Verlauf nicht eingegangen, weshalb sie nicht im mathematischen Grundlagenabschnitt 2.1 zu finden sind. Aber zunächst zur Funktion:
Die Funktion hat drei Eingabeparameter: a,b und c. Dies sind die Koeffizienten einer Polynomgleichung dritten Grades.

$$x^3 + ax^2 + bx + c = 0$$

Zur Berechnung der Nullstellen eines Polynomes dritten Grades gibt es vier verschiedene Konstellationen:

1. Eine reelle und zwei komplexe Lösungen. ($\Delta > 0$)
2. Eine doppelte reelle und eine einfache Lösung.($\Delta = 0$ und $p \neq 0$)

3. Eine dreifache reelle Lösung. ($\Delta = 0$ und $p = 0$)

4. Drei unterschiedliche reelle Lösungen. ($\Delta < 0$)

Für jeden Fall gibt es nun eigene Berechnungsmöglichkeiten. Dafür werden drei Variablen p, q, Δ definiert:

$$p := b - \frac{a}{3}$$
$$q := \frac{2a^3}{27} - \frac{ab}{3} + c$$
$$\Delta := \left(\frac{q}{2}\right)^2 + \left(\frac{p}{3}\right)^3$$

Da die Hesse-Matrix symmetrisch ist und somit nach Satz 2.18 nur reelle Lösungen hat, fällt der erste Fall von oben weg. Alle anderen sind aber weiterhin möglich. In Klammern stehen die Bedingungen unter welchen die Fälle eintreten.
In den folgenden Codezeilen werden die Variablen p, q, Δ implementiert.

```
p = b-a.^2./3;
q = 2*a.^3./27-a.*b./3+c;
delta = (q/2).^2+(p/3).^3;
```

Danach wird zunächst Fall 4 umgesetzt. Dies geschieht in den folgenden Zeilen:

```
acosinus = acos(-q./2.*sqrt(-27./p.^3));
predisk = sqrt(-4/3.*p);
x1 = predisk.*cos(1/3.*acosinus)-a./3;
x2 = -predisk.* cos(1/3.*acosinus+pi/3)-a./3;
x3 = -predisk.* cos(1/3*acosinus-pi/3)-a./3;
```

Die Werte von x_1, x_2 und x_3 werden nun als *double*-Wert zurückgegeben. Im Anschluss findet die Berechnung der Nullstelle statt, wenn eine dreifache reelle Nullstelle vorliegt. Dafür muss gelten: $\Delta = 0$ und $p = 0$. Die folgenden Zeilen sind für diesen Fall zuständig:

```
z = -a/3;
threeTime = delta==0 & p == 0;
x1(threeTime) = z(threeTime);
x2(threeTime) = z(threeTime);
x3(threeTime) = z(threeTime);
```

Im letzten für uns möglichen Fall (also eine zweifache und einfache reelle Nullstelle) ist der Fall, dass $\Delta = 0$ und $p \neq 0$. Der letzte Codeausschnitt in diesem Kapitel ist für diese Konstellation zuständig:

```
z1 = 3*q./p;
z2 = -3*q./(2*p);
twoPlusOne = delta==0 & p ~= 0;
x1(twoPlusOne) =z1(twoPlusOne);
x2(twoPlusOne) = z2(twoPlusOne);
x3(twoPlusOne) = z2(twoPlusOne);
```

Nachdem wir alle Fälle im Code abgearbeitet haben, können wir uns nun der Berechnung des charakteristischen Polynoms widmen.

5.2.3 getEigenValues3D.m

In Abschnitt 5.2.1 haben wir die Hesse-Matrix im 3-Dimensionalen und in 5.2.2 die Nullstellen einer Funktion dritten Grades mit ausschließlich reellen Lösungen berechnet. Kommen wir zurück zur Berechnung des charakteristischen Polynoms. Sei

$$\mathcal{H}V = \begin{pmatrix} dxx & dxy & dxz \\ dxy & dyy & dyz \\ dxz & dyz & dyy \end{pmatrix}$$

Dann gilt für das charakteristische Polynom $\chi_{\mathcal{H}V}$:

$$\begin{aligned} \det(\lambda \cdot I_3 - \mathcal{H}V) &= \det \begin{pmatrix} \lambda - dxx & -dxy & -dxz \\ -dxy & \lambda - dyy & -dyz \\ -dxz & -dyz & \lambda - dzz \end{pmatrix} \\ &= (\lambda - dxx)(\lambda - dyy)(\lambda - dzz) - 2 \cdot dxy \cdot dxz \cdot dyz \\ &+ (\lambda - dxx) \cdot dyz^2 + (\lambda - dyy) \cdot dxz^2 + (\lambda - dzz) \cdot dxy^2 \\ &= \lambda^3 \underbrace{-(dxx + dyy + dzz)}_{a} \lambda^2 \\ &\underbrace{+(dxx \cdot dyy + dxx \cdot dzz + dyy \cdot dzz - dxy^2 - dxz^2 - dyz^2)}_{b} \lambda \\ &\underbrace{- \det(\mathcal{H}V)}_{c} \end{aligned}$$

Das charakteristische Polynom ist nun berechnet und die Koeffizienten sind bestimmt. Diese benutzen wir zur Berechnung der Nullstellen:

```
a = -(Dxx+Dyy+Dzz);
b = Dxx.*Dyy + Dxx.*Dzz+Dyy.*Dzz-Dxy.^2-Dxz.^2-Dyz.^2;
c = -(Dxx.*Dyy.*Dzz + 2*Dxy.*Dxz.*Dyz - Dxy.^2.*Dzz-Dyz.^2.*Dxx-Dxz
    .^2.*Dyy);
[lam1, lam2, lam3] = get3DZeroPoints(a,b,c);
```

Die restlichen Zeilen des Codes sind im Wesentlichen nicht mehr neu für uns. Die Eigenwerte werden lediglich nach ihren Absolutbeträgen sortiert.
Jetzt können wir uns der Berechnung des 3D-Frangi-Filters widmen.

5.2.4 frangi3D.m

Die dreidimensionale Frangi-Funktion hat im Vergleich zum zweidimensionalen Pendant einen Eingabeparameter mehr: *alpha*. Dieser hat dieselbe Funktion wie *beta* und ist auch in Abschnitt 3.3 ausführlich erläutert. Das Vorgehen ist nahezu komplett identisch. Man berechnet für verschiedene Skalare die Hesse-Matrix, versetzt sie mit dem Lindeberg-Parameter und berechnet davon die Eigenwerte. Erst ab der Berechnung der Konstanten $\mathcal{R}_A$,$\mathcal{R}_B$ und $\mathcal{S}$ und der Definition von Frangis 3D-Vesselnessfunktion wird die Implementierung angepasst:

```
    Rb = abs(lam1)./sqrt(abs(lam2).*(abs(lam3)));
    Ra = abs(lam2)./abs(lam3);
    S = sqrt(lam1.^2+lam2.^2+lam3.^2);
```

```
V0 = (1-exp(-Ra.^2/(2*alpha.^2))).*exp(-Rb.^2/(2*beta.^2)).*(1-exp(-S
    .^2/(2*c.^2)));
```

Die letzte Anpassung geschieht bei der Berechnung der auszuschließenden Voxel. Hier wird die Bedingung auf die Eigenwerte λ_2 und λ_3 erweitert:

```
% The other case of the vesselness function.
if brightness == 0
    V0(lam2>0 | lam3>0)=0;
else
    V0(lam2<0 | lam3<0)=0;
end
```

Nachdem wir uns meine Implementierung genau angeschaut haben, können wir uns nun im folgenden Kapitel mit der Analyse der Anwendung, sowie mit den Stärken und Schwächen auseinandersetzen.

6 Analysen und Experimente am Frangi-Filter

In den nächsten beiden Unterkapiteln wird die Anwendung des Filters auf verschiedenste Arten geprüft. An dieser Stelle sei bereits gesagt, dass in diesem Abschnitt der Fokus deutlich mehr auf der 2-Dimensionalen als auf der 3-Dimensionalen Ausführung liegt. Das hängt mit der leichteren Visualisierung von Bildern im Vergleich zu Volumen zusammen. Die Erkenntnisse lassen sich mithilfe der 2D-Variante aussagekräftiger darstellen. In einer Live-Vorführung mit einem ordentlichen Volumen-Renderer könnte die 3D-Version erheblich besser analysiert werden. Nichts desto trotz wird auf die 3D-Version - wenn auch in deutlich geringerem Maßen - eingegangen.

6.1 Analyse der Implementierung mit unterschiedlichen Eingabeparametern

Im ersten Schritt der Analyse wenden wir den Filter auf ein und dasselbe Bild an. Lediglich die Eingabeparameter werden verändert. Bevor wir mit der eigentlichen Analyse beginnen, sollten wir kurz einen Blick auf die Standardparameter werfen, die wir - wenn nicht anders definiert- verwenden, wenn die Funktion *frangi2D* aufgerufen wird:

- $sigmas = 1, ..., 15$
- $\beta = 0.5$
- $c = 15$
- $lindeberg = 2$
- $gaussfactor = 3$
- $brightness = 0$

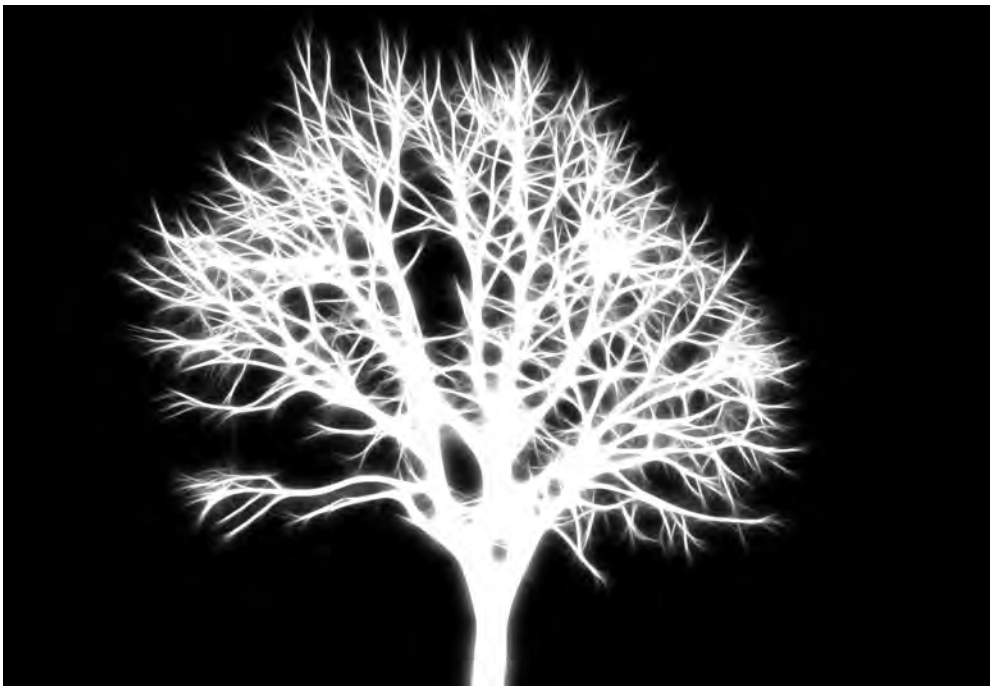


Abbildung 9: Originalbild, das für alle Untersuchungen in diesem Abschnitt verwendet wird.

Als Ursprungsbild nehmen wir das aus Abbildung 4 des Implementierungskapitels 5. In Abb. 9 ist es noch einmal größer abgedruckt. Dort wurde bereits die Entwicklung des neuen Bildes anhand verschiedener Skalare pro Iterationsschritt der Funktion erläutert. Ergänzend sollte man zu dieser Entwicklung noch feststellen, dass die Skalare geschickt gewählt werden sollten. Rein theoretisch könnte sich der Anwender nun denken: Ich wende den Filter nun mit möglichst vielen und unterschiedlich großen Skalaren σ an. In Abbildung 10 wurde der Filter mit einer Skalarmenge von $\sigma = 1, ..., 100$ angewandt. Dies sollte aber vermieden werden. Der Grund lautet: Die dickeren Äste verschwimmen gänzlich und die dünneren Zweige sind in manchen Fällen nur noch leicht zu erahnen oder werden gänzlich mit weiß überdeckt. Außerdem gilt folgende Faustregel: Mit je mehr Skalare die Funktion ausgeführt wird, desto länger ist deren Laufzeit. Deshalb muss die Menge der Skalare wohl bedacht sein. Diese sollte sich nach dem Zweck und der zu untersuchenden tubulären Struktur richten.

Den nächsten Parameter, den wir unter die Lupe nehmen wollen, ist der *brightness*-Operator. Bisher haben wir immer nur versucht helle Strukturen auf dunklem Hintergrund hervorzuheben. Nun versuchen wir dunkle auf hellem Fundament zu betonen. Dafür bilden wir das Komplement des Ursprungsbildes aus Abbildung 9. Dieses ist in 11a dargestellt. Alle weißen Äste sind nun schwarz und der schwarze Hintergrund ist nun weiß. Aber was passiert nun, wenn man den Filter mit den Standardparametern und *brightness* $= 0$ darauf laufen lässt? Interessanterweise ändert sich nicht viel am Ergebnis im Vergleich zur Anwendung mit *brightness*$= 1$. In Abb. 11b ist das Resultat abgedruckt. Meine Empfehlung ist bei Hervorhebung von dunklen Strukturen anschließend noch das Komplement zu bilden, um dem Originalbild etwas näher zu kommen. In Abb. 11c wäre jenes wiedergegeben. Zwei

Parameter, die wir noch betrachten sollten sind β und c. Dafür ist es hilfreich eine Serie von Bildern mit unterschiedlichen β-Werten zu betrachten. Kehren wir zurück zu den Standardparametern. Wir wenden nun den Filter mit den Parametern $\beta \in \{0.01, 0.02, 0.05, 0.10, 0.15, 0.30, 0.50, 0.70\}$ an. Die Entwicklung der Resultate ist in den Abbildungen 12a bis 12h dargestellt. Dabei stellt man fest, dass bei einem geringen β (Abb. 12a bis 12c) die Konturen nur schwach erkannt werden. Erst ab einem Wert von $\beta = 0.10$ werden die tubulären Strukturen erkennbarer dargestellt. In den Figuren 12e bis 12g wird die Tubularstruktur immer deutlicher und kräftiger. Die letzte große Erkenntnis ist, dass im Übergang von 12g auf 12h keine großartigen Änderungen mehr geschehen. Das lässt darauf schließen, dass der Wert $\beta = 0.50$ ideal für die Anwendung des Filters ist, was im Übrigen genau derjenige Wert ist, der in Abschnitt 3.4 von Frangi vorgeschlagen wurde.

Eine letzte Analyse, die wir anstellen wollen, ist die Abhängigkeit vom Parameter c. Dafür fertigen wir wieder eine Serie mit unterschiedlichen Werten für c an. ($c \in \{0, 1, 2, 3, 10, 15, 30, 100\}$)
Die Ergebnisse dieser sind wieder in den Teilabbildungen 13a bis 13h aufgelistet. Dabei kann man Folgendes sehr deutlich erkennen: Je höher der c-Wert ist, desto dunkler ist das Resultat. Während in 13a alle möglichen, noch so kleinsten Details hervorgehoben werden, verschwinden diese im Verlauf immer mehr. In 13c werden sämtliche unnötige Details nicht mehr hervorgehoben und in 13f sind diese bereits gänzlich verschwunden. Erhöht man den Wert von c noch weiter, so erblasst das Bild wieder (vgl. 13g) und erlischt in 13h nahezu gänzlich. An dieser Stelle sei erwähnt, dass der c-Wert auch spezifisch vom Objekt abhängen kann.

Halten wir dies fest. Die Idealwerte lauten in unserem Fall:

- $\beta = 0.50$
- $c = 15$

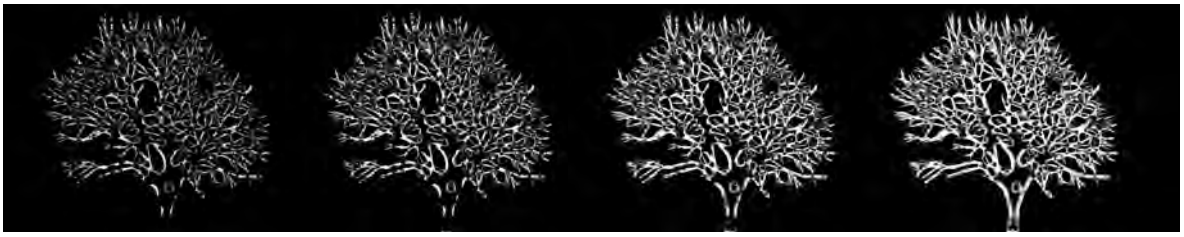
Die Parameter *lindeberg* und *gaussfactor* werden nicht weiter beachtet.

Jetzt haben wir uns genug mit der 2D-Variante beschäftigt. Es ist Zeit für einen kurzen Blick auf die Dreidimensionale Ausführung. Aufgrund der schwierigen Darstellbarkeit eines 3D-Objekts auf einem 2D-Objekt und der ähnlichen Funktion wird

Abbildung 10: Eine Menge mit großen Skalaren muss nicht immer von Vorteil sein.

(a) Tubuläre Struktur mit dunklen Verzweigungen **(b)** Bild nach Anwendung des Frangi-Filters **(c)** Bild nach Bildung des Komplementes

Abbildung 11: Anwendung des Filters mit dunklen tubulären Strukturen



(a) $\beta = 0.01$ **(b)** $\beta = 0.02$ **(c)** $\beta = 0.05$ **(d)** $\beta = 0.10$

(e) $\beta = 0.15$ **(f)** $\beta = 0.30$ **(g)** $\beta = 0.50$ **(h)** $\beta = 0.70$

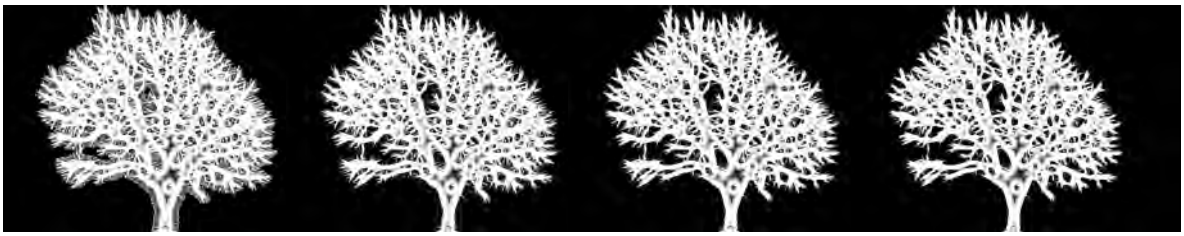
Abbildung 12: Anwendung des Filters mit verschiedenen β-Werten

dieser Teil nur im Gröbsten behandelt. In Abbildung 14a ist unsere dreidimensionale tubuläre Beispielstruktur zu sehen. Wir wenden hierzu unsere *frangi3D*-Funktion mit den Standardparametern

- $\alpha = 0.5$
- $\beta = 0.5$
- $c = 500$
- $lindeberg = 2$
- $gaussfactor = 3$
- $brightness = 0$

an. Lediglich die Skalare ändern wir ab. Im ersten Versuch (Abb. 14b) verwenden wir nur das Skalar $\sigma = 1$. Dort lässt sich eine leichte Hervorhebung der Konturen erkennen. Vergrößern wir die Skalarmenge, dann werden auch die dickeren Strukturen besser dargestellt (vgl.14c). Wie bereits erwähnt, könnte man die 3D-Variante mit einem Volumen-Renderer beeindruckender darstellen.

Nachdem wir nun die Eingabeparameter des Filters analysiert haben, können wir uns nun den Stärken und Schwächen des Filters zuwenden.



(a) $c = 0$ **(b)** $c = 1$ **(c)** $c = 2$ **(d)** $c = 3$

(e) $c = 10$ **(f)** $c = 15$ **(g)** $c = 30$ **(h)** $c = 100$

Abbildung 13: Anwendung des Filters mit verschiedenen c-Werten

(a) Tubuläre 3D Struktur **(b)** Bild nach Anwendung des Frangi-Filters mit Skalar $\sigma = 1$ **(c)** Bild nach Anwendung des Frangi-Filters mit den Skalaren $\sigma = 1, ..., 5$

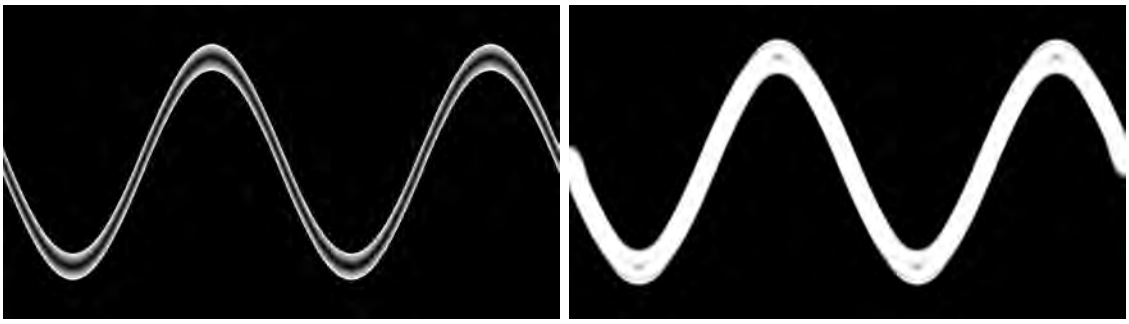
Abbildung 14: Ergebnis des 3D-Filters mit unterschiedlichen Skalaren

6.2 Stärken und Schwächen des Frangi-Filters

Dazu werden wir mit fünf unterschiedlichen Objekten experimentieren und anhand dieser die Tauglichkeit des Filters analysieren. Aus den vorangegangenen Kapiteln wissen wir bereits, dass der Filter gut für Bilder mit gefäßartigen Strukturen funktioniert. Deshalb wollen wir nun andere Muster filtern. Vorab sei gesagt, dass bei Anwendung des Filters immer versucht wurde die Parameter $sigmas, \beta$ und c so zu wählen, dass das Ergebnis optimal ist, die aber im weiteren Laufe der Arbeit nicht explizit angegeben werden. Es kann natürlich je nach Situation auch bessere Parameterkonstellationen geben.

Das erste Muster, an dem wir unseren Filter ausprobieren wollen, ist eine doppelte sinus-Kurve auf schwarzem Hintergrund. Das in Abbildung 15 dargestellte wellenförmige Muster zeigt im Vergleich zu den folgenden drei Beispielbildern am meisten Ähnlichkeiten zu einer Tubularstruktur auf. (Das letzte Objekt läuft außer Konkurrenz.) Während im Originalbild 15a noch eine doppelte Sinuskurve, die zur Mitte immer mehr ins Schwarze verschwimmt zu sehen ist, führt der Filter diese weißen Randlinien der Kurven in 15b zu einer einzigen zusammen. Diese Verstärkung ist genau Sinn und Zweck des Versuchs. Man kann nun sehr sicher sagen, dass das Er-

kennen von leicht verschwimmenden Strukturen eine Stärke des Frangi-Filters ist.

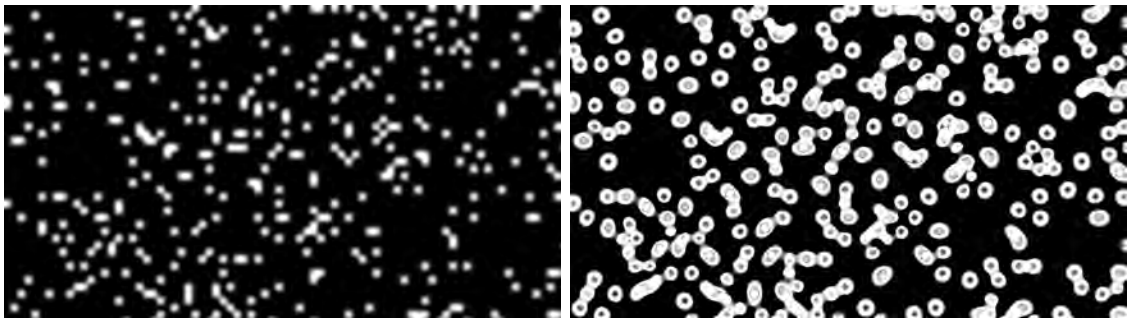


(a) Originalbild:
Doppelte sinus-Kurve auf schwarzem Hintergrund

(b) Mit Frangi:
Die doppelte Kurve wurde zu einer einzigen dicken Kurve verstärkt.

Abbildung 15

Als nächstes wollen wir uns ein schwarzes Bild mit unterschiedlich verteilten weißen, unscharfen Punkten ansehen (vgl. Abb. 16). Das in 16a dargestellte Originalbild ähnelt mehr einem Sternenhimmel. Die Herausforderung an den Frangi-Filter könnte an dieser Stelle das Fehlen einer tubulären Struktur sein. Es liegen zwar Punkte enger beieinander als andere, aber das liegt daran, dass jene zufällig erstellt und verteilt worden sind. Es existiert also kein Schema, mit dem dieses Bild berechnet hätte werden können. Aber die interessantere Frage ist nun: Was macht der Filter mit den ungeordneten und unscharfen Punkten? In Abbildung 16b ist das Ergebnis ersichtlich. Aus den verschwommenen, kleinen Punkten entstehen wesentlich größere, klare Kreise. Bei genauerem Betrachten dieser Kreise fällt auf, dass alle denselben Aufbau haben. Alle sind von einem dicken weißem Rand umgeben und haben einen kleinen schwarzen Fleck in der Mitte, der ungefähr die Größe des Originals besitzt. Eine weitere Auffälligkeit kann man vor allem im unteren Bildbereich feststellen. Dort liegen viele Punkte eng beieinander. Was macht Frangi damit? Die kleinen Kreise werden im gefilterten Bild bereits als tubuläre Struktur angenommen und verbunden. Zusammenfassend kann man sagen: Im Vergleich zur sinus-Kurve aus Abbildung 15 funktioniert der Filter bei zufällig verteilten Punkten noch gut. Allerdings fällt die Genauigkeit des Filters durch überdimensional große Proportionalität der dargestellten Kreise und das Zusammenführen von engen Punkten deutlich ab. Trotzdem kann man zumindest das Erkennen bzw. Filtern von einzelnen kleinen Punkten noch als Stärke des Filters aufführen. Die Schwäche liegt dann bei eng zusammenliegenden Tupfen.
Nachdem wir nun festgestellt haben, dass einzelne kleine Punkte vom Filter erkannt werden, können wir uns nun der Frage widmen, was mit einzelnen großen Objekten (z.B. Kreisen) passiert. Dazu betrachten wir zunächst das Originalbild in Abbildung 17a. Dort sind 16 graue Kreise in je vier Spalten und Zeilen angeordnet. Anzuführen ist auch, dass die Kreise in vertikaler Sichtweise enger zueinander stehen als in horizontaler Ebene. Warum wir uns nun im Gegensatz zu den vorherigen Beispielen Strukturen mit grauen anstelle von weißen Objekten ansehen, hat einen einfachen Grund: Eine Stärke des Filters, die anhand dieses Experiments gezeigt werden soll, ist die Farbunabhängigkeit von der Originalstruktur. Denn, wie man im Resultat 17b sieht, werden die grauen Stellen zumindest am Rand weiß hervorgehoben ab-

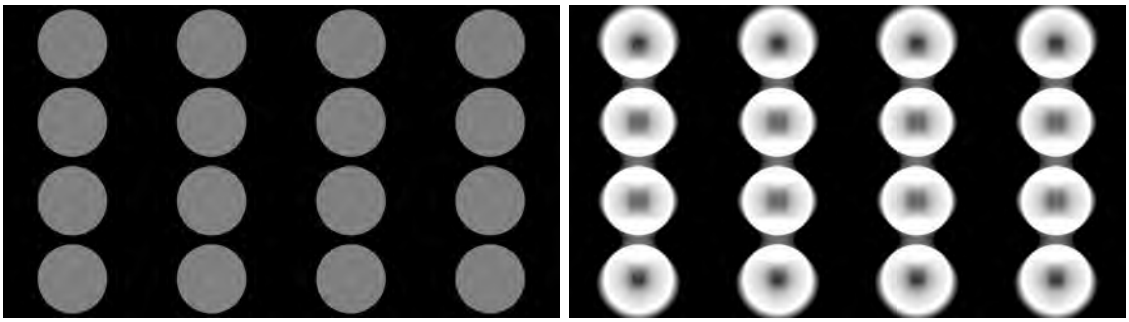


(a) Originalbild:
Ungleichmäßig verteilte verschwommene weiße Punkte auf schwarzem Hintergrund

(b) Mit Frangi:
Die Punkte werden scharf, aber auch überproportioniert dargestellt

Abbildung 16

gebildet. Allerdings kann man auch erkennen, dass in jedem einzelnen Kreis des Ergebnisses die Ränder verschwommen sind. Auffallend ist auch, wenn man sich die Erkenntnis aus dem vorangegangen Beispiel noch einmal ins Gedächtnis ruft, dass ebenfalls in der Mitte der Kreise bzw. Punkte ein schwarzer Fleck vorhanden ist. Zudem kann man auch feststellen, dass im Vergleich zu vorher die eng aneinanderliegenden Kreise (also die in horizontaler Richtung) leicht verbunden werden. Mögen zwar die angewandten Skalare in diesem Beispiel etwas zu groß gewählt worden sein, so offenbart dieses Beispiel eine weitere Schwäche des Frangi-Filters. Die Genauigkeit des Filters schwindet mit zunehmender Größe der Skalarwerte. Dies wurde bereits in Abschnitt 6.1 angeschnitten. Also könnte man ja eigentlich sagen: Da der Filter bei diesem Bild ungenau wird, genau dann wenn man die Skalarmenge vergrößert, sollte man es einfach mit einer kleineren Menge versuchen. Der Grundgedanke dahinter macht durchaus Sinn. Der Rand wird schärfer dargestellt. Allerdings hat diese Modifizierung eine Konsequenz. Die schwarzen Flecken in der Mitte jedes einzelnen Kreises werden größer, was auch einen logischen Grund hat. Je kleiner das Skalar, desto geringer ist der Bereich, der beim Berechnen der Hesse-Matrix miteinbezogen wird, wodurch im Inneren die Änderungen nahe null sind. Daraus folgt, dass sich die Eigenwerte auch nahe der Null sind und somit keine hervorzuhebende Struktur erkannt wird. Das Finden einer (nahezu) optimalen Skalarmenge kann als weitere Schwäche aufgeführt werden. Um die letzten Zeilen zusammenzufassen: Eng beieinander liegende große Kreise sollte man nicht mithilfe des Frangi-Filters versuchen hervorzuheben. Dunklere Konturen auf noch dunklerem Hintergrund eignen sich allerdings schon im Allgemeinen.
Was mit kleinen und großen Kreis-bzw. Punktmustern passiert, haben wir anschaulich dargestellt. Nun können wir uns einer anderen interessanten Struktur widmen. Die Rede ist von einem Schachbrettmuster. Die interessanteste Frage, die sich vor Anwendung des Filters auf das in Abb. 18a sich befindende Originalbild ist: Welchen Parameter wähle ich für *brightness*? Da die hellen, wie auch die dunklen Teile des Bildes exakt gleich verteilt sind, kann man an dieser Stelle nicht vernünftig entscheiden welcher Parameter besser ist. Aus diesem Grund wurde der Filter für beide Szenarien (vgl. 18b und 18c) angewandt. Was beim ersten Vergleich schnell auffällt ist, dass beim mit *brightness*=0 gefilterten Bild, das Karo-Muster exakt dem Originalbild entspricht, während es beim anderen invertiert ist. Sprich alle schwarzen Felder sind dort weiß und umgekehrt. Dies ist aber unter Miteinbezug vom vorherigen

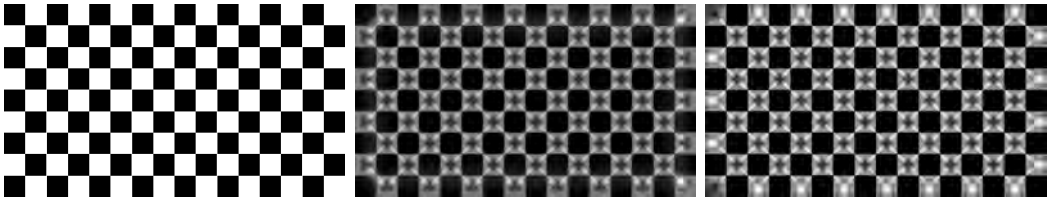


(a) Originalbild:
Gleich verteilte große grau gefüllte Kreise auf schwarzem Hintergrund

(b) Mit Frangi:
Die Punkte werden erkannt, aber nicht mehr mit allergrößter Genauigkeit dargestellt

Abbildung 17

Unterabschnitt dieses Kapitels nicht sonderlich überraschend, da bei *brightness*=1 die dunklen Strukturen hervorgehoben werden sollten. Viel auffallender sind aber die hinzugekommenen Diagonallinien in beiden Anwendungen. In 18b sind diese an der Stelle, die im Originalbild weiß sind und in 18c sind sie dort, wo die schwarzen Kästchen ursprünglich waren. Die Erklärung der Existenz dieser Diagonallinien ist für beide Beispiele dieselbe. Nehmen wir uns nun einen Teil mit diesen Diagonalen aus 18b heraus. Dabei gilt bei Schachbrettmustern: Während links, recht, oben und unten von einem Kästchen immer Felder der anderen Farbe sind, so sind auf den Diagonalen Felder der gleichen Farbe. Und auf diese Diagonalfelder reagiert nun der Filter. Er „meint“, dass diese zur existierenden Struktur gehören und verstärkt die Diagonalen. Würden die Skalarwerte deutlich erhöht werden, so wären die Diagonalen deutlich dicker dargestellt. Im Allgemeinen kann man aber klar sagen: Eine Anwendung auf Schachbrettmustern, auf Strukturen mit nahezu gleich großer Schwarz/Weiß-Auslastung sowie auf rechteckigen Grundgerüsten ist nicht zu empfehlen und muss als Schwäche aufgeführt werden.



(a) Originalbild:
Schachbrettmuster

(b) Mit Frangi
(*brightness*=0)

(c) Mit Frangi
(*brightness*=1)

Abbildung 18

Nachdem wir nun eine sinus-Kurve, zufällige Punkte, geordnete große graue Kreise und auch ein Schachbrettmuster analysiert haben, können wir uns an dieser Stelle einem „Bonus“ widmen. Denn bisher haben wir stets Schwarz-Weiß-Bilder betrachtet. Aber was passiert mit Farbbildern? Funktioniert der Filter dort auch? Dazu habe ich nun in Abbildung 19 die tubuläre Struktur aus 4 in eine möglichst bunte und alle Farben umfassende Tubularstruktur umgewandelt und den Frangi-Filter darauf angewandt. Dazu muss vorab erwähnt werden, dass Farbbilder einen anderen Aufbau

als Schwarz-Weiß-Bilder haben. Während bei letzteren in dem zwei-dimensionalen Array nur der Grauwert gespeichert wird, wird bei Farbbildern ein 3-Tupel der RGB-Werte hinterlegt. Die Anwendung des Frangi-Filters läuft nun leicht verändert ab. Sämtliche Parameter, die zur Anwendung des Filters benötigt werden, bleiben gleich. Lediglich die Filterung wird dreimal hintereinander ausgeführt. Beim ersten Durchlauf werden alle Rot-Werte gefiltert, beim Zweiten alle Grün-Werte und beim Dritten alle Blau-Werte. Am Schluss werden die drei Ergebnisse wieder zu einem Bild zusammengesetzt und ausgegeben.
Die alles entscheidende Frage lautet nun: Funktioniert der Farb-Frangi-Filter? Insofern eine tubuläre Struktur vorliegt, kann der Filter einwandfrei auch auf Farbbildern angewandt werden. Wie man in Abbildung 19b sehen kann, sind die gefäßartigen Strukturen gut hervorgehoben. Auch die farbliche Blässe im Originalbild bereitet dem Filter keine Probleme. Fazit: Der Filter kann auch farbige tubuläre Strukturen erkennen und hervorheben.

(a) Originalbild:
Alle weißen Flächen aus Abbildung 4 wurden durch bunte und vielfältige Flächen ersetzt.

(b) Mit Frangi:
Der Filter erkennt auch die Farbunterschiede und hebt die in 19a noch relativ verblassten Farben deutlich hervor.

Abbildung 19

Die Stärken und Schwächen sind in Tabelle 3 noch einmal zusammengefasst.

7 Zusammenfassung

Fassen wir die wesentlichsten Punkte zusammen. Ziel dieser Arbeit war eine detaillierte Erläuterung der Funktionsweise zu geben. In Kapitel 2 wurden die mathematischen, sowie bildverarbeitungsspezifischen Grundlagen des Frangi-Filters kennengelernt. Dazu zählt die Berechnung von Eigenwerten einer Matrix. Außerdem können wir nun die zweite Ableitung von mehrdimensionalen Funktionen berechnen. Des Weiteren wurde gezeigt, dass die Frobenius-Norm einer Matrix unter gewissen Umständen die euklidische Norm ihrer Eigenwerte ist. Zudem widmeten wir uns der Definition eines Ellipsoiden. Mit der Einführung der Faltung lernten wir ein wichtiges Werkzeug der Bildverarbeitung kennen.

Diese Mittel verwendeten wir nun im wichtigsten Kapitel 3 dieser Arbeit. Dort wurde die Funktionsweise des Frangi-Filters im Detail besprochen. Nach einer kur-

Stärken	Schwächen
• Erkennen von tubulären Strukturen. • Schärfen von weit auseinanderliegenden, kleinen und unscharfen Punkten. • Anwendung auf Farbbildern bei vorhandener Tubularstruktur funktioniert. • Unterscheidung von minimal helleren Umgebungen klappt.	• Kleine eng zusammenliegende Punkte werden als Tubularstruktur aufgefasst. • Große kreisförmige Muster werden unscharf und ungenügend erkannt. • Anwendung mit großen und vielen Skalarwerten erhöht die Laufzeit erheblich. • Rechteckformen werden ebenfalls nicht erkannt. • Zusammenführung von Objekten, die nicht zusammengehören. • Auswahl der verwendeten Skalare is oft schwierig.

Tabelle 3: Auflistung der Stärken und Schwächen des Frangi-Filters

zen Einführung behandelten wir die Eigenwerte eines Bildes nach dem Prinzip von Frangi und analysierten die erhaltenen Eigenwerte im Anschluss. Dort wurde erläutert, welche Voraussetzungen optimal für eine tubuläre Struktur ist. Mithilfe der Vesselness-Funktion konnten wir nun diese Erkenntnisse über die Eigenwerte nutzen und einen neuen Pixel- bzw. Voxelwert für ein gegebenes Skalar bestimmen. Wir lernten außerdem, dass der Filter sowohl für das Zweidimensionale als auch für das Dreidimensionale funktioniert. Der einzige Unterschied ist die Definition der Vesselness-Funktion. Im letzten Teil dieses Kapitels beschäftigten wir uns noch mit der Berechnung des endgültigen Bild- oder Volumenpunktes.

Der nächste Abschnitt 4 sollte dann Optimierungsmöglichkeiten skizzieren. Wir sahen einen Ansatz, der den Frangi-Filter mithilfe eines neuronalen Netzwerkes ausführt. Außerdem behandelten wir überblicksartig eine Variante zur Dämpfung von Atemwegswänden in Gefäßstrukturen. Danach untersuchten wir die beigefügte Implementierung im Detail (Kapitel 5). Dort wurde zunächst die 2D-Variante in Einzelheiten erklärt. Es folgte eine Darstellung der Abänderungen zur Dreidimensionalen Option. Zuletzt wurde im Kapitel 6 der Filter überwiegend im Zweidimensionalen analysiert. Mithilfe eines Beispielbildes konnte man sämtliche Parameterkonstellationen unter die Lupe nehmen. Anschließend wurde durch Experimente versucht herauszufinden, auf welchen Objekten der Filter hervorragend bzw. schlecht funktioniert. Dabei lautete eine Erkenntnis, dass dieser wie zu erwarten bei Tubularstrukturen (sowohl in Graustufen als auch in Farbumgebungen) sehr gut arbeitet. Bei großen Kreis- oder Schachbrettmustern und anderen Strukturen ist die Anwendung des Filters nicht geeignet und zu empfehlen.

Abschließend kann man sagen, dass der Frangi-Filter ein funktionierendes Konzept

ist. Er hebt die tubulären Strukturen gut hervor und kann auch schwache Graustufenunterschiede problemlos erkennen. Ein Aspekt für zukünftige Arbeiten könnte sein, dass die Auswahl der Skalarmengen nicht von Hand erfolgen muss, sondern automatisch erfolgt.

8 Anhang - Implementierungsquelltext

8.1 2D-Frangi-Filter

Datei 8.1: convolution2D.m

```
%% Algorithm for performing the "convolution" of two matrices in R^2.
%% "A" and "B" are two R^2 matrices that have to be convoluted.
function C = convolution2D(A,B)
% Rotate B by 180 ° around its own axis. This makes the calculation easier.
B= rot90(B,2);

% Calculate the x-sizes and y-sizes of matrice A.
sizeA = size(A);
sizeAy=sizeA(1);
sizeAx=sizeA(2);

% Calculate the x-sizes and y-sizes of matrice B.
sizeB = size(B);
sizeBy=sizeB(1);
sizeBx=sizeB(2);

% Find the center of the y-size and x-size on the right and left of B.
% (If the size is even, half will be rounded down to the smaller center.)
middleByRight = floor(sizeBy/2);
middleByLeft = sizeBy - middleByRight-1;
middleBxRight = floor(sizeBx/2);
middleBxLeft = sizeBx - middleBxRight-1;

% Creates a matrix T of size 2 * x-size of B + the x-size of A and
% 2 * y-size of B + the y-size of A. A is set in the middle of T.
startTy = sizeBy+1;
endTy = sizeBy+sizeAy;
startTx = sizeBx+1;
endTx = sizeBx+sizeAx;
T = zeros(sizeAy + 2* sizeBy, sizeAx + 2*sizeBx);
T(startTy:endTy,startTx:endTx) = A;

% Calculates the convolution.
C=zeros(size(T));
for y = startTy:endTy
    for x = startTx:endTx
        tempA = T(y-middleByLeft:y+middleByRight,x-middleBxLeft:x+
            middleBxRight);
        prod = tempA.*B;
        C(y,x)=sum(sum(prod));
    end
end
```

```
42 C=C(startTy:endTy,startTx:endTx);
43 end
```

Datei 8.2: hesseMatrix2D.m

```
%% The Hessian matrix calculates an image using the formula specified by
%% Frangi. "Image" is the image given by the user, "s" is the scalar that
%% Frangi needs for the second derivation and the "gauss factor" is another
%% factor that is needed to calculate the semi-discrete convolution.
%%
%% The function should be called like this:
%% Image = imread('XXX');
%% [hessXX, hesXY, hesYY] = hesseMatrix2D(Image, 2, 3);
function [hesXX, hesXY, hesYY] = hesseMatrix2D(Image, s, gaussfactor)
rest = @(x,y) exp(-(x.^2+y.^2)/(2*s.^2))/(2*pi*s.^6);
% The second deriviatives of the gaussian distribution.
dxx = @(x,y) (x.^2-s.^2).*rest(x,y);
dxy = @(x,y) x.*y.*rest(x,y);
dyy = @(x,y) (y.^2-s^2).*rest(x,y);

[X, Y] = meshgrid(-gaussfactor*s:gaussfactor*s, -gaussfactor*s:gaussfactor*s);

hesFilXX = dxx(X,Y);
hesFilXY = dxy(X,Y);
hesFilYY = dyy(X,Y);

% This is where the "convolution" is carried out.
hesXX = imfilter(double(Image), hesFilXX, 'conv');
hesXY = imfilter(double(Image), hesFilXY, 'conv');
hesYY = imfilter(double(Image), hesFilYY, 'conv');
end
```

Datei 8.3: getEigenValues2D.m

```
%% Calculates the eigenvalues of the Hessian matrix.
%% Dxx is the top left entry, Dxy the top right and bottom left entry
%% and Dyy the bottom right entry. lam1 is an array of all eigenvalues that
%% are smaller in magnitude and lam2 is an array of all eigenvalues that
%% are larger in magnitude. These two arrays are returned.
function [lam1,lam2]=getEigenValues2D(Dxx,Dxy,Dyy)
% Implementing the midnight formula in case of symmetric Hessian
% matrice.
discriminant = sqrt((Dxx - Dyy).^2 + 4*Dxy.^2);
lam1 = (Dxx + Dyy + discriminant)/2;
lam2 = (Dxx + Dyy - discriminant)/2;

% If the absolutly value of lam1 is bigger than the absolutly value of
% lam2.
flip =abs(lam1)>abs(lam2);
temp = lam1;
lam1(flip)=lam2(flip);
lam2(flip)=temp(flip);
end
```

Datei 8.4: frangi2D.m

```
%% Function that implements the 2-dimensional Frangi filter. "Image" is the
%% input image on which the Frangi filter is to be executed. "sigmas" is a
%% vector of scalars to be applied. "beta" and "c" are the parameters that
%% you need to adjust. "lindeberg" is the parameter that classifies the
%% importance of individual scalars. "brightness" can be either "1" for dark or
%% "0" for bright. The last parameter indicates by how much the scalar should
    be
%% amplified during the "convolution".
%%
%% Example:
%% Image = imread('XXX.jpg');
%% sigmas=1:1:15;
%% beta =0.5;
%% c =15;
%% lindeberg = 2;
%% outIm = frangi2D(Image, sigmas, beta, c, lindeberg ,1,3) ;
%% figure, subplot (2,1,1) ; imshow(Im); subplot (2,1,2) ; imshow(outIm);
function MaxPicture = frangi2D(Image, sigmas, beta, c, lindeberg, brightness ,
    gaussfaktor)
% Constructing a picture with negativ infinity values.
MaxPicture = -Inf * ones(size(Image));
if brightness<0 || brightness>1
    disp("No valid brightness is given. Please input either '1' for dark or '0'
        for bright ");
    return;
end
% Calculate the Frangi-Filter for each sigma.
for i = 1:length(sigmas)
    % Gets the second deriviative of the Image.
    [Dxx,Dxy,Dyy] = hesseMatrix2D(Image,sigmas(i),gaussfaktor);

    % Multiplicate it with the scalar pow to the lindeberg parameter.
    Dxx = (sigmas(i)^lindeberg)*Dxx;
    Dxy = (sigmas(i)^lindeberg)*Dxy;
    Dyy = (sigmas(i)^lindeberg)*Dyy;

    % Gets the eigenvalues of the Hessian matrice. lam1 is an array with
    % the absolutly lower eigenvalues an lam2 an array with the absolutly
    % higher eigenvalues.
    [lam1,lam2]=getEigenValues2D(Dxx,Dxy,Dyy);
    %  clear Dxx Dxy Dyy;

    % To invoid division by zero, we take lam2 in case of zero to
    % epsilon.
    lam2(lam2==0) = eps;

    % Calculate Frangis constants.
```

```
45      Rb = lam1./lam2;
46      S = sqrt(lam1.^2+lam2.^2);
47
48      % Frangis 2 dimensional Vesselness function .
49      V0 = exp(-Rb.^2/(2*beta.^2)).*(1-exp(-S.^2/(2*c.^2)));
50      clear Rb S;
51      % The other case of the vesselness function .
52      if brightness == 0
53          V0(lam2>0)=0;
54      else
55          V0(lam2<0)=0;
56      end
57      clear lam1 lam2;
58      % The 4th step of Vesselness of Frangi is the maximation of the
59      % pictures of each scalar .
60      MaxPicture = max(MaxPicture, V0);
61      clear V0;
62  end
63  clear Image;
64  end
```

8.2 3D-Frangi-Filter

Datei 8.5: get3DZeroPoints.m

```
%% Calculates the zeros of a 3rd degree polynomial.
%% The input parameters a, b, c are as follows.
%% f(x) = x^3 + ax^2 + bx + c
%% Every third-degree polynomial can be transformed into this notation.
%% Since the Hessian matrix is symmetrical, the 4th case of the Cardan
%% formula can be used. There are only real solutions.
function [x1, x2, x3] = get3DZeroPoints(a,b,c)
% Calculate p and q.
p = b-a.^2./3;
q = 2*a.^3./27-a.*b./3+c;

delta = (q/2).^2+(p/3).^3;

% Calculate the arcosinus part per default.
acosinus = acos(-q./2.*sqrt(-27./p.^3));

% Calculates the factor before the equation.
predisk = sqrt(-4/3.*p);

% Calculates the zero points.
x1 = predisk.*cos(1/3.*acosinus)-a./3;
x2 = -predisk.* cos(1/3.*acosinus+pi/3)-a./3;
x3 = -predisk.* cos(1/3*acosinus-pi/3)-a./3;

% Converts into double values. (In reality there are real numbers.)
x1 = real(x1);
x2 = real(x2);
x3 = real(x3);

% If the polynomial has three times the same zeropoint.
z = -a/3;
threeTime = delta==0 & p == 0;
x1(threeTime) = z(threeTime);
x2(threeTime) = z(threeTime);
x3(threeTime) = z(threeTime);

% If the polynomial has zwo same zeropoints and one other at another
% point.
z1 = 3*q./p;
z2 = -3*q./(2*p);
twoPlusOne = delta==0 & p ~= 0;
x1(twoPlusOne) =z1(twoPlusOne);
x2(twoPlusOne) = z2(twoPlusOne);
x3(twoPlusOne) = z2(twoPlusOne);
end
```

Datei 8.6: hesseMatrix3D.m

```
%% The Hessian matrix calculates an volume using the formula specified by
%% Frangi. "Volume" is the image given by the user, "s" is the scalar that
%% Frangi needs for the second derivation and the "gauss factor" is another
%% factor that is needed to calculate the semi-discrete convolution.
%%
%% The function should be called like this:
%% [hesXX, hesXY, hesXZ, hesYY,hesYZ,hesZZ] = hesseMatrix3D(Volume, [1..10], 3)
function [hesXX, hesXY, hesXZ, hesYY,hesYZ,hesZZ] = hesseMatrix3D(Volume,
    s, gaussfactor)
% The part that is in all expressions equal.
rest = @(x,y,z) exp(-(x.^2+y.^2+z.^2)/(2*s.^2))/(sqrt(8*pi^3)*s^7);

% The second deriviatives of the gaussian distribution.
dxx = @(x,y,z) (x.^2-s.^2).*rest(x,y,z);
dxy = @(x,y,z) (x.*y).*rest(x,y,z);
dxz = @(x,y,z) (x.*z).*rest(x,y,z);
dyy = @(x,y,z) (y.^2-s.^2).*rest(x,y,z);
dyz = @(x,y,z) (y.*z).*rest(x,y,z);
dzz = @(x,y,z) (z.^2-s.^2).*rest(x,y,z);

% The range that the convolution should be evaluated
range = -gaussfactor*s:gaussfactor*s;
[X, Y, Z] = meshgrid(range,range,range);


hesFilXX = dxx(X,Y,Z);
hesFilXY = dxy(X,Y,Z);
hesFilXZ = dxz(X,Y,Z);
hesFilYY = dyy(X,Y,Z);
hesFilYZ = dyz(X,Y,Z);
hesFilZZ = dzz(X,Y,Z);

%Volume = double(Volume);

% This is where the "convolution" is carried out.
hesXX = imfilter(Volume, hesFilXX, 'conv');
hesXY = imfilter(Volume, hesFilXY, 'conv');
hesYY = imfilter(Volume, hesFilYY, 'conv');
hesXZ = imfilter(Volume, hesFilXZ, 'conv');
hesYZ = imfilter(Volume, hesFilYZ, 'conv');
hesZZ = imfilter(Volume, hesFilZZ, 'conv');
end
```

Datei 8.7: getEigenValues3D.m

```
%% Calculates the eigenvalues of the 3D Hessian matrix.
%% Dxx is the top left entry, Dxy the top middle and left middle entry, Dxz
%% the top right and left down entry. Dyy is the centered entry. Dyz is the
%% right centered and bottom centered entry. lam1 is an array of all
%% eigenvalues that are smaller in magnitude and lam3 is an array of all
%% eigenvalues that are the largest in magnitude. lam2 is in the center.
%% These three arrays are returned.
function [lam1,lam2, lam3]=getEigenValues3D(Dxx,Dxy,Dxz,Dyy, Dyz, Dzz)
% Calculate the characteristic polynomial.
a = -(Dxx+Dyy+Dzz);
b = Dxx.*Dyy + Dxx.*Dzz+Dyy.*Dzz-Dxy.^2-Dxz.^2-Dyz.^2;
c = -(Dxx.*Dyy.*Dzz + 2*Dxy.*Dxz.*Dyz - Dxy.^2.*Dzz-Dyz.^2.*Dxx-Dxz
    .^2.*Dyy);

% Calculates the zero points.
[lam1, lam2, lam3] = get3DZeroPoints(a,b,c);

% Sorts it on absolutly magnitude.
flip12 =abs(lam1)>abs(lam2);
temp12 = lam1;
lam1(flip12)=lam2(flip12);
lam2(flip12)=temp12(flip12);

flip13 =abs(lam1)>abs(lam3);
temp13 = lam1;
lam1(flip13)=lam3(flip13);
lam3(flip13)=temp13(flip13);

flip23 =abs(lam2)>abs(lam3);
temp23 = lam2;
lam2(flip23)=lam3(flip23);
lam3(flip23)=temp23(flip23);
end
```

Datei 8.8: frangi3D.m

```
%% Function that implements the 3-dimensional Frangi filter . "Volumen" is the
%% input volumen on which the Frangi filter has to be executed. "sigmas" is a
%% vector of scalars to be applied. "alpha", "beta" and "c" are the parameters
    that
%% you need to adjust. "lindeberg" is the parameter that classifies the
%% importance of individual scalars. "brightness" can be either "1" for dark or
%% "0" for bright. The last parameter indicates by how much the scalar should
    be
%% amplified during the "convolution".
%%
%% Example:
%% load('XXX.mat'); % Loads the 3D Volume in the variable data
%% sigmas = 1:1:5;
%% alpha = 0.5;
%% beta = 0.5;
%% c = 500;
%% data = double(data);
%% filteredVolume=frangi3D(data,sigmas,alpha, beta, c, 2,1,3) ;
%% figure,
%% volshow(filteredVolume,'Backgroundcolor',[0.5 0.5 0.5]) ;
function MaxVolumen = frangi3D(Volumen, sigmas,alpha, beta, c, lindeberg,
    brightness, gaussfaktor)
% Constructing a volume with negativ infinity values.
MaxVolumen = -Inf * ones(size(Volumen));

% Checking the values.
if brightness<0 || brightness>1
    disp("No valid brightness is given. Please input either '1' for dark or '0'
        for bright");
    return;
end

% Calculating the 3D-Frangi-Filter for each sigma.
for i = 1:length(sigmas)
    % Gets the second deriviative of the volume.
    [Dxx,Dxy,Dxz,Dyy, Dyz, Dzz] = hesseMatrix3D(Volumen,sigmas(i),
        gaussfaktor);

    % Multiplicate it with the scalar pow to the lindeberg parameter.
    Dxx = (sigmas(i)^lindeberg)*Dxx;
    Dxy = (sigmas(i)^lindeberg)*Dxy;
    Dxz = (sigmas(i)^lindeberg)*Dxz;
    Dyy = (sigmas(i)^lindeberg)*Dyy;
    Dyz = (sigmas(i)^lindeberg)*Dyz;
    Dzz = (sigmas(i)^lindeberg)*Dzz;

    % Gets the eigenvalues of the Hessian matrice. lam1 is an array with
```

```
43      % the absolutly lower eigenvalues an lam2 an array with the absolutly
44      % higher eigenvalues .
45      [lam1,lam2, lam3]=getEigenValues3D(Dxx,Dxy, Dxz,Dyy,Dyz, Dzz);
46      lam1 = double(lam1);
47      clear Dxx Dxy Dxz Dyy Dyz Dzz;
48
49      % To invoid division by zero , we take lam1, lam2 and lam3 in case of
50      % zero to epsilon .
51      lam1(lam1==0) = eps;
52      lam2(lam2==0) = eps;
53      lam3(lam3==0) = eps;
54
55      % Calculate Frangis constants .
56      Rb = abs(lam1)./sqrt(abs(lam2).*(abs(lam3)));
57      Ra = abs(lam2)./abs(lam3);
58      S = sqrt(lam1.^2+lam2.^2+lam3.^2);
59
60      % Frangis 3 dimensional Vesselness function .
61      V0 = (1-exp(-Ra.^2/(2*alpha.^2))).*exp(-Rb.^2/(2*beta.^2)).*(1-exp(-S
            .^2/(2*c.^2)));
62      clear Ra Rb S;
63
64      % The other case of the vesselness function .
65      if brightness == 0
66          V0(lam2>0 | lam3>0)=0;
67      else
68          V0(lam2<0 | lam3<0)=0;
69      end
70      clear lam1 lam2 lam3;
71
72      % The 4th step of Vesselness of Frangi is the maximation of the
73      % volumes of each scalar .
74      MaxVolumen = max(MaxVolumen, V0);
75      clear V0;
76  end
77  clear Volumen;
78  end
```

9 Literaturverzeichnis

Literatur

[1] *Angiografie*. https://www.internisten-im-netz.de/untersuchungen/angiografie.html. letzter Zugriff am: 06.07.2021.

[2] Alejandro F. Frangi u. a. "Multiscale vessel enhancement filtering". In: *Medical Image Computing and Computer-Assisted Intervention — MICCAI'98*. Hrsg. von William M. Wells, Alan Colchester und Scott Delp. Berlin, Heidelberg: Springer Berlin Heidelberg, 1998, S. 130–137. ISBN: 978-3-540-49563-5.

[3] Gerd Fischer und Boris Springborn. "Eigenwerte". In: *Lineare Algebra: Eine Einführung für Studienanfänger*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2020, S. 241–306. ISBN: 978-3-662-61645-1. DOI: 10.1007/978-3-662-61645-1_6. URL: https://doi.org/10.1007/978-3-662-61645-1_6.

[4] Gerd Fischer. "5 Euklidische und unitäre Vektorräume". In: *Lineare Algebra: Eine Einführung für Studienanfänger*. Wiesbaden: Springer Fachmedien Wiesbaden, 2014, S. 274–330. ISBN: 978-3-658-03945-5. DOI: 10.1007/978-3-658-03945-5_6. URL: https://doi.org/10.1007/978-3-658-03945-5_6.

[5] Jan Glaubitz, Daniel Rademacher und Thomas Sonar. "Metrik, Norm, Topologie". In: *Lernbuch Analysis 1: Das Wichtigste ausführlich für Bachelor und Lehramt*. Wiesbaden: Springer Fachmedien Wiesbaden, 2019, S. 389–411. ISBN: 978-3-658-26937-1. DOI: 10.1007/978-3-658-26937-1_13. URL: https://doi.org/10.1007/978-3-658-26937-1_13.

[6] Gerd Fischer und Boris Springborn. "Bilinearformen und Skalarprodukte". In: *Lineare Algebra: Eine Einführung für Studienanfänger*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2020, S. 307–370. ISBN: 978-3-662-61645-1. DOI: 10.1007/978-3-662-61645-1_7. URL: https://doi.org/10.1007/978-3-662-61645-1_7.

[7] Tim Jerman u. a. "Beyond Frangi: An improved multiscale vesselness filter". In: 9413 (März 2015). DOI: 10.1117/12.2081147.

[8] Daniel Jimenez-Carretero u. a. "3D Frangi-based lung vessel enhancement filter penalizing airways". In: *2013 IEEE 10th International Symposium on Biomedical Imaging*. 2013, S. 926–929. DOI: 10.1109/ISBI.2013.6556627.

[9] Antonia Longo u. a. "Assessment of hessian-based Frangi vesselness filter in optoacoustic imaging". In: *Photoacoustics* 20 (2020), S. 100200. ISSN: 2213-5979. DOI: https://doi.org/10.1016/j.pacs.2020.100200. URL: https://www.sciencedirect.com/science/article/pii/S2213597920300409.

[10] Weilin Fu u. a. *Frangi-Net: A Neural Network Approach to Vessel Segmentation*. Nov. 2017.

[11] *Was ist ein Neuronales Netz?* https://www.bigdata-insider.de/was-ist-ein-neuronales-netz-a-686185/. letzter Zugriff am: 29.06.2021.

[12] Dirk-Jan Kroon. *Hessian based Frangi Vesselness filter*. https://www.mathworks.com/matlabcentral/fileexchange/24409-hessian-based-frangi-vesselness-filter.

[13] Gerd Altmann. letzter Zugriff am: 11.06.2021. URL: https://pixabay.com/de/illustrations/baum-stamm-ast-kahl-beschnitten-105200/.

10 Eidesstattliche Erklärung

Hiermit erkläre ich, Severin Primbs, dass ich die vorliegende Arbeit eigenständig und ohne fremde Hilfe angefertigt habe. Textpassagen, die wörtlich oder dem Sinn nach auf Publikationen oder Vorträgen anderer Autoren beruhen, sind als solche kenntlich gemacht. Die Arbeit wurde bisher keiner anderen Prüfungsbehörde vorgelegt und auch noch nicht veröffentlicht.

Ort, Datum	Unterschrift