Lehrstuhl für Mathematik mit Schwerpunkt
Digitale Bildverarbeitung

Volumenähnlichkeit von 3D-CT-Daten auf der Grundlage der Haar-Wavelet-Transformation

Masterarbeit in Informatik

von

Manuel Pauli

1. Prüfer	2. Prüfer
Prof. Dr. Tomas Sauer	Prof. Dr. Michael Granitzer

---

15. Mai 2023

Inhaltsverzeichnis

Zusammenfassung

In dieser Arbeit wurden Ähnlichkeitsmaße zum Vergleich von 3D-CT-Daten auf Basis der Wavelet-Transformation entwickelt. Dazu wurde ein bestehendes Ähnlichkeitsmaß für 2D-Daten so angepasst, dass es auf 3D-CT-Daten anwendbar ist: HaarPSI3D. Zusätzlich wurden zwei weitere Ähnlichkeitsmaße für 3D-CT-Daten entwickelt: HaarVectorPSI und HaarHistSim. Diese bestimmen die Ähnlichkeit auf Basis von Gradienten, die sich aus der Wavelet-Transformierten der Daten ableiten lassen. HaarVectorPSI berechnet die Ähnlichkeit zweier Signale durch den positionsweisen Vergleich der Gradientenvektoren. Um diese zu vergleichen, wurde das Ähnlichkeitsmaß ‘‘längensensitive Kosinusähnlichkeit’’ entwickelt, welches die Ähnlichkeit zwischen den Vektoren eines Gradienten bezüglich des eingeschlossenen Winkels und der Vektorenlängen ermittelt. Außerdem wurde eine Variante von HaarVectorPSI mit einer schnellen Wavelet-Transformation (FWT) vorgeschlagen. Diese liefert annähernd gleichwertige Ergebnisse wie HaarVectorPSI, hat aber eine erheblich kürzere Berechnungszeit. HaarHistSim betrachtet die Ähnlichkeit der Gradientenvektoren zweier Signale positionsunabhängig. Dazu werden aus den Gradienten Histogramme erstellt, anhand derer die Ähnlichkeit ermittelt wird. Während HaarVectorPSI im Vergleich zu HaarHistSim gute Ergebnisse im Bereich IQA liefert, ist HaarHistSim geeignet, die Ähnlichkeit zweier Signale bezüglich der Textur zu bestimmen. HaarPSI und HaarVectorPSI erzielen hier nur schlechte Resultate. Es wurde zudem gezeigt, dass mit HaarVectorPSI und HaarPSI Pattern-Matching prinzipiell möglich ist. Alle entwickelten Algorithmen wurden in der Programmiersprache ‘‘python’’ implementiert. Schließlich wurden diese drei Ähnlichkeitsmaße hinsichtlich ihrer Empfindlichkeit gegenüber verschiedenen Transformationen und ihrer Berechnungszeit verglichen.

Abbildungsverzeichnis

Tabellenverzeichnis

Python Code

1 Einleitung

Die computergestützte Analyse und Verarbeitung von Computertomographie-Daten (CT) spielt in der Industrie eine immer größer werdende Rolle. Zunächst vorwiegend in der Medizin eingesetzt, hat CT mittlerweile ein breites industrielles Anwendungsfeld, wobei immer schneller immer mehr CT-Daten produziert werden [Han10]. Die Einsatzgebiete reichen dabei, dank immer schnellerer Scann-Verfahren und besserer Auflösungen bis hin zu sub-μ CT-Scans, von geometrischer Analyse von Werkstücken oder ganzen Autos zu zerstörungsfreier Prüfverfahren in der Qualitätssicherung. Es gibt daher einen großen Bedarf an software-basierten analytischen Werkzeugen, um die durch einen CT-Scan entstehenden 3D-Voxel Daten zu verarbeiten [Rei+04]. Der Vergleich von CT-Daten ist dabei eine fundamentale Aufgabe auf dem viele komplexere Anwendungen aufbauen können. Der Prozess zur Bestimmung der Ähnlichkeit zwischen Objekten oder Formen wird als Matching bezeichnet [TV08] (und ist Grundlage des 3D content based shape retrival).
Matching lässt sich prinzipiell in area-based, feature-based und NN-methods unterteilen. Bei Area-Based matching werden Bilder direkt auf Pixelebene mit Hilfe einer Similarity Funktion verglichen, wo hingegen bei der feature-based Methode durch feature detection die verschiedenen Strukturen aus dem Bild extrahiert werden [TV08]. Diese Matching Prozeduren können dann z.B. in Form von Content based image retrival (CBIR) oder content based shape retrival genutzt werden. Seit Kurzem gibt es auch Anstrengungen IQA (Image Quality Assessment) Techniken für Image Retrival (IR) einzusetzen [PPT21]. Grundlegend lassen sich IQA Verfahren in subjektiv (menschlicher Betrachter) und objektiv (algorithmisch) unterteilen, wobei sich die Anwendungsbereiche von IQA in 3 Kategorien einteilen lassen. Full Reference (FR) beurteilt die Qualität des Zielbildes mit Zugang zu dem originalen Referenzbild, wobei prinzipiell bei FR IQA nach einem preprocessing aus den zu vergleichenden Bildern features extrahiert werden um eine Distanz ermitteln zu können, die zu einem Quality score führt. Bei RR (Reduced Reference) besteht kein Zugang zu einem Referenzbild, sondern nur zu bereits extrahierten features. NR (No Reference) Ansätze bestimmen die Qualität lediglich Anhand des zu bewertenden Bildes [ZM20]. Einen umfassenden Überblick

wurde dazu in einer Arbeit von Guangtoa Zhai erstellt[ZM20]. Bezüglich IR sind vor allem FR IQA Varianten interessant, die bei der feature extraction strukturelle Informationen des Bildes berücksichtigen, sodass die Betrachtung der Ähnlichkeit in Richtung des ”content based” Ansatzes geht.[PPT21]. Ein interessanter Teilbereich sind hierbei transformed-based FR IQA in der es Ansätze auf Basis der Wavelet-Domain gibt, die in Teilen auch an Konzepte des Human visual systems (HSV) angelehnt sind.
Kürzlich wurden hier Erfolge erzielt einen Haar-wavelet- basierten Ansatz in Anlehnung zu FSIM (Feature Similarity Index) zu entwickeln. Dieser Ansatz eines Haar wavelet-based perceptual similarity index for image quality assessment (HaarPSI) [Rei+18] konnte nicht nur bessere Ergebnisse liefern, sondern war zudem auch noch performanter als vergleichbare FR IQAs. Im Bereich FR IQA für IR konnte bereits gezeigt werden, dass FSIM im Bezug auf CBIR durchaus Anwendung finden kann [Pha+17]

1.1 Forschungsfragen

Das Ziel dieser Masterarbeit ist es, den HaarPSI Algorithmus [Rei+18], der für 2D-Signale konzipiert wurde, auf 3D-Signale zu erweitern. Dabei soll untersucht werden, inwiefern sich die Eigenschaften von HaarPSI aus den Untersuchungsergebnissen zu dem 2D-Fall auf den 3D-Fall übertragen lassen. Ein weiteres Ziel dieser Arbeit ist die Entwicklung von vektorbasierten Ansätzen für die Signalähnlichkeit. Diese Ansätze sind angelehnt an HaarPSI und definieren die Ähnlichkeit eines Signals s durch den Vergleich der Vektoren seines Gradientenfeldes. Das Gradientenfeld von s kann durch die Haar-Wavelet-Transformation approximiert werden. Hierbei soll jeweils untersucht werden, inwiefern ein solcher Ansatz Vorteile bezüglich der Ermittlung von IQ und möglicherweise auf Pattern-Matching von 2D und 3D-Daten hat. Dafür soll auch betrachtet werden ob ein FWT (Schnelle Wavelet Transformation) Ansatz vergleichbar geeignet ist wie ein UWT (undezimierte Wavelet-Transformation) Ansatz.

1.2 Struktur der Arbeit

Die vorliegende Arbeit besteht aus fünf Kapiteln. In Kapitel 2 werden zunächst die theoretischen Grundlagen beschrieben. Das Hauptaugenmerk liegt dabei auf der Einführung der Haar-Wavelets, der diskreten Wavelet-Transformation und wie auf dieser Basis der Gradient

eines Signals approximiert werden kann. In Kapitel 3 werden die Algorithmen ”HaarPSI3D”, ”HaarVectorPSI” und seine Varianten und ”HaarHistSim” definiert und beschrieben. Im Unterkapitel 3.2.1 wird das Ähnlichkeitsmaß ”längensensitive Kosinusähnlichkeit”, welches für ”HaarVectorPSI” notwendig ist definiert. Schließlich werden die Implementationen der Algorithmen vorgestellt. In Kapitel 4 werden die Algorithmen evaluiert. Dabei wird im Unterkapitel 4.1 das Verhalten bezüglich Translation, Rotation, Skalierung und Empfindlichkeiten bezüglich verschiedener Verzerrungen behandelt. In 4.2 werden Unterschiede der Algorithmen in der Bestimmung der Ähnlichkeit genauer betrachtet. Weiter wird im Unterkapitel 4.3 die prinzipielle Tauglichkeit bezüglich Pattern Matching behandelt und im Unterkapitel 4.4 die Fähigkeiten bezüglich IQA. Die Analyse der Laufzeit bezüglich Datengrößen und Parameter der Algorithmen wird im Unterkapitel 4.5 durchgeführt. Letztlich werden im Kapitel 5 die Ergebnisse genauer eingeordnet und bewertet.

2 Theoretische Grundlagen

2.1 Fourier

Bevor auf das Konzept der Wavelet-Transformation eingegangen wird, soll zunächst die Fourier-Transformation beleuchtet werden. Sie spielt bei der Berechnung der diskreten Wavelet-Transformation (DWT) eine wichtige Rolle, denn sie ermöglicht erst eine effiziente Implementation. Die Fouriertransformierte einer Funktion ist in der Signalanalyse von hoher Bedeutung. Sie gibt Aufschluss darüber welche Frequenzen im gesamten Signal vorkommen und welchen Anteil sie darin haben.

Definition 2.1.1 *Die Menge aller p summierbaren Funktionen $L_p(\mathbb{R})$ ist definiert durch*

$$L_p(\mathbb{R}) := \left\{ f \in L(\mathbb{R}) : \|f\|_p := \int |f(t)|^p \, dt < \infty \right\}$$

und die Menge aller p summierbaren Folgen durch

$$l_p(\mathbb{Z}) := \left\{ c \in l(\mathbb{Z}) : \|f\|_p := \sum_{k \in \mathbb{Z}} |c(k)|^p < \infty \right\}.$$

Definition 2.1.2 (Fourier-Transformation) *Für eine Funktion $f \in L_1(\mathbb{R})$ bezeichnet man*

$$\widehat{f}(\xi) := \int_{\mathbb{R}} f(t) e^{-i\xi t} dt, \quad \xi \in \mathbb{R}$$

als die Fourier-Transformierte $\widehat{f} : \mathbb{R} \to \mathbb{C}$. Für eine Folge $c \in l_1(\mathbb{Z})$ ist sie als

$$\widehat{c}(\xi) := \sum_{k \in \mathbb{Z}} c(k) e^{-ik\xi}, \quad \xi \in \mathbb{R}$$

definiert.

Die Fourier-Transformation gibt, wie bereits erwähnt, Aufschluss über die Frequenzen die im Signal stecken, aber nicht an welcher Stelle bzw. zu welchem Zeitpunkt die entsprechende Frequenz auftritt. Es gibt keine Lokalität, da über ganz $\mathbb{R}$ integriert wird. Bei der Fourier-Transformation einer Folge c ist zu beachten, dass c nicht wieder in eine Folge, sondern in eine Funktion $\widehat{c}$ überführt wird und ist zudem auch noch 2π periodisch:

$$\widehat{c}(\xi + 2\pi) = \sum_{k\in\mathbb{Z}} c(k)e^{-i(\xi+2\pi)k} = \sum_{k\in\mathbb{Z}} c(k)e^{-i\xi k} \underbrace{e^{-i2\pi k}}_{=1} = \sum_{k\in\mathbb{Z}} c(k)e^{-i\xi k} = \widehat{c}(\xi).$$

Die Faltung einer Funktion f mit einer Folge c lässt sich auch als Produkt ihrer Fourier-Transformierten darstellen. Die Faltung hat in der Signalverarbeitung hohe Bedeutung, da die Anwendung von Filtern im Wesentlichen eine Faltung ist. Auch bei der Wavelet-Transformation spielt die Faltung eine wichtige Rolle.

Definition 2.1.3 (Faltung) *Seien $f, g \in L(\mathbb{R})$ und $c, d \in l(\mathbb{Z})$. Dann ist die Faltung zweier Funktionen definiert als*

$$f * g := \int_{\mathbb{R}} f(\bullet - t) \cdot g(t)\,\mathrm{d}t \quad \in L(\mathbb{R})$$

und die Faltung zweier Folgen als

$$c * d := \sum_{k\in\mathbb{Z}} c(\bullet - k) \cdot d(k) \quad \in l(\mathbb{Z}).$$

Die Faltung einer Funktion f mit einer Folge c ist definiert durch

$$c * f := f * c := \sum_{k\in\mathbb{Z}} f(\bullet - k) \cdot d(k) \quad \in L(\mathbb{R}).$$

Man kann sich die Faltung als eine Methode vorstellen, mit der ermittelt werden kann, wie lokal ähnlich sich zwei Funktionen oder Folgen sind.

Definition 2.1.4 (Faltungseigenschaft Fourier-Transformation) *Für $f, g \in L_1(\mathbb{R})$ bzw. $c, d \in l_1(\mathbb{Z})$ sind $f * g \in L_1(\mathbb{R})$, $c * d \in l_1(\mathbb{Z})$ und $f * c \in L_1(\mathbb{R})$ und es gilt:*

$$(f * g)^\wedge(\xi) = \widehat{f}(\xi)\widehat{g}(\xi), \quad (c * d)^\wedge(\xi) = \widehat{c}(\xi)\widehat{d}(\xi) \quad \textit{und} \quad (f * c)^\wedge(\xi) = \widehat{f}(\xi)\widehat{c}(\xi).$$

Der Vorteil der Berechnung der Faltung über die Multiplikation der Fourier-Transformierten liegt nun darin, dass diese nun wesentlich effizienter und schneller berechnet werden kann.

Diskrete Fourier-Transformation für 2D/3D-Signale
Die Diskrete Fourier-Transformation lässt sich auch auf n-dimensionale Signalen anwenden. Da im Rahmen dieser Masterarbeit zweidimensionale, bzw. dreidimensionale Signale verarbeitet werden, sollen diese Signalklassen nun kurz definiert werden.

Definition 2.1.5 *Die Menge aller Signale $l(\mathbb{Z}^2)$ sind definiert durch*

$$c = (c(\alpha) : \alpha \in \mathbb{Z}^2) = (c(j,k) : j,k \in \mathbb{Z})$$

Definition 2.1.6 *Die Menge aller Signale $l(\mathbb{Z}^3)$ sind definiert durch*

$$c = (c(\gamma) : \gamma \in \mathbb{Z}^3) = (c(j,k,l) : j,k,l \in \mathbb{Z})$$

Definition 2.1.7 *Die Menge aller Signale $l(\mathbb{Z}^s)$ sind definiert durch*

$$c = (c(\beta) : \beta \in \mathbb{Z}^s) = (c(j,k,..) : j,k,.. \in \mathbb{Z})$$

Die diskrete zweidimensionale Fouriertransformation (DFT) zu einem Signal $c \in l(\mathbb{Z}_n^2)$ lässt sich dann wie folgt beschreiben.

$$\begin{aligned}
\widehat{c}(\beta) &= \sum_{\alpha \in \mathbb{Z}^2{}_n} = c(\alpha) \cdot e^{-2\pi i \alpha^T \beta / n} = \sum_{\alpha_1 \in \mathbb{Z}_n} \sum_{\alpha_2 \in \mathbb{Z}_n} c(\alpha_1, \alpha_2) \cdot e^{-2\pi i \alpha_1 \beta_1 / n} e^{-2\pi i \alpha_2 \beta_2 / n} \\
&= \sum_{\alpha_1 \in \mathbb{Z}_n} e^{-2\pi i \alpha_1 \beta_1 / n} \sum_{\alpha_2 \in \mathbb{Z}_n} c(\alpha_1, \alpha_2) \cdot e^{-2\pi i \alpha_2 \beta_2 / n} \\
&= \sum_{\alpha_1 \in \mathbb{Z}_n} (c(\alpha_1, \bullet))^{\wedge}(\beta_2) e^{-2\pi i \alpha_1 \beta_1 / n}
\end{aligned}$$

Durch diese Darstellung lässt sich nun auch erkennen, dass die 2D DFT berechnet werden kann, indem bei der Anwendung auf ein Bild die eindimensionale DFT zunächst auf alle Zeilen angewendet wird und auf den entstandenen Vektor nochmals angewendet wird. Dieses vorgehen lässt sich auch analog auf den 3D Fall übertragen. Zusammen mit der Faltungseigenschaft der Fouriertransformation 2.1.4 kann die DFT oder die darauf aufbauende schnelle Fourier Transformation (FFT) dazu verwendet werden, Filter effizient

zu implementieren [Sau16]. Somit lässt sich auch die bei der Haar-DWT durchgeführte Filterung mit FIR Haar-Filtern effizient implementieren.

2.2 Haar-Wavelet

Da die Wavelet-Transformation auf Wavelets basiert, sollen diese nun kurz eingeführt werden.

Definition 2.2.1 (Wavelet) *Eine Funktion $\psi \in L_2(\mathbb{R})$ wird als* Wavelet *bezeichnet, wenn sie* mittelwertfrei *ist, d.h. wenn gilt:*

$$\int_{\mathbb{R}} \psi(t)\,\mathrm{d}t = 0.$$

Ein Wavelet ψ heißt normalisiert, *wenn*

$$\|\psi\|_2 = 1.$$

Ein Wavelet ψ heißt zulässig, *wenn*

$$C_\psi := \int_{\mathbb{R}} \frac{|\widehat{\psi}(\xi)|^2}{|\xi|}\,\mathrm{d}\xi < \infty.$$

Die Zulässigkeitsbedingung sorgt dafür, dass $\widehat{\psi}(\xi)$ am Übergang $\xi \longrightarrow 0$ hinreichend schnell gegen 0 geht. Das heißt insbesondere, dass

$$0 = \widehat{\psi}(0) = \int_{\mathbb{R}} \psi(x) e^{-i\cdot 0\cdot x}\,\mathrm{d}x = \int_{\mathbb{R}} \psi(x)\,\mathrm{d}x$$

gelten muss. Daraus folgt sofort, dass jede zulässige Funktion *automatisch* per Definition ein Wavelet ist.

Im Rahmen dieser Arbeit wird das Haar-Wavelet verwendet, welches bereits 1910 von Alfred Haar definiert wurde [Haa10]. Es ist das älteste und einfachste aller Wavelets und

ist definiert als

$$\psi := \begin{cases} 1, & x \in [0, \frac{1}{2}) \\ -1, & x \in [\frac{1}{2}, 1) \\ 0, & sonst. \end{cases}$$

In Abbildung 2.1 ist das Haar-Wavelet dargestellt.

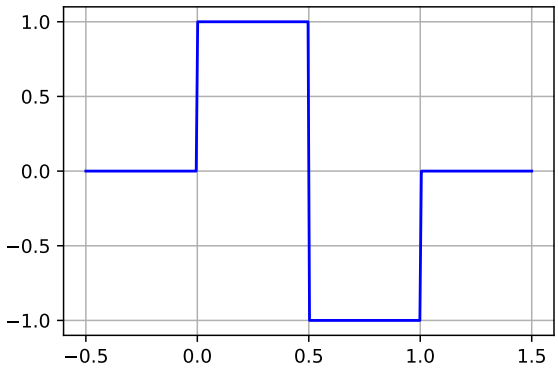


Abbildung 2.1: Graphische Darstellung des 1D Haar-Wavelets

Seine Einfachheit macht es zu dem idealen Kandidaten für eine performante Implementierung der Wavelet-Transformation. Das Haar-Wavelet-System ist zudem das einzige seiner Art, welches orthonormal und symetrisch ist und kompakten Träger hat [Mey93]. Haar-Wavelets für $\mathbb{R}^s$ lassen sich aus Tensorprodukten der Skalierungs- und Wavelet- Funktionen generieren. Für den Fall, dass $s = 1$ ist die Skalierungsfunktion definiert als

$$\psi_0(x) = \begin{cases} 1, & x \in [0, 1) \\ 0, & sonst \end{cases}$$

und die Wavelet-Funktion als

$$\psi_1(x) = \psi_0(2x) - \psi_0(2x - 1) = \begin{cases} 1, & x \in [0, \frac{1}{2}) \\ -1, & x \in [\frac{1}{2}, 1) \\ 0, & sonst. \end{cases}$$

Die Skalierungsfunktion und Waveletfunktion stehen orthogonal zu einander. Die Skalierungsfunktion $\psi_0(x)$ und die normalisierte Haar-Wavelet Familie

$$\psi^n_{1,k} := 2^{\frac{n}{2}} \psi(2^n \bullet -k), k \in \mathbb{Z}, n \in \mathbb{N}_0$$

bilden somit eine orthonormale Basis in $L_2(\mathbb{R})$, sodass jede Funktion $f \in L_2(\mathbb{R})$ beschrieben werden kann als

$$f = \sum_{k\in\mathbb{Z}} c_k(f)\psi_{0,k} + \sum_{n=0}^{\infty}\sum_{k\in\mathbb{Z}} d_k^n(f)\psi_{1,k}^n, k \in \mathbb{Z}, n \in \mathbb{N}_0,$$

wobei c_k und d_k^n den Skalierungs, bzw. Wavelet- Koeffizienten entsprechen.

$$c_k(f) \coloneqq \int_{\mathbb{R}} f(t)\psi_0(t-k)dt$$

$$d_k^n(f) \coloneqq 2^{\frac{n}{2}} \int_{\mathbb{R}} f(t)\psi_1(2^n t-k)dt$$

Durch den Parameter n werden die Wavelets gestreckt, bzw. gestaucht, was maßgeblichen Einfluss auf die Lokalisation des Wavelets hat. Aufbauend darauf lässt sich eine Multiresolution Analyisis definieren, auf die in Kapitel 2.3 näher eingegangen wird.

2.3 Multiresolution Analysis mit Haar Wavelets

Die Mulitresolution Analysis bildet das Werkzeug, auf dem die schnelle Wavelet-Transformation (FWT) aufbaut. Die Idee dahinter ist, das Signal, welches transformiert werden soll, in verschiedene Auflösungen zu approximieren. Dies geschieht mit Hilfe eines Tiefpassfilters, wobei in der jeweiligen Auflösung die Details mit einem Hochpassfilter extrahiert werden. Tief- und Hochpassfilter entsprechen beispielsweise den im Kapitel 2.2 dargestellten Skalierungsfunktion und Wavelet-Funktionen. Im Folgenden soll nun erstmal definiert werden, welche Eigenschaften einer MRA zu Grunde liegen. [Tra12]

Definition 2.3.1 (Multiresolution Analysis (MRA)) *Sei $f(x) \in L^2(\mathbb{R}^N), N \geqq 1$. Eine MRA von $L^2(\mathbb{R}^N)$ is ein Paar $((V_j)_{j\in\mathbb{Z}}, \varphi)$ wo V_j ein geschachtelter Unterraum (closed Subspace) von $L^2(\mathbb{R}^N)$ ist und φ eine Funktion in $L^2(\mathbb{R}^N)$ ist und folgende Eigenschaften erfüllt:*

Increasing: $\forall j \in \mathbb{Z} : V_j \subset V_{j+1}$

Scaling: $\forall j \in \mathbb{Z} : f(x) \in V_j \leftrightarrow f(2x) \in V_{j+1}$

Separability $\bigcap_{j\in\mathbb{Z}} V_j = \{0\}$

Density $\bigcup_{j\in\mathbb{Z}} V_j = L^2(\mathbb{R}^N)$

Die Familie $\{\varphi(\bullet - k), k \in \mathbb{Z}^N\}$ *ist eine orthonormale basis von* V_0

Ein Signal soll also bei der MRA so in Unterräume zerlegt werden, dass Unterräume mit höherem Index feiner sind als mit niedrigeren. Des Weiteren sollen diese Unterräume dicht in $L^2(\mathbb{R}^N)$ sein, somit den gesamten Raum ausfüllen und dabei keine Redundanz aufweisen (Separability). Die in Definition 2.3.1 beschriebene "Scaling" Eigenschaft beschreibt, dass ein Signal in verschiedenen Auflösungsstufen betrachtet wird, wobei sich die "Auflösung" zwischen V_j und V_{j+1} um den Faktor 2 unterscheidet. Die Information, die bei der Approximation von Raum V_{j+1} zu V_j verloren geht, beschreibt die Räume W_j, welche orthogonal komplementär zu V_j in V_{j+1} sind, sodass gilt

$$W_j \oplus V_j = V_{j+1}.$$

Ein Raum W_j wird auch als Detailraum bezeichnet [Tra12]. Die Skalierungsfunktionen und Waveletfunktionen für $\mathbb{R}^s$ können leicht mit Hilfe von Tensorprodukten der Basisfunktionen von Kapitel 2.2 wie folgt generiert werden.

$$\psi_\theta(x) := \prod_{j=1}^{s} \psi_{\theta_j}(x_j), \theta \in \{0,1\}^s \setminus \{0\}.$$

Daraus ergeben sich dann die Skalierungs-Koeffizienten

$$c_\alpha(f) = \int_{\mathbb{R}^s} f(t)\psi_0(t-\alpha)dt, \alpha \in \mathbb{Z}^s$$

und die Wavelet- Koeffizienten mit $\theta \in \{0,1\}^s \setminus \{0\}$

$$d^n_{\theta,\alpha}(f) = 2^{\frac{ns}{2}} \int_{\mathbb{R}^s} f(t)\psi_\theta(2^n t - \alpha)dt, \alpha \in \mathbb{Z}^2, n \in \mathbb{N}_0,$$

worauf sich die orthogonale Repräsentation ergibt

$$f = \sum_{\alpha\in\mathbb{Z}^s} c_\alpha(f)\psi_{0,\alpha} + \sum_{n=0}^{\infty}\sum_{\theta\neq 0}\sum_{\alpha\in\mathbb{Z}^s} d^n_{\theta,\alpha}(f)\psi^n_{\theta,\alpha}, \psi^n_{\theta,\alpha} := 2^{\frac{ns}{2}}\psi_\theta(2^n \bullet - \alpha)\ .$$

In der praktischen Anwendung wird natürlich nur bis zu einem maximalen Level n_{max} summiert [SS22].

2.4 Diskrete 2D Wavelet Transformation

Wie in Kapitel 2.2 erwähnt lassen sich Haar Wavelets für die mehrdimensionale Wavelet-Transformation durch Tensorprodukte generieren. Die 2D-Haar Wavelets sind dann wie folgt definiert

$$\psi_{(0,0)}(x,y) = \psi_0(x)\psi_0(y)$$
$$\psi_{(0,1)}(x,y) = \psi_0(x)\psi_1(y)$$
$$\psi_{(1,0)}(x,y) = \psi_1(x)\psi_0(y)$$
$$\psi_{(1,1)}(x,y) = \psi_1(x)\psi_1(y)$$

Die Haar-Wavelets müssen noch bezüglich der Energie normiert werden. Das heißt insbesondere, dass

$$\sum_{\alpha\in\mathbb{Z}} |\psi_{\theta_j}(\alpha)|^2 = 1.$$

Die Normalisierungsfaktoren für die Haar Wavelets lassen sich aufgrund dessen für R^s und in Abhängigkeit von der Skalierung n dann so ermitteln

$$A_{n,s} = 2^{\frac{ns}{2}}, n \in \mathbb{N} .$$

Somit ergeben sich für den 2D Fall Vorfaktoren von $A_{n,2} = 2^n$. Die Wavelet-Familie für die 2D Haar Wavelet-Transformation besteht somit aus 3 Wavelet Funktionen, die wie folgt definiert sind.

$$\psi^n_{\theta,\alpha} := 2^n \psi_\theta(2^n \bullet - \alpha) .$$

In Abbildung 2.2 sind beispielhaft 2D Haar-Wavelets verschiedener Skalierungen abgebildet. Durch die Orientierung kann man erkennen, dass das Wavelet $\psi_{(0,1)}(x,y)$ Informationen über horizontale Kanten bei der Filterung hervorhebt und $\psi_{(1,0)}(x,y)$ bzw. $\psi_{(1,1)}(x,y)$ vertikale bzw. diagonale Kanten. $\psi_{(0,0)}(x,y)$ entspricht der Skalierungsfunktion.

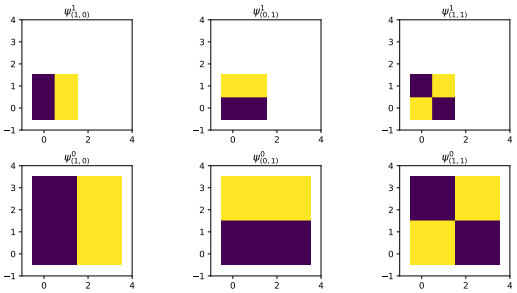


Abbildung 2.2: Beispiele von 2D Haar-Wavelets für zwei Skalierungen

2.5 Diskrete 3D Wavelet Transformation

Die in 2.4 beschriebene 2D DWT lässt sich auch leicht auf den 3D Fall anpassen. Die 3D-Haar Wavelets sind dann wie folgt definiert

$$\psi_{(0,0,0)}(x,y,z) = \psi_0(x)\psi_0(y)\psi_0(z)$$
$$\psi_{(0,0,1)}(x,y,z) = \psi_0(x)\psi_0(y)\psi_1(z)$$
$$\psi_{(0,1,0)}(x,y,z) = \psi_0(x)\psi_1(y)\psi_0(z)$$
$$\psi_{(0,1,1)}(x,y,z) = \psi_0(x)\psi_1(y)\psi_1(z)$$
$$\psi_{(1,0,0)}(x,y,z) = \psi_1(x)\psi_0(y)\psi_0(z)$$
$$\psi_{(1,0,1)}(x,y,z) = \psi_1(x)\psi_0(y)\psi_1(z)$$
$$\psi_{(1,1,0)}(x,y,z) = \psi_1(x)\psi_1(y)\psi_0(z)$$
$$\psi_{(1,1,1)}(x,y,z) = \psi_1(x)\psi_1(y)\psi_1(z)$$

Die Wavelets müssen ebenfalls normiert werden, somit ergeben sich für den 3D Fall Vorfaktoren von $A_{n,3} = 2^{\frac{n3}{2}}$. Die Wavelet-Familie für die 3D Haar Wavelet-Transformation besteht daher aus 7 Wavelet Funktionen, die definiert sind als

$$\psi^n_{\theta,\alpha} := 2^{\frac{n3}{2}}\psi_\theta(2^n \bullet - \alpha)$$

In Abbildung 2.3 sind beispielhaft 3D Haar-Wavelets, so wie Skalierungswavelet dargestellt. Es ergeben sich also 7 Detail-Koeffizienten für jeden Approximationslevel.

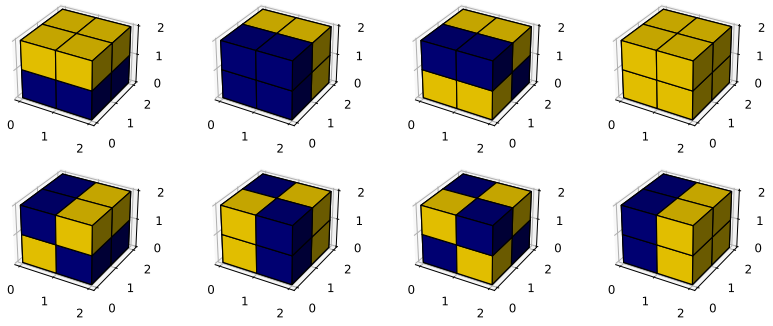


Abbildung 2.3: Darstellung der 3D Haar Wavelets für verschiedene Orientierungen

2.6 Schnelle Diskrete Wavelet Transformation

Bei der Berechnung diskreten Wavelet Transformation wird oft auf das im Kapitel 2.3 beschriebene Konzept der Mulitresolution Analysis zurückgegriffen, da hierdurch eine sehr performante Implementation möglich wird. Die schnelle Wavelet-Transformation (FWT) teilt ein Signal durch eine kaskadenartige Anwendung von Filtern in Sub-Bänder auf. Die Filter L und H entsprechen hierbei idealen Tief- und Hochpassfiltern, die gemeinsam Sub-Band Filter bilden. Nach jedem Filterungsschritt wird ein Subsampling mit Faktor 2 vorgenommen, sodass bei jedem nächsten Schritt mit abnehmender Auflösung gearbeitet wird. Dies lässt sich dadurch begründen, dass nach jedem Filterungsschritt die maximale auftretende Frequenz im Approximationsanteil nur noch halb so hoch ist. Die Down- bzw. Upsampling-Operatoren sind definiert als

Definition 2.6.1 (Downsampling-Operator) *der Ordung n Sei $n \geq 2$. Der Operator $\downarrow_n$, der eine Folge $c \in l(\mathbb{Z})$ abbildet auf*

$$\downarrow_n c = c(n \cdot \bullet),$$

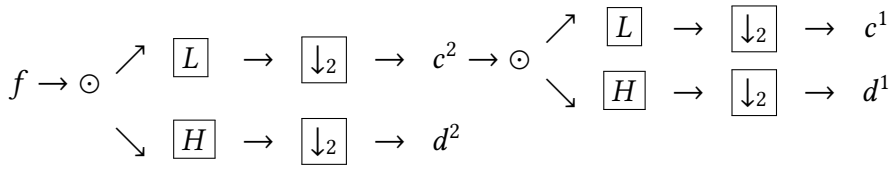
heißt Downsampling-Operator *der Ordnung n.*

Definition 2.6.2 (Upsampling-Operator) *der Ordung n. Der Operator $\uparrow_n$ mit*

$$\uparrow_n c(j) = \begin{cases} c(j/n), & j \in n\mathbb{Z} \\ 0, & j \in Z \setminus n\mathbb{Z}, \end{cases}$$

heißt Upsampling-Operator *der Ordnung n.*

Im Folgenden wird hierzu die Analyse- Filterbank dargestellt. Die Filter werden pyramidenartig angewendet, wobei der nächste Dekompositionsschritt immer auf dem Tiefpass-Anteil ausgeführt wird.



Die Ergebnisse aus der Filterung mit den Hochpassfiltern werden auch Wavelet-Koeffizienten genannt. Aufgrund des Subsamplings nach jedem Filterungsschritts ist diese Art einer DWT zwar sehr performant, jedoch nicht verschiebungsinvariant [She+92] [Bra03]. Dies kann zu Aliasing Effekten führen, welche Probleme bei Kantenerkennung verursachen. Eine Möglichkeit um eine verschiebungsinvariante DWT zu definieren, ist den Subsampling-Schritt wegzulassen.[She+92] [Bra03] Diese Version wird als ”à trous” DWT bezeichnet. Der Berechnungsaufwand steigt jedoch dadurch signifikant an. Es gibt auch Anstrengungen beide Herangehensweisen miteinander zu verheiraten, um so bei moderatem Berechnungsaufwand Aliasing Effekte zu reduzieren. Bei einer solchen ”The Over-complete” DWT (OCDWT) wird bei einer Waveletzerlegung mit N Leveln bis zu einem gewissen Level M die DWT mit Subsampling angewandt, für die restlichen N-M Level wird auf die ”à trous” Variante gewechselt [Bra03]. Bei einer Haar- FWT wird das Bild nun kaskadenartig gefiltert, wobei bei jedem Schritt beim 2D Fall 3 Detailkoeffizienten generiert werden. Das Ergebnis diese Prozedur ist in Abbildung 2.4 dargestellt.

c entspricht dem Ergebnis durch Filterung mit der Skalierungs-Funktion. Die Detailkoeffizienten $d^n_{(0,1)}$, $d^n_{(1,0)}$, und $d^n_{(1,1)}$ entprechen den Koeffizienten für horizontalen, vertikalen und diagonalen Informationen. Der letztlich überbleibende Skalierungskoeffizient c entspricht dabei dem Mittelwert des gesamten Bildes. Ein Beispiel einer schnellen 2D DWT mit Haar Wavelets für die ersten drei Level wird dazu in Abbildung 2.5 dargestellt. Das Bild wurde dabei nach dem Muster in Abbildung 2.4 in 3 Detailkoeffizienten pro Level aufgespalten. Für die 3D DWT ergibt sich analog zu Abbildung 2.4 nun ein dreidimensionales Dekompositionsschema, welches in Abbildung 2.6 für die ersten 2 Level dargestellt ist, wobei die Skalierungs- und Detail-Koeffizienten nun von kubischer Form sind. Wenn man annimmt, dass ein Signal f von quadratischer, bzw kubischer Größe mit jeweils Kantenlänge $N = 2^x, x \in \mathbb{N}$ ist, dann kann die maximale Anzahl an Level n_{max} wie folgt berechnet werden:

$$n_{max} = \lfloor log_2(N) \rfloor \ .$$

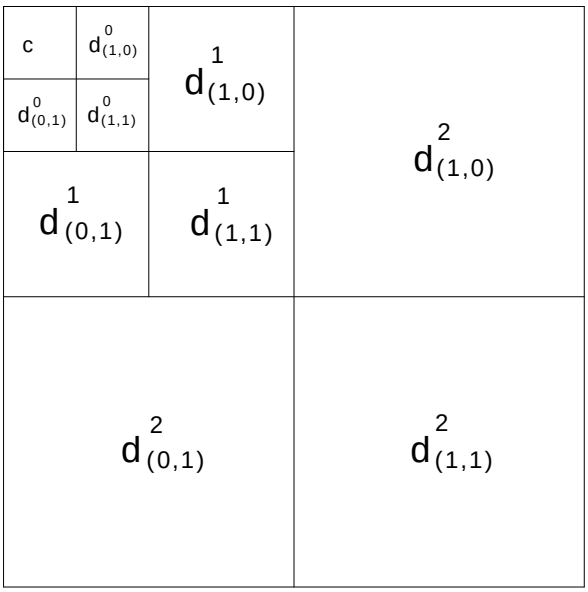


Abbildung 2.4: 2D Wavelet Decomposition

Ab einem gewissen Level lassen sich bei einer DWT kaum noch sinnvoll Information, bzw. Frequenzen, extrahieren. In Abbildung 2.7 sind für jedes Level der entsprechende Skalierungskoeffizient und das zugehörige normalisierte Histogramm dargestellt. Man kann erkennen, dass in Level 1 und Level 0 eigentlich keine relevanten Features des Bildes mehr vorhanden ist. Somit macht es durchaus Sinn einen Level n_t zu bestimmen, ab dem die DWT abgebrochen werden kann.

Bei der Filterung eines Signals mit Haar-Filtern muss das Signal an den Rändern erweitert werden, sodass die Filter angewandt werden können. Dazu gibt es im Prinzip drei Möglichkeiten:

Padding mit Nullen Die einfachste Möglichkeit ist es ein Signal durch anfügen von Nullen unendlich zu erweitern. Dies ist leicht zu implementieren, hat aber den Nachteil, dass die WT an den Rändern ungenau wird.

Periodisierung Bei dieser Variante wird das Signal an den Rändern periodisch wiederholt

Symmetrische Erweiterung Das Signal wird an den Rändern symmetrisch, also gespiegelt, fortgesetzt

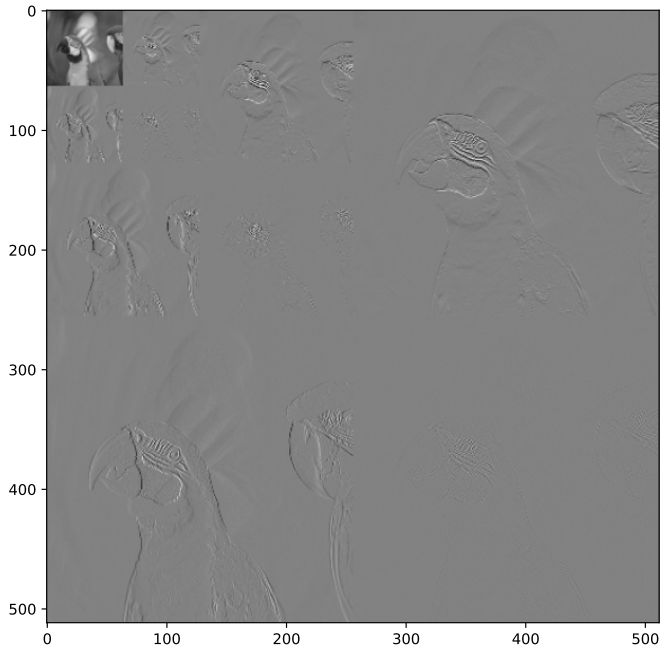


Abbildung 2.5: Beispiel einer schnellen 2D diskreten Wavelet-Transformation mit Haar Wavelets für 3 Level. In der linken oberen Ecke befindet sich der restliche Approximationsanteil

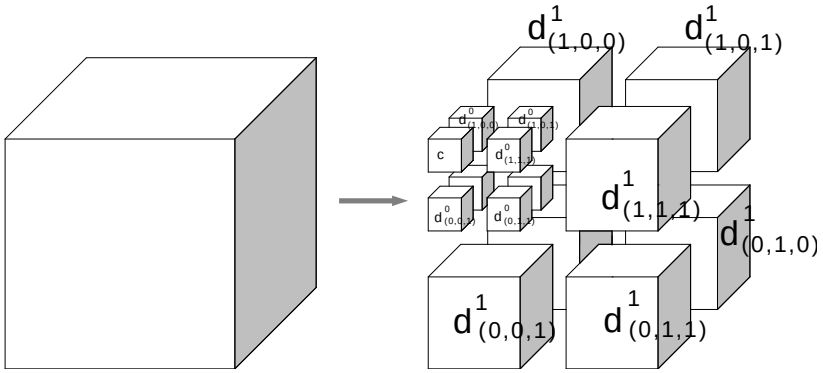


Abbildung 2.6: 3D Wavelet Decomposition für die ersten 2 Level [Tra12]

2.7 Gradienten mit Haar Wavelets

Die im Rahmen dieser Arbeit entwickelten Ähnlichkeitsmaße betrachten die Ähnlichkeit zwischen zweier Signale auf Basis der Gradientenfelder, die sich durch Anwendung der Haar-DWT approximieren lassen. Die Basis dazu bildet ein durch die Wavelet-Koeffizienten definiertes Vektorfeld.

Aus den im Kapitel 2.6 beschriebenen Wavelet-Koeffizienten lässt sich allgemein für $\beta \in \mathbb{Z}^s$

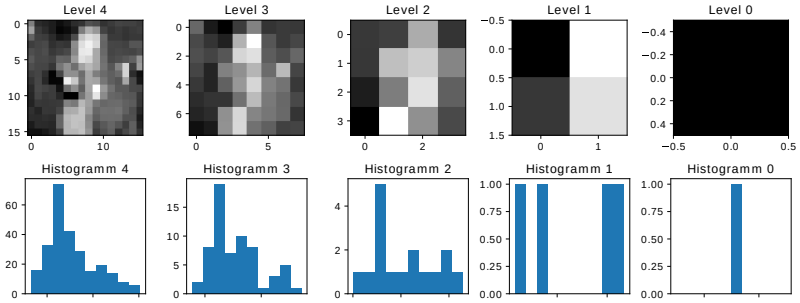


Abbildung 2.7: Skalierungskoeffizienten nach Approximationslevel. Ab einem gewissen Level können nicht mehr sinnvoll Informationen extrahiert werden.

und $n \in \mathbb{N}_0$ die Vektoren wie folgt definieren:

$$d_\alpha^n(f) := (d_{\theta_j}^n(f) : j = 1, ..., s) \in \mathbb{R}^s \ .$$

Das renormalisierte Vektorfeld

$$\hat{d}_\alpha^n(f) := -2^{n(1+\frac{s}{2})+2} d_\alpha^n(f)$$

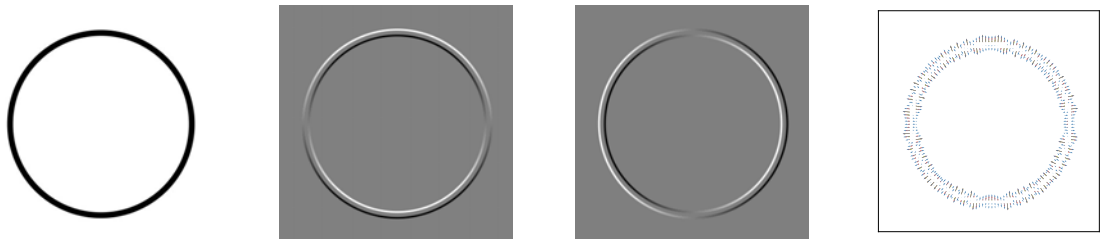
ist nach [SS22] dabei eine gute Approximation des gesampelten Gradientenfeldes einer Funktion f, welches definiert ist als

$$\nabla f(x_\alpha^n), \alpha \in \mathbb{Z}^s, n \in \mathbb{N}_0 \ .$$

Der Renormalisationsfaktor $-2^{n(1+\frac{s}{2})+2}$ bewirkt eine von der Skala n abhängigen Skalierung der Vektoren, wobei der Gradient nochmals durch eine weitere Skalierung mit dem Faktor 2^n angepasst wird. Dadurch werden die Längenunterschiede zwischen den Skalen kompensiert [SS22], sodass das Vektorfeld nun folgendermaßen definiert ist:

$$\hat{d}_\alpha^n(f) := -2^{\frac{n \cdot s}{2}+2} d_\alpha^n(f) \ .$$

In Abbildung 2.8 ist ein Beispiel eines so erzeugten Gradienten für eine Skala zu einem 2D Signal dargestellt. Dazu werden aus dem Bild, wie in Abbildung 2.8 (a) dargestellt, die Wavelet-Koeffizienten $d_{\theta_j}^n(f)$ extrahiert, welche die horizontalen und vertikalen Informationen aus dem Bild enthalten. Aus diesen Wavelet-Koeffizienten kann nun das in Abbildung 2.8 (d) gezeigte Vektorfeld gebildet werden, welches eine gute Abschätzung des Gradienten $\nabla f(x_\alpha^n)$ darstellt.



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

Abbildung 2.8: Aus den Haar-Wavelet- Koeffizienten (b)(c) des Referenzbildes (a) kann eine Approximation des Gradienten (d) gebildet werden. Es sind die Detail-Koeffizienten für Level 2 dargestellt. Die Farben der Vektoren in (d) stehen für die verschiedenen Stufen der Approximation.

2.8 Ähnlichkeit und Distanz

Ähnlichkeitsfunktionen sowie Distanzfunktionen sind Funktionen, die den Unterschied zwischen zwei Objekten quantifizieren. Eine Distanzfunktion muss dabei folgende Eigenschaften erfüllen [Sar19] [Ont20]:

Definition 2.8.1 (Distanz- Metrik) *Eine Distanzmetrik d ist eine Funktion die folgende Eigenschaften $\forall x, y$ erfüllt:*

Nicht-Negativität $d(x, y) \geq 0$

Identiät $d(x, y) = 0 \Leftrightarrow x = y$

Symmetrie $d(x, y) = d(y, x)$

Dreiecksungleichung $d(x, z) \leq d(x, y) + d(y, z)$

Während Distanzfunktionen zwei ähnlichen Objekten einen eher kleinen Wert zuordnet, ordnet eine Ähnlichkeitsfunktion den selben Objekten einen eher großen Wert zu. Ähnlichkeitsmaße spielen eine wichtige Rolle bei der Bewertung von Bildqualität (IQA), da sie es ermöglichen, die visuelle Wahrnehmung von Menschen zu modellieren und zu vergleichen. Ähnlichkeitsfunktionen sind wie folgt definiert [Sar19] [Ont20]:

Definition 2.8.2 (Ähnlichkeitsfunktion) *Eine Ähnlichkeitsfunktion ist eine Funktion $s(x, y)$, die die Ähnlichkeit zwischen zwei Objekten x und y misst und $\forall x, y$ folgende Eigenschaften erfüllt:*

Nicht-Negativität $s(x, y) \geq 0$

Begrenztheit $s(x, y) \leq u$

Symmetrie $s(x, y) = s(y, x)$

Reflexivität $s(x, x) \geq s(x, y)$

Die Eigenschaft der Symmetrie gewährleistet, dass die Ähnlichkeit zweier Objekte unabhängig von der Reihenfolge sein muss. Ein Objekt sollte auch immer die höchste Ähnlichkeit mit sich selbst haben. Dies wird durch die Reflexivität gewährleistet. Ähnlichkeiten werden typischerweise immer in einem Bereich $[0, 1]$ angegeben, sodass $u = 1$ gilt. Das grundlegende Ähnlichkeitsmaß, auf dem die in Kapitel 3 beschriebenen HaarVectorPSI Algorithmen aufbauen, heißt Kosinus-Ähnlichkeit und ist wie folgt definiert:

Definition 2.8.3 (Kosinus-Ähnlichkeit) *Für zwei Vektoren a und b entspricht die Kosinus-Ähnlichkeit dem Kosinus des eingeschlossenen θ bezüglich dem Nullvektor*

$$S_{Kosinus}(a, b) = cos(\theta) = \frac{a \cdot b}{\|a\|\|b\|} = \frac{\sum_{i=1}^{n} a_i b_i}{\sqrt{\sum_{i=1}^{n} a_i^2}\sqrt{\sum_{i=1}^{n} b_i^2}},$$

wobei a_i und b_i den jeweiligen Komponenten an Position i entsprechen.

3 Methoden

Das Hauptziel dieser Arbeit ist es, verschiedene Ähnlichkeitsmaße zu entwickeln, die geeignet sind, die Volumenähnlichkeit zwischen 3D-CT-Daten zu bestimmen. Die vorgestellten Algorithmen sind für die Anwendung auf Graustufensignalen geeignet. In Kapitel 3.1 wird der in [Rei+18] beschriebene HaarPSI-Algorithmus für Voxeldaten angewendet. In Kapitel 3.2 wird auf Basis der Möglichkeit den Gradienten eines Signals aus den Haar-Wavelet-Koeffizienten zu approximieren (wie in Kapitel 2.7 beschrieben), das Ähnlichkeitsmaß HaarVectorPSI definiert. Dazu werden auch verschiedene Varianten der Gewichtung vorgestellt sowie eine HaarVectorPSI-Variante, die die FWT implementiert. Kapitel 3.3 beschreibt eine Möglichkeit die Ähnlichkeit zwischen zwei Signalen bezüglich ihrer Verteilung von Kantenstärken und Richtungen der Gradienten zu bestimmen. Die hierzu vorgestellte HaarHistSim implementiert eine positionsunabhängige Bestimmung von Ähnlichkeit. In Kapitel 3.4 werden die Implementierungen der Algorithmen in Python vorgestellt.

3.1 HaarPSI3D

3.1.1 Definition

Der in [Rei+18] beschriebene HaarPSI Algorithmus ist für zweidimensionale Daten definiert worden. Im Folgenden soll nun HaarPSI3D ausgehend von HaarPSI hergeleitet werden, welcher auf 3D-Voxel-Daten anwendbar ist. Die Wavelet-Filter werden nun nach der Auflösungstufe j definiert. Dabei entspricht $j = 1$ der feinsten Auflösung und $j + 1$ der nächst gröberen. Zunächst werden, wie bei HaarPSI, ausgehend von den 1D-Haar-Filtern

$$\psi_0^1 = [1, 1] \text{ und } \psi_1^1 = [-1, 1] \ ,$$

durch Tensorprodukte folgende 3D Haar-Filter konstruiert:

$$\psi^j_{(0,0,1)} = 2^{-\frac{3j}{2}} \psi^j_0 \otimes \psi^j_0 \otimes \psi^j_1$$
$$\psi^j_{(0,1,0)} = 2^{-\frac{3j}{2}} \psi^j_0 \otimes \psi^j_1 \otimes \psi^j_0$$
$$\psi^j_{(1,0,0)} = 2^{-\frac{3j}{2}} \psi^j_1 \otimes \psi^j_0 \otimes \psi^j_0 \ .$$

Bei der Anwendung eines 3D-Haar-Filters auf ein Signal f werden Kanteninformationen, die in der jeweiligen Ebene liegen, extrahiert. Beispielweise würde man im Falle einer Filterung mit $\psi^j_{(0,0,1)}$ den durch die xy-Ebene definierten Anteil der Kanteninformationen erhalten. Als Ähnlichkeitsmaß wird, wie in HaarPSI, das Ähnlichkeitsmaß S verwendet, welches eine Variante des Sørensen–Dice coefficient darstellt.

Definition 3.1.1 (Ähnlichkeitsmaß S) *Für 2 skalare Parameter a und b ist das Ähnlichkeitsmaß S definiert als*

$$S(a, b, C) = \frac{2ab + C}{a^2 + b^2 + C} \ ,$$

wobei durch den skalaren Parameter C eine Gewichtung erreicht werden kann.

Für zwei Signale $f_1, f_2 \in l^2(\mathbb{Z}^3)$ ergeben sich für jede Ebene jeweils eine feature-map, welche definiert sind durch

$$HS^{(k)}_{f_1,f_2}(x) = l_\alpha \left(\frac{1}{2} \sum_{j=1}^{2} S\left(\left| (\psi^j_{(k)} * f_1)(x) \right|, \left| (\psi^j_{(k)} * f_2)(x) \right|, C \right) \right) \ .$$

Durch den Parameter $k \in O,\ O = \{(0,0,1), (0,1,0), (1,0,0)\}$ werden die Wavelet-Filter für die Ebenen $\{xy, xz, yz\}$ ausgewählt. Diese werden dann jeweils mit den Signalen f_1, f_2 gefaltet und positionsweise durch den Sørensen–Dice coefficient verglichen. Anschließend wird über die Auflösungsstufen $j \in \{1, 2\}$ gemittelt, sodass für jede Ebene eine feature-map $HS^{(k)}_{f_1,f_2}(x)$ generiert wird. Wie in [Rei+18] beschrieben, kann es sinnvoll sein das Ergebnis der lokalen Ähnlichkeit durch eine logistische Funktion nicht-linear abzubilden. Diese Funktion ist definiert als

$$l_\alpha(x) = \frac{1}{1 + e^{-\alpha x}},\ \alpha > 0 \ .$$

Mit Hilfe der Haar-Filter $\psi^3_{(k)}$, die eher niedere Frequenzen erfassen, wird analog zu HaarPSI folgende Gewichtungs-map für beide Signale f_1, f_2 definiert:

$$W_f^{(k)}(x) = \left|(\psi^j_{(k)} * f)(x)\right| \ .$$

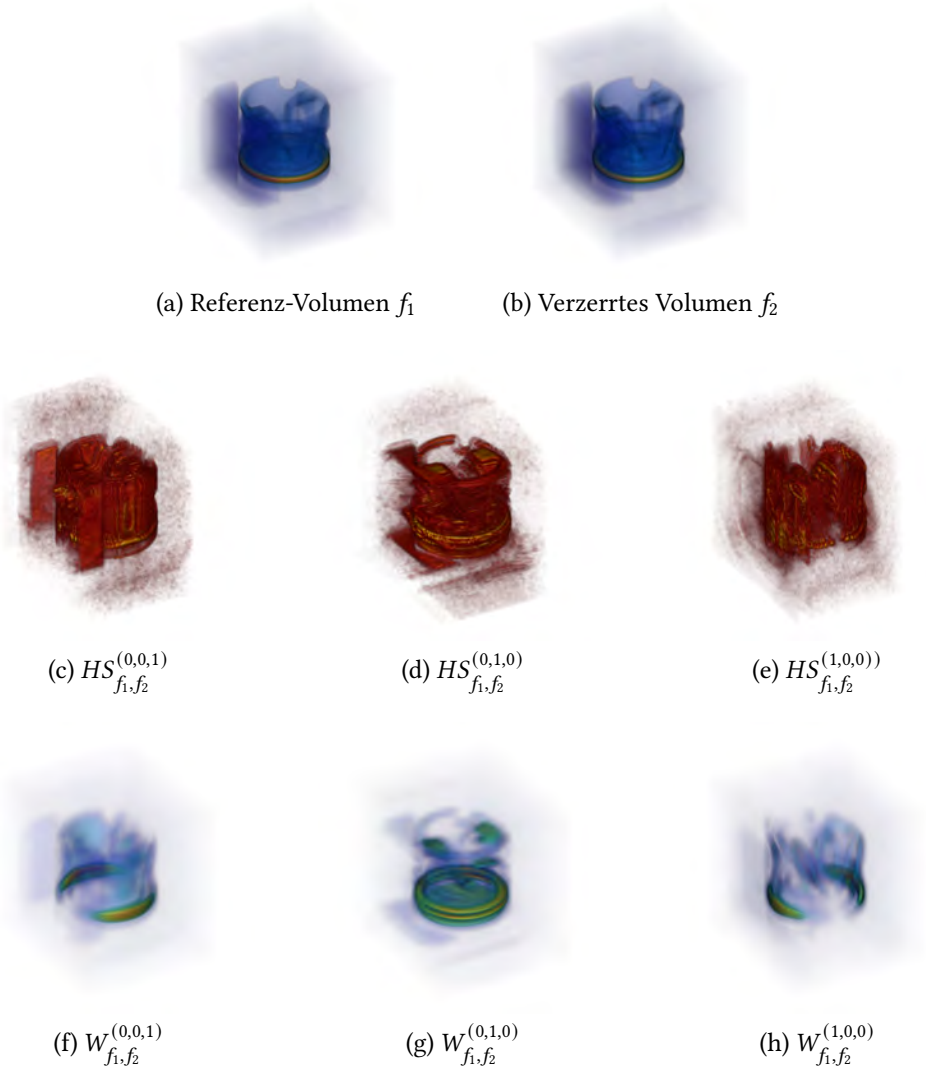
Die Berechnung des HaarPSI3D kann also analog zu HaarPSI durch Gewichtung und Mittelung der $HS^{(k)}_{f_1,f_2}(x)$ feature-maps mit der Gewichtungs-Map $W^{(k)}_{f_1,f_2}(x) = max\left(W^{(k)}_{f_1}(x), W^{(k)}_{f_2}(x)\right)$ erfolgen, sodass HaarPSI3D definiert ist als:

$$\text{HaarPSI3D}_{f_1,f_2} = l_\alpha^{-1}\left(\frac{\sum_x \sum_{k\in O} HS^{(k)}_{f_1,f_2}(x) \cdot W^{(k)}_{f_1,f_2}(x)}{\sum_x \sum_{k\in O} W^{(k)}_{f_1,f_2}(x)}\right)^2 \ .$$

Die feature-Maps und Gewichtungs-maps, die aus einem CT-Scan eines Kolbens entstehen, sind in Abbildung 3.1 zu sehen.

3.1.2 Parameterbestimmung

Das Ziel dieses Kapitels ist es, die Parameter α und C zu bestimmen, die für HaarPSI3D verwendet werden. Der Parameter C steuert die Empfindlichkeit der Ähnlichkeitsfunktion S gegenüber Unterschieden zwischen den zu vergleichenden skalaren Werten. Je größer C ist, desto weniger werden kleine Unterschiede berücksichtigt und desto höher sind die Werte von S. Der Parameter α beeinflusst die Form der S-Kurve der logistischen Funktion $L_\alpha(x)$. Je größer α ist, desto steiler ist die Kurve und desto stärker werden kleine Unterschiede zwischen den Werten von S betont. Wenn zwei Bilder eine unterschiedliche Streuung um den Mittelwert der Werte von S haben, kann die Anwendung der logistischen Funktion zu einer Änderung der Ordnung führen. Dies liegt daran, dass die logistische Funktion die Werte von S nicht linear transformiert, sondern nach 1 hin abflacht. Das bedeutet, dass zwei Werte nahe 1 auf der ursprünglichen Skala, z.B. 0.8 und 0.9, auf der logistischen Skala näher zusammen sind, als zwei Werte nahe 0, z.B. 0.1 und 0.2. Betrachtet man als Beispiel zwei Maps $S_1 = [0.5, 0.5]$ und $S_2 = [0, 1]$ stellt man fest, dass beide einen Mittelwert von 0.5 haben. Wendet man nun $L_\alpha(x)$ auf beide an, so ist der Mittelwert in S_2 kleiner als der von S_1. Man kann sagen, je größer die Streuung um den Mittelwert war, desto kleiner wird auch der Mittelwert nach Anwendung der logistischen Funktion. Da HaarPSI3D direkt von HaarPSI abgeleitet ist, soll der Parameter α den gleichen Wert ($\alpha = 4.2$) haben, wie in [Rei+18] als Standardwert für alle Datenbanken festgelegt wurde. Der Parameter C muss



(a) Referenz-Volumen f_1 (b) Verzerrtes Volumen f_2

(c) $HS_{f_1,f_2}^{(0,0,1)}$ (d) $HS_{f_1,f_2}^{(0,1,0)}$ (e) $HS_{f_1,f_2}^{(1,0,0))}$

(f) $W_{f_1,f_2}^{(0,0,1)}$ (g) $W_{f_1,f_2}^{(0,1,0)}$ (h) $W_{f_1,f_2}^{(1,0,0)}$

Abbildung 3.1: CT-Scan eines Kolbens (a). Durch gaußsche Unschärfe veränderte Version des Kolbens (b). Die Ähnlichkeitsmaps HS_{f_1,f_2} für jede Ebene (c)(d)(e). Die Gewichtungen W_{f_1,f_2} für jede Ebene (f)(g)(h)

jedoch an den Wertebereich der 3D-CT Voxeldaten angepasst werden. Der Wertebereich eines graustufigen Bildes ist $[0, 255]$, der von 3D-CT Voxeldaten ist viel breiter. Der in Kapitel 4 verwendete 3D-CT-Scan eines Kolben hat einen maximalen Wert von ca. 57000.

Der Parameter C wird nun so angepasst, dass er für den neuen Wertebereich sinnvoll gewählt ist. Damit die Funktion S im neuen Wertebereich exakt das selbe Verhalten hat, muss C so gewählt werden, dass das Verhältnis von C zum Quadrat des maximalen Wertes von a und b gleich bleibt. Das heißt, wenn C=30 für einen Wertebereich von 0-255 gilt, dann

muss C für einen Wertebereich von 0-57000 gelten:

$$\frac{C}{255^2} = \frac{C'}{57000^2} \ .$$

Daraus folgt:

$$C' = \frac{C \cdot 57000^2}{255^2} = \frac{30 \cdot 57000^2}{255^2} \approx 1500000 \ .$$

Also muss C ungefähr 1500000 gewählt werden, damit die Funktion S im neuen Wertebereich exakt das selbe Verhalten hat.

Mit diesen Parametern wurde HaarPSI3D auf verschiedene 3D-CT-Daten angewendet und mit anderen Methoden verglichen. Die Ergebnisse werden in Kapitel 4 präsentiert.

3.2 HaarVectorPSI

Die im Kapitel 3.1 vorgestellte Definition eines Ähnlichkeitsmaßes basiert im Grunde auf dem Vergleich der durch die Haar-Wavelet-Transformation generierten Waveletkoeffizienten. In Kapitel 2.7 wird dargelegt, dass es möglich ist, aus den Wavelet-Koeffezienten $d^n_{\theta_j,\alpha}(f)$ Vektorfelder $\hat{d}^n_\alpha(f)$ zu erzeugen, welche gute Annäherungen an die Gradientenfelder $\triangledown f(x^n_\alpha)$ darstellen. Die hier entwickelten Varianten von HaarVectorPSI zielen darauf ab, ein Ähnlichkeitsmaß auf Basis genau dieser Gradientenfelder für graustufige 2D und 3D Daten zu definieren. Hierbei entspricht die Länge eines Vektors der Stärke (Kontrast) einer Kante und die Richtung eines Vektors gibt Information über die Ausrichtung der Kante. Durch die Getrenntheit dieser Informationen ist es möglich diese bei der Berechnung der Ähnlichkeit unterschiedlich zu Gewichten.
Aufgrund des vektor-basierten Ansatzes lässt sich HaarVectorPSI zudem auch leicht auf n-dimensionale Signale anwenden. Der in Kapitel 3.2.2 beschriebene HaarVectorPSI Algorithmus verwendet das in Kapitel 2.6 dargestellte Konzept einer DWT ohne Subsampling um Aliasing Effekten vorzubeugen. In Kapitel 3.2.3 wird eine Variante von HaarVectorPSI präsentiert, die auf der schnellen Wavelet-Transformation (FWT), wie in Kapitel 2.6 beschrieben, aufbaut. Bevor nun die drei verschiedenen Arten eines solchen gradientenbasierten Ansatzes beschrieben werden, soll zunächst auf die Auswahl einer geeigneten Ähnlichkeitsfunktion für Gradientenvektoren eingegangen werden.

3.2.1 Auswahl Ähnlichkeitsfunktion

Im Folgenden soll ein Ähnlichkeitsmaß definiert werden, welches geeignet ist eine Ähnlichkeit zwischen den Vektoren zweier Gradientenfelder zu bestimmen. Da, wie bereits erwähnt, zwei Vektoren verglichen werden sollen, bietet sich die weit verbreitet eingesetzte Kosinusähnlichkeit als logische Wahl an, welche in Kapitel 2.8 definiert ist. Sie gibt den Kosinus des eingeschlossenen Winkels zweier Vektoren an, sodass sie auch deshalb als passende Wahl erscheint, da sie durch diese geometrische Betrachtung sehr gut zum Anwendungsfall von HaarVectorPSI passt. Die Kosinusähnlichkeit gibt nur Aufschluss darüber, wie ähnlich sich zwei Vektoren bezüglich ihrer Richtung sind und bezieht hierbei nicht die Länge der Vektoren mit ein. Im Anwendungsfall von HaarVectorPSI ist dies aber gewünscht, da die Länge eines Vektors die Stärke einer Kante darstellt, sodass im speziellen folgende Eigenschaften erfüllt werden sollen:

Längensensibilität Betrachtet man zwei Vektoren v_1 und v_2, die einen festen Winkel $\beta > 0$ einschließen, so soll die Ähnlichkeitsfunktion folgende Eigenschaften bezüglich deren Längen l_1 und l_2 aufweisen:

- $\Delta l = |l_1 - l_2|$. Je größer Δl desto unähnlicher sind die Vektoren v_1 und v_2
- $l_1 = l_2 = 0 + \epsilon, \epsilon >= 0$. Für $\epsilon \longrightarrow 0$ soll die Ähnlichkeit zwischen den Vektoren v_1 und v_2 zunehmen und für den Fall $\epsilon \longrightarrow \infty$ gegen 1 gehen. Für $\epsilon \longrightarrow \infty$ soll sich die Ähnlichkeit zwischen den Vektoren v_1 und v_2 vermindern und dem Wert der reinen winkelabhängigen Kosinusähnlichkeit annähern.

Winkelsensibilität Betrachtet man zwei Vektoren v_1 und v_2 mit festen Längen $l_1 > 0$ und $l_2 > 0$, so sollen sie sich umso unähnlicher sein, je größer der eingeschlossene Winkel β ist.

Im Anwendungsfall des Vergleichs zweier Bilder begründet sich die Längensensibilität durch zwei Punkte: Zum einen soll der Unterschied der Stärke von Kanten auch Auswirkung auf die Ähnlichkeit haben. Zum anderen sollen zwei sehr schwache Kanten, also zwei Gradientenvektoren kleiner Länge, eine eher hohe Ähnlichkeit haben. Sind die Kantenstärken sehr klein, so ist man im Bereich von Rauschen. In diesem Fall soll die Ähnlichkeit unbeachtet vom eingeschlossenen Winkel sehr hoch sein. Die Winkelsensibilität ergibt sich aus der reinen Tatsache, dass zwei Kanten umso unähnlicher sind, je größer der eingeschlossene Winkel ist. Ausgehend von diesen Voraussetzungen kann eine längenabhängige Kosinusähnlichkeit nun so definiert werden, dass den zu vergleichenden Vektoren eine

zusätzliche Dimension hinzugefügt wird, die anhand der Länge der Vektoren definiert wird, sodass die Ähnlichkeitsfunktion nun so definiert ist:

Definition 3.2.1 (Längensensitive Kosinus-Ähnlichkeit) *Für zwei Vektoren a und b, mit Dimension n, entspricht die längensensitive Kosinus-Ähnlichkeit dem Betrag des Kosinus des eingeschlossenen Winkels θ der Vektoren $\hat{a}$ und $\hat{b}$ mit*

$$\begin{aligned}\hat{a} &= \left(a_1', ..., a_n', L_\alpha(\|a\|, C)\right)\\ \hat{b} &= \left(b_1', ..., b_n', L_\alpha(\|b\|, C)\right)\ ,\end{aligned}$$

wobei a' und b' die Einheitsvektoren zu a und b sind und die Funktion L definiert ist als

$$L_A(x, C) = \frac{1}{(A \cdot x)^C}\ ,$$

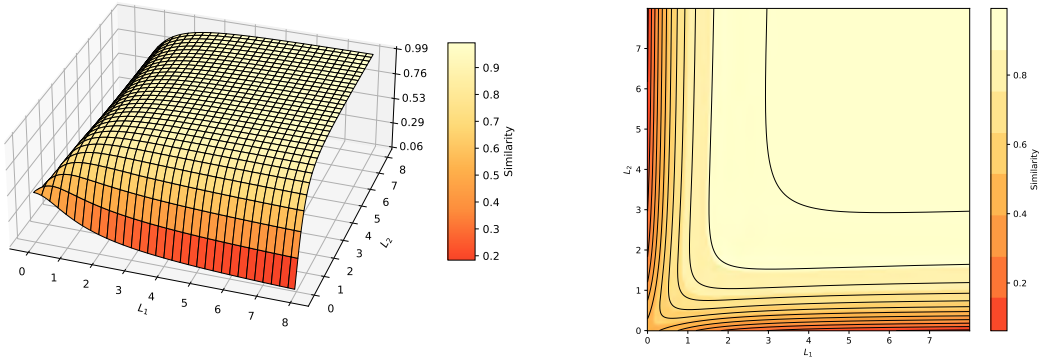
sodass die längensensitive Kosinusähnlichkeit dann wie folgt berechnet werden kann:

$$S_{LSKosinus}(a, b) = \left|\cos(\theta)\right| = \left|\frac{\hat{a} \cdot \hat{b}}{\|\hat{a}\|\ \|\hat{b}\|}\right| = \left|\frac{\sum_{i=1}^{n} \hat{a}_i \hat{b}_i}{\sqrt{\sum_{i=1}^{n} \hat{a}_i^2}\sqrt{\sum_{i=1}^{n} \hat{b}_i^2}}\right|\ ,$$

wobei $\hat{a}_i$ und $\hat{b}_i$ den jeweiligen Komponenten an Position i entsprechen.

Die grundlegende Idee, eine zusätzliche Dimension d_L zu einem Vektor a hinzuzufügen, besteht darin, die Länge des Vektors a so zu berücksichtigen, dass der Wert in d_L umgekehrt proportional zur Länge des Vektors ist. Je größer der Wert in Dimension d_L ist, desto kleiner ist der Winkel zwischen den Vektoren $\hat{a}$ und $\hat{b}$, was eine höhere Ähnlichkeit impliziert. Dieses Prinzip ist in Abbildung 3.2 veranschaulicht.

Dargestellt wird hier die Ähnlichkeit zweier Vektoren a und b unter einem festen Winkel β in Abhängigkeit der jeweiligen zusätzlichen Dimensionen L_1 und L_2. Es lässt sich erkennen, dass die geforderten Eigenschaften der Längensensibilität erfüllt werden. Wird der Längenunterschied zweier Vektoren größer, so nimmt auch die Ähnlichkeit der Vektoren ab. Betrachtet man zwei Vektorenpaare $p_1 = (v_1, v_2)$ und $p_2 = (v_3, v_4)$ mit gleichem eingeschlossenen Winkel β, wobei $\|v_1\| < \|v_3\|$ und $\|v_2\| < \|v_4\|$ gelten soll, dann ist die Ähnlichkeit von p_1 höher als p_2, da die resultierenden Dimensionen L_1 und L_2 von p_1 größer sind.



(a) (b)

Abbildung 3.2: Kosinus-Ähnlichkeit für zwei Vektoren a und b unter Konstantem Winkel $\beta = 60$ aber veränderlichen zusätzlichen Dimensionen L1 und L2

Der Wert der Zusatzdimension wird durch die Funktion L_A berechnet, wobei sich aus der Tatsache, dass $x > 0$ und

$$\lim_{x \to 0} L_A(x, C) = \infty$$
$$\lim_{x \to \infty} L_A(x, C) = 0$$

der Wertebereich $(0, \infty)$ ergibt, auf dem die Länge des Vektor abgebildet wird. Der Parameter A bewirkt hierbei eine Skalierung in x-Richtung und der Faktor C beeinflusst die Stärke der Krümmung der Funktion. Für einen Vektor a und $d_L \to \infty$ geht der eingeschlossene Winkel β von $\hat{a}$ mit dem Vektor $z = (0, ..., 0, 1)$ gegen 0. Die Richtung von a spielt also keine Rolle mehr. Somit gilt bezüglich der längensensitiven Kosinusähnlichkeit $\hat{a} = z$, für den Fall, dass a ein Nullvektor ist. In Abbildung 3.3 ist hierzu die resultierende längensensitive Kosinusähnlichkeit in Abhängigkeit von Winkel und Länge dargestellt. Es lässt sich gut erkennen, dass die geforderten Eigenschaften der Längen- und Winkelsensibilität erfüllt sind. Bei steigendem eingeschlossenen Winkel β nimmt die Ähnlichkeit ab, wobei die Abnahme bei gleichem Winkel umso flacher ist, je kürzer die Vektoren sind. Somit ist die längensensitive Kosinusähnlichkeit ein geeignetes Ähnlichkeitsmaß für den Anwendungsfall von HaarVectorPSI.

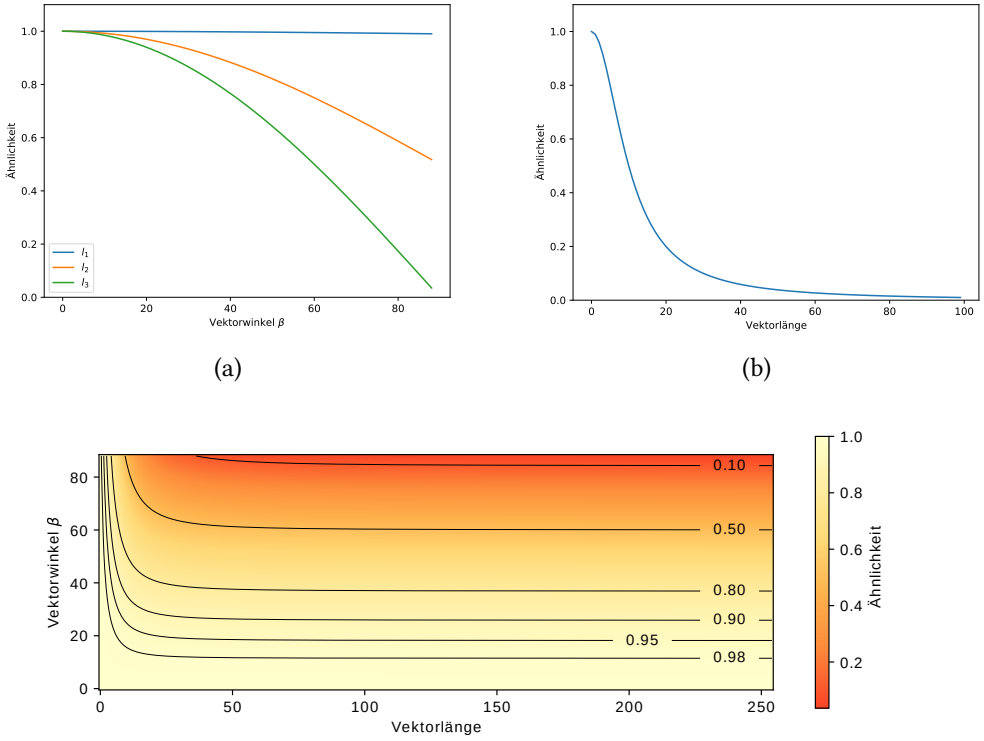


Abbildung 3.3: Darstellung der Ähnlichkeit in Abhängigkeit vom eingeschlossenen Winkel β zweier Vektoren und deren Länge. In (a) werden Vektoren mit festen Längen $l_1 < l_2 < l_3$ verglichen. In (b) werden Vektoren mit festen eingeschlossenen Winkel $\beta = 90$ verglichen. In (c) wird die Abhängigkeit der Ähnlichkeit von β und Länge zugleich dargestellt.

3.2.2 HaarVectorPSI

Der Ansatz des vektorbasierten HaarVectorPSI ähnelt in seiner Grundstruktur anderen Ähnlichkeitsmaßen, wie HaarPSI oder auch FSIM, in dem Sinne, dass auch hier feature-maps und Gewichtungs-maps zu Bestimmung der Ähnlichkeit berechnet werden. Im Folgenden soll HaarVectorPSI für den allgemeinen s-dimensionalen Fall betrachtet werden. Analog zu Kapitel 3.1, werden nun die Wavelet-Filter nach der Auflösungsstufe j definiert, wobei $j = 1$ der feinsten und $j + 1$ der nächst gröberen Auflösung entspricht. Die zur Berechnung der Waveletkoeffizienten nötigen Haar-Filter lassen sich, wie in Kapitel 2.3 beschrieben, aus den eindimensionalen diskreten Haar Wavelets

$$\psi_0^1 = [1, 1] \text{ und } \psi_1^1 = [-1, 1]$$

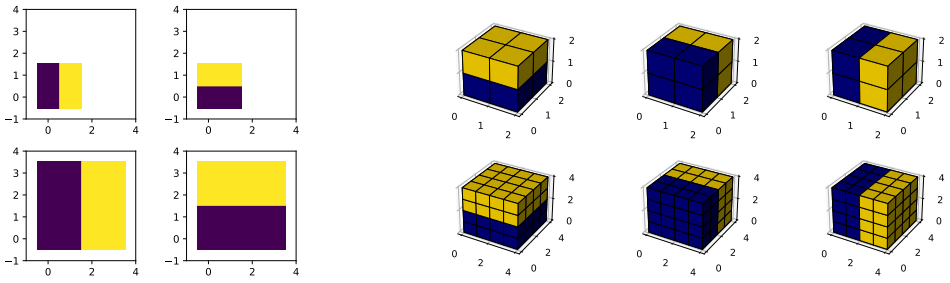
durch Tensorprodukte berechnen, wobei ψ_0^1 dem Tiefpassfilter und ψ_1^1 dem Hochpassfilter für die Auflösungstufe $j = 1$ entsprechen. Für die Bildung des s-dimensionalen Gradienten sind genau s Wavelet Filter notwendig, die bei der Filterung des Signals die Anteile in den jeweiligen Dimensionen erzeugen. Für den 2D-Fall ($s = 2$) sind für die Gradientenbildung nötigen normierten Wavelets für horizontale und vertikale Anteile am Bild wie folgt bestimmt:

$$\psi_{(0,1)}^j = 2^{-j} \psi_0^j \otimes \psi_1^j$$
$$\psi_{(1,0)}^j = 2^{-j} \psi_1^j \otimes \psi_0^j ,$$

wobei ψ_0^j und ψ_1^j den für eine Auflösungsstufe $j \in \mathbb{N}$ skalierten 1D Haar-Wavelets entsprechen. Im Falle eines dreidimensionalen Signals sind zur Ermittlung des 3D-Gradienten die Haar-Wavelets nötig, aus denen sich die Anteile in den Ebenen (x,y), (x,z) und(y,z) ergeben, also:

$$\psi_{(0,0,1)}^j = 2^{-\frac{3j}{2}} \psi_0^j \otimes \psi_0^j \otimes \psi_1^j$$
$$\psi_{(0,1,0)}^j = 2^{-\frac{3j}{2}} \psi_0^j \otimes \psi_1^j \otimes \psi_0^j$$
$$\psi_{(1,0,0)}^j = 2^{-\frac{3j}{2}} \psi_1^j \otimes \psi_0^j \otimes \psi_0^j$$

Die 2D und 3D Haar-Filter für eine Skalierung sind in Abbildung 3.4 dargestellt.



(a) (b)

Abbildung 3.4: Die 2D und 3D Haar Filter für die ersten zwei Auflösungsstufen, welche für die Berechnung des Gradienten genutzt werden

Die Wavelet-Koeffizienten für ein Signal f sind für eine Auflösungsstufe j dann definiert durch

$$d^j_{f,k}(x) = \left(\psi^j_k * f\right)(x), k \in O \ .$$

Die Menge O enthält die Richtungen der Wavelets, die für die Bildung des Gradienten benötigt werden. Für den 2D-Fall gilt $O = \{(0,1),(1,0)\}$ und für den 3D-Fall $O = \{(0,0,1),(0,1,0),(1,0,0)\}$. Der Operator $*$ entspricht hierbei dem Faltungsoperator, sodass sich nach Definition 2.1.4 die Faltung zwischen dem Signal f und dem Wavelet-Filter ψ^n_k auch als Multiplikation der jeweiligen Fouriertransformierten beschreiben lässt. Die Wavelet-Koeffizienten lassen sich unter Anwendung der DFT aufgrund dessen auch wie folgt berechnen:

$$d^j_{f,k}(x) = DFT^{-1}\left(DFT(\psi^j_k) \cdot DFT(f)\right)(x), k \in O \ ,$$

wobei DFT^{-1} der umgekehrten DFT entspricht. Dadurch lässt sich bei der Implementation auf Bibliotheken zurückgreifen, die sehr gut optimierte Versionen der DFT oder gleich FFT abbilden.

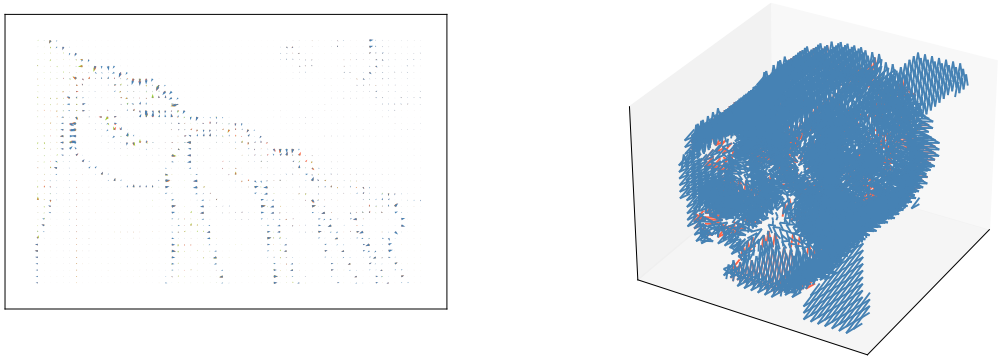
Analog zu Kapitel 2.7 wird nun aus den gewählten Waveletkoeffizienten für jede Auflösungsstufe j ein Vektorfeld $GM^j_f(x)$ gebildet, welches, wie in Kapitel 2.7 dargelegt, eine gute Annäherung an den Gradienten $\nabla f(x^j_\alpha)$ des Signals f mit Dimension s für die Auflösung j darstellt.

$$GM^j_f(x) = 2^{-\frac{js}{2}} \begin{pmatrix} d^j_{f,k_1}(x) \\ \vdots \\ d^j_{f,k_s}(x) \end{pmatrix}$$

In Abbildung 3.5 ist beispielhaft ein 2D sowie 3D Gradient dargestellt. Die Gradientenmaps werden nun für zwei Signale f_1 und f_2 gebildet und mit Hilfe des in Kapitel 3.2.1 ausgewählten Ähnlichkeitsmaßes der längensensitiven Kosinusähnlichkeit verglichen. Die daraus resultierenden feature similarity maps $CM^j_{f1,f2}$ können im Bezug zu HaarPSI mit der $HS^k_{f_1,f_2}$ Map verglichen werden und sind für $j \in \{1,2\}$ definiert durch:

$$CM^j_{f1,f2}(x) = S_{\text{LSKosinus}}\left(GM^j_{f_1}(x), GM^j_{f_2}(x)\right) \ .$$

In Abbildung 3.6 sind similarity maps zwischen 2D-Signalen für 2 verschiedene Auflösungen dargestellt, wobei jeder Wert x dabei dem Ähnlichkeitswert zwischen den Vektoren $GM^j_{f_1}(x)$ und $GM^j_{f_2}(x)$ an Position x entspricht. Genau wie HaarPSI, betrachtet HaarVectorPSI die Ähnlichkeit zwischen zwei Objekten aus der Perspektive der menschlichen Wahrnehmung.



(a) (b)

Abbildung 3.5: 2D Gradientenmap des Referenzbildes ”Caps” aus der LIVE Image database für drei Auflösungsstufen (a) und 3D Gradientenmap eines Kolbens (4.1)

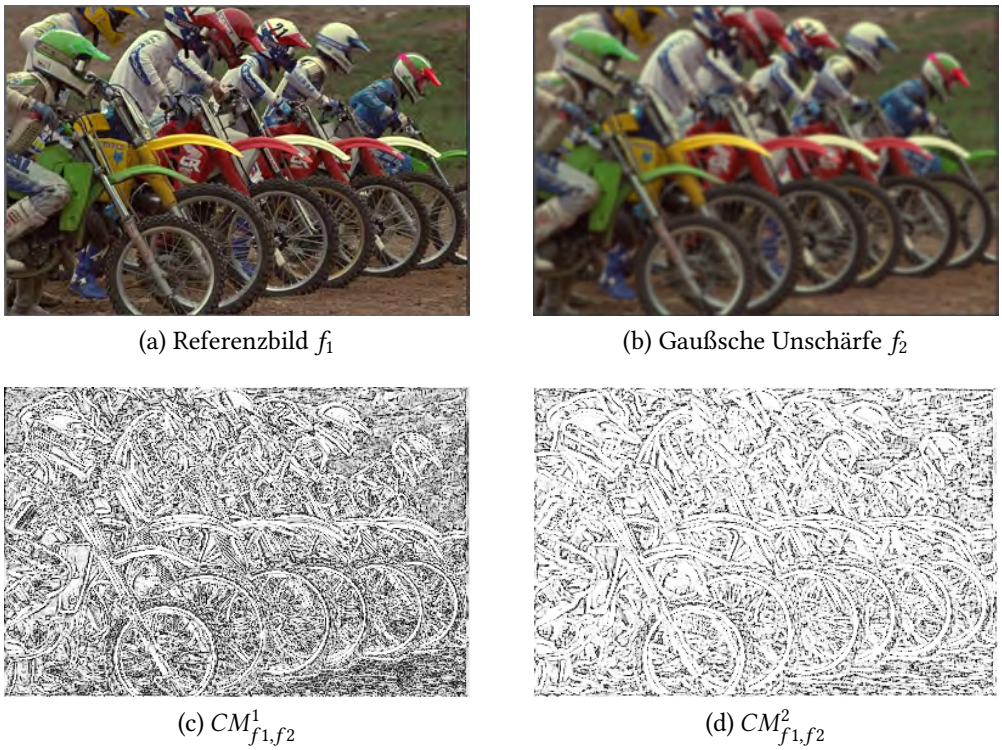
Das HVS (human visual system) verarbeitet ein visuelles Signal gleichzeitig in mehreren Auflösungstufen. Dies wird durch den MRA Ansatz der WT mit Haar-Wavelets abgebildet. Wie bei HaarPSI, sollen auch bei HaarVectorPSI Abweichungen zwischen zwei Bildern vor allem im Bereich grober starker Kanten mehr Gewichtung erfahren. Dies wird im Falle von HaarVektorPSI für s-dimensionales Signal f_1 und f_2 durch folgende Gewichtungs-map abgebildet:

$$W_{f_1,f_2}(x) = \max\left(\left\|GM^3_{f_1}(x)\right\|, \left\|GM^3_{f_2}(x)\right\|\right) ,$$

wobei $GM^3_{f_1,}(x)$ und $GM^3_{f_2}(x)$ den Gradientenfeldern durch Filterung mit den niederfrequentesten Haar-Wavelets der Auflösung $j = 3$ entsprechen. Es werden also grobe niederfrequente Kanten erkannt, die die maßgebliche Struktur des Bildes definieren. Aufgrund der durch die Skalierung des Wavelets entstehenden Unschärfe, wird ein Bereich um diese Kanten definiert, dem erhöhte Gewichtung zugewiesen werden soll. Der Wert der Gewichtungsmap an Position x entspricht dabei der maximalen Länge zweier verglichener Vektoren der gebildeten Gradientenmaps. In Abbildung 3.7 ist dies beispielhaft für ein 2D Bild dargestellt.

Die Berechnung des HaarVectorPSI zwischen zwei s-dimensionalen Signalen f_1 und f_2 über $J = 2$ Auflösungsstufen ist dann wie folgt definiert

$$HaarVectorPSI_{f_1,f_2} = l_\alpha^{-1}\left(\frac{\sum_x \sum_{j=1}^{J} l_\alpha\left(CM^j_{f1,f2}(x)\right) W_{f1,f2}(x)}{J \sum_x W_{f_1,f_2}(x)}\right)^2$$



(a) Referenzbild f_1 (b) Gaußsche Unschärfe f_2

(c) $CM^1_{f1,f2}$ (d) $CM^2_{f1,f2}$

Abbildung 3.6: Aus dem Referenzbild (a) und einem Vergleichsbild mit Gaußsche Unschärfe (b) ergeben sich die similarity maps $CM_{f1,f2,j}$ für Auflösungsstufen j=1 (c) und j=2 (d). Die Bilder sind Teil der LIVE-Datenbank.

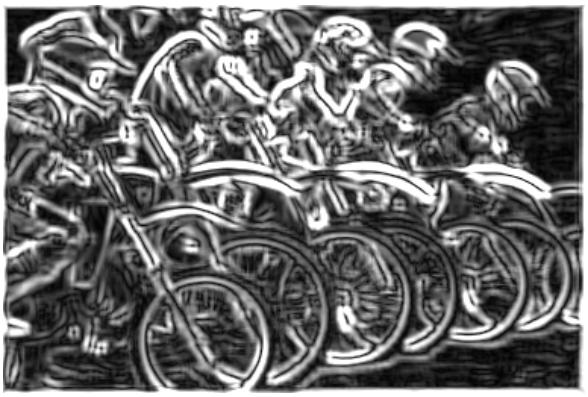
Die ermittelten Ähnlichkeitswerte der $CM^j_{f_1,f_2}$ Map für $j \in \{1,2\}$ werden hierbei durch $W_{f1,f2}(x)$ gewichtet und zu einer Gesamtähnlichkeit gemittelt. Die feature similarity-map $CM^j_{f_1,f_2}$ wird dabei, analog zu HaarPSI, durch die logistische Funktion[1] $l_\alpha(\cdot)$ nicht-linear in den Wertebereich $[\frac{1}{2}, l_\alpha(1)]$ abgebildet, um das Ansprechverhalten von Neuronen zu simulieren. Da HaarVectorPSI die Ähnlichkeit im Bereich $[0,1]$ definiert, wird durch $l^{-1}_\alpha(\cdot)$ der Wertebereich dahingehend angepasst. $l^{-1}_\alpha(\cdot)$ sowie $(\cdot)^2$ haben dabei keinen Einfluss auf die resultierende Ordnung zwischen den verglichenen Signalen, sondern dienen dazu den Wertebereich so zu strecken, sodass die reinen Zahlenwerte der Ergebnisse für einen menschlichen Betrachter sinnvoll erscheinen.

Da $S_{\text{LSKosinus}}$ sowie HaarVectorPSI auf s-dimensionalen Signalen definiert sind, lässt sich dieses Ähnlichkeitsmaß somit auch leicht auf 3D-CT Voxel Daten anwenden. Durch die Gewichtungs-map W_{f_1,f_2} wird, wie bereits erwähnt, den groben prägenden Kanten-Strukturen von Daten mehr Gewichtung zugewiesen. Ein alternativer Ansatz einer Gewichtung,

[1] Die logistische Funktion wird oft bei neuralen Netzen eingesetzt, um das Ansprechverhalten von Neuronen zu simulieren.

(a) Referenzbild f_1

(b) Gaußsche Unschärfe f_2



(c) $W_{f1,f2}$

Abbildung 3.7: Gewichtungs-map zwischen zwei Bilder aus der 'LIVE' Datenbank [She+]. Aus (a) und (b) wird eine Gewichtungs-map (c) gebildet die grobe und starke Kanten betont.

der auch feine Kantenstrukturen berücksichtigt, lässt sich durch eine skalenabhängige Gewichtungs-map realisieren. Diese wird nun wie folgt berechnet:

$$W^j_{f_1,f_2}(x) = \max\left(\left\|GM^j_{f_1}(x)\right\|, \left\|GM^j_{f_2}(x)\right\|\right) .$$

Für jede Auflösungsstufe j gibt es nun eine Gewichtungs-map, wobei für unterschiedliche Auflösungen verschieden grobe Kanten erfasst werden. Der Bereich um die Kanten wird stärker gewichtet. Dieser Bereich wird kleiner, je feiner die Auflösungsstufe ist, weil die Unschärfe durch den Haar-Filter auch kleiner wird. HaarVectorPSI mit auflösungsabhängigen Gewichtungs-maps ist dann definiert durch:

$$HaarVectorPSI^{\text{multiweight}}_{f_1,f_2} = l_\alpha^{-1}\left(\frac{\sum_x \sum_{j=1}^{J} l_\alpha\left(CM^j_{f1,f2}(x)\right) \cdot W^j_{f1,f2}(x)}{\sum_x \sum_{j=1}^{J} W^j_{f_1,f_2}(x)}\right)^2 .$$

Ein weiterer Aspekt der untersucht werden soll, ist, inwiefern die Gewichtungs-map vor allem für 3D-CT Daten aber auch 2D-Daten relevant ist. Deshalb soll nun eine gewich-

tungslose Variante von HaarVectorPSI definiert werden. Es wird zudem auf die logistische Funktionen l_α verzichtet um ein Ähnlichkeitsmaß zu definieren, welches ausschließlich auf der längensensitiven Kosinusähnlichkeit basiert und keine sonstige Gewichtung hat. Für zwei s-dimensionale Signale f_1 und f_2 der Länge p ist der gewichtungslose HaarVectorPSI für $J = 2$ definiert durch:

$$HaarVectorPSI_{f_1,f_2}^{\text{noweight}} = \left(\frac{1}{pJ} \sum_{x}^{p} \sum_{j=1}^{J} CM_{f1,f2}^{j}(x) \right)^2 .$$

Dabei wird die Ähnlichkeit durch das arithmetische Mittel der Einzelähnlichkeiten der feature-similarity map $CM_{f1,f2}^{j}(x)$ ermittelt. Da sich durch den Verzicht auf eine Gewichtung Performance-Vorteile bezüglich der Berechnungszeit ergeben, soll später unter anderem untersucht werden, inwiefern Auswirkungen auf die resultierende Ordnung bezüglich Ähnlichkeit zwischen Daten bestehen und welche Auswirkungen die verschiedenen Gewichtungsmethoden haben.

3.2.3 FWT HaarVectorPSI

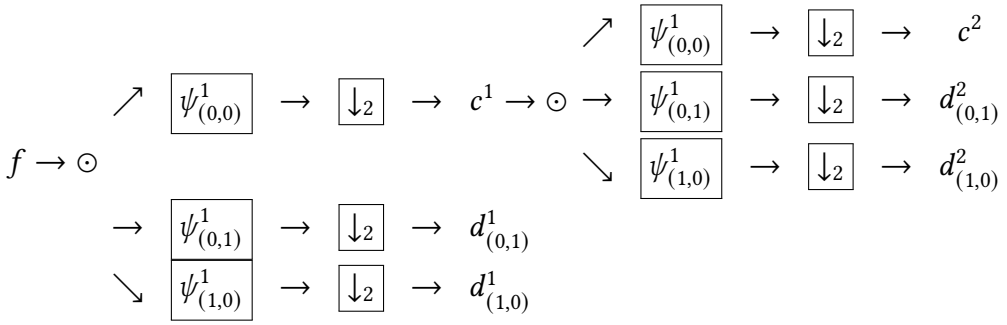
Im folgenden soll der FWT HaarVectorPSI Algorithmus beschrieben werden, welcher auf dem in Kapitel 3.2.2 vorgestellten HaarVectorPSI aufbaut, jedoch das im Kapitel 2.6 vorgestellte Prinzip der FWT benutzt. Die Wavelet-Filter werden analog zu Kapitel 3.2 bezüglich der Auflösungsstufe j definiert. HaarVectorPSI oder auch HaarPSI setzen den MRA- Ansatz insofern um, dass ein Signal mit verschieden skalierten Haar Wavelets gefiltert wird. Bei FWT HaarVectorPSI hingegen werden die Haar-Filter rekursive auf die Approximationsanteile angewendet, um eine MRA zu realisieren. Die nötigen Haar-Filter zur Bildung des Gradienten lassen sich für den 2D-Fall wie folgt definieren:

$$\begin{aligned}
\psi_{(0,0)}^{1} &= 2^{-1} \psi_0^1 \otimes \psi_0^1 \\
\psi_{(0,1)}^{1} &= 2^{-1} \psi_0^1 \otimes \psi_1^1 \\
\psi_{(1,0)}^{1} &= 2^{-1} \psi_1^1 \otimes \psi_0^1
\end{aligned}$$

und für den 3D-Fall

$$\psi^1_{(0,0,0)} = 2^{-\frac{3}{2}}\psi^1_0 \otimes \psi^1_0 \otimes \psi^1_0$$
$$\psi^1_{(0,0,1)} = 2^{-\frac{3}{2}}\psi^1_0 \otimes \psi^1_0 \otimes \psi^1_1$$
$$\psi^1_{(0,1,0)} = 2^{-\frac{3}{2}}\psi^1_0 \otimes \psi^1_1 \otimes \psi^1_0$$
$$\psi^1_{(1,0,0)} = 2^{-\frac{3}{2}}\psi^1_1 \otimes \psi^1_0 \otimes \psi^1_0$$

wobei ψ^1_0 und ψ^1_1 den in Kapitel 3.2.2 definierten 1D- Haar Wavelets mit Auflösungsstufe $j = 1$ entsprechen. Diese Haar-Wavelets werden dann kaskadenartig auf ein Signal angewandt, wobei für den 2D-Fall diese dann folgendermaßen definiert ist:



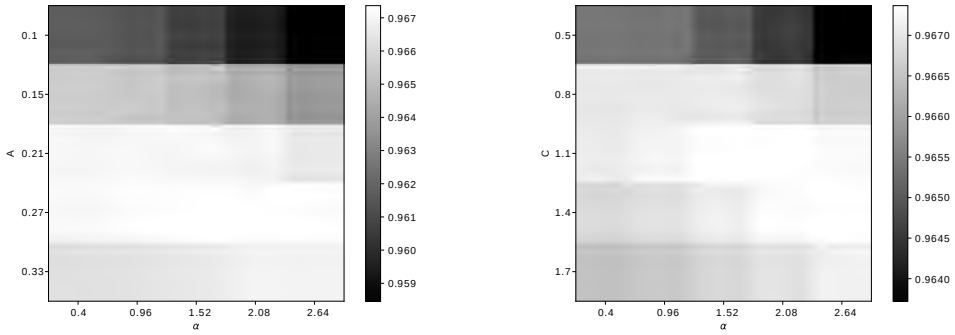
Es lässt sich erkennen, dass jeder Filterungsschritt $n + 1$ auf dem gedownsampleten Ergebnis des Filterungsschrittes n mit dem Tiefpassfilter $\psi^1_{(0,0)}$ aufbaut. Dadurch reduzieren sich nach jedem Filterungsschritt die Datenmengen, die dann noch weiter analysiert werden müssen. Bei einem Signal mit Dimension s muss nach jedem Filterschritt nach dem downsampling mit $\downarrow_2$ nur noch $\frac{1}{2^s}$ der Datenmenge weiter verarbeitet werden, was zu erheblichen Vorteilen bezüglich der Berechnungszeit führt. Vor allem im 3D-CT-Bereich, wo ohnehin große Datenmengen verarbeitet werden, ist ein solcher Ansatz interessant. Da nun feature-maps und Gewichtungs-maps mit steigender Skalierung auch eine kleinere Auflösung besitzen, wird im Weiteren die multiweight HaarVectorPSI Variante angewandt, damit die Gewichtungs-maps und feature-maps pro Auflösungsstufe die selbe Auflösung haben. Die weitere Definition des FWT HaarVectorPSI lässt sich analog zu HaarVetorPSI betrachten, sodass FWT HaarVectorPSI definiert ist als:

$$HaarVectorPSI_{f_1,f_2} = l_\alpha^{-1}\left(\frac{\sum_{j=1}^{J}\sum_{x=0}^{p_j} l_\alpha\left(CM^j_{f1,f2}(x)\right)\cdot W^j_{f1,f2}(x)}{\sum_{j=1}^{J}\sum_{x=0}^{p_j} W^j_{f_1,f_2}(x)}\right)^2,$$

wobei J der Anzahl der Auflösungsstufen und p_j der Anzahl der Datenpunkte für eine Auflösungsstufe j entsprechen.

3.2.4 Parameterbestimmung

Für HaarVectorPSI, welche die in Kapitel 3.2.1 beschriebene längensensitiven Kosinusähnlichkeit verwendet, gibt es drei Parameter, die eingestellt werden müssen: α, A und C. Diese Parameter beeinflussen die Berechnung der Funktion L, welche die Länge der Vektoren in die Ähnlichkeitsberechnung einbeziehen. Der Parameter C bestimmt die Krümmung der Funktion L, der Parameter A bewirkt eine Skalierung bezüglich der x-Achse. Im Folgenden sollen nun die Parameter für 2D-Bilder und 3D-CT-Daten bestimmt werden. Diese werden für alle HaarVectorPSI Varianten verwendet. Für den 2D-Fall werden dazu die Parameter bezüglich der ”LIVE Image Database” [She+] optimiert, sodass eine maximal große Korrelation zwischen dem Human Visual Score und dem des Algorithmus besteht. Es wurde dazu eine Rastersuche durchgeführt und die beste Parameter- Kombination gewählt. Dazu wurden die Wertebereiche A= $[0.1, 0.33]$, C = $[0.5, 1.7]$ α= $[0.4, 2.64]$ gewählt. Das Ergebnis der Rastersuche für HaarVectorPSI ist in Abbildung 3.8 dargestellt.



(a) (b)

Abbildung 3.8: Ergebnis der Rastersuche in den Wertebereichen A= $[0.1, 0.33]$, C = $[0.5, 1.7]$ α= $[0.4, 2.64]$ bezüglich der ”LIVE Image Database” [She+]. In (a) ist die Korrelation bei festem C und variablen A und α dargestellt und in (b) C und α bei festgelegtem A

Für den 2D-Fall werden daher folgende Parameter gewählt: $A = 0,27$, $C = 1,1$, $\alpha = 2,08$. Ausgehend von diesen Parametern werden die Parameter für den 3D-Fall abgeleitet. Die Parameter C und α werden direkt aus dem 2D-Fall übernommen. Da 3D-CT-Daten einen anderen Wertebereich als Bilder haben, muss eine Anpassung der Parameter vorgenommen

werden. Der Parameter A wurde für den 2D-Fall für einen Wertebereich von $[0, 255]$ optimiert und wird nun so angepasst, dass er für einen Wertebereich von $[0, 57000]$ sinnvoll gewählt ist. Dazu wird der Parameter A mit einem Faktor $\frac{255}{57000}$ angepasst, sodass der Parameter A für den 3D-Fall gewählt wird als

$$A' = \frac{255}{57000} A = 0,00120 \ .$$

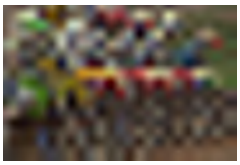
3.3 HaarHistSim

3.3.1 Definition

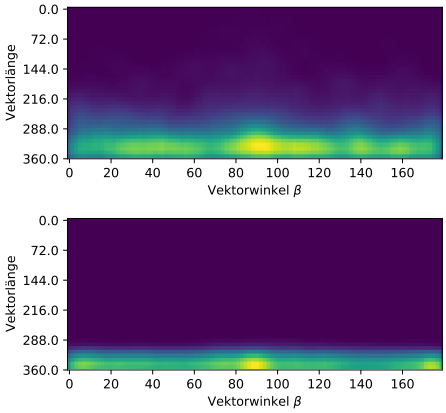
Die bisher vorgestellten Varianten von HaarVectorPSI basieren auf den positionsweisen Vergleich von Vektoren der Gradientenfelder von Referenzbild und Vergleichsbild. Betrachtet man die Ähnlichkeit zweier Signale aus der Sichtweise, inwiefern diese sich strukturell ähnlich sind, wobei die Positionen der Kanten keine Rolle spielen sollen, so sind die HaarVectorPSI Varianten dazu nicht geeignet. Der im Folgenden vorgestellte HaarHistSim Algorithmus ist für die positionsunabhängige Bestimmung der Ähnlichkeit zwischen Gradientenstrukturen geeignet. Die grundlegende Idee dahinter ist der Vergleich von Vektorhistogrammen zur Bestimmung der Ähnlichkeit.

Für jedes Signal f wird zunächst ein Vektorfeld $GM_f^j(x)$ gebildet. Dieses Vektorfeld ist eine gute Annäherung an den Gradienten $\triangledown f(x_\alpha^j)$ des Signals f mit Dimension s für die Auflösungsstufe j. Die Bildung des Vektorfelds erfolgt analog zu Kapitel 3.2.2 oder als FWT-Variante analog zu Kapitel 3.2.3. In einem ersten Schritt werden hier sehr kleine Vektoren $\|v\| \leq 0,1$ bereits aussortiert. Um die Ähnlichkeit zwischen zwei Signalen f_1 und f_2 zu bestimmen, werden die Vektorfelder der beiden Signale verglichen. Ein Vektor v mit Dimension s ist definiert durch seine Länge l und den Winkeln $W_v = \{\alpha_1, \ldots, \alpha_{s-1}\}$, die seine Richtung beschreiben. Somit resultieren für ein Signal f mit Dimension s für eine Auflösungsstufe j, Histogramme $H_f^{j,s}$ mit s Dimensionen. Die erste Dimension wird für die Länge eines Vektors und die restlichen für die richtungsbestimmenden Winkeln verwendet. In Abbildung 3.9 ist dazu ein Beispiel für die Auflösungsstufe $j = 2$ dargestellt, in dem ein Referenzbild mit einem Bild mit starkem fastfading- Fehler verglichen wird. Damit zwei Histogramme miteinander sinnvoll verglichen werden können, müssen diese die selbe Intervall- Aufteilung und Breite haben. Die Winkel W_v befinden sich alle im Intervall

(a) Referenzbild f_1



(b) Fastfadingfehler f_2



(c)

Abbildung 3.9: (c) Zeigt die resultierenden Histogramme für das (a) Referenzbild und ein (b) Vergleichsbild mit sehr starkem fastfading Fehler. Die Bilder stammen aus der 'LIVE' Datenbank [She+]

$[0, 180]$, sodass auch das Histogramm in diesem Bereich aufgespannt wird. Die Längen der Gradientenvektoren sind in einem Bereich $[0, l_{max}]$. Als Anpassung an den Wertebereich der Signale wird für 2D-Daten $l_{max} = 255$ und für 3D-CT-Daten $l_{max} = 56000$ gesetzt. Bei der Erstellung eines Histogramms sind im wesentlichen der Intervall- Startpunkt t_0 und die Intervall- Breite h von Bedeutung. Je nach Wahl von t_0 kann sich das Erscheinungsbild des Histogramms bei gleicher Datengrundlage stark verändern. Um das Problem der Wahl von t_0 zu Lösen werden im Folgenden Histogramme nach dem im [Sco09] vorgeschlagenen Prinzip des "Averaged Shifted Histogram (ASH)" gebildet, welches ein Mittel zur Schätzung der Kerndichte darstellt. Dabei wird der Ursprung t_0 mehrfach um den Faktor $\delta = \frac{h}{m}$ verschoben, wobei m der Anzahl der Verschiebungen entspricht. Das ASH entspricht einer Dichteschätzung, die sich einer dreieckigen Kerndichteschätzung annähert, in der die Gewichte für ein Histogramm mit $2m - 1$ Unterteilungen, wie in [Sco09] beschrieben, so gewählt, dass es im 1D-Fall einem Sampling der Dreiecksfunktion

$$K(t) = (1 - |t|)I_{-1,1}(t)$$

mit

$$t = \frac{1-m}{m}\ \frac{2-m}{m}\ \dots\ \frac{-1}{m}\ 0\ \frac{-1}{m}\ \dots\ \frac{2-m}{m}\ \frac{1-m}{m}$$

entspricht. Für den s-dimensionalen Fall ist die Funktion zur Berechnung der Gewichtung folgendermaßen definiert

$$K^s(x, y, \dots) = (1 - |x|)I_{-1,1}(x) \cdot (1 - |y|)I_{-1,1}(y) \cdot \dots \ .$$

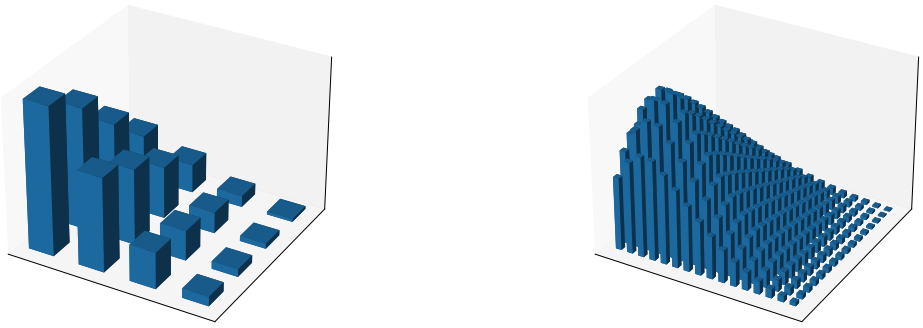
Für den Fall, dass $m = 2$ würde sich für den 2D-Fall somit folgender Kern F^2 ergeben:

$$\begin{bmatrix} 0.25 & 0.5 & 0.25 \\ 0.5 & 1 & 0.5 \\ 0.25 & 0.5 & 0.25 \end{bmatrix}$$

Dieser wird zur Gewichtung zwischen den verschobenen Histogrammen verwendet. Das ASH kann also berechnet werden, indem aus den Gradientenfeldern $GM_f^j(x)$ eines Signals f die m-fach verschobenen und s-dimensionalen Histogramme $H_{f,m}^{j,s}(x)$ generiert werden. Diese werden dann mit Hilfe eines Filters gewichtet. Für ein Signal $f \in \mathrm{l}(\mathbb{Z}^s)$ mit N Datenpunkten ist die Berechnung des gewichteten ASH für eine Auflösungsstufe j dann folgendermaßen definiert:

$$\hat{H}_f^{j,s}(x) = \frac{F^s * H_{f,m}^{j,s}(x)}{N \prod_{h \in S} h} \ .$$

Die Menge $S = \{h_1, h_2, \dots h_s\}$ entspricht dabei der Menge der Intervallbreiten der jeweiligen Dimension des Histogramms. Das m-fach verschobene Histogramm $H_{f,m}^{j,s}(x)$ wird durch Filterung mit dem Kern F^s gewichtet, wobei der Operator $*$ dem s-dimensionalen Faltungsoperator entspricht. In Abbildung 3.10 ist beispielhaft ein ASH dargestellt. Das ASH löst das Problem der Wahl des Startpunktes t_0, jedoch nicht die Wahl der Intervall-Breite h. Je kleiner h gewählt wird, desto genauer ist die Unterteilung der Vektoren bezüglich Winkel und Länge. Somit ist die Wahl von h entscheidend für die Sensibilität der Histogrammähnlichkeit und wird auf einen festen Wert gesetzt, da die Sensibilität nicht von der Datenanzahl oder deren Verteilung abhängig sein soll. Die experimentelle Bestimmung von h wird in Kapitel 3.3.2 betrachtet. Zwischen zwei Histogrammen H_1 und H_2 soll nun ein Ähnlichkeitsmaß definiert werden. Dieses basiert auf der Überschneidung, die sich ergibt, wenn die Histogramme überlagert werden und ist folgendermaßen definiert:



(a) (b)

Abbildung 3.10: In (a) ist ein zweidimensionales Histogramm H_1 dargestellt. In (b) ist das Histogramm $H_2 = ASH(H_1)$ abgebildet mit $m = 4$ Verschiebungen

Definition 3.3.1 (überschneidungs-basierte Histogrammähnlichkeit) *Für zwei Histogramme H_1 und H_2, mit Dimension s, ist die normierte überschneidungs-basierte Histogrammähnlichkeit definiert als*

$$S_{HistSim}(H_1, H_2) = 1 - B(H_1, H_2) \ ,$$

wobei die Funktion $B(\cdot)$ definiert ist als

$$B(H_a, H_b) = \frac{\sum_l \sum_{x \in \mathbb{Z}_{s-1}} \left| H_a(l,x) - H_b(l,x) \right| \cdot L(l)}{\sum_l \sum_{x \in \mathbb{Z}_{s-1}} (H_a(l,x) + H_b(l,x)) \cdot L(l)}$$

und die Funktion $L(\cdot)$ als

$$L(l) = l \cdot h_L \ .$$

Mit Hilfe der Funktion $B(\cdot)$ wird die Distanz zwischen zwei Histogrammen mit Dimension s ermittelt, welche der Überschneidungsstärke entspricht. Der Wertebereich von $B(\cdot)$ ist definiert durch das Intervall $[0, 1]$, wobei 0 für eine totale Übereinstimmung und 1 für keine Überschneidung steht. Die Funktion $L(\cdot)$ dient zur Gewichtung der Histogramme bezüglich der Längendimension. Der Wert $z = l \cdot h_L$ entspricht dabei der Obergrenze des l-ten Intervalls der Längendimension der Histogramme. Somit wird der Häufungsunterschied kleinerer Vektoren weniger stark gewichtet als der größerer Vektoren. Zur Berechnung der Ähnlichkeit, wird in der Funktion $S_{\text{HistSim}}(\cdot)$ die errechnete Distanz dann zu einem Ähnlichkeitswert durch Subtraktion von 1 umgewandelt. Die hier verwendete Variante hat den Vorteil relativ einfach und schnell berechenbar zu sein. Die Histogrammähnlichkeit

wird für jede Auflösungsstufe j separat berechnet und zu einer Gesamtähnlichkeit gemittelt, sodass HaarHistSim für J Auflösungsstufen definiert ist als

$$\text{HaarHistSim}_{f_1,f_2} = \frac{\sum_j S_{\text{HistSim}}(\hat{H}_{f_1}^{j,s}(x)\ , \hat{H}_{f_2}^{j,s}(x))}{J}\ .$$

Der dadurch ermittelte Ähnlichkeitswert entspricht der Ähnlichkeit zweier Signale bezüglich der Verteilung ihrer Kantenstärken und Richtungen der Gradienten und ist zudem eine positionsunabhängige Betrachtung von Ähnlichkeit.

3.3.2 Parameterbestimmung

HaarHistSim hat drei Parameter, die bestimmt werden sollen. Die Parameter *bin_Angles* und *bin_Lengths* bestimmen die Feinheit der Klassenunterteilung bezüglich Winkel und Länge der Gradientenvektoren. Sie steuern daher, wie sensibel HaarHistSim bezüglich Unterschieden in der Verteilung der Gradienten reagiert. Dabei ist zu beachten, dass beide Parameter Einfluss auf die Berechnungszeiten haben. Hohe Werte führen zu einer erhöhten Berechnungsdauer, da der Aufwand zur Berechnung der Ähnlichkeit zwischen den Histogrammen zunimmt. Der Parameter $shifts$ bestimmt, wie viele Verschiebungen bei der Berechnung des ASH angewandt werden und beeinflusst somit die Glättung der Histogramme. Auch dieser Parameter hat Einfluss auf die Berechnungszeit, da $shifts$ den Berechnungsaufwand des ASH und auch der Ähnlichkeit zwischen den Histogrammen beeinflusst. Um einen Kompromiss zwischen Berechnungszeit und Glättung zu finden wird $shifts = 5$ gesetzt. Die Paramter *bin_Angles* und *bin_Lengths* werden bezüglich der ”LIVE Image Database” [She+] optimiert. Zur Ermittelung der Werte für *bin_Angles* und *bin_Lengths* wird eine Rastersuche durchgeführt, bei der die Wertebereiche wie folgt gewählt wurden: $bin\_Angles \in [2, 5, 10, 20, 30, 40]$, $bin\_Lengths \in [20, 50, 80, 90, 100]$. In Abbildung 3.11 ist das Ergebnis der Rastersuche dargestellt.

Basierend darauf werden die Parameter entsprechend der höchsten Korrelation gesetzt: $bin\_Angles = 5$ und $bin\_Lengths = 50$. Diese Werte werden sowohl für Bilder als auch 3D-CT-Daten verwendet, da kein Parameter direkt auf die Länge der Gradientenvektoren angewandt wird.

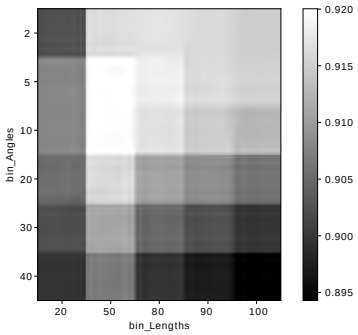


Abbildung 3.11: Ergebnis der Rastersuche in den Wertebereichen $bin\_Angles \in [2, 5, 10, 20, 30, 40]$, $bin\_Lengths \in [20, 50, 80, 90, 100]$ bezüglich der ”LIVE Image Database” [She+]

3.4 Implementation

3.4.1 Rahmenbedingungen

Im Folgenden soll nun die Implementierung der in Kapitel 3 beschriebenen Algorithmen dargelegt werden. Diese erfolgt in der Programmiersprache Python in der Version 3.9 und unter Verwendung folgender Bibliotheken:

`numpy` Erlaubt eine effiziente Implementierung mathematischer Operationen auf große Datenmengen. Im Besonderen wird es hier zur Generierung der Wavelets durch Tensorprodukte, zur Erstellung von Histogrammen und mathematische Berechnungen auf großen Arrays eingesetzt.

`scipy` wird zur Implementation der Faltung verwendet. `scipy` bietet zudem die Möglichkeit die Faltung mit Hilfe der FFT zu berechnen. Es wird zudem zur statistischen Auswertung der Daten verwendet.

Zur graphischen Darstellung der Resultate wurde auf die Bibliotheken `vedo, matplotlib` zurückgegriffen. In den folgenden Kapiteln soll nun die 3D- Implementation der Algorithmen auszugsweise beschrieben werden.

3.4.2 Implementation Haar-Wavelet-Transformation

Wie in Kapitel 2.2 beschrieben, können mehrdimensionale Haar Wavelets durch Tensorprodukte der eindimensionalen Wavelets konstruiert werden. Dies ist in der Methode `get3DHaarWavelets` in 3.1 implementiert.

Python Code 3.1: Berechnung der 3D Haar Wavelets

```
def get3DHaarWavelets(scale):
  haar1 = np.ones(2 ** scale)
  haar2 = np.ones(2 ** scale)
  half = haar1.shape[0] // 2
  haar2[:half] = -haar1[:half]

  normfactor = 2 ** (-(3 * scale) / 2)

  wavelet1 = np.tensordot(haar1, haar2, 0) # oy
  wavelet2 = np.tensordot(haar2, haar1, 0) # yo
  wavelet3 = np.tensordot(haar2, haar2, 0) # yy
  wavelet4 = np.tensordot(haar1, haar1, 0) # oo

  wavelet3D1 = np.tensordot(haar1, wavelet1, 0) * normfactor # ooy
  wavelet3D2 = np.tensordot(haar1, wavelet2, 0) * normfactor # oyo
  wavelet3D3 = np.tensordot(haar1, wavelet3, 0) * normfactor # oyy
  wavelet3D4 = np.tensordot(haar1, wavelet4, 0) * normfactor # ooo
  wavelet3D5 = np.tensordot(haar2, wavelet1, 0) * normfactor # yoy
  wavelet3D6 = np.tensordot(haar2, wavelet2, 0) * normfactor # yyo
  wavelet3D7 = np.tensordot(haar2, wavelet3, 0) * normfactor # yyy
  wavelet3D8 = np.tensordot(haar2, wavelet4, 0) * normfactor # yoo

  return np.array([wavelet3D1, wavelet3D2, wavelet3D3, wavelet3D4, ↩
      wavelet3D5, wavelet3D6, wavelet3D7, wavelet3D8])
```

Die Funktion `get3DHaarWavelets` berechnet die 3D-Haar-Wavelets für eine bestimmte Skala `scale`. Zu Beginn werden die 1D- Haar Wavelets `haar1` und `haar2` erstellt, die bereits die Skalierung `scale` haben. Anschließend werden durch Tensorprodukte die 3D-Haar-Wavelets `wavelet3D1` bis `wavelet3D8` generiert. Bei der Haar-Wavelet-Transformation werden diese generierten Filter mit einem 3D- Signal gefaltet. Die Faltung ist in 3.2 in der Methode `convolve3D` implementiert. Dabei wird auf die `scipy` interne `signal` Bibliothek zurückgegriffen, welche den mehrdimensionalen Faltungsoperator implementiert.

Python Code 3.2: Anwendung der Haar- Filter

```
1 def convolve3D(data, wavelet):
2     result = signal.convolve(
3       data,
4       wavelet,
5       mode='same',
6       method='auto'
7   )
8   return result
```

Der Parameter `mode = 'same'` in Zeile 5 setzt dabei fest, welcher Bereich des Ergebnisses nach der Filterung verwendet wird. In diesem Fall so, dass das Ausgangssignal die selbe Größe hat wie das Eingangssignal. Wie in Kapitel 2.1 erwähnt, lässt sich die Faltung zweier Signale auch als Multiplikation ihrer Fourier-Transformierten beschreiben. Durch den Parameter `method='auto'` in Zeile 6 wird automatisch ausgewählt, ob für den aktuellen Fall die FFT Variante oder die direkte Faltung optimal ist.

3.4.3 Implementation HaarPSI3D

Folgend soll nun auszugsweise die Implementation der `haar_psi_numpy` Methode des HaarPSI3D Algorithmus beschrieben werden. Zunächst werden zu dem Referenzbild `reference_image` und dem Vergleichsbild `distorted_image` die Haar-Wavelet-Koeffizienten für die ersten drei Skalen berechnet (3.3).

Python Code 3.3: Berechnung der Haar Wavelet-Koeffizienten

```
1 number_of_scales = 3
2 coefficients_reference_image = haar_wavelet_decompose(reference_image, ↩
      number_of_scales)
3 coefficients_distorted_image = haar_wavelet_decompose(distorted_image, ↩
      number_of_scales)
```

Anschließend werden die in Kapitel 3.1 dargelegten Maps für die drei Orientierungen im 3D-Fall erzeugt. Der Code in 3.4 beschreibt die Berechnung des HaarPSI3D. Das in Zeile 3 erstellte `weights` array entspricht dabei der Gewichtungs-map $W^{(k)}_{f_1,f_2}(x)$ und das array `local_similarities` der Ähnlichkeitsmap $HS^{(k)}_{f_1,f_2}(x)$, welche die lokalen Ähnlichkeiten

enthält. In Zeile 19 werden die zuvor erstellen Ähnlichkeits- und Gewichtungs-maps zu der Gesamtähnlichkeit verrechnet.

Python Code 3.4: Berechnung des HaarPSI3D

```
1  # Computes the weights and similarities for each orientation
2  for orientation in range(3):
3    weights[:, :, :, orientation] = numpy.maximum(
4      numpy.abs(coefficients_reference_image[:, :, 2 + orientation * ↩
           number_of_scales]),
5      numpy.abs(coefficients_distorted_image[:, :, 2 + orientation * ↩
           number_of_scales])
6    )
7    coefficients_reference_image_y_magnitude = numpy.abs(
8    coefficients_reference_image[:, :, (orientation * number_of_scales,↩
         1 + orientation * number_of_scales)])
9    coefficients_distorted_image_y_magnitude = numpy.abs(
10   coefficients_distorted_image[:, :, (orientation * number_of_scales,↩
         1 + orientation * number_of_scales)])
11
12   local_similarities[:, :, :, orientation] = numpy.sum(
13     (2 * coefficients_reference_image_y_magnitude * ↩
           coefficients_distorted_image_y_magnitude + C) / (
14       coefficients_reference_image_y_magnitude ** 2 + ↩
             coefficients_distorted_image_y_magnitude ** 2 + C),
15     axis=3
16   ) / 2
17
18 # Calculate the final score
19 similarity = logit(numpy.sum(sigmoid(local_similarities[:], alpha) * ↩
       weights[:]) / numpy.sum(weights[:]),
20            alpha) ** 2
```

3.4.4 Implementation HaarVectorPSI

HaarVectorPSI basiert auf der Ähnlichkeit der Gradienten der zu vergleichenden Signalen. Die Berechnung dieser Gradienten ist in 3.5 implementiert. Es ist zu beachten, dass bereits hier die in Kapitel 3.2.1 beschriebene zusätzliche Dimension zu den Gradientenvektoren berechnet wird. Die Methode `createVectorMaps` berechnet für ein Signal `data` die Gradienten für `scales` Skalen.

Python Code 3.5: Berechnung Haar Gradient

```
1  def createVectorMaps_extraDim(image, scales, scaleStart=0):
2    mapcount = scales - scaleStart
3    vectormaps = np.zeros((mapcount, image.shape[0], image.shape[1], image.shape[2], 4), dtype=np.float64)
4    dimensions = 3
5    for s in range(mapcount):
6      wavelets = waveletsPool[s + scaleStart]
7      e1 = convolve3D(image, wavelets[0])
8      e2 = convolve3D(image, wavelets[1])
9      e3 = convolve3D(image, wavelets[7])
10
11     lengths = np.sqrt(
12     np.einsum("ijk,ijk->ijk", e1, e1) + np.einsum("ijk,ijk->ijk", e2, e2) + np.einsum("ijk,ijk->ijk", e3, e3))
13
14     e1 = np.divide(e1, lengths, out=np.zeros_like(e1), where=lengths != 0)
15     e2 = np.divide(e2, lengths, out=np.zeros_like(e2), where=lengths != 0)
16     e3 = np.divide(e3, lengths, out=np.zeros_like(e3), where=lengths != 0)
17
18     lengths = lengths * 2 ** ((-s * dimensions) / 2)
19     extraDim = np.divide(1.0, (A * lengths) ** C, where=lengths != 0, out=np.ones_like(lengths))
20
21     vectormaps[s] = np.stack((e1, e2, e3, extraDim), axis=3)
22 return vectormaps
```

Für jede Skala `s` werden zunächst die drei nötigen Waveletkoeffizienten `e1` bis `e3` berechnet. Zur Vorbereitung für die Anwendung der in Kapitel 3.2.1 beschriebenen Ähnlichkeitsfunktion werden die Koeffizienten anschließend normalisiert. Die Länge der Vektoren wird, wie in Kapitel 2.7 beschrieben, durch einen Faktor $2^{-\frac{j \cdot s}{2}}$ in Zeile 18 angepasst. Anschließend wird die zusätzliche Dimension `extraDim` berechnet. Ein Vektor v des arrays `vectormaps` entspricht also einem Vektor $\hat{v}$ der längensensitiven Kosinusähnlichkeit aus Kapitel 3.2.1, welche durch den Code in 3.6 implementiert wird. Die Methode `cosine_similarity_extraDim_elementwise3D` berechnet aus zwei zuvor erstellten `vectormaps` die längensensitive Kosinusähnlichkeit zwischen den Vektoren.

Python Code 3.6: Berechnung längensensitiver Kosinus Ähnlichkeit

```
def cosine_similarity_extraDim_elementwise3D(vmap_ref, vmap_dist):
  cossim_map = np.abs(
    np.einsum("sijkm,sijkm->sijk", vmap_ref, vmap_dist, dtype=numpy.↩
        float64) /(
    np.sqrt(np.einsum("sijkm,sijkm->sijk", vmap_ref, vmap_ref , dtype=↩
        numpy.float64))
    * np.sqrt(np.einsum("sijkm,sijkm->sijk", vmap_dist, vmap_dist , ↩
        dtype=numpy.float64)))
  )
return cossim_map
```

Dabei wird die Funktion `numpy.einsum` verwendet, die eine effiziente Implementation für mulitdimensionale Arrays ermöglicht. In dem zurückgegebenen Array `cossim_map` befinden sich dann die Werte der lokalen Ähnlichkeit für alle Auflösungsstufen. Die darauf basierende Berechnung von HaarVectorPSI für 3D-CT-Daten ist in 3.7 implementiert.

Python Code 3.7: Berechnung HaarVectorPSI3D

```
def haar_vector_psi(reference_image, distorted_image, subsampling=True):
  if subsampling:
    reference_image = subsample3D(reference_image)
    distorted_image = subsample3D(distorted_image)

  scales = 2

  vmaps_ref = createVectorMaps_extraDim(reference_image, scales)
  vmaps_dist = createVectorMaps_extraDim(distorted_image, scales)

  cmaps = cosine_similarity_extraDim_elementwise3D(vmaps_ref, vmaps_dist↩
      )

  vmaps_ref_big = createVectorMaps(reference_image, scales + 1, ↩
      scaleStart=scales)
  vmaps_dist_big = createVectorMaps(distorted_image, scales + 1, ↩
      scaleStart=scales)
  weights = np.maximum(np.sqrt(np.einsum("sijkl,sijkl->sijk", ↩
      vmaps_ref_big, vmaps_ref_big)),
    np.sqrt(np.einsum("sijkl,sijkl->sijk", vmaps_dist_big, ↩
        vmaps_dist_big)))

```

```
18    similarity = logit(np.sum(sigmoid(cmaps, alpha) * weights) / np.sum(↩
          weights * scales), alpha) ** 2
19
20    return similarity, vmaps_ref, vmaps_dist, cmaps, weights
```

In Zeile 2 wird zunächst ein optionales Subsampling der Signale durchgeführt, welche verglichen werden sollen. Anschließend werden in Zeile 8 die Gradienten für alle Skalen berechnet. Danach wird in Zeile 11 die längensensitive Kosinusähnlichkeit der Gradienten punktweise berechnet, sodass `cmaps` die lokale Ähnlichkeit für alle Auflösungsstufen enthält. Dies entpricht $CM^{j}_{f1,f2}$ aus Kapitel 3.2. Diese werden dann unter Verwendung der Gewichtungs-map, welche in Zeile 16 berechnet wird, zu einer Gesamtähnlichkeit verrechnet. Der Wert in `similarity` entspricht dem finalen Ähnlichkeitswert zwischen den Signalen `reference_image` und `distorted_image`

3.4.5 Implementation HaarHistSim

HaarHistSim basiert auf dem Vergleich von Histogrammen, die aus den Gradienten zweier Signale gebildet werden. In der zugehörigen Implementation in 3.8 werden dazu für jede Skala `s` die nötigen Werte berechnet, anhand derer die Histogramme gebildet werden. Es wird für jeden Vektor der Gradienten `vmap_ref_f` und `vmap_dist_f` die Länge und die Winkel, welche die Richtung des Vektors beschreiben, bestimmt. Die Gradienten werden analog zu 3.5 berechnet, jedoch ohne Zusatzdimension, da diese hier nicht benötigt wird. Nachdem in Zeile 21 das ASH zu den Signalen berechnet worden ist, wird in Zeile 23 die überschneidungsbasierte Histogrammähnlichkeit zwischen den gebildeten Histogrammen `refHist` und `distHist` berechnet. Der Wert des HaarHistSim entspricht dann dem arithmetischen Mittel der Histogrammähnlichkeiten, die für jede Skala bestimmt wurden.

Python Code 3.8: Berechnung HaarHistSim

```
1   [...]
2  for s in range(scales):
3   [...]
4   #calculate vector lengths
5   refL = np.sqrt(np.einsum("ij, ij->i",vmap_ref_f,vmap_ref_f), dtype=np.↩
        float64)
6   distL = np.sqrt(np.einsum("ij, ij->i", vmap_dist_f, vmap_dist_f), ↩
        dtype=np.float64)
7
```

```
8    #calculate angles plane 1
9    refAngles_p1 = np.arctan2(vmap_ref_f[:,2], vmap_ref_f[:,0] )* (180.0/↩
         np.pi)
10   distAngles_p1 = np.arctan2(vmap_dist_f[:,2] ,vmap_dist_f[:,0]) * ↩
         (180.0 / np.pi)
11
12   #calculate angles plane 2
13   refAngles_p2 = np.arctan2(vmap_ref_f[:,1], vmap_ref_f[:,0] )* (180.0/↩
         np.pi)
14   distAngles_p2 = np.arctan2(vmap_dist_f[:,1] ,vmap_dist_f[:,0]) * ↩
         (180.0 / np.pi)
15
16   lengthRangeMax = 56000
17   dimensions = 3
18   refL = refL * 2 ** ((-s * dimensions) / 2)
19   distL = distL * 2 ** ((-s * dimensions) / 2)
20
21   refHist, distHist = createASH(refAngles_p1,distAngles_p1,refL,distL,↩
         refAngles_p2,distAngles_p2,bins_Angles, bins_Lengths, rangeLengths↩
         =(0,lengthRangeMax))
22
23   sim = multidim_histogram_Similarity(refHist, distHist)
24   sim_map[s] = sim
25
26 haarHistSim = np.sum(sim_map) / scales
```

Der Python-Code in 3.9 zeigt die Implementation zur Berechnung des ASH. Der Parameter `shift` entspricht dabei der Anzahl der Verschiebungen und die Parameter `binsAngles`, `binsLengths` der Klassenunterteilungen der Histogramme. Zur Bildung des ASH werden zunächst die neuen Klassenaufteilungen `binAngles_p1, binAngles_p2, binLength` berechnet. Dazu werden die vorherigen Klassenanzahlen um den Faktor `shifts` erhöht. Dann werden in den Zeilen 5 und 6 die multidimensionalen Histogramme `refHist` und `distHist` berechnet. Um das ASH zu erhalten müssen die Histogramme durch einen Filter F^s gewichtet werden.

Python Code 3.9: Berechnung des ASH `createASH`

```
1 binAngles_p1 = np.histogram_bin_edges(refAngles_p1, bins=binsAngles*↩
      shifts, range=rangeAngles)
2 binAngles_p2 = np.histogram_bin_edges(refAngles_p2, bins=binsAngles * ↩
      shifts, range=rangeAngles)
```

```
3  binLength = np.histogram_bin_edges(refLengths, bins=binsLengths*shifts, ↩
       range=rangeLengths)
4
5  refHist = np.histogramdd((refAngles_p1,refAngles_p2, refLengths), bins=(↩
       binAngles_p1,binAngles_p2, binLength))
6  distHist = np.histogramdd((distAngles_p1,distAngles_p2, distLengths), ↩
       bins=(binAngles_p1,binAngles_p2, binLength))
7
8  kernel = calculateKernel(shifts)
9
10 ref_ash = signal.convolve(refHist[0], kernel, mode="same")
11 dist_ash = signal.convolve(distHist[0],kernel,mode="same")
12 [...]
13 ref_ash = ref_ash / (dx * dy * dz * np.sum(ref_ash)) if np.sum(ref_ash) ↩
       > 0 else ref_ash
14 dist_ash = dist_ash / (dx * dy * dz * np.sum(dist_ash)) if np.sum(↩
       dist_ash) > 0 else dist_ash
15 return [ref_ash, refHist[1]], [dist_ash, distHist[1]]
```

In den Zeilen 10 und 11 wird dazu der Filter `kernel` auf die Histogramme angewandt. Anschließend werden die erstellten ASHs `ref_ash` und `dist_ash` noch normalisiert. Zwischen diesen Histogrammen soll die überschneidungsbestimmte Ähnlichkeit berechnet werden. Diese ist in 3.10 implementiert. Die Methode `multidim_histgram_Similarity` entspricht der Funktion $S_{HistSim}$ aus Kapitel 3.3.

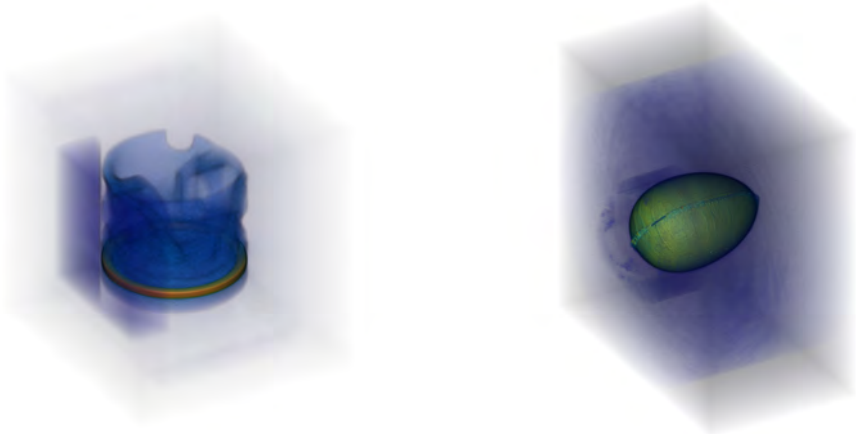
Python Code 3.10: Berechnung der Ähnlichkeit zwischen Histogrammen

```
1 def multidim_histogram_Similarity(hist1, hist2):
2 hist1Values, hist2Values = lengthweight(hist1,hist2)
3 histSum = np.sum(hist1Values + hist2Values)
4 diff = np.sum(np.abs(hist1Values[:] - hist2Values[:])) / histSum if ↩
      histSum > 0 else 0
5 similarity = 1-diff
6 return similarity
```

In Zeile 2 werden zunächst die Histogramme bezüglich der Längendimension gewichtet, was der Funktion L aus Kapitel 3.3 entspricht. In Zeile 4 wird die normierte Differenz aus den zuvor gewichteten Histogrammen berechnet. Der Ähnlichkeitswert des HaarHistSim wird bestimmt, indem der Wert `diff` von 1 subtrahiert wird.

4 Evaluation

In diesem Kapitel werden die in Kapitel 3 vorgestellten Algorithmen zur Berechnung der Ähnlichkeit zwischen 3D-CT-Daten und Bildern evaluiert. Um die Algorithmen zu testen, werden verschiedene Experimente durchgeführt, bei denen die Empfindlichkeit der Algorithmen zu Translation, Rotation, Rauschen und Unschärfe untersucht wird. Außerdem wird die Performance der Algorithmen in Bezug auf die Laufzeit gemessen. Für die Experimente werden, neben synthetisch generierten Signalen, Daten aus folgenden Quellen verwendet: Für den Test von Bildern wurde auf Daten der ”LIVE image database release 2” [She+], sowie der ”Salzburg Texture Image Database (STex)” [Kwi+11], zurückgegriffen. Da als Input der Algorithmen nur Graustufen- Daten verwendet werden können, werden die Farbbilder analog zu der matlab Funktion `rgb2gray` zu einem Graustufenbild umgewandelt. Zum Test von 3D-CT-Voxel-Daten dient ein Scan eines Kolben (aus einem Automotor) und ein Scan eines Überraschungseis, welche in Abbildung 4.1 dargestellt sind.



(a) (b)

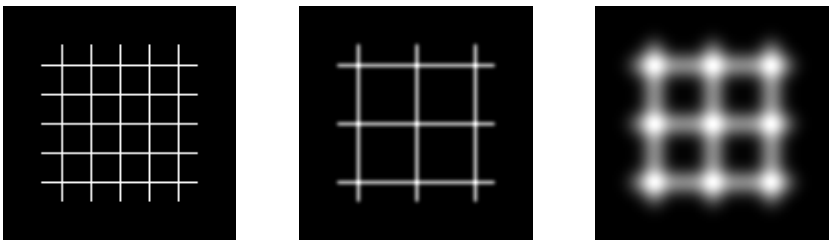
Abbildung 4.1: 3D-CT Scans eines (a) Kolbens eines Motors und (b) eines Überraschungseis

4.1 Empfindlichkeit gegenüber Transformationen und Verzerrungen

In den folgenden Kapiteln soll die Sensibilität der in Kapitel 3 vorgestellten Algorithmen bezüglich verschiedenen Transformationen für 2D und 3D Daten untersucht werden. Die Auswahl an Transformationen wird hierbei auf Translation, Rotation und Skalierung festgelegt.

4.1.1 Translation

Im Folgenden soll nun das Verhalten der Ähnlichkeitsbestimmung der in Kapitel 3 vorgestellten Algorithmen bezüglich Translation für 2D und 3D Daten untersucht werden. Dabei wird auch darauf eingegangen, inwiefern die jeweiligen Parameter der Algorithmen Einfluss auf die Sensibilität bezüglich Translation haben. Da die haar-wavelet-basierten Algorithmen ein Signal in verschiedenen Auflösungen betrachten, soll zunächst untersucht werden, ob es einen Unterschied der Sensibilität in Bezug auf verschobene feine und grobe Informationen gibt. In Abbildung 4.2 und 4.3 sind dazu drei verschiedene Referenzbilder für feine, mittlere und grobe Informationen dargestellt, die verschoben werden sollen. Jedes



(a) (b) (c)

Abbildung 4.2: Die drei verschiedenen Detail-Stufen, die für die 2D-Translationsanalyse benutzt werden. In Bild (a) sind feine Strukturen dargestellt, in (b) weniger fein und in (c) sehr grobe Strukturen.

dieser Bilder wird nun in verschiedene Richtungen und Positionen verschoben und mit dem originalen Referenzbild verglichen. Das Ergebnis dazu sind für 2D-Daten in Tabelle 4.1 und für 3D-Daten in Tabelle 4.2 bildlich dargestellt.

Die in den Tabellen dargestellten Abbildungen zeigen die Empfindlichkeit der einzelnen Algorithmen bezüglich Translation, wobei x und y- Achse der Tanslation des Referenzbildes

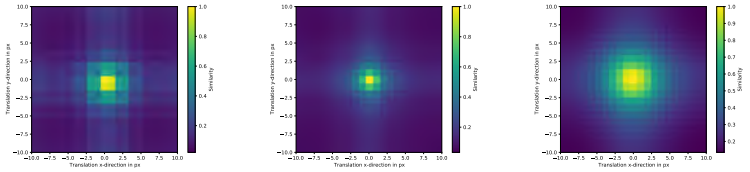
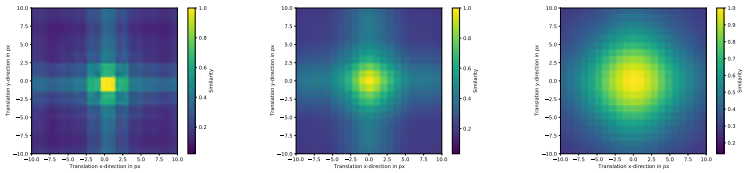
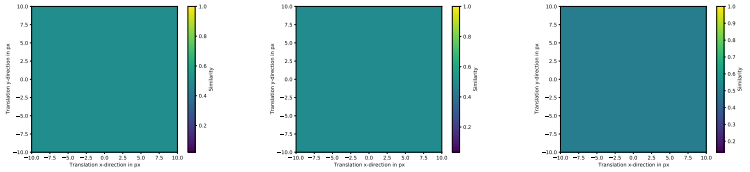
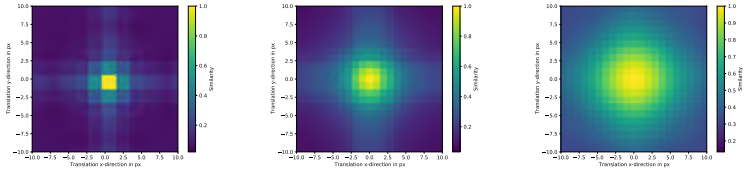
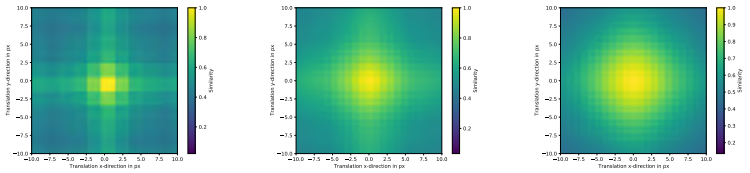
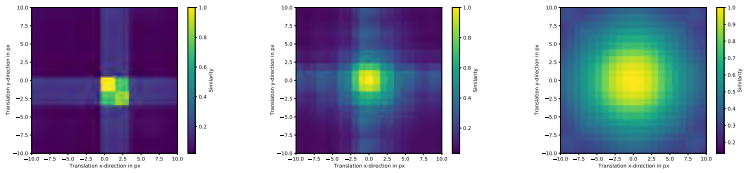
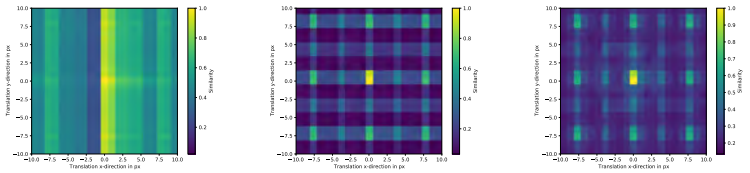
	Translation-Detail Test		
	fein	mittel	grob
HaarPSI			
HaarVectorPSI			
HaarVectorPSI BigWeight			
HaarVectorPSI Multiweight			
HaarVectorPSI No weight			
HaarVectorPSI FWT			
Haar Hist Sim			

Tabelle 4.1: Empfindlichkeit der verschiedenen Algorithmen bezüglich Translation von 2D-Daten für drei Arten von Strukturen (fein, mittel, grob).

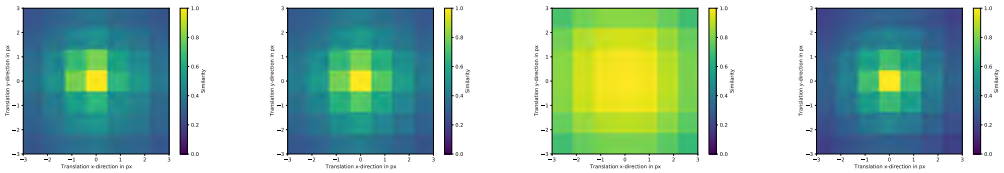
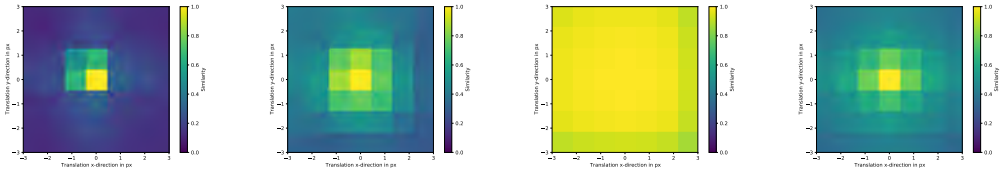
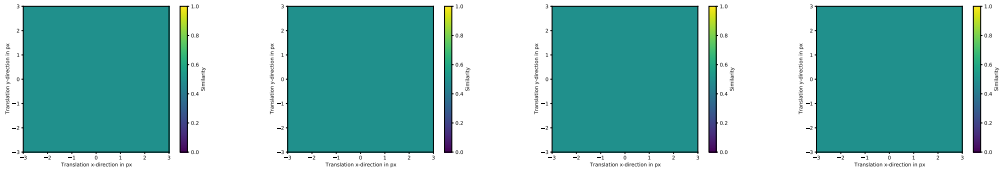
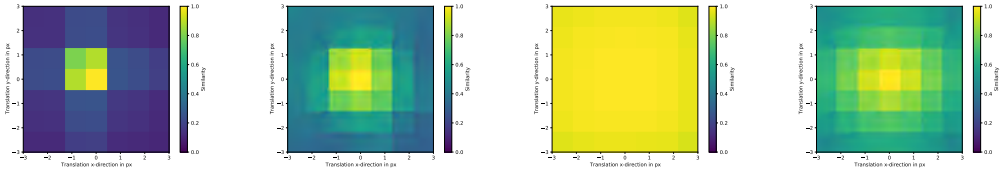
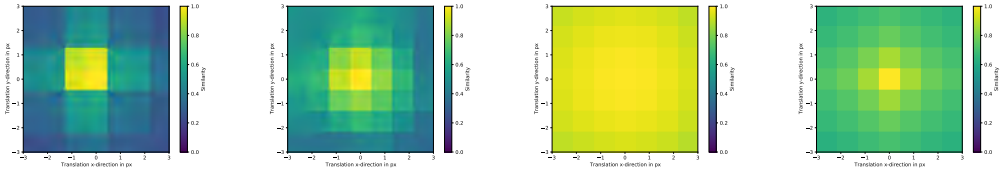
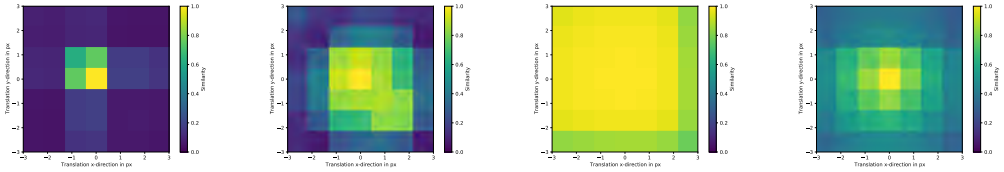
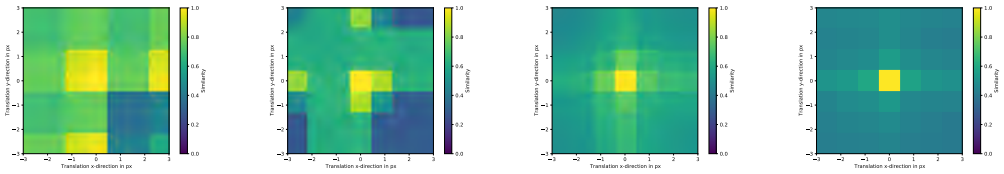
	Translation-Detail Test 3D			
	fein	mittel	grob	Kolben
HaarPSI				
HaarVectorPSI				
HaarVectorPSI BigWeight				
HaarVectorPSI Multiweight				
HaarVectorPSI No weight				
HaarVectorPSI FWT				
Haar Hist Sim				

Tabelle 4.2: Empfindlichkeit der verschiedenen Algorithmen bezüglich Translation von 3D-Daten für drei Arten von Strukturen (fein, mittel, grob) und einen realen 3D-CT-Scan (Kolben) aus Abbildung 4.1. Dargestellt ist die XY-Ebene der Translation.

(a) (b) (c)

Abbildung 4.3: Die drei verschiedenen Detail-Stufen, die für die 3D-Translationsanalyse benutzt werden. In Bild (a) sind feine Strukturen dargestellt, in (b) weniger fein und in (c) sehr grobe Strukturen.

in Pixeln entspricht. Der resultierende Ähnlichkeitswert ist farblich dargestellt, wobei die Farbe gelb einer Ähnlichkeit von 1 entspricht. Allgemein betrachtet, lässt sich feststellen, dass alle Algorithmen mit Ausnahme von HaarHistSim eine eher starke Empfindlichkeit bezüglich Translation haben. Betrachtet man zunächst HaarPSI, stellt man für alle drei Strukturen fest, dass eine vertikale bzw. horizontale Translation einen geringeren Ähnlichkeitsabfall erzeugt als diagonale Verschiebungen. HaarPSI ist somit für horizontale und vertikale Verschiebungen weniger translationsempfindlich. Auch für HaarVectorPSI und seinen Varianten lässt sich dieses Verhalten feststellen. Dieses Verhalten lässt sich dadurch erklären, dass sich diagonale Verschiebungen immer aus mehreren horizontalen und vertikalen Verschiebungen beschreiben lassen, wodurch sich beim Vergleich ein höherer Ähnlichkeitsabfall ergibt. Die Strukturart beeinflusst, wie stark die Ähnlichkeit bei Verschiebung abfällt. Wie Abbildung 4.2 zeigt, ist der Abfall bei groben Strukturen geringer als bei feinen Strukturen. Dies lässt sich dadurch erklären, dass bei groben Stukturen sich der Gradient von einem Pixel zum nächsten weniger stark verändert, sodass eine höhere Ähnlichkeit besteht. Im speziellen Fall von HaarVectorPSI FWT stellt sich zudem eine Asymmetrie im Ähnlichkeitsabfall ein, die sich durch das mehrfache Downsampling des Signals bei der Wavelet-Zerlegung erklären lässt.
Obwohl man bei HaarHistSim, aufgrund der positionsunabhängigen Definition der Ähnlichkeit, keine Translationsempfindlichkeit erwarten würde, ruft eine Verschiebung dennoch eine Änderung der Ähnlichkeit hervor. Betrachtet man hierzu das Ergebnis für den 2D-Fall aus Tabelle 4.2 lässt sich ein Kachelmuster in der Ähnlichkeitsschwankung erkennen. HaarHistSim ermittelt die Gradienten auf Basis der Haar-FWT, sodass die Ursache hierfür wieder das Downsampling nach jedem Filterungsschritt ist. In Abbildung 4.4 ist die Translationsempfindlichkeit einer Variante des HaarHistSim dargestellt, in der kein Downsampling ausgeführt wird. Unter diesen Umständen lässt sich eine totale Unempfindlichkeit bezüglich

Translation erkennen. Im Folgenden soll nun die Translationsempfindlichkeit für reale 3D-

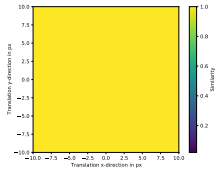


Abbildung 4.4: Abgebildet ist hier das Ergebnis der Translationsempfindlichkeit für Haar-HistSim, jedoch ohne FWT. Das Referenzbild aus 4.2 (b) wurde wie beim Test aus Tabelle 4.1 verschoben.

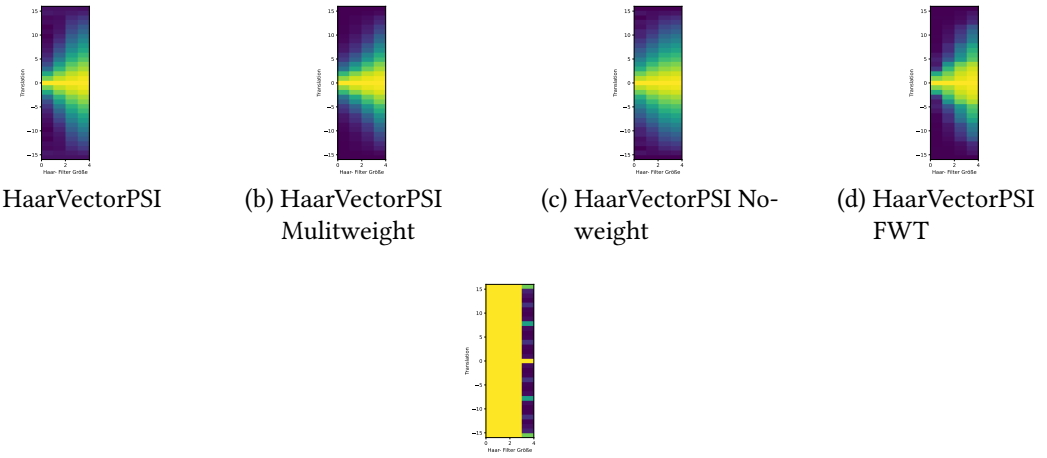
CT-Daten betrachtet werden. Dazu wird der 3D-CT-Scan eines Kolbens aus Abbildung 4.1 mehrfach in verschiedene Richtungen verschoben und die Ähnlichkeit ermittelt. Damit bei der Verschiebung keine Informationen verloren gehen, da über den Rand hinaus verschoben wird, werden die CT-Daten so bearbeitet, dass die Werte im Randbereich des Volumens auf 0 gesetzt werden. In Tabelle 4.2 sind die Ergebnisse dazu unter ”Kolben” aufgelistet. Es lässt sich feststellen, dass das Verhalten der Algorithmen ähnlich zu den synthetisch generierten Signalen ist. Die durch Anwendung der FWT auftretenden Artefakte sind auch beim realen 3D-CT-Scan vorhanden, aber weniger stark ausgeprägt. Bei der Histogrammähnlichkeit ist ebenfalls eine Translationsabhängigkeit erkennbar, sodass die Variante ohne FWT gewählt werden muss, falls eine translationsunabhängige Ähnlichkeitsbestimmung erwünscht ist. Die Haar-Wavelet-Transformation zerlegt ein Signal in Frequenzbänder, die durch die Wahl der Waveletskalierungen bestimmt werden. Die Wahl der Skalierung ist daher für die Empfindlichkeit bezüglich Verschiebungen im Signal relevant. Im Folgenden soll nun dargestellt werden, inwiefern die Skalierung der Wavelet-Filter Auswirkung auf die Translationsempfindlichkeit hat. Die Translation wird dabei auf die Testbilder, wie in Abbildung 4.5 dargestellt, ausgeführt. Die Anzahl der Haar-Filter bleibt dabei konstant, nur

(a) (b)

Abbildung 4.5: Die Referenzbilder für (a) 2D und (b) 3D-Daten, die zur Ermittelung der Translationsempfindlichkeit bezüglich Skalierung der Haar-Filter.

deren Skalierung wird schrittweise erhöht. Beispielsweise entspricht eine Erhöhung um 0

der Standartfilterbank mit Skalierungen $[1, 2, 3]$ und eine Erhöhung um 1 der mit $[2, 3, 4]$. In den Abbildungen 4.6 und 4.7 sind die Ergebnisse dazu für 2D und 3D-Daten dargestellt.



(a) HaarVectorPSI (b) HaarVectorPSI Mulitweight (c) HaarVectorPSI No-weight (d) HaarVectorPSI FWT

(e) HaarHistSim

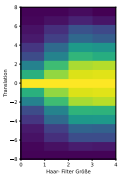
Abbildung 4.6: Translationsempfindlichkeit für 2D-Daten in Abhängigkeit von der Wahl der Skalierung der Haar-Filter. Je größer die Filter, desto geringer ist die Empfindlichkeit bezüglich Verschiebung. Die Translation wurde bezüglich einer Achse durchgeführt.

Es lässt sich sowohl für 2D und 3D-Daten feststellen, dass je kleiner die Filtergrößen sind, desto empfindlicher die Algorithmen auf Verschiebungen reagieren. Bei HaarHistSim lässt sich dieser Zusammenhang nicht feststellen, da die Vektoren der Gradienten positionsunabhängig verglichen werden. Nur die durch das Downsampling entstehenden Artefakte sind sichtbar.

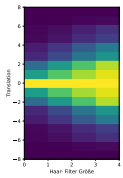
4.1.2 Rotation

Im Folgenden soll auf das Verhalten bezüglich Rotation eingegangen werden. Eine Rotation, bei der das gesamte Signal oder Teile davon bezüglich einer oder mehrerer Achsen gedreht wird, beinhaltet auch immer eine Translation, sodass diesbezüglich Effekte wie in Kapitel 4.1.1 eine maßgebliche Rolle spielen. Je weiter man vom Drehpunkt entfernt ist, desto mehr nehmen diese Effekte zu.

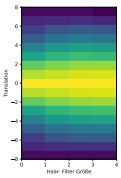
Die in Abbildung 4.8 dargestellten Referenzbilder werden nun mit gedrehten Versionen ihrer selbst verglichen. Dabei wird jeweils das gesamte Signal schrittweise bezüglich einer



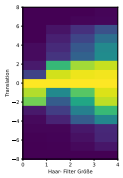
(a) HaarVectorPSI



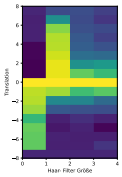
(b) HaarVectorPSI Mulitweight



(c) HaarVectorPSI No-weight

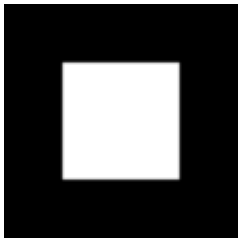


(d) HaarVectorPSI FWT

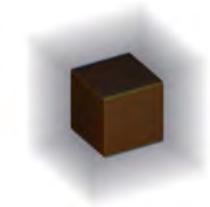


(e) HaarHistSim

Abbildung 4.7: Translationsempfindlichkeit für 3D-Daten bezüglich der Wahl der Skalierung der Haar-Filter. Je größer die Filter, desto geringer ist die Empfindlichkeit bezüglich Verschiebung. Die Translation wurde bezüglich einer Achse durchgeführt.



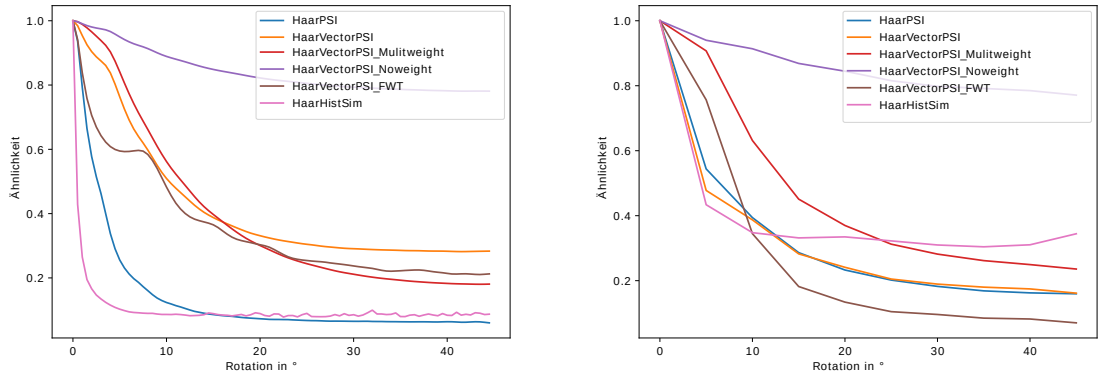
(a)



(b)

Abbildung 4.8: Die Referenzbilder für (a) 2D und (b) 3D-Daten, die zur Ermittelung der Rotationsempfindlichkeit.

Drehachse bis 45 gedreht. In Abbildung 4.9 ist die daraus resultierende Ähnlichkeit dargestellt. Bei allen Algorithmen im 2D und 3D-Fall, lässt sich beobachten, dass die Ähnlichkeit bei steigender Rotation abnimmt und sich ab einer gewissen Rotation stark abflacht. Eine weitere Rotation bewirkt hier, begründet durch die in 4.1.1 beschriebene Translationsempfindlichkeit, keinen nennenswerten Unterschied in der Ähnlichkeit. Daher lassen sich Ähnlichkeitsunterschiede bezüglich Rotation bei eher kleinen Rotationen bestimmen.



(a) (b)

Abbildung 4.9: Darstellung der Ähnlichkeit in Abhängigkeit der Rotation für (a) 2D und (b) 3D-Daten.

4.1.3 Skalierung

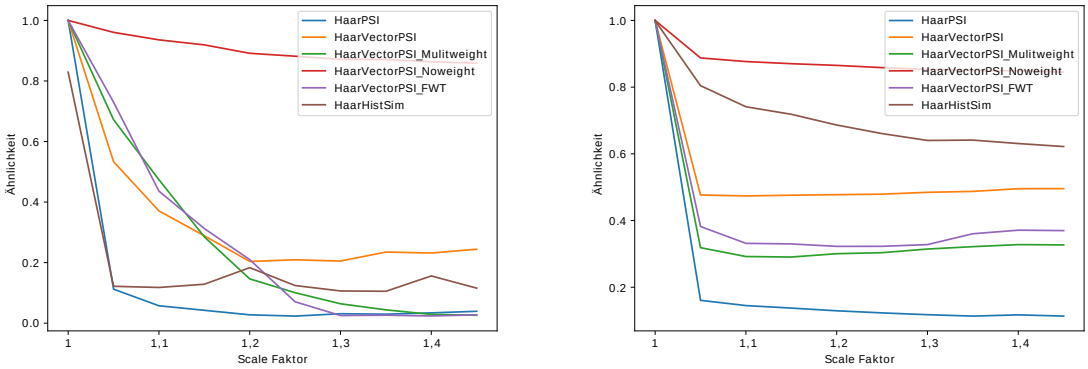
Im Folgenden soll auf die Empfindlichkeit der Algorithmen aus Kapitel 3 bezüglich Skalierung eingegangen werden. Die Referenzbilder, die dazu verwendet werden, sind in Abbildung 4.10 dargestellt. Es handelt sich dabei um je zwei generierte und zwei reale Signale. Damit bei der Skalierung keine Informationen verloren gehen, hat jedes der Referenzbilder einen Rahmen, sodass die Signale bei der Skalierung nicht über den Rand hinaus skaliert werden. Jedes dieser Referenzbilder wird nun bis zu einem Skalierungsfaktor

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

Abbildung 4.10: Referenzbilder für den Test der Empfindlichkeit bezüglich Skalierung. (a) Quadrat, (b) Würfel, (c)Bild ”Caps” aus der LIVE Image Database, (d) Kolben

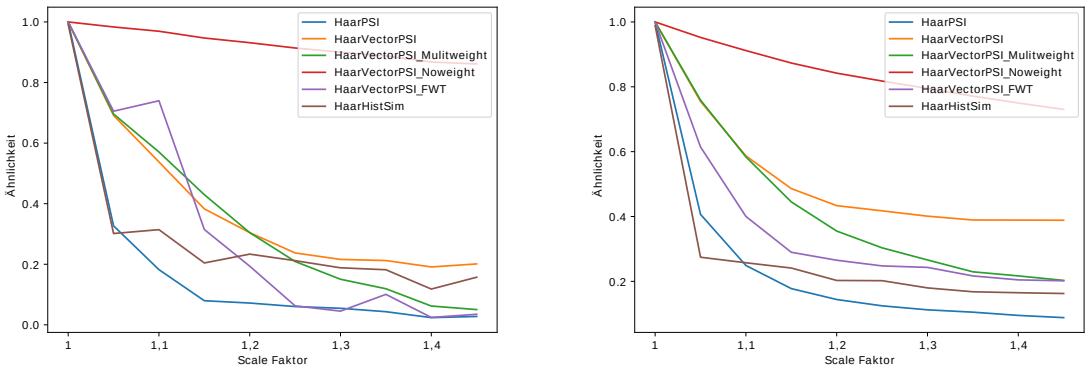
$f = 1,5$ in 10 Abstufungen skaliert und mit dem Referenzbild verglichen. Die Resultate für die 2D-Signale sind in 4.11 und für 3D-Signale in 4.12 dargestellt.

Da eine Skalierung immer auch eine Translation von Kanten bewirkt, spielen die Effekte aus Kapitel 4.1.1 hier eine Rolle. Dies ist auch aus den Resultaten erkennbar, da schon bei



(a) (b)

Abbildung 4.11: Empfindlichkeit bezüglich Skalierung für 2D-Daten. (a) Quadrat, (b) ”Caps”



(a) (b)

Abbildung 4.12: Empfindlichkeit bezüglich Skalierung für 3D-Daten. (a) Würfel, (b) Kolben

kleiner Skalierung die ermittelte Ähnlichkeit aller Algorithmen stark abfällt. Das qualitative Verhalten ist dabei für alle Algorithmen gleich. Der Algorithmus ”HaarVectorPSI noweight” zeigt hier jedoch einen weitaus flacheren Abfall als die restlichen Algorithmen und ist somit vergleichsweise unempfindlich bezüglich Skalierung. Da HaarVectorPSI noweight keine Gewichtungs-map zur Ermittlung der Ähnlichkeit verwendet, spielen Unterschiede in den skalierten und verschobenen groben Strukturen eine weniger starke Rolle als bei den HaarVectorPSI Varianten mit Gewichtungs-map. Dies ist in Abbildung 4.13 bezüglich dem Bild ”Caps” veranschaulicht. Bei der lokalen Ähnlichkeit (a) steht die Farbe ”Weiß” für eine hohe Ähnlichkeit und ”Schwarz” für niedrige Ähnlichkeit. Bei der Gewichtungs-map (b) steht eine hellere Farbe für eine stärkere Gewichtung. In (a) ist zu erkennen, dass die lokale Ähnlichkeit zwischen dem Referenzbild und der skalierten Variante sehr hoch ist.

(a) (b)

Abbildung 4.13: Darstellung der (a) lokalen Ähnlichkeit und (b) Gewichtung des HaarVectorPSI Algorithmus

Die Gewichtungs-map (b) gewichtet jedoch überwiegend genau die Stellen stärker, die in der lokalen Ähnlichkeit (a) als unähnlich betrachtet werden. Da HaarVectorPSI noweight die Ähnlichkeit nur auf Basis von (a) bestimmt, ist auch die berechnete Ähnlichkeit höher.

4.1.4 Empfindlichkeit gegenüber Verzerrung

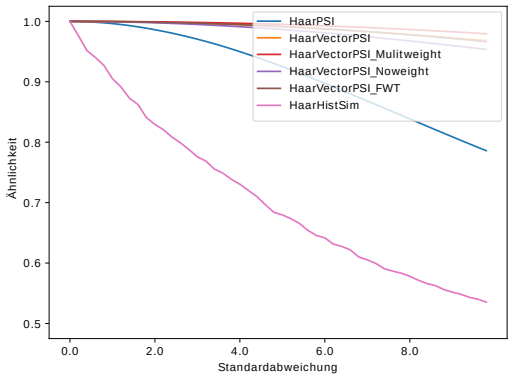
Im Folgenden soll das Verhalten der Algorithmen auf Bildstörungen betrachtet werden. Im Speziellen soll hier auf Bildrauschen und Unschärfe eingegangen werden. Zum Testen der Sensibilität auf Bildrauschen werden die Typen ”Gaußsches Rauschen”, ”Salt & Pepper” und ”Speckle” betrachtet. Diese Typen gehören zu den häufigen Rauscharten die in CT-Scans vorkommen können [Rav+17]. Bei ”Speckle” Rauschen handelt es sich, im Gegensatz zum additiven gaußschem Rauschen, um multiplikatives Rauschen, sodass hellere Regionen stärker verrauscht werden als dunklere. Bei ”Salt & Pepper” Rauschen werden zufällig Pixel, bzw. Voxel eines Signals auf den Minimal- oder Maximalwert gesetzt. Solche Fehler können beispielweise bei Fehlern in einem Bildsensor auftreten. Zu jeder dieser Typen werden aus einem Referenzbild, wie in 4.14 dargestellt, verrauschte Bilder in 50 Intensitätsstufen erzeugt und mit dem Referenzbild verglichen. Abbildung 4.15 zeigt die Resultate für Gaußsches Rauschen.

Bei allen Algorithmen, die getestet wurden, sinkt die Ähnlichkeit zum Referenzbild, wenn die Standardabweichung σ des gaußschen Rauschens steigt. Dabei nimmt die Stärke der Abnahme zu, wenn die Standartabweichung größer wird. Bei HaarHistSim nimmt die Ähnlichkeit schon bei kleiner Standardabweichung stark ab und flacht dann leicht ab. Das hängt

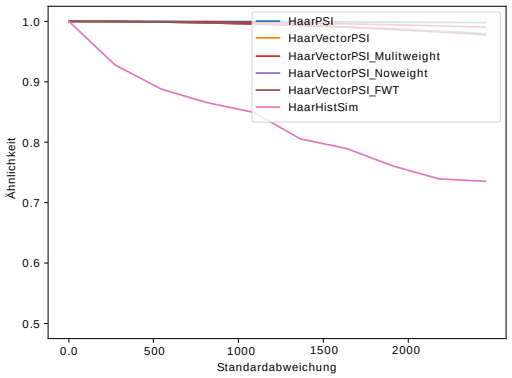
(a)

(b)

Abbildung 4.14: Generierte Referenzbilder für (a) 2D und (b) 3D Daten.



(a)



(b)

Abbildung 4.15: Darstellung der Ähnlichkeit bezüglich gaußschem Rauschen für (a) 2D und (b) 3D-Daten.

damit zusammen, dass die durch das Rauschen entstehenden Kanten bei der Erstellung des Histogramms anders gewichtet werden als beispielsweise bei HaarVectorPSI. Während bei HaarVectorPSI eine sehr schwache Kante nur sehr schwach Gewichtet wird, hängt die Gewichtung sehr kleiner Kanten bei HaarHistSim von der Intervallbreite der Längendimension ab. Für ein Intervall $[0, l]$ bekommen alle Vektoren die hier eingegliedert sind die selbe Gewichtung. Daher reagiert HaarHistSim stärker auf Rauschen.
Betrachtet man das Ergebnis zu ”salt and pepper” Rauschen in Abbildung 4.16, stellt man fest, dass sich alle Algorithmen ähnlich Verhalten. Wieder nimmt bei steigender Intensität des Rauschens die Ähnlichkeit ab. Da bei dieser Rauschart durch setzen der Pixel auf maximalen und minimalen Wert eher starke Kanten gesetzt werden reagieren alle Algorithmen nun auch schon bei kleinerer Intensität stark auf das Rauschen, wobei bei hohen Standardabweichungen der Ähnlichkeitsabfall abflacht. Bei der Rauschart ”speckle” verhält es sich ähnlich. In Abbildung 4.17 sind die Ergebnisse dazu dargestellt. Da es sich um multiplikatives Rauschen handelt sind bei gleicher Standardabweichung Daten an hellen

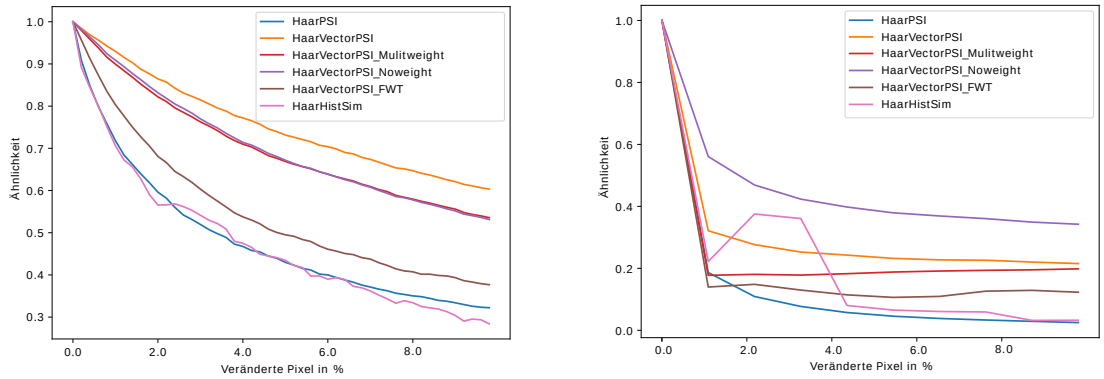


Abbildung 4.16: Darstellung der Ähnlichkeit bezüglich ”Salt & Pepper” Rauschen für (a) 2D und (b) 3D-Daten.

Stellen stärker verrauscht als bei additiven gaußschem Rauschen, wie in den Ergebnissen in Abbildung 4.15. Daher reagieren alle Algorithmen bei gleichem σ sensibler auf ”speckle” Rauschen als auf gaußschem Rauschen.
Für die hier betrachteten Rauscharten, soll nun das Verhalten der Algorithmen auf Anwendung auf ein reales 3D-CT-Signal betrachtet werden. Dazu wird der in Abbildung 4.1 dargestellte Kolben mit den verschiedenen Rauscharten verrauscht. Die Ergebnisse dazu sind in Abbildung 4.18 dargestellt. Die Ergebnisse stimmen im Wesentlichen mit den Rauschtests mit generierten Daten überein. Die Algorithmen mit Außnahme von HaarHistSim reagieren im Vergleich zu den anderen beiden Rauscharten eher unempfindlich auf gaußsches Rauschen. Bei ”salt and pepper” und ”speckle” Rauschen ist, wie bei den vorherigen Tests, ein ähnlicher Empfindlichkeitsverlauf feststellbar.
Im Folgenden soll nun untersucht werden, inwiefern die Wahl der Parameter der Algorithmen Einfluss auf die Empfindlichkeit gegenüber gaußschem Rauschen hat. Dabei werden für HaarVectorPSI und seine Varianten die Parameter im Wertebereichen $\alpha = [0.1, 10]$ und $C = [0.1, 3]$ gewählt. Im Speziellen wird für den 2D-Fall $A = [0.01, 0.1]$ und im 3D Fall $A = [0.0004, 0.004]$ gewählt. Die Ergebnisse für 2D und 3D sind in den Tabellen 4.3 und 4.4 dargestellt. Von den drei Parametern haben nur A und C Einfluss auf die Rauschempfindlichkeit. Der Parameter α hat keinen signifikanten Einfluss. Je größer A und je kleiner C gewählt wird, desto empfindlicher reagieren die HaarVectorPSI- Varianten auf gaußsches Rauschen. Beide Parameter haben Einfluss auf die Berechnung der längensensitiven Kosinusähnlichkeit aus Kapitel 3.2.1, welche in allen HaarVectorPSI- Varianten verwendet wird. Gaußsches Rauschen fügt einem Signal typischerweise viele schwache Kanten hinzu. Im

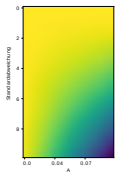
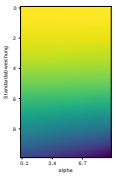
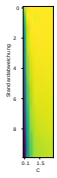
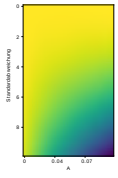
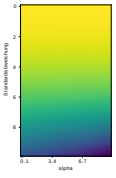
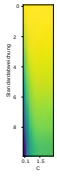
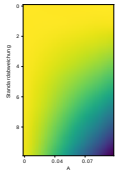
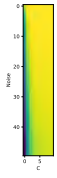
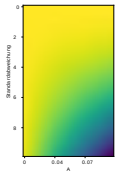
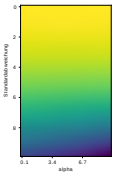
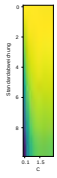
Gaußsches Rauschen - Parametertest 2D			
	A	α	C
HaarVectorPSI			
HaarVectorPSI Multiweight			
HaarVectorPSI No weight			
HaarVectorPSI FWT			

Tabelle 4.3: Empfindlichkeit der verschiedenen Algorithmen bezüglich gaußschem Rauschen in 2D-Daten für die Parameter A, α und C.

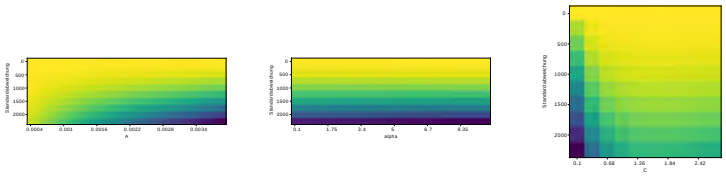
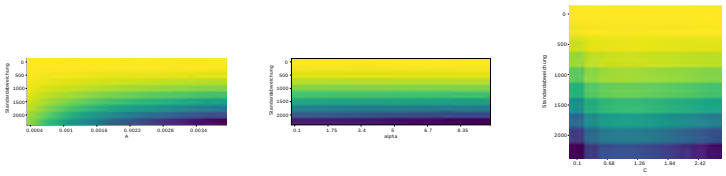
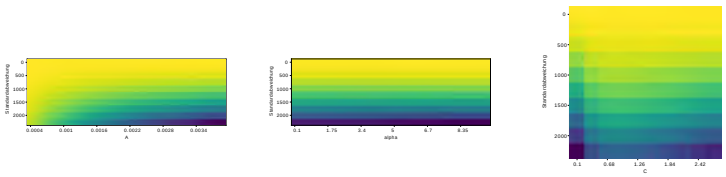
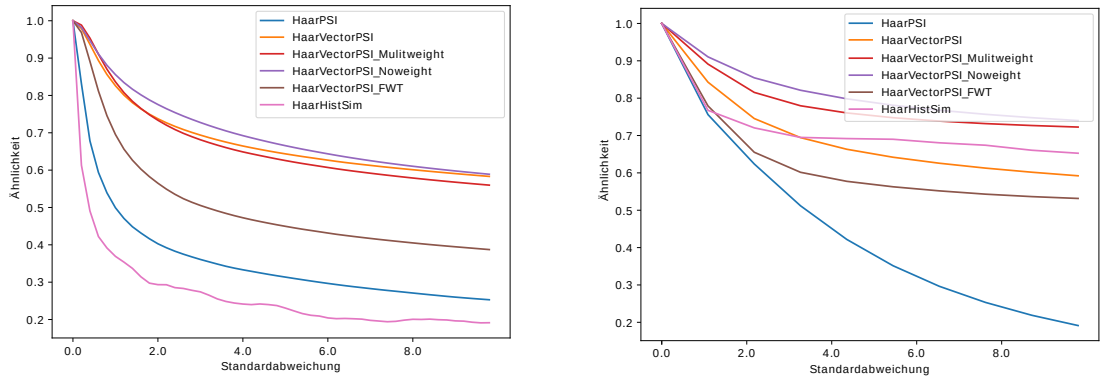
	Gaußsches Rauschen - Parametertest 3D		
	A	α	C
HaarVectorPSI			
HaarVectorPSI Multiweight			
HaarVectorPSI No weight			
HaarVectorPSI FWT			

Tabelle 4.4: Empfindlichkeit der verschiedenen Algorithmen bezüglich gaußschem Rauschen in 3D Daten für die Parameter A, α und C.

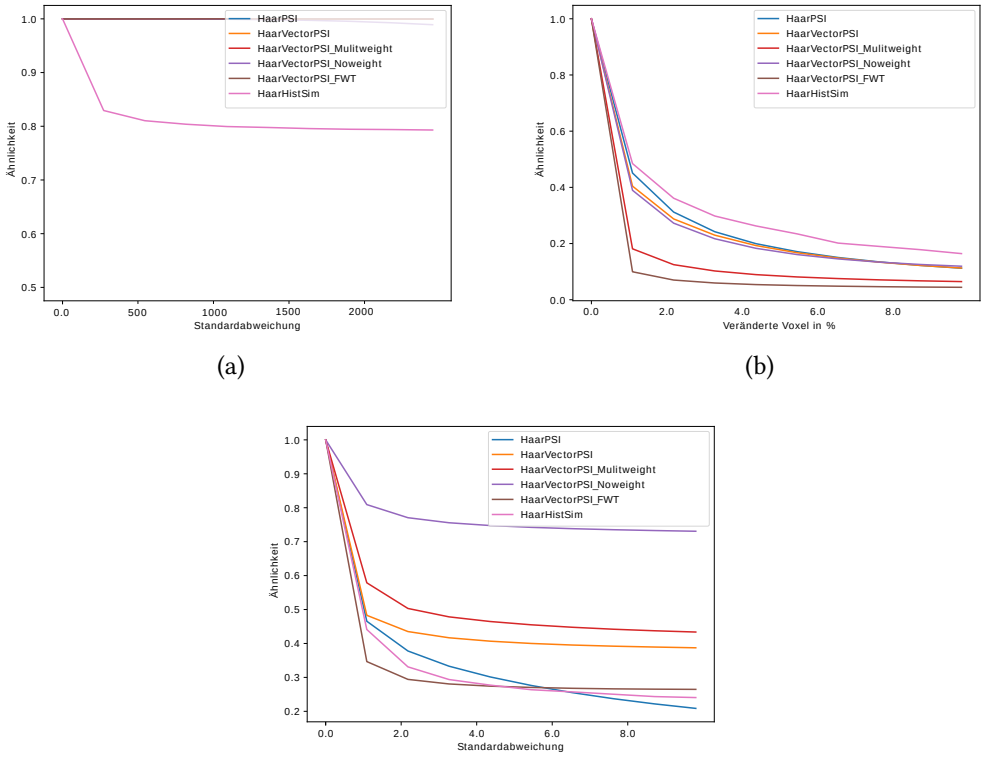


(a) (b)

Abbildung 4.17: Darstellung der Ähnlichkeit bezüglich ”speckle” Rauschen für (a) 2D und (b) 3D-Daten.

Gradienten des Signals macht sich das durch zusätzliche Vektoren kleiner Länge bemerkbar. Ein kleines A und ein großes C machen die Berechnung unempfindlicher gegenüber den zusätzlichen Vektoren mit kleiner Länge, die durch das Rauschen entstehen. Bei HaarHistSim wird die Rauschempfindlichkeit durch die Wahl der Unterteilung der Histogramme beeinflusst. In Abbildung 4.19 ist das Ergebnis dargestellt. Es lässt sich feststellen, dass eine feinere Klasseneinteilung der Histogramme, sowohl bezüglich Länge und Winkel zu einer erhöhten Sensibilität bezüglich gaußschem Rauschen führt. Die Wahl der Klassenbreite bestimmt die Auflösung und die Glättung der Daten. Eine eher große Klassenbreite kann die Details der Verteilung verdecken, während eine eher kleine Klassenbreite die Sensibilität verstärkt. Für die Algorithmen HaarPSI und HaarPSI3D hat der Parameter α, wie auch bei HaarVektorPSI, keine Auswirkung auf die Rauschempfindlichkeit. Die Ergebnisse zur Untersuchung des Parameters C sind in Abbildung 4.20 dargestellt. Es lässt sich feststellen, dass die Ähnlichkeitsbestimmung weniger anfällig für Rauschen ist, wenn der Parameter C größer gewählt wird. In dem von HaarPSI und HaarPSI3D verwendeten Änhlichkeitsmaß kann durch den Parameter C das Verhalten der Ähnlichkeitsberechnung bezüglich eher kleiner Werte gesteuert werden. Je größer der Parameter C gewählt wird, desto ähnlicher werden die verglichenen Werte a und b betrachtet. Dies ist vor allem der Fall, wenn $a, b < C$. Die durch das gaußsche Rauschen entstandenen schwachen Kanten bekommen so bei erhöhtem C weniger Einfluss in die Ähnlichkeitsbestimmung.

Ein weiterer häufiger Effekt, der die Ähnlichkeit zwischen Daten beeinflussen kann ist Unschärfe. Daher soll nun die Sensibilität bezüglich gaußscher Unschärfe betrachtet werden.

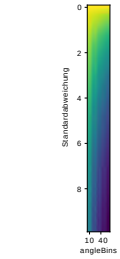


(a) (b)

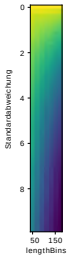
(c)

Abbildung 4.18: Darstellung der Ähnlichkeit des 3D-CT-Scans eines Kolbens aus Abbildung 4.1 bezüglich verrauschten Varianten mit den Rauscharten (a) gaußsches Rauschen (b) salt and pepper und (c) speckle.

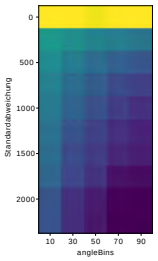
Dazu werden aus den Referenzbildern, aus Abbildung 4.5, Versionen mit zunehmender gaußscher Unschärfe erzeugt. Für den 2D-Fall wurde σ in 50 Schritten erhöht und für den 3D-Fall in 10 Schritten. Die Ergebnisse dazu sind in Abbildung 4.21 dargestellt. Sowohl für 2D und 3D-Daten verhalten sich die getesteten Algorithmen ähnlich. Je größer σ und damit die Unschärfe ist, desto niedriger die Ähnlichkeitswerte. Dabei flacht der Abfall der Ähnlichkeit für größer werdende Unschärfe immer mehr ab. Nun soll das Verhalten der Algorithmen auf Anwendung auf ein reales 3D-CT Signal betrachtet werden. Aus dem in Abbildung 4.1 dargestellten Kolben werden dazu 10 Varianten mit steigender Unschärfe generiert und verglichen. Die Ergebnisse dazu sind in Abbildung 4.22 dargestellt. Auch hier ergibt sich ein vergleichbares Verhalten zu den Ergebnissen aus Abbildung 4.21. Mit steigender Unschärfe des Vergleichsbildes nimmt auch die Ähnlichkeit immer weiter ab, wobei der Abfall der Ähnlichkeit für große σ abflacht. Die Anwendung eines Unschärfefilters auf ein Signal hat vor allem Einfluss auf die enthaltenden Details. Bei der Filterung mit verschieden skalierten Wavelets betrachtet man das Signal in verschiedenen Auflösungsstu-



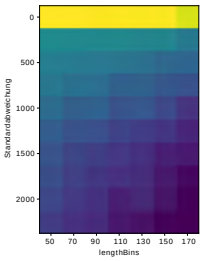
(a) angleBins(2D)



(b) lengthBins(2D)



(c) angleBins(3D)



(d) lengthBins(3D)

Abbildung 4.19: Empfindlichkeit von HaarHistSim bezüglich gaußschem Rauschen in 2D und 3D-Daten für die Parameter *binAngle* und *binLength*.

fen. Es liegt daher nahe, dass die Skalierung der Wavelet-Filter die Ähnlichkeit bezüglich gaußscher Unschärfe beeinflussen kann. Im Folgenden soll nun untersucht werden, ob und inwiefern die Wahl der Skalierung die Ähnlichkeitsbestimmung dahingehend beeinflusst. Der Testaufbau ist dabei analog zu den Skalierungstests für die Translationsempfindlichkeit in Kapitel 4.1.1. In Tabelle 4.5 sind hierzu die Ergebnisse dargestellt. Die Ergebnisse zeigen, dass alle getesteten Algorithmen weniger empfindlich auf gaußsche Unschärfe reagieren, wenn die Skalierungen der Haar-Filter der Filterbank größer gewählt werden. Dies liegt daran, dass die gaußsche Unschärfe die Details im Signal reduziert, die bei einer größeren Skalierung der Filter weniger relevant sind, da das Signal in einer gröberen Auflösungsstufe betrachtet wird.

4.2 Vergleich der Ähnlichkeitsbewertungen

Dieses Kapitel beschreibt die Unterschiede der Algorithmen zur Ähnlichkeitsermittlung, die in Kapitel 3 vorgestellt wurden. Dabei werden insbesondere die Auswirkungen der verschiedenen Gewichtungsmethoden des HaarVectorPSI und der Unterschied zwischen

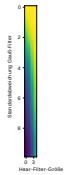
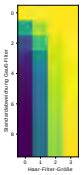
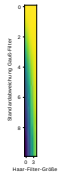
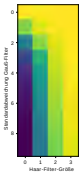
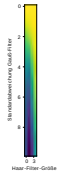
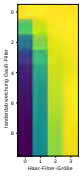
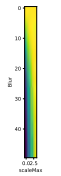
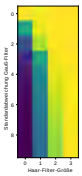
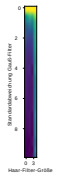
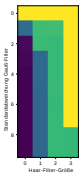
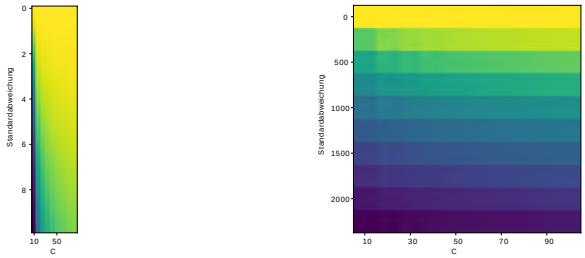
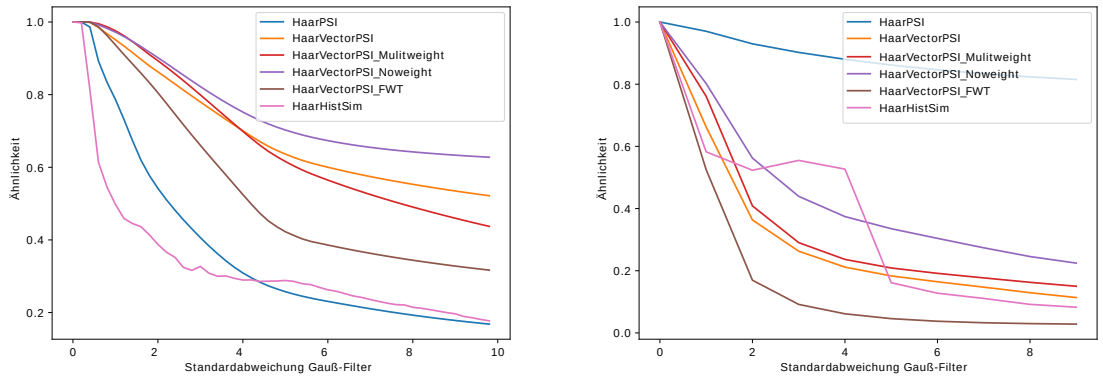
Unschärfe - Haar Skalierungstest		
	2D	3D
HaarVectorPSI		
HaarVectorPSI Multiweight		
HaarVectorPSI No weight		
HaarVectorPSI FWT		
Haar Hist Sim		

Tabelle 4.5: Empfindlichkeit der verschiedenen Algorithmen bezüglich gaußscher Unschärfe in 2D und 3D-Daten in Abhängigkeit der Skalierung der Haar-Filter



(a) 2D (b) 3D

Abbildung 4.20: Empfindlichkeit von HaarPSI und HaarPSI3D bezüglich Rauschen in 2D und 3D-Daten für den Parameter C



(a) (b)

Abbildung 4.21: Darstellung der Ähnlichkeit bezüglich gaußsche Unschärfe für (a) 2D und (b) 3D Daten.

HaarHistSim und HaarVectorPSI analysiert. Zunächst wird der Einfluss einer Gewichtungsmap anhand von HaarVectorPSI untersucht, der grobe Strukturen höher gewichtet, und HaarVectorPSI noweight, der keine Gewichtung anwendet. Zum Vergleich der Ähnlichkeitsbewertung werden 2D-Daten aus Abbildung 4.23 und 3D-Daten aus Abbildung 4.24 herangezogen. In den Rauschbildern A sind leere Bereiche durch gaußsches Rauschen verändert und in den Rauschbildern B ist die abgebildete Struktur durch ”speckle” Rauschen verändert. Für die verrauschten Daten A und B wird nun die Ähnlichkeit zum Referenzbild ermittelt. Die ermittelten Werte für den 2D-Fall sind HaarVectorPSI : (A = 0.9270 , B = 0.8721), HaarVectorPSI noweight : (A = 0.8729 , B = 0.9632) und für den 3D-Fall HaarVectorPSI : (A = 0.9352 , B = 0.7120), HaarVectorPSI noweight : (A = 0.7866 , B = 0.9421). Es zeigt sich ein Unterschied in der Ordnung der Ähnlichkeitswerte, da HaarVectorPSI A ähnlicher zum Referenzsignal betrachtet als B, während HaarVectorPSI noweight die verrauschten Daten

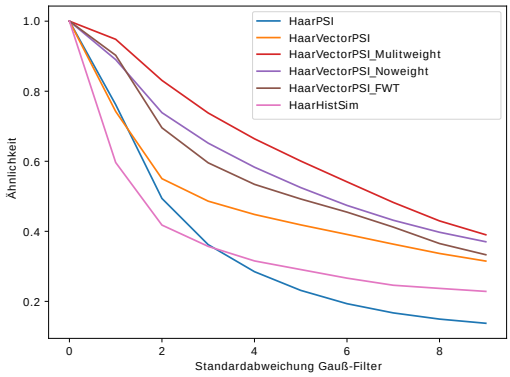
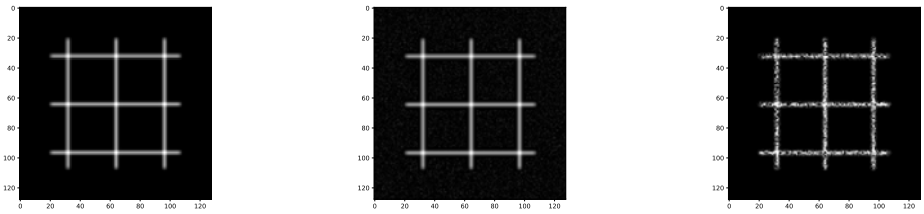


Abbildung 4.22: Darstellung der Ähnlichkeit des 3D-CT-Scans eines Kolbens aus Abbildung 4.1 bezüglich gaußsche Unschärfe



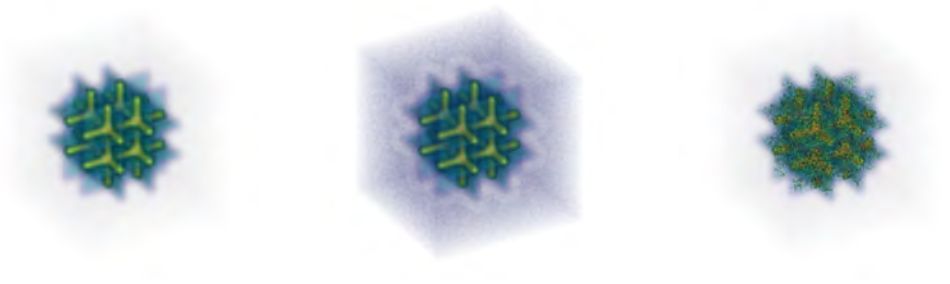
(a) (b) (c)

Abbildung 4.23: Referenzbild (a), Rauschbild A (b), Rauschbild B (c). In A sind die leeren Flächen von einem gaußschem Rauschen mit $\sigma = 10$ überlagert. Bild B ist mit ”speckle” Rauschen mit $\sigma = 0,5$ verändert.

B als ähnlicher zum Referenzbild als die verrauschten Daten A bewertet. Dies lässt sich dadurch erklären, dass HaarVectorPSI Bereiche um Kanten mit hohem Kontrast durch die Gewichungsmap stärker gewichtet. Das Rauschen im leeren Bereich der Bilder fällt dadurch weniger stark ins Gewicht. Auf der anderen Seite gewichtet HaarVectorPSI noweight die Ähnlichkeit an jedem Punkt gleich stark, sodass das Rauschen im leeren Bereich nun im Vergleich mehr Gewicht bekommt als bei HaarVectorPSI.

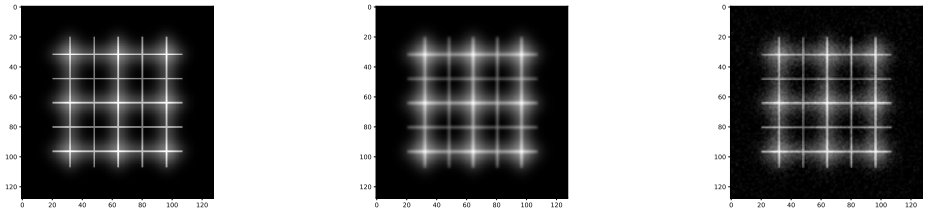
Im Folgenden soll nun untersucht werden, inwiefern die Art der Gewichtung die Ähnlichkeitsbestimmung beeinflusst. Dazu werden HaarVectorPSI und HaarVectorPSI Multiweight miteinander verglichen. Bei HaarVectorPSI Multiweight wird, im Gegensatz zu HaarVectorPSI, jede Detailstufe mit einer separaten Gewichtungs-map gewichtet. Zum Testen möglicher Unterschiede in der Ähnlichkeitsbewertung werden die Daten aus den Abbildungen 4.25 und 4.26 verwendet.

Zu einem Referenzsignal sind veränderte Versionen A und B dargestellt. Signal A ist eine



(a) (b) (c)

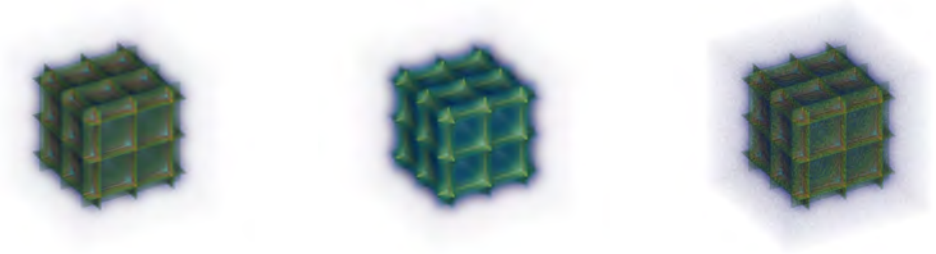
Abbildung 4.24: Referenzvolumen (a), Rauschvolumen A (b), Rauschvolumen B (c). In A sind die leeren Bereiche von einem gaußschem Rauschen mit $\sigma = 2500$ überlagert. Bild B ist mit ”speckle” Rauschen mit $\sigma = 0,5$ verändert.



(a) (b) (c)

Abbildung 4.25: (a) Referenzbild, (b) Bild A mit gaußscher Unschärfe, (c) Rauschbild B. Bild A ist durch gaußsche Unschärfe mit $\sigma = 0,8$ verändert und Bild B durch gaußsches Rauschen mit $\sigma = 10$ verzerrt.

durch gaußsche Unschärfe veränderte Version und Signal B eine durch gaußsches Rauschen verrauschte Version. Die Ergebnisse der Bewertung durch die Algorithmen sind für den 2D-Fall wie folgt: HaarVectorPSI : (A = 0.8827 , B = 0.8597), HaarVectorPSI Mulitweight : (A = 0.9075 , B = 0.9345) und für den 3D-Fall HaarVectorPSI : (A = 0.8904 , B = 0.8599), HaarVectorPSI Mulitweight : (A = 0.9201 , B = 0.9313). Es ergibt sich also ein Unterschied in der Ordnung der Ähnlichkeitswerte. Die gaußsche Unschärfe in den Signalen A hat vor allem Einwirkung auf die Detail-Struktur der Referenzsignale, das gaußsche Rauschen in den Signalen B fügt zwar Rauschen hinzu, die Strukturen der Bilder bleiben aber besser erhalten als bei gaußscher Unschärfe. HaarVectorPSI sieht in Signal A eine höhere Ähnlichkeit als in B, da die verwendete Gewichtungs-map den Detailstrukturen weniger Gewichtung gibt als den groben Strukuren, die durch die Gaußsche Unschärfe weniger stark beeinflusst werden. HaarVectorPSI Mulitweight hingegen, sieht in Signal B eine höhere Ähnlichkeit als in A, da jede Detailebene eine eigene Gewichtungs-map hat. Es werden also Kanten im Detailbereich stärker gewichtet als bei HaarVectorPSI. Daher hat die durch die gaußsche

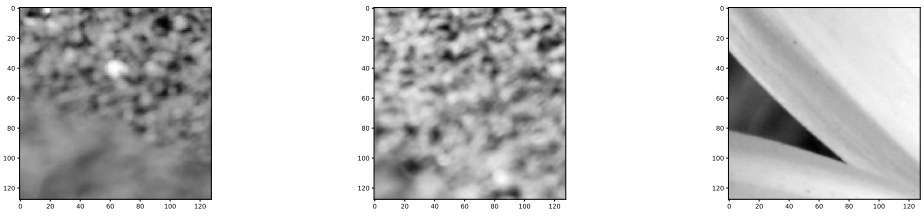


(a) (b) (c)

Abbildung 4.26: (a) Referenzvolumen, (b) Volumen mit gaußscher Unschärfe A, (c) Rauschvolumen B. Volumen A ist durch gaußsche Unschärfe mit $\sigma = 0,8$ verändert und Volumen B durch gaußsches Rauschen mit $\sigma = 1900$ verzerrt.

Unschärfe bewirkte Änderung einen stärkeren Einfluss auf die Ähnlichkeitsbestimmung. HaarVectorPSI Multiweight reagiert also auf Änderungen in den Detail-Strukturen sensibler als HaarVectorPSI.

HaarVectorPSI misst die Ähnlichkeit zwischen zwei Signalen anhand der Übereinstimmung ihrer Gradienten an jeder Position. HaarHistSim misst die Ähnlichkeit unabhängig von der Position, indem sie die Übereinstimmung der Histogramme vergleicht, die in Kapitel 3.3 erläutert werden. Die Ähnlichkeit zwischen zwei Signalen basiert also auf der Übereinstimmung der Struktur ihrer Gradienten, die eine andere Art der Ähnlichkeit darstellt als die positionsabhängige Ähnlichkeit bei HaarVectorPSI. Anhand der in Abbildung 4.27 dargestellten Bilder, welche aus der Salzburg Texture Image Database [Kwi+11] stammen, sollen die Unterschiede in der Ähnlichkeitsbestimmung verdeutlicht werden. Als Referenzbild



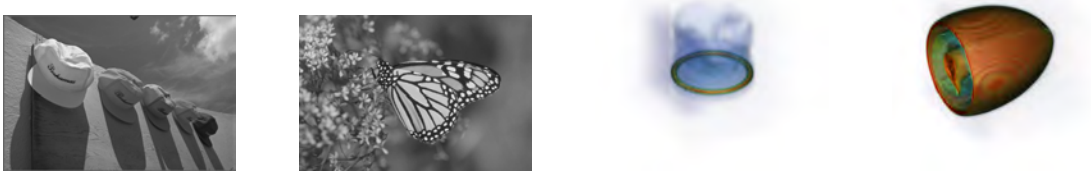
(a) (b) (c)

Abbildung 4.27: (a)Referenzbild (Kies), (b) Bild A (Kies), (c) Bild B (Blume) aus der Salzburg Texture Image Database [Kwi+11]

wurde zufällig ein Kies-Bild ausgewählt. Die Bilder A und B stammen aus den Kategorien 'Kies' und 'Blumen' und werden mit dem Referenzbild verglichen. Die Ergebnisse der Ähnlichkeitsbewertung hierzu sind: HaarVectorPSI : (A = 0.4673 , B = 0.5275), HaarHistSim : (A

= 0.6672 , B = 0.2972). Das Referenzbild und Bild A sind Bilder der selben Kategorie und beinhalten ähnliche Strukturen, während Bild B keine Kies-Strukturen aufweist. Dennoch bewertet HaarVectorPSI Bild B als ähnlicher zum Referenzbild als Bild A. HaarHistSim erkennt in Bild A eine höhere Ähnlichkeit zum Referenzbild als zu Bild B. Die Ordnung der Ähnlichkeiten ist also unterschiedlich. Da HaarVectorPSI die Ähnlichkeit positionsweise betrachtet, kann zwischen zwei Bildern mit ähnlicher Textur nur dann eine hohe Ähnlichkeit ermittelt werden, wenn die darin enthaltenen Strukturen auch in ihrer Position gut übereinstimmen. Dabei ist HaarVectorPSI sehr empfindlich gegenüber Translation, wie in Kapitel 4.1.1 gezeigt. Sind die Texturen zweier Bilder ähnlich, jedoch verschoben, ergibt sich daher eine vergleichsweise niedrige Ähnlichkeit. Das Referenzbild und Bild A haben eine ähnliche Textur und daher auch eine ähnliche Verteilung von Kantenstärken und Richtungen der Gradienten. Aufgrund dessen stellt HaarHistSim, wegen der positionsunabhängigen Betrachtung der Ähnlichkeit, eine höhere Übereinstimmung zu Bild A fest. Betrachtet man also die Ähnlichkeit zweier Signale bezüglich der Übereinstimmung ihrer Texturen, so ist der positionsunabhängige Ansatz des HaarHistSim vorteilhaft.

Im nun folgenden Beispiel soll der Ähnlichkeitsverlauf betrachtet werden, wenn ein Signal A mit einem Signal C verglichen wird, wobei C eine lineare Interpolation von A nach B darstellt. In Abbildung 4.28 sind hierzu die genutzten Referenzbilder dargestellt. Für



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

Abbildung 4.28: Referenzbilder: (a) ”Caps”, (b) ”Monarch” aus der LIVE image database [She+] und die 3D-CT-Scans (c) Kolben, (d) Ü-Ei

den Test von 2D-Daten wird nun (a) schrittweise mit einem Bild verglichen, welches von (a) nach (b) interpoliert wird. Für 3D-Daten wird (c) ebenfalls schrittweise mit einem Volumen verglichen, welches eine Interpolation von (c) nach (d) darstellt. Die Interpolation ist dabei prozentual und im Bereich $[0, 100]$. Die Ergebnisse dazu sind in Abbildung 4.29 zu sehen. HaarPSI und die HaarVectorPSI Varianten zeigen sowohl für 2D als auch für 3D-Daten ein ähnliches verhalten. Je weiter von A nach B interpoliert wird, desto geringer der Ähnlichkeitswert. Die Stärke des Abfalls der Ähnlichkeit nimmt zu Beginn stetig zu

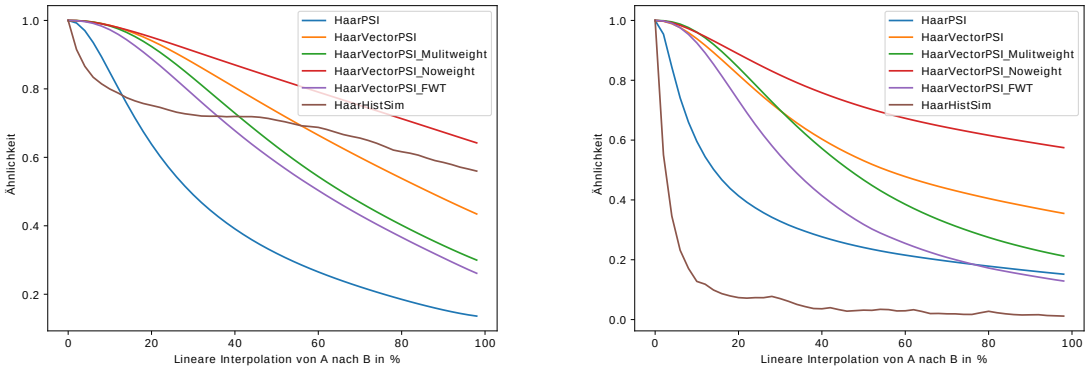


Abbildung 4.29: Ähnlichkeit eines Signals A zu einem Signal C. C ist eine lineare Interpolation von A nach B. (a) Ergebnis für 2D-Daten. (b) Ergebnis für 3D-Daten

und flacht zum Ende hin ab. Da die lineare Interpolation positionsweise berechnet wird und die HaarVectorPSI Varianten die Ähnlichkeit ebenfalls positionsweise bestimmen, ergibt sich glatter Ähnlichkeitsabfall. HaarHistSim zeigt hier ein abweichendes Verhalten. Im 2D-Fall ist im Ähnlichkeitsverlauf zur Mitte ein Plateau zu erkennen, bevor bei weiterer Interpolation die Ähnlichkeit wieder stärker abfällt. Im 3D-Fall ist dieses Verhalten nicht sichtbar. Die Ähnlichkeit fällt hier sehr schnell ab. Außerdem ist im Vergleich zu den anderen Algorithmen eine stärke Schwankung der Ähnlichkeit erkennbar. Dieses Verhalten von HaarHistSim erklärt sich aus der positionsunabhängigen Bestimmung der Ähnlichkeit der Textur der Signale. Abhängig davon, wie die Texturen der Referenzsignale A und B aufgebaut sind, kann bei steigendem Interpolationsgrad eine Interpolation C von A nach B eine Textur haben, welche nicht immer unähnlicher zu der Textur von A ist. Daher ist das Verhalten von HaarHistSim in diesem Testfall vergleichsweise unregelmäßig.

4.3 Pattern Matching

Die in Kapitel 3 vorgestellten Algorithmen zielen vor allem auf den Bereich des IQA (Image Quality Assessment) ab. In diesem Kapitel soll untersucht werden, ob die Algorithmen auch prinzipiell für Pattern Matching geeignet sein könnten. Dazu werden zum Testen der 2D-Algorithmen handschriftliche Zahlen aus der MNIST Datenbank und Bilder von Kleidung aus der Fashion-MNIST verwendet. Die MNIST Datenbank ist eine Sammlung von 70.000 Bildern von handgeschriebenen Ziffern von 0 bis 9, die oft als Benchmark für maschinelles

Lernen und Bildverarbeitung verwendet wird [LCB10]. Die Fashion-MNIST ist eine an an MNIST angelehnte Datenbank von ebenfalls 70.000 Bildern, jedoch von Kleidungstücken aus Zalando, welcher ein Modeanbieter im Onlinehandel ist. Für jede Bildkategorie wird ein Matching durchgeführt und die ähnlichsten fünf Bilder zurückgegeben. Dabei wird immer unter allen Kategorien gesucht, sodass für ein Referenzbild einer Kategorie A die Matches in einer Menge an Bilder mit allen Kategorien gesucht werden. Die Ergebnisse sind in Tabelle 4.6 für die Kategorie ”Zahl 8” und in Tabelle 4.7 für die Kategorie “Hose” aufgelistet.

Pattern Matching						
	Referenzbild	Ergebnis				
HaarPSI						
HaarVectorPSI						
HaarVectorPSI Multiweight						
HaarVectorPSI No weight						
HaarVectorPSI FWT						
Haar Hist Sim						

Tabelle 4.6: Pattern Matching für 2D Daten aus der MNIST Datenbank (handschriftliche Zahlen)

Betrachtet man die Ergebnisse bezüglich der MNIST Datenbank, so haben alle getesteten Algorithmen außer HaarHistSim unter den besten fünf Matches Bilder aus derselben Kategorie wie das Referenzbild (Kategorie 8) bestimmt. Bei den Ergebnissen bezüglich der MNIST-Fashion Datenbank ist dies für alle Algorithmen der Fall. Im Folgenden soll nun untersucht werden inwiefern die Algorithmen bezüglich Texture-Pattern-Matching geeignet sind. Die Daten dazu stammen aus der ”Salzburg Texture Image Database (STex)” [Kwi+11]. Sie beinhaltet eine umfangreiche Sammlung von 476 Farbtexturbildern, die in

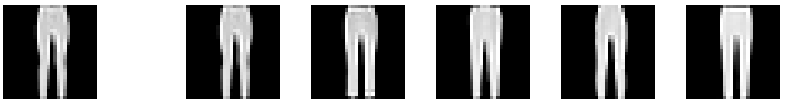
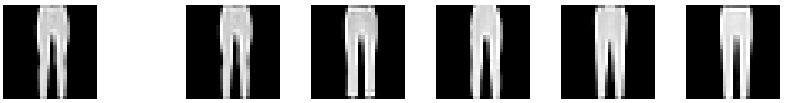
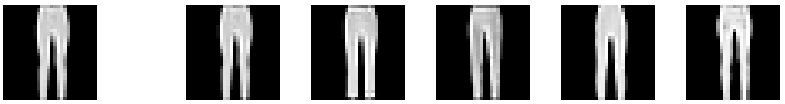
Pattern Matching		
	Referenzbild	Ergebnis
HaarPSI		
HaarVectorPSI		
HaarVectorPSI Multiweight		
HaarVectorPSI No weight		
HaarVectorPSI FWT		
Haar Hist Sim		

Tabelle 4.7: Pattern Matching für 2D-Daten aus der MNIST FASHION Datenbank (Kleindung aus Zalando)

der Umgebung von Salzburg, Österreich, aufgenommen wurden. Diese werden für den Anwendungszweck dieser Arbeit in Graustufenbilder umgewandelt. In Abbildung 4.8 sind die Ergebnisse für ein Referenzbild vom Typ ”Wood” dargestellt.

Die Ergebnisse zeigen deutlich, dass HaarPSI und HaarVectorPSI und seine Varianten nicht für Pattern-Matching bezüglich Textur geeignet sind. In den fünf ähnlichsten Matches sind Bilder verschiedener Kategorien vorhanden. HaarHistSim erzielt hier jedoch aufgrund der positionsunabhängigen Betrachtung der Ähnlichkeit gute Resultate. Alle fünf dargestellten Matches sind aus der Kategorie ”Wood” aus der auch das Referenzbild stammt.

Um 3D Pattern Matching zu untersuchen, werden die nötigen Daten aus den Referenzsignalen in Abbildung 4.30 generiert. Diese Referenzen werden mit verschiedenen Methoden verzerrt und so 200 veränderte Signale erzeugt. Die Verzerrungsarten umfassen Translation (0px - 5px), Rotation (0° - 5°), gaußsche Unschärfe ($\sigma \in [0, 1]$) und gaußsches Rauschen ($\sigma \in [0, 2500]$) mit zufälliger Intensität. In Tabelle 4.9 sind die Ergebnisse für Pattern-Matching bezüglich des Kolben aus Abbildung 4.30 dargestellt. In der Tabelle 4.9 sind für

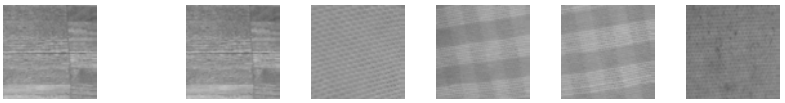
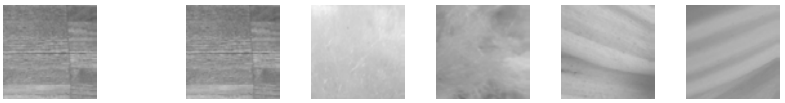
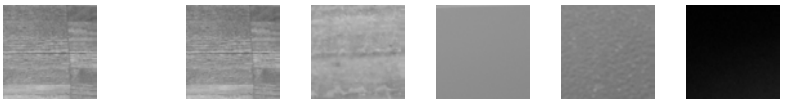
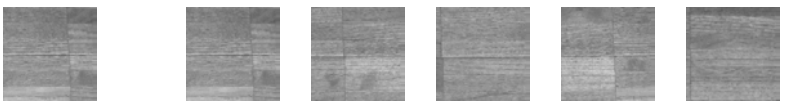
Texture Pattern Matching		
	Referenzbild	Ergebnis
HaarPSI		
HaarVectorPSI		
HaarVectorPSI Multiweight		
HaarVectorPSI No weight		
HaarVectorPSI FWT		
Haar Hist Sim		

Tabelle 4.8: Pattern Matching für 2D-Daten aus der ”Salzburg Texture Image Database (STex)” [Kwi+11]

jeden Algorithmus die fünf ähnlichsten Signale dargestellt. Es lässt sich feststellen, dass für jeden Algorithmus die besten fünf Resultate jeweils ebenfalls vom Typ ”Kolben” sind.

4.4 IQA

In diesem Kapitel wird die Korrelation zwischen den Rankings der in Kapitel 3 vorgestellten Algorithmen und einem gewählten Vergleichsranking für den 2D-Fall vorgestellt. Als Vergleichsordnung wird der ’Human Score’ aus der ”LIVE image database Release 2” [She+] verwendet. Dabei wird die Spearman-Korrelation als Maß für die Übereinstimmung zwischen den Rankings verwendet. Die ”LIVE Image Quality Database Release 2” enthält 982 Bilder, die mit verschiedenen Arten von Verzerrungen versehen sind, wie z.B. JPEG-Kompression, JPEG2000-Kompression, Gaußsche Unschärfe, weißes Rauschen und fastfading-Fehler. Die Bilder wurden von 29 bis 39 menschlichen Betrachtern bewertet, die ihre subjektive Einschätzung der Bildqualität auf einer Skala von 1 (schlecht) bis 9

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

Abbildung 4.30: Würfel (a), Zylinder (b), Ei (c), Kolben (d)

Pattern Matching						
	Referenzbild	Ergebnis				
HaarPSI						
HaarVectorPSI						
HaarVectorPSI Multiweight						
HaarVectorPSI No weight						
HaarVectorPSI FWT						
Haar Hist Sim						

Tabelle 4.9: Pattern Matching für 3D-Daten aus einem generierten 3D-Dataset

(ausgezeichnet) angegeben haben.

Die Tabelle 4.10 zeigt die Spearman-Rangkorrelationskoeffizienten zwischen den Bewertungen der getesteten Algorithmen und den menschlichen Bewertungen für fünf Verzerrungsarten und die gesamte LIVE Database release 2. Die Verzerrungsarten sind JPEG2000 (jpg2k), JPEG (jpg), Gaussian white noise (gwn), Gaussian blur (gblur) und fastfading (ff).

Die Ergebnisse zeigen, dass alle Algorithmen eine hohe Korrelation mit den menschlichen Bewertungen haben, was darauf hinweist, dass sie die Bildqualität gut messen können.

Order Correlation LIVE Database						
	jpg2k	jpg	gwn	gblur	ff	gesamt
HaarPSI	**0.9702**	**0.9847**	**0.9799**	0.9676	0.9552	**0.9689**
HaarVectorPSI	0.9676	0.9835	0.9611	0.9677	0.9646	0.9674
HaarVectorPSI Mulitweight	0.9594	0.9808	0.9535	**0.9702**	0.9622	0.9581
HaarVectorPSI Noweight	0.9612	0.9726	0.9378	0.9647	0.9540	0.9362
HaarVectorPSI FWT	0.9509	0.9789	0.9305	0.9530	**0.9718**	0.9579
HaarHistSim	0.9578	0.9750	0.9534	0.9557	0.9434	0.9200

Tabelle 4.10: Spearman Korrelation LIVE Database. Die Korrelationen werden mit HaarPSI verglichen

Höhere Korrelation als HaarPSI mit statistischer Signifikanz ($p < 0.05$)

Niedrigere Korrelation als HaarPSI mit statistischer Signifikanz ($p < 0.05$)

Der Algorithmus mit der höchsten Korrelation für die gesamte Datenbank ist HaarPSI mit einem Wert von 0.9590. Der Algorithmus mit der niedrigsten Korrelation für die gesamte Datenbank ist HaarHistSim mit einem Wert von 0.920. Insgesamt haben HaarPSI und HaarVectorPSI eine ähnlich gute Korrelation. HaarPSI liefert insgesamt das beste Ergebnis, doch je nach Verzerrung schneiden andere Algorithmen besser ab. Besonders bei gaußscher Unschärfe und fastfading-Fehler zeigen die HaarVectorPSI Varianten etwas bessere Ergebnisse.

4.5 Laufzeit

Die Berechnungszeit ist ein wichtiger Faktor bei der Algorithmuswahl. In diesem Kapitel wird die Performance der in Kapitel 3 entwickelten Algorithmen analysiert. Die Berechnungen wurden auf einem Rechner mit Intel Core i7-7700HQ @ 2,80 GHz ausgeführt. Dabei wird die Berechnungszeit der Algorithmen in Abhängigkeit der Datengröße, bzw. Auflösung der zu vergleichenden Daten und der Wahl der Skalierung der Haar-Filter analysiert. Die Ergebnisse zur Berechnungszeit bezüglich Auflösung sind in Abbildung 4.31 dargestellt. Dabei wurden randomisiert 2D und 3D-Signale steigender Auflösung erzeugt und miteinander verglichen. Die erzeugten Daten sind dabei jeweils immer quadratisch, bzw. kubisch. Die dargestellten Ergebnisse entsprechen dem Mittelwert aus drei verschiedenen Durchläufen. Da bei steigender Datengröße immer mehr Informationen verarbeitet werden müssen,

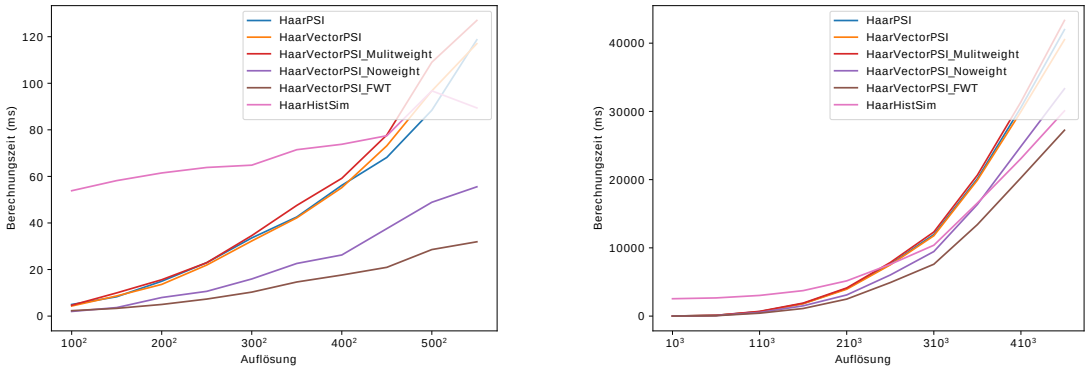


Abbildung 4.31: Laufzeit der verschiedenen Algorithmen bei steigender Auflösung für (a) 2D und (b) 3D Daten.

steigt die Berechnungszeit logischerweise bei allen Algorithmen an. Da eine Verdoppelung der Kantenlänge eines quadratischen Bildes eine vierfache Datenmenge ergibt, steigen auch die Berechnungszeiten quadratisch an. FWT HaarVectorPSI ist dabei aufgrund der Verwendung der FWT deutlich schneller als alle anderen getesteten Algorithmen, da durch das Downsampling nach jedem Filterungsschritt die zu verarbeitende Datenmenge halbiert wird. Ebenfalls hat HaarVectorPSI ohne Gewichtung eine deutlich schnellere Berechnungszeit als die HaarVectorPSI Varianten mit Gewichtung. HaarHistSim hat schon bei geringer Datengrößen eine vergleichsweise hohe Berechungszeit, jedoch steigt diese langsamer bei steigender Datengröße an, als bei den restlichen Algorithmen. Dies führt dazu, dass ab einer gewissen Datengröße HaarHistSim schneller berechnet werden kann als die restlichen Algorithmen. Für HaarVectorPSI ist dies für 2D-Daten ungefähr bei einer Kantenlänge von 500, für 3D-Daten bei 250 der Fall. Dieses Verhalten resultiert aus der Tatsache, dass die Berechnung der Histogramm Ähnlichkeit nach Bildung der Histogramme unabhängig von der Datengröße des Inputs ist. Für kleine Inputgrößen dominiert die Berechnung der Ähnlichkeit der Histogramme die Laufzeit. Ab einer gewissen Datengröße wird die Berechnungszeit hauptsächlich von der Berechnung der Gradienten und der Bildung der Histogramme bestimmt. Die Berechnung der Ähnlichkeit der Histogramme spielt dann nur noch eine untergeordnete Rolle.

Die Berechnungszeit der Algorithmen wird auch von der Wahl der Skalierung der Haar-Filter beeinflusst. In Abbildung 4.32 ist dieser Zusammenhang dargestellt. Die in dieser Abbildung angetragene Filter Skalierung entspricht dabei der maximalen Skalierung der

Haar Filter. Für HaarVectorPSI und seine Varianten bleibt die Anzahl der Filter der Filterbank dabei gleich, für die FWT Varianten wird ein Signal bis zu dieser maximalen Skalierung zerlegt. Die Größe des Signals ist dabei auf einen konstanten Wert festgelegt. Erwartungs-

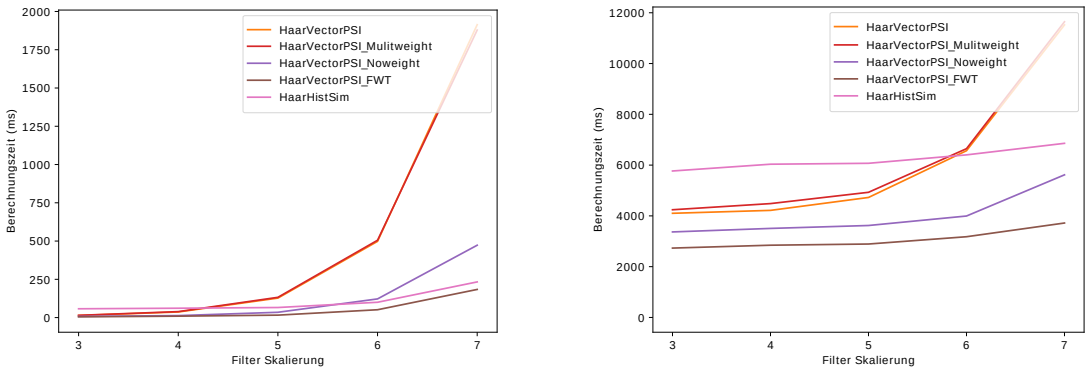


Abbildung 4.32: Laufzeit der verschiedenen Algorithmen bei steigender maximalen Filter Skalierung für (a) 2D und (b) 3D Daten
bei einer konstanten Inputgröße mit Seitenlängen von 200 (2D) und 200 (3D)

gemäß steigt die Berechnungszeit für alle Algorithmen bei steigender Skalierung an, da bei den Varianten ohne FWT die Faltung der Filter mit dem Signal berechnungsaufwändiger wird und bei den FWT Varianten zusätzliche Skalen berechnet werden müssen. Da bei den FWT Varianten das Signal nach jedem Filterungsschritt einem Subsampling mit Faktor 2 unterzogen wird, nimmt die zusätzliche Berechnungszeit für jede weitere Auflösungsstufe ab. Für die HaarVectorPSI Varianten ohne FWT nimmt die zusätzliche Berechnungszeit für jede Erhöhung der Filterskalierung zu, da die Filter immer auf das Eingangssignal mit voller Auflösung angewendet werden. Die Anzahl der Klassenunterteilungen beeinflusst die Berechnungszeit von HaarHistSim, aber der Einfluss nimmt ab, wenn die Inputgröße zunimmt. Abbildung 4.33 zeigt den Zusammenhang für die Parameter 'lengthBins' und 'angleBins' für eine feste Inputgröße. Bezüglich des Paramteters 'lengthBins' zeigt sich sowohl für den 2D und 3D-Fall ein linearer Zusammenhang von Klassenanzahl und Berechnungszeit. Für den Parameter 'angleBins', welcher die Anzahl der Klassen für die Richtung der Vektoren festlegt, stellt sich für den 3D-Fall ein nicht-linearer Zusammenhang ein. Im 3D-Fall wird die Richtung eines Vektors durch zwei Winkel beschrieben. Eine Erhöhung von 'angleBins' führt deshalb zu einer feineren Klassenunterteilung in zwei Dimensionen des Histogramms. Die Berechnungszeit steigt somit quadratisch an.

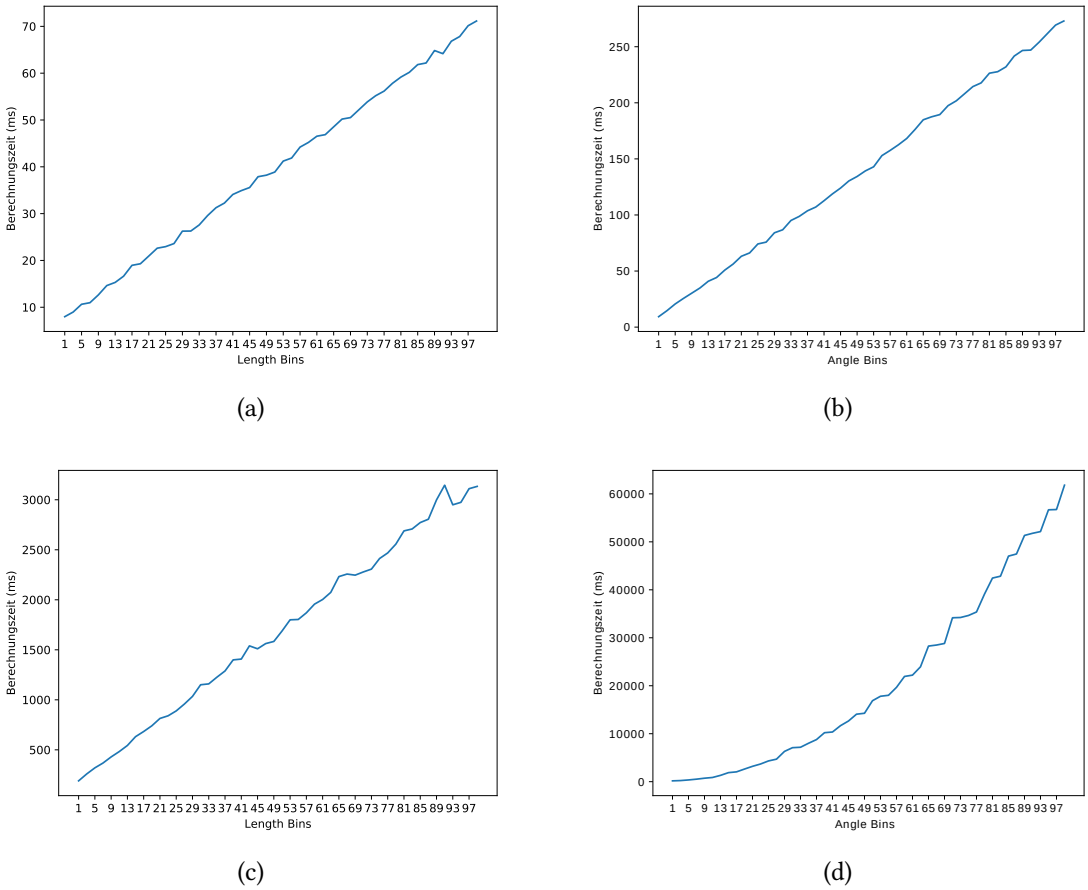


Abbildung 4.33: Laufzeit der verschiedenen Algorithmen bei steigender Feinheit der Klassenunterteilungen für HaarHistSim bezüglich (a)(b) 2D und (c)(d) 3D-Daten bei einer konstanten Inputgröße mit Seitenlängen von 200 (2D) und 64 (3D)

5 Fazit

Im Rahmen dieser Arbeit wurden erfolgreich Algorithmen entwickelt, die zur Bestimmung der Ähnlichkeit zwischen 3D-CT-Daten verwendet werden können. Es wurden drei Algorithmen vorgestellt: HaarPSI3D, HaarVectorPSI und HaarHistSim. Diese Algorithmen basieren auf der diskreten Haar-Wavelet-Transformation und unterscheiden sich in der Art und Weise, wie sie die Ähnlichkeit zwischen den Daten definieren.

HaarPSI3D ist eine Erweiterung des bestehenden HaarPSI Algorithmus aus [Rei+18] für Voxeldaten. HaarVectorPSI definiert die Ähnlichkeit auf Basis der Gradientenvektoren, die ebenfalls durch die Haar-DWT berechnet werden können. Für den Vergleich von Gradientenvektoren wurde das Ähnlichkeitsmaß "längensensitive Kosinusähnlichkeit" entwickelt. HaarHistSim definiert die Ähnlichkeit auf Basis von Histogrammen, die aus den Gradientenvektoren erstellt werden.

Die Algorithmen wurden anhand verschiedener Kriterien evaluiert, wie z.B. ihrer Empfindlichkeit gegenüber Translation, Rotation, Skalierung und Bildfehler, ihrer Leistung in der Bildqualitätsbewertung (IQA) und ihrer Anwendbarkeit für Pattern-Matching. Die Ergebnisse zeigten, dass HaarVectorPSI ein ähnliches Verhalten wie HaarPSI bezüglich der Empfindlichkeiten der getesteten Verzerrungen aufweist, aber bessere Eigenschaften in IQA bezüglich Unschärfe und fastfading hat. Allerdings war HaarPSI in IQA insgesamt besser als HaarVectorPSI. Die Wahl der Gewichtungsmethode bei HaarVectorPSI hat einen Einfluss auf die Ähnlichkeitsermittlung und Empfindlichkeit bezüglich Verzerrungen. HaarVectorPSI noweight gewichtet jeden Datenpunkt gleich, ist jedoch empfindlicher bezüglich gaußschem Rauschen als HaarVectorPSI. Ist das Signal jedoch durch "speckle" Rauschen verzerrt, dann reagiert HaarVectorPSI empfindlicher als die gewichtungslose Variante. HaarHistSim ohne FWT ist für Verschiebungen des Signals unempfindlich und daher nützlich, wenn die Ähnlichkeit unabhängig von Translationen im Signal betrachtet werden soll. Mit HaarVectorPSI und HaarPSI ließ sich grundlegendes Pattern-Matching durchführen, was mit HaarHistSim nicht gelang. Dafür zeigte HaarHistSim eine gute Leistung bei Texture

Pattern-Matching, was die anderen Algorithmen nicht erreichten. HaarVectorPSI FWT, eine Variante von HaarVectorPSI mit Subsampling, zeigte trotz Artefaktbildung nur geringfügig schlechtere Ergebnisse als HaarVectorPSI im Bereich IQA. Teilweise sogar bessere (fastfading). Das Pattern-Matching war mit HaarVectorPSI FWT vergleichbar zu HaarVectorPSI. Daher ist HaarVectorPSI mit FWT eine interessante Wahl, wenn sehr große 3D-CT-Daten verglichen werden sollen, da bei ähnlich guten Ergebnissen die Berechnungszeit erheblich sinkt.

Ein möglicher Ansatz für zukünftige Forschung wäre die Anwendung von IQA auf 3D-Daten, um die Qualität von 3D-CT-Scans zu bewerten. Außerdem könnte man andere Arten von Wavelets testen, um zu untersuchen, ob sie bessere Ergebnisse für Pattern-Matching liefern. Ein zusätzlicher Forschungsbereich für die Zukunft wäre die Verwendung der Algorithmen für Objekterkennung, um die Signalinhalte zu interpretieren und zu klassifizieren.

Literatur

[Bra03] Andrew Bradley. „Shift-Invariance in the Discrete Wavelet Transform“. In: 1 (Jan. 2003) (siehe S. 14).

[Haa10] Alfred Haar. „Zur Theorie der orthogonalen Funktionensysteme“. In: *Mathematische Annalen* 69.3 (Sep. 1910), S. 331–371. ISSN: 1432-1807. DOI: 10.1007/BF01456326. URL: https://doi.org/10.1007/BF01456326 (siehe S. 7).

[Han10] Randolf Hanke. „Computertomographie in der Materialprüfung Stand der Technik und aktuelle Entwicklungen“. In: *DGZfP-Jahrestagung, Erfurt/Germany* (2010) (siehe S. 1).

[Kwi+11] Roland Kwitt u. a. *Salzburg Texture Image Database (STex)*. https://www.wavelab.at/sources/STex/. Accessed: 2023-03-31. 2011 (siehe S. 51, 73, 76, 78).

[LCB10] Yann LeCun, Corinna Cortes und CJ Burges. „MNIST handwritten digit database“. In: *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist* 2 (2010) (siehe S. 76).

[Mey93] Yves Meyer. „Wavelets - Algorithms and Applications“. In: (1993) (siehe S. 8).

[Ont20] Santiago Ontañón. „An overview of distance and similarity functions for structured data“. In: *Artificial Intelligence Review* 53.7 (Okt. 2020), S. 5309–5351. ISSN: 1573-7462. DOI: 10.1007/s10462-020-09821-w. URL: https://doi.org/10.1007/s10462-020-09821-w (siehe S. 18).

[PPT21] Baisakhi Sur Phadikar, Amit Phadikar und Subro S Thakur. „A comprehensive assessment of content-based image retrieval using selected full reference image quality assessment algorithms“. In: *Multimedia Tools and Applications* 80.10 (2021), S. 15619–15646 (siehe S. 1, 2).

[Pha+17] Baisakhi Sur Phadikar u. a. „Content-based image retrieval for big visual data using image quality assessment model“. In: *CSI transactions on ICT* 5.1 (2017), S. 45–51 (siehe S. 2).

[Rav+17] Aditya Ravishankar u. a. „A survey on noise reduction techniques in medical images“. In: *2017 International conference of Electronics, Communication and Aerospace Technology (ICECA)*. Bd. 1. 2017, S. 385–389. DOI: 10.1109/ICECA.2017.8203711 (siehe S. 61).

[Rei+04] Christof Reinhart u. a. „Modern voxel based data and geometry analysis software tools for industrial CT“. In: *CD-ROM Proceedings of 16th WCNDT–World Conference on NDT, Montreal, Canada*. 2004 (siehe S. 1).

[Rei+18] Rafael Reisenhofer u. a. „A Haar wavelet-based perceptual similarity index for image quality assessment“. In: *Signal Processing: Image Communication* 61 (2018), S. 33–43 (siehe S. 2, 20–22, 84).

[Sar19] Gerhard Sartorius. „Distanz- und Ähnlichkeitsmaße“. In: *Erfassen, Verarbeiten und Zuordnen multivariater Messgrößen*. Springer Vieweg, 2019, S. 111–133. DOI: 10.1007/978-3-658-23576-5_8 (siehe S. 18).

[Sau16] Tomas Sauer. „Skript zur Vorlesung: Einführung in die Signal- und Bildverarbeitung SS 2016“. In: (2016) (siehe S. 7).

[SS22] Tomas Sauer und Andreas Michael Stock. „Haar Waveletes, Gradients and approximate total variation regularization“. In: (2022) (siehe S. 10, 17).

[Sco09] David Scott. „Averaged Shifted Histogram“. In: *Wiley Interdisciplinary Reviews: Computational Statistics* 2 (Dez. 2009), S. 160–164. DOI: 10.1002/wics.54 (siehe S. 38).

[She+] HR Sheikh u. a. *LIVE image quality assessment database release 2*. URL: http://live.ece.utexas.edu/research/quality (siehe S. 33, 36, 38, 41, 42, 51, 74, 78).

[She+92] Mark J Shensa u. a. „The discrete wavelet transform: wedding the a trous and Mallat algorithms“. In: *IEEE Transactions on signal processing* 40.10 (1992), S. 2464–2482 (siehe S. 14).

[TV08] Johan WH Tangelder und Remco C Veltkamp. „A survey of content based 3D shape retrieval methods“. In: *Multimedia tools and applications* 39.3 (2008), S. 441–471 (siehe S. 1).

[Tra12] Minh-Phuong Tran. „3D image analysis with variational methods and wavelets : applications to medical image processing“. Theses. Université d'Orléans, Sep. 2012. URL: https://tel.archives-ouvertes.fr/tel-00772308 (siehe S. 9, 10, 16).

[ZM20] Guangtao Zhai und Xiongkuo Min. „Perceptual image quality assessment: a survey“. In: *Science China Information Sciences* 63.11 (2020), S. 1–52 (siehe S. 1, 2).

Eidesstattliche Erklärung

Ich erkläre, dass ich die vorliegende Arbeit selbständig, ohne fremde Hilfe und ohne Benutzung anderer als der angegebenen Quellen und Hilfsmittel verfasst habe und dass alle Ausführungen, die wörtlich oder sinngemäß übernommen wurden, als solche gekennzeichnet sind. Mit der aktuell geltenden Fassung der Satzung der Universität Passau zur Sicherung guter wissenschaftlicher Praxis und für den Umgang mit wissenschaftlichem Fehlverhalten bin ich vertraut. Ich erkläre mich einverstanden mit einer Überprüfung der Arbeit unter Zuhilfenahme von Dienstleistungen Dritter (z.B. Anti-Plagiatssoftware) zur Gewährleistung der einwandfreien Kennzeichnung übernommener Ausführungen ohne Verletzung geistigen Eigentums an einem von anderen geschaffenen urheberrechtlich geschützten Werk oder von anderen stammenden wesentlichen wissenschaftlichen Erkenntnissen, Hypothesen, Lehren oder Forschungsansätzen.

Passau, 15. Mai 2023

Manuel Pauli