

EBERHARD KARLS UNIVERSITÄT TÜBINGEN  
FACHBEREICH INFORMATIK

MASTERARBEIT  
IM STUDIENGANG MEDIENINFORMATIK

**Analyse und plattformunabhängige  
Implementierung zum effizienten  
Auflösen des Bayer-Mosaiks**

*Andreas Reiter*

Matrikelnr.: 5739000

Prüfer:

Prof. Dr. Thomas WALTER  
Prof. Dr. Andreas SCHILLING

Bearbeitungszeitraum:

01.09.2022 - 28.02.2023

## **Ehrenwörtliche Erklärung**

Hiermit versichere ich, dass ich die vorliegende Masterarbeit selbstständig und nur mit den angegebenen Hilfsmitteln angefertigt habe und dass alle Stellen, die dem Wortlaut oder dem Sinne nach anderen Werken entnommen sind, durch Angaben von Quellen als Entlehnung kenntlich gemacht worden sind. Diese Masterarbeit wurde in gleicher oder ähnlicher Form in keinem anderen Studiengang als Prüfungsleistung vorgelegt.

---

Ort, Datum

---

Unterschrift

## Zusammenfassung

Aufnahmen von digitalen Kameras werden oft zu fertigen Bildern verarbeitet, ohne zusätzliche Einflussnahme des Nutzers. Mit RAW-Formaten kann eine digitale Aufnahme nahezu unverarbeitet abgespeichert werden. Die meisten RAW-Formate sind allerdings proprietär. Dies trifft nicht auf das Digital Negative (DNG)-Format zu. Dieses Format ist öffentlich dokumentiert und mit einem kostenlosen Konverter können andere RAW-Formate in das DNG-Format überführt werden. Um eine Aufnahme im DNG-Format in ein fertiges Bild zu überführen, wurde Java Extended NEF Image File Format Editor (JENIFFER) 2 entwickelt. Dieses Programm ermöglicht eine transparente Verarbeitung des Bildes, im Gegensatz zu vielen kommerziellen Anwendungen.

Digitale Kamerasensoren sind in der Regel monochrom und können deswegen nur Graustufenbilder erzeugen. Um ein Farbbild zu erzeugen, müssen in jedem Bildpunkt drei Farbwerte vorhanden sein. Dazu wird am häufigsten das Bayer-Mosaik verwendet. Das Bayer-Mosaik besteht aus sich wiederholenden Blöcken aus vier Farbfiltern, einmal für die Farben Rot, zweimal Grün und einmal Blau. Die fehlenden zwei Farbwerte in jedem Bildpunkt müssen durch sogenannte Demosaicing-Algorithmen berechnet werden.

In dieser Arbeit wurde JENIFFER2 um sechs Demosaicing-Algorithmen erweitert. Zwei dieser Algorithmen wurden in jeweils zwei unterschiedlichen Variationen implementiert, was die Gesamtzahl der in JENIFFER2 verfügbaren Demosaicing-Algorithmen auf zwölf bringt. Die neu implementierten Algorithmen wurden aus existierenden Algorithmen ausgewählt, die in wissenschaftlicher Literatur oft referenziert, oder in anderen Open-Source RAW-Konvertern genutzt werden. Zusätzlich wurde ein Kombinationsalgorithmus entworfen, welcher aus Teilen von zwei bereits existierenden Algorithmen besteht. Die implementierten Algorithmen wurden abschließend bezüglicher ihrer erzeugten Bildqualität und Laufzeit untersucht. Hier konnte festgestellt werden, dass in Abhängigkeit von den Bildeigenschaften, Ratio Corrected Demosaicing (RCD) oder der Kombinationsalgorithmus die beste Bildqualität erzeugen, allerdings auch die höchsten Rechenzeiten benötigen.

## Abstract

Recordings from digital cameras are often processed into finished images without any additional influence from the user. With RAW-formats, a digital image can be saved almost unprocessed. However, most RAW-formats are proprietary. This is not the case with the Digital Negative (DNG)-format. This format is publicly documented and with a free converter other RAW-formats can be converted into the DNG-format. To convert a recording in DNG-format into a finished image, Java Extended NEF Image File Format Editor (JENIFFER) 2 was developed. This program allows a transparent processing of the image, unlike many commercial applications.

Digital camera sensors are usually monochrome and therefore can only produce grayscale images. To create a color image, three color values must be present in each pixel. The Bayer-Mosaic is most commonly used for this purpose. The Bayer-Mosaic consists of repeating blocks of four color filters, once for the colors red, twice green and once blue. The missing two color values in each pixel, must be calculated with so-called Demosaicing-Algorithms.

In this work, JENIFFER2 was extended with six Demosaicing-Algorithms. Two of these algorithms were implemented in two variations each, bringing the total number of Demosaicing-Algorithms available in JENIFFER2 to twelve. The newly implemented algorithms were selected from existing algorithms, which are often referenced in scientific literature or used in other open-source RAW-converters. Additionally, a combination algorithm was designed, which consists of parts of two already existing algorithms. Finally, the implemented algorithms were examined with regard to their generated image quality and runtime. It could be determined that, depending on the image properties, Ratio Corrected Demosaicing (RCD) or the combination algorithm produce the best image quality, but also require the highest computing times.

# Inhaltsverzeichnis

<b>Zusammenfassung</b>	<b>3</b>
<b>Abstract</b>	<b>4</b>
<b>Abbildungsverzeichnis</b>	<b>7</b>
<b>Tabellenverzeichnis</b>	<b>9</b>
<b>Abkürzungsverzeichnis</b>	<b>10</b>
<b>1. Einleitung</b>	<b>12</b>
1.1. Motivation . . . . .	12
1.2. Ziele . . . . .	13
1.3. Überblick . . . . .	14
<b>2. Grundlagen</b>	<b>15</b>
2.1. Kamerasensoren . . . . .	16
2.2. Bayer-Filter . . . . .	17
2.3. Demosaicing . . . . .	19
2.3.1. Kategorien von Demosaicing-Algorithmen . . . . .	20
2.3.2. Demosaicing-Artefakte . . . . .	22
2.4. RAW-Format . . . . .	23
2.5. DNG-Format . . . . .	26
2.5.1. Adobe DNG Converter . . . . .	28
2.5.2. DNG Konvertierungspipeline . . . . .	29
<b>3. JENIFFER2</b>	<b>34</b>
3.1. Verarbeitungsschritte und bisherige Ausführungszeit . . . . .	35
3.2. Implementierte Demosaicing-Algorithmen . . . . .	37
3.2.1. Keine Interpolation . . . . .	38
3.2.2. Nearest Neighbour . . . . .	38
3.2.3. Bilineare Interpolation mit Mittelwert . . . . .	38
3.2.4. Bilineare Interpolation mit Median . . . . .	38
3.2.5. Bikubische Interpolation . . . . .	39
3.2.6. Hamilton-Adams Interpolation . . . . .	39
3.2.7. Malvar-He-Cutler lineare Interpolation . . . . .	39
3.2.8. Chuan-Kai Lin Patterned Pixel Grouping . . . . .	40

3.2.9. Zhang-Wu Directional Linear Minimum Mean-Square Error . . . . .	41
3.2.10. Luis Sanz Rodríguez Ratio Corrected Demosaicing . . . . .	42
3.2.11. DLMMSE Grünkanalinterpolation mit RCD Rot- und Blaukanal- interpolation . . . . .	43
3.3. Stand der Demosaicing-Algorithmen . . . . .	44
3.4. Weitere Änderungen an JENIFFER2 . . . . .	46
<b>4. Vergleich der Demosaicing-Algorithmen</b>	<b>48</b>
4.1. Teststrategien . . . . .	48
4.2. Image Dataset . . . . .	51
4.3. Subjektiver Vergleich . . . . .	57
4.4. Messung . . . . .	68
4.5. Diskussion . . . . .	74
4.5.1. Auswertung der Bildqualität der implementierten Demosaicing- Algorithmen . . . . .	74
4.5.2. Auswertung der Laufzeiten der Demosaicing-Algorithmen . . . . .	78
4.5.3. Vorteile des kombinierten DLMMSE + RCD Algorithmus . . . . .	79
<b>5. Fazit und Ausblick</b>	<b>80</b>
5.1. Fazit . . . . .	80
5.2. Ausblick . . . . .	82
<b>Literaturverzeichnis</b>	<b>84</b>
<b>Anhang</b>	<b>90</b>
<b>A. Vollständige Messdaten</b>	<b>90</b>
<b>B. Installationsanleitung/ Installation Guide</b>	<b>106</b>
B.1. Prerequisites . . . . .	106
B.2. Building the DNG reader . . . . .	106
B.3. Building the UI . . . . .	106
<b>C. Benutzerhandbuch</b>	<b>107</b>
<b>D. User Manual</b>	<b>111</b>
<b>E. Rechtlicher Hinweis</b>	<b>115</b>

## Abbildungsverzeichnis

1.	Allgemeine Verarbeitungspipeline eines Bildes [5] . . . . .	15
2.	(a) Auslesearchitektur eines interline transfer Charge-Coupled Device (CCD) und (b) Complementary Metal Oxide Semiconductor (CMOS) Bildsensors [7, S. 8] . . . . .	16
3.	Beispiel eines Bayer-Filters [10] . . . . .	17
4.	Profilansicht von einzelnen Pixel in einem Bayer-Filter [10] . . . . .	18
5.	Unverarbeitete Aufnahme unter Verwendung eines Bayer-Filters . . . . .	19
6.	Framework der residualen Interpolation [25, S. 5] . . . . .	21
7.	Beispiele von Demosaicing-Artefakten. Im linken Bild treten Zipper-Artefakte entlang der Motorhaube auf. Im rechten Bild treten False-Color-Artefakte am Außenspiegel und entlang der Windschutzscheibe auf. [19, S. 12] . . .	23
8.	Methoden zum Erstellen von Digital Negative (DNG)-Dateien und Ein- bettung von Kameradetails [41] . . . . .	27
9.	Adobe DNG Converter und seine Einstellungsmasken [43] . . . . .	29
10.	Verarbeitungsschritte einer DNG-Datei in Java Extended NEF Image File Format Editor (JENIFFER)2 [3, S. 31] . . . . .	30
11.	Verarbeitungsschritte einer DNG-Datei in JENIFFER2 mit Beispielbil- dern nach jedem Schritt [3, S. 31] . . . . .	31
12.	Bibliothek Tab von JENIFFER2 . . . . .	34
13.	Editor Tab von JENIFFER2 . . . . .	35
14.	Ausführungszeiten der einzelnen Verarbeitungsschritte von JENIFFER2 vor dieser Masterarbeit [3, S. 74] . . . . .	36
15.	Color peak signal-to-noise ratio Performanz verschiedener representati- ver Demosaicing-Algorithmen auf dem IMAX und Kodak Dataset (vgl. Kapitel 4.2) [25, S. 2] . . . . .	44
16.	Visueller Vergleich verschiedener state-of-the-art Algorithmen [64, S. 227]	45
17.	Evaluationsprozedur von Demosaicing-Algorithmen mit Referenzbildern [68, S. 104] . . . . .	49
18.	Ausschnitte von Bildern der Kodak Lossless True Color Image Suite . .	52
19.	Ausschnitte von Bildern des McMaster Dataset . . . . .	54
20.	Ausschnitte von Bildern des Microsoft Research Cambridge Demosaicing Dataset . . . . .	56
21.	Verarbeitungsschritte zur Vorbereitung einer Portable Network Graphics (PNG)-Datei für das Demosaicing . . . . .	57

22.	Ausschnitte von Bild 13 aus dem Kodak Set von verschiedenen Algorithmen und Differenzbilder . . . . .	61
23.	Ausschnitte von Bild 19 aus dem Kodak Set von verschiedenen Algorithmen und Differenzbilder . . . . .	62
24.	Ausschnitte von Bild 5 aus dem McMaster Set von verschiedenen Algorithmen und Differenzbilder . . . . .	63
25.	Ausschnitte von Bild 18 aus dem MSR Set von verschiedenen Algorithmen und Differenzbilder . . . . .	64
26.	Ausschnitte von Bild 500 aus dem MicroSoft Research (MSR) Set von verschiedenen Algorithmen und Differenzbilder . . . . .	65
27.	Ausschnitte von Bild 18 aus dem MSR Set mit Rauschen von verschiedenen Algorithmen und Differenzbilder . . . . .	66
28.	Ausschnitte von Bild 500 aus dem MSR Set mit Rauschen von verschiedenen Algorithmen und Differenzbilder . . . . .	67
29.	Ausführungszeiten der Demosaicing-Algorithmen analog zu [3, S. 75] . . .	72
30.	Ausführungszeiten der in JENIFFER2 implementierten Demosaicing-Algorithmen gegenüber ihrer durchschnittlichen Peak Signal to Noise Ratio (PSNR)-Werte . . . . .	73

## Tabellenverzeichnis

1.	Mean Square Error (MSE), PSNR und Structural Similarity Index Measure (SSIM) Mittlwerte in den drei Bildersammlung Kodak, McMaster und MSR. Das jeweils beste Ergebnis in einer Spalte ist hervorgehoben. . . . .	69
2.	MSE, PSNR und SSIM Mittlwerte in den MSR mit Rauschen und die Differenz zu dem MSR Set ohne Rauschen. Das jeweils beste Ergebnis in einer Spalte ist hervorgehoben. . . . .	70
3.	MSE, PSNR und SSIM Werte mit einem verarbeiteten Bild aus Adobe Photoshop und Capture One als Referenz. MSE-Werte wurden mit dem Faktor $10^{-7}$ herunterskaliert. . . . .	71
4.	Ausführungszeiten in ms der einzelnen Verarbeitungsschritte für jeden Demosaicing-Algorithmus analog zu [3, S. 74] . . . . .	72
5.	MSE des Kodak Sets . . . . .	91
6.	PSNR des Kodak Sets . . . . .	92
7.	SSIM des Kodak Sets . . . . .	93
8.	MSE des McMaster Sets . . . . .	94
9.	PSNR des McMaster Sets . . . . .	95
10.	SSIM des McMaster Sets . . . . .	96
11.	MSE des MSR Sets . . . . .	97
12.	PSNR des MSR Sets . . . . .	98
13.	SSIM des MSR Sets . . . . .	99
14.	MSE des MSR Sets mit Rauschen . . . . .	100
15.	PSNR des MSR Sets mit Rauschen . . . . .	101
16.	SSIM des MSR Sets mit Rauschen . . . . .	102
17.	MSE Differenz des MSR Sets mit Rauschen - MSR Set . . . . .	103
18.	PSNR Differenz des MSR Sets mit Rauschen - MSR Set . . . . .	104
19.	SSIM Differenz des MSR Sets mit Rauschen - MSR Set . . . . .	105

## **Abkürzungsverzeichnis**

**ARI** Adaptive Residual Interpolation

**CCD** Charge-Coupled Device

**CFA** Color Filter Array

**CMOS** Complementary Metal Oxide Semiconductor

**CNN** Convolutional Neural Network

**DLMMSE** Directional Linear Minimum Mean-Square-Error

**DLMMSE-CodeEst** Directional Linear Minimum Mean-Square-Error mit Code Estimator

**DLMMSE-PaperEst** Directional Linear Minimum Mean-Square-Error mit Paper Estimator

**DNG** Digital Negative

**HA** Hamilton-Adams Interpolation

**JENIFFER** Java Extended NEF Image File Format Editor

**JPEG** Joint Photographic Experts Group

**MHC** Malvar-He-Cutler

**MSE** Mean Square Error

**MSR** MicroSoft Research

**NEF** Nikon Electronic Format

**NN** Nearest Neighbour

**PNG** Portable Network Graphics

**PPG** Patterned Pixel Grouping

**PSNR** Peak Signal to Noise Ratio

**RCD** Ratio Corrected Demosaicing

**SSIM** Structural Similarity Index Measure

**TIFF** Tagged Iage File Format

**VNG** Variable Number of Gradients

**XMP** eXtensible Metadata Platfrom

# **1. Einleitung**

Zwischen der Aufnahme und der Betrachtung eines Bildes, finden oft automatisch, eine Vielzahl an Verarbeitungsschritten statt. Um in einem professionellen Einsatz bestmögliche Ergebnisse zu erhalten, ist eine umfangreiche Bearbeitung des Bildes notwendig. Dies lässt sich nur ermöglichen, wenn das Bild in einem möglichst unverarbeiteten Zustand vorliegt. Zahlreiche proprietäre Dateiformate, die übergreifend als Rohdatenformat oder RAW-Daten bezeichnet werden, sollen diese Verarbeitung ermöglichen [1].

Die Verarbeitung von RAW-Dateien wird mittels RAW-Konvertern außerhalb der Kamera vorgenommen. RAW-Konverter verfügen über eine Vielzahl von Anpassungsmethoden, wie dem Demosaicing, den Weißabgleich oder der Farbraumtransformation. Jedoch wird in kommerziellen RAW-Konvertern meist nicht verraten, welches Demosaicingverfahren genutzt wird. Der Open Source RAW-Konverter Java Extended NEF Image File Format Editor (JENIFFER) wurde zu diesem Zweck entwickelt [2]. JENIFFER soll eine transparente und möglichst hohe Einflussnahme auf die Verarbeitungsschritte ermöglichen. In Anlehnung an JENIFFER wurde JENIFFER2 entwickelt [3]. Ziel von JENIFFER2 war das Auslesen und Verarbeiten von Digital Negative (DNG) Dateien, sowie dessen Steuerung und Begutachtung über eine moderne Benutzeroberfläche.

## **1.1. Motivation**

Der wesentliche Nachteil von JENIFFER war, dass es nur das RAW-Format Nikon Electronic Format (NEF) des Herstellers Nikon verarbeiten konnte. Außerdem konnte JENIFFER nur NEF Dateien älterer Nikon Kameramodelle verarbeiten. Dies schränkte die Anwendungsfälle von JENIFFER stark ein.

JENIFFER2 umgeht diesen Nachteil, indem es RAW-Dateien im nicht proprietären DNG Format einliest. Das Unternehmen Adobe stellt die Spezifikation des DNG Formats, sowie einen DNG Konverter frei zugänglich zur Verfügung. Der Adobe DNG Konverter kann eine Vielzahl an proprietären RAW-Formate in eine DNG Datei konvertieren. Damit kann JENIFFER2, über den Zwischenschritt des Adobe DNG Konverters, RAW-Dateien in proprietären Formaten verarbeiten.

In der initialen Implementierung von JENIFFER2 wurden allerdings nur wenige, grundlegende Algorithmen für den Verarbeitungsschritt des Demosaicings bereitgestellt. Durch das Bereitstellen von mehreren und aktuelleren Algorithmen kann eine höhere Anpassungsmöglichkeit und Qualität des Resultatbildes ermöglicht werden. Dazu wurden Algorithmen, welche auch in Open Source RAW-Konvertern wie RawTherapee, Darktable oder auch Matlab vorhanden sind, implementiert.

## 1.2. Ziele

JENIFFER2 mit Demosaicing-Algorithmen zu erweitern ist das primäre Ziel dieser Arbeit. Im Idealfall liefern diese Algorithmen bestmögliche Ergebnisse, in einer möglichst kurzen Berechnungszeit. Algorithmen mit verschiedenen Stärken und Anwendungsfällen, sowie Algorithmen, die in anderer Software mit RAW-Konverter-Funktionen enthalten sind, sind hierbei von besonderem Interesse. Der Nutzer soll über die vorhandene Dialogfunktion in JENIFFER2 den gewünschten Demosaicing-Algorithmus auswählen können.

Um dieses Ziel zu erreichen, ist zunächst eine Recherche über den Stand der aktuellen Demosaicing-Algorithmen notwendig. Welche Algorithmen in anderen Open-Source-Raw-Konvertern verwendet werden, ist hier ein guter Anhaltspunkt, da die dort verwendeten Algorithmen meist gut dokumentiert sind. Ein weiterer Kernpunkt ist die Untersuchung und Optimierung von JENIFFER2 bezüglich ihrer Demosaicing-Funktion. Neuere Demosaicing-Algorithmen können wesentlich längere Ausführungszeiten, als bisher implementierte Algorithmen besitzen und damit auch zusätzliche Hilfsfunktionen benötigen. Bei der Auswahl der Algorithmen mussten also auch die durch JENIFFER2 gesetzten Rahmenbedingungen beachtet werden.

Abschließend wird eine Analyse der implementierten Algorithmen durchgeführt. Dies erfolgt anhand von objektiven Parametern und subjektiven Vergleichen anhand markanter Stellen. Es wird ebenfalls der Frage nachgegangen, ob und wie groß das Verhältnis von Laufzeit und Qualität der Algorithmen im Vergleich ist. Zusätzlich wurden auch verschiedene Optimierungen und Bugfixes an JENIFFER2 vorgenommen, sowie kleine Usability-Funktionen hinzugefügt.

### **1.3. Überblick**

In Kapitel 2 werden Grundlagen erklärt, wie Kamerasensoren und das Bayer-Mosaik, aus denen die Notwendigkeit von Demosaicing-Algorithmen hervorgeht. Was Demosaicing-Algorithmen sind und welche Eigenschaften diese haben können, wird ebenfalls in diesem Kapitel behandelt. Außerdem werden RAW-Formate und das DNG-Format erläutert sowie die Verarbeitungsschritte, die JENIFFER2 nutzt, um eine DNG-Datei zu verarbeiten.

JENIFFER2, die Strukturierung des Programms, ihre Nutzeroberfläche und die Ausführungszeiten ihrer Komponente, werden zu Beginn von Kapitel 3 erläutert. Anschließend werden die Demosaicing-Algorithmen, die in JENIFFER2 implementiert sind, aufgelistet und jeweils in ihren wesentlichen Eigenschaften und ihrer Funktionsweise beschrieben. Zudem werden in diesem Kapitel weitere state-of-the-art Demosaicing-Algorithmen präsentiert und zusätzliche in dieser Arbeit vorgenommene Änderungen an JENIFFER2 aufgelistet.

Die Analyse und Auswertung der implementierten Demosaicing-Algorithmen erfolgt in Kapitel 4. Hier wird zunächst die Methodik erläutert, die genutzt wird, um die Algorithmen zu bewerten. Darauf folgt eine subjektive Analyse der durch die Algorithmen erzeugten Bilder. Zuletzt werden die Messwerte präsentiert und ausgewertet.

Neben einer Zusammenfassung dieser Arbeit findet sich in Kapitel 5 auch eine Erläuterung von möglichen zukünftigen Erweiterungen von JENIFFER2.

## 2. Grundlagen

Vor dem fertigen Erhalt eines Bildes sind mehrere Verarbeitungsschritte notwendig. Eine grobe Darstellung der Pipeline ist in Abbildung 1 veranschaulicht. Zuerst misst ein Bildsensor das einfallende Licht. Dieser erzeugt ein Signal, welches nach Bedarf verstärkt und anschließend in ein digitales Signal umgewandelt wird. Das Kamerasystem kann nun automatisch weitere Verarbeitungsschritte vornehmen und das Bild in einem geeigneten Bildformat gespeichert werden. Alternativ kann das Bild auch vor der weiteren kamerainternen Verarbeitung, in einem Rohdatenformat abgespeichert werden und auf einem Computer verlustfrei bearbeitet werden, bis die erwünschten Qualitätskriterien erreicht werden [4, S. 7]. Ein notwendiger Verarbeitungsschritt bei Bildsensoren mit Bayer-Mosaik ist das Demosaicing.

In den folgenden Unterkapiteln, werden die Komponente und Bildformate, die für das Demosaicing in JENIFFER2 relevant sind, erläutert. Auf den Demosaicing-Vorgang, die Kategorisierung seiner Algorithmen sowie mögliche dabei entstehende Artefakte wird ebenfalls eingegangen.

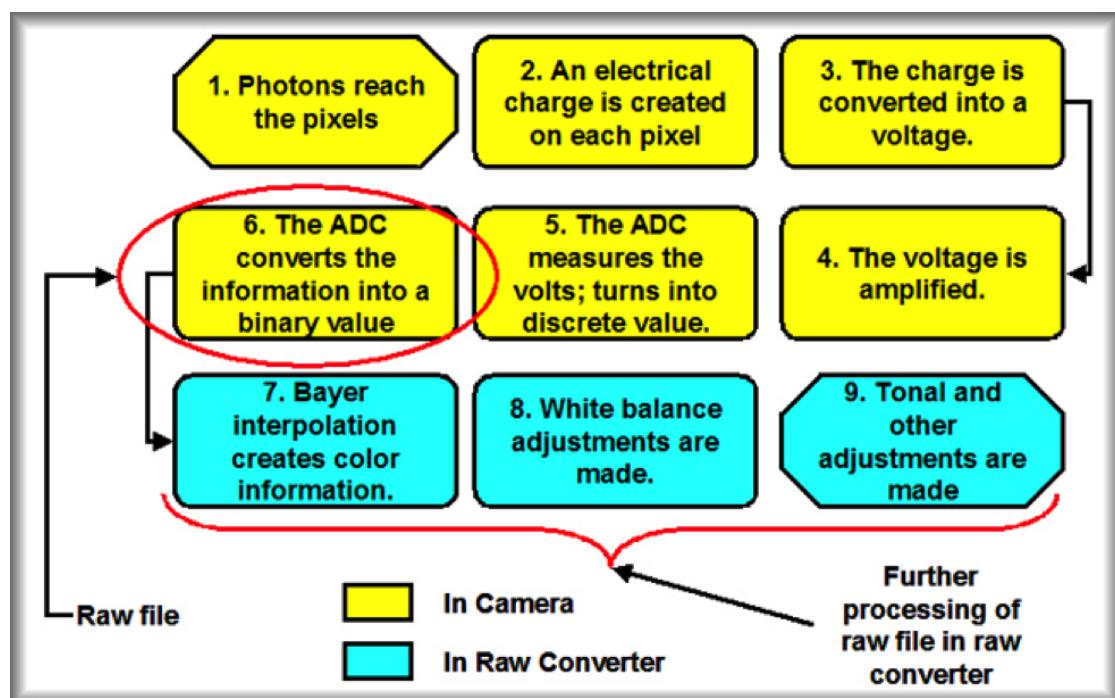


Abbildung 1: Allgemeine Verarbeitungspipeline eines Bildes [5]

## 2.1. Kamerasensoren

Nachdem Licht durch ein optisches System geleitet wird, trifft es auf die Oberfläche des Kamerasensors. Von diesem wird das eintreffende Signal gesampled. Die von dem Sensor gemessenen Werte werden anschließend von einem analog-zu-digital-Wandler in eine digitale Repräsentation gebracht, die nun weiterverarbeitet werden kann [4, S. 2–3]. Dazu werden lichtempfindliche Halbleiter genutzt, die das einfallende Licht, je nach Intensität, in ein entsprechendes elektrisches Signal umwandeln [6, S. 19].

Der Kamerasensor selbst besteht aus vielen solcher, meist quadratischen Einheiten, die als Pixel bezeichnet werden und in einem zweidimensionalen Array angeordnet sind [7, S. 7]. Diese Art von Sensor wird deswegen auch als Flächensor und ist die wichtigste Form für den Einsatz in Kameras [8, S. 80]. Fotodioden sind die lichtempfindliche Komponente auf einem Pixel. Die zwei relevantesten Typen von Kamerasensoren sind der Charge-Coupled Device (CCD) und Complementary Metal Oxide Semiconductor (CMOS). Ihr größter Unterschied ist die Art, mit der das Signal gemessen und weiterkommuniziert wird [4, S. 36].

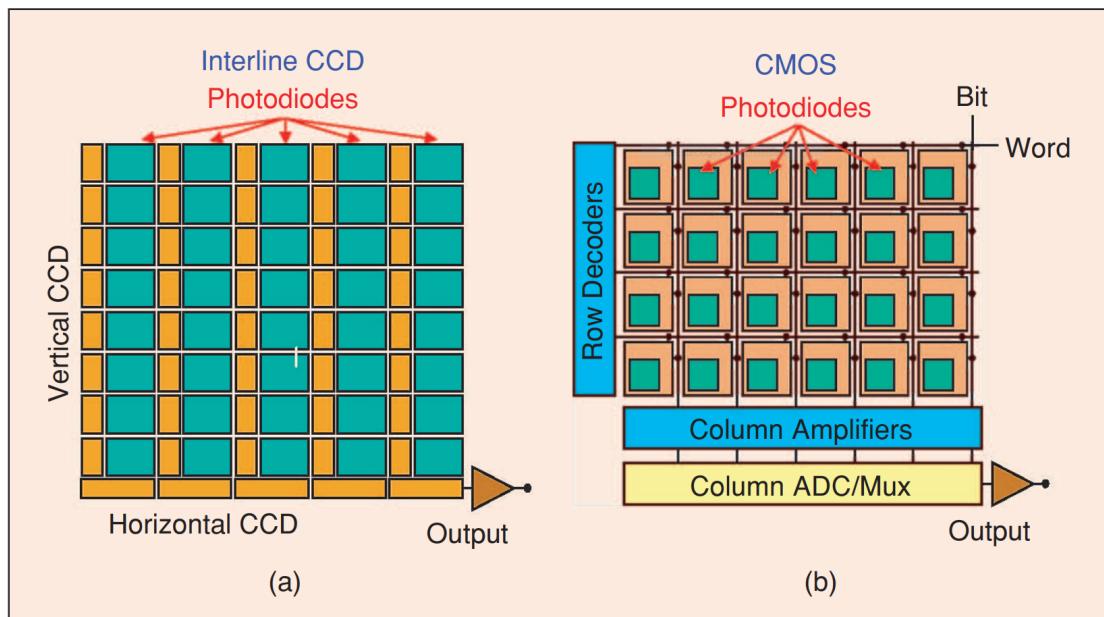


Abbildung 2: (a) Ausleearchitektur eines interline transfer CCD und (b) CMOS Bildsensors [7, S. 8]

Bei interline CCDs wird das Signal von der Fotodiode auf einen Impuls hin an direkt angrenzende, lichtgeschützte Register. Diese lichtgeschützten Register verhindern einen

"Schmiereffekt" der sonst während der weiteren Signalübertragung entstehend könnte. Aus den vertikalen Registern werden die Signale anschließend in ein horizontales Register übertragen, von wo sie zeilenweise ausgelesen werden und dabei alle denselben Ausgabeverstärker nutzen [6, S. 19]. CCDs sind im Vergleich zu CMOS die ältere Technologie und geben ein Signal wieder, das weniger von Rauschen belastet ist. Dafür besitzen sie keine integrierte digitale Logik, verbrauchen mehr Energie und haben eine niedrigere Bildrate, wegen des umständlichen Ausleseprozesses [8, 9, S. 85–89].

CMOS Sensoren sind die neuere Technologie. Sie haben keinen umständlichen Ausleseprozess und jedes Element kann einzeln adressiert werden, was zu einer schnelleren Auslesegeschwindigkeit führt. Sie verbrauchen nur etwa 10-33 % der Energie eines vergleichbaren CCD Sensor. Durch ihre integrierte Logik, können verschiedene Operationen wie Analog-Digital-Wandlung, Belichtungskontrolle, Weißabgleich oder Kontrastkorrektur direkt auf dem Chip vorgenommen werden. Dafür haben CMOS Sensoren eine höhere Rauschanfälligkeit und liefern damit Bilder mit einer niedrigeren Qualität [8, S. 87][6, S. 23][9].

## 2.2. Bayer-Filter

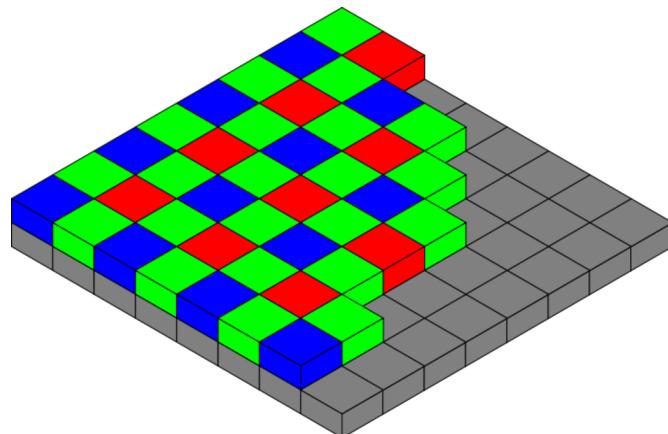


Abbildung 3: Beispiel eines Bayer-Filters [10]

Kamerasensoren sind grundsätzlich monochrome Sensoren. Das bedeutet, dass sie ein Signal für sämtliches Licht wiedergeben, dass sich innerhalb ihres sensitiven Spektrums befindet. Man kann damit also nur Graustufenbilder aufzeichnen. Um Farbbilder aufzuzeichnen, müssen zusätzliche Methoden verwendet werden [11, S. 62]. Es ist möglich, alle Farben des sichtbaren Spektrums aus drei Grundfarben zu erzeugen. Dieses Prinzip wird bei Farffilmen oder im Druck verwendet. Die additive Farbmischung des

RGB-Farbmodells wird in den meisten Fällen bei digitalen Kameratasensoren verwendet [6, S. 34]. Der in der Regel verwendete Lösungsansatz zur Farbtrennung ist der Bayer-Filter. Der Filter wurde nach seinem Erfinder Bruce E. Bayer benannt, der das Patent 1976 im Namen der Eastman Kodak Company einreichte [12].

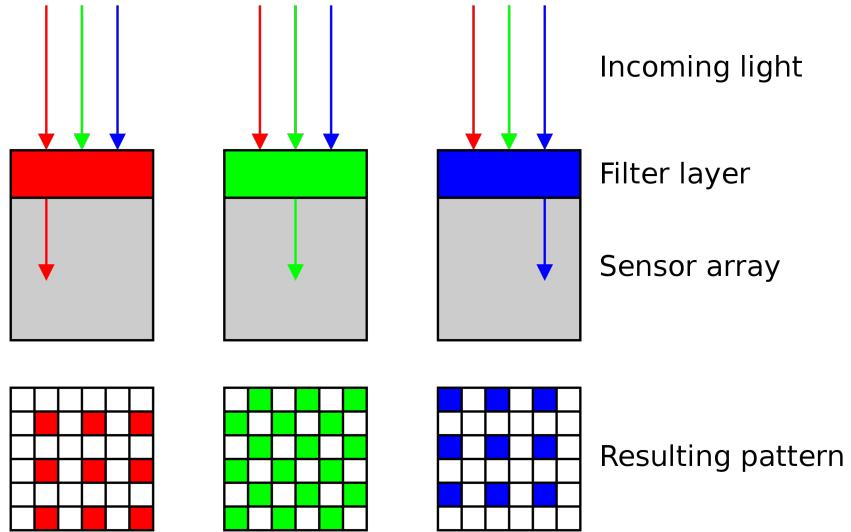


Abbildung 4: Profilansicht von einzelnen Pixeln in einem Bayer-Filter [10]

Alternativ wird der Bayer-Filter auch als Bayer-Gitter, Bayer-Mosaik, Bayer-Muster oder Bayer-Color Filter Array (CFA) bezeichnet. Jedem Pixel wird ein Farbfilter vorgeschaltet oder aufgedampft [6, 8, S. 38, 85]. Dabei werden die drei Farben Rot, Grün und Blau des RGB-Farbmodells verwendet. Nun produziert jeder Pixel nur noch einen Messwert, der sich proportional zu dem jeweiligen einfallenden Licht einer der drei Farben verhält. Dieser Messwert ist eigentlich auch ein Graustufenwert. Erst durch die Zuordnung eines Decoder-Rings in den Metadaten des Bildes wird klargestellt, welcher Pixel welche Farbe repräsentiert [13, S. 2]. Dasselbe Muster eines quadratischen Blocks aus vier Pixeln, wiederholt sich horizontal und vertikal über den gesamten Sensor, wie in Abbildung 3. Es gibt vier Permutationen des Bayer-Filters. Betrachtet man einen quadratischen Viererblock, so ergeben sich die Varianten RGGB, BGGR, GBRG und GRBG. In welcher dieser Varianten der Bayer-Filter angeordnet ist, wird ebenfalls in den Metadaten des Bildes gespeichert. In der Abbildung 4 wird dargestellt, wie jeder Filter nur den entsprechenden Teil des Lichtspektrums passieren lässt. Im unteren Teil der Abbildung 4 lässt sich die Verteilung der jeweiligen Farbfilter gut erkennen. 50 % der Sensoren sind mit einem grünen Filter belegt und jeweils 25 % mit einem roten oder

blauen Filter. Grund dafür ist, dass die Farbe Grün, die sich in der Mitte des Spektrums befindet, für die menschliche Wahrnehmung am relevantesten ist. Grün hat hier den größten Einfluss auf die Luminanzwahrnehmung [8, S. 85–86] und somit auch auf die Schärfewahrnehmung und Kontrastwahrnehmung [6, S. 39–40]. Rot und Blau bilden die Chrominanzinformation [12] [14, S. 76].

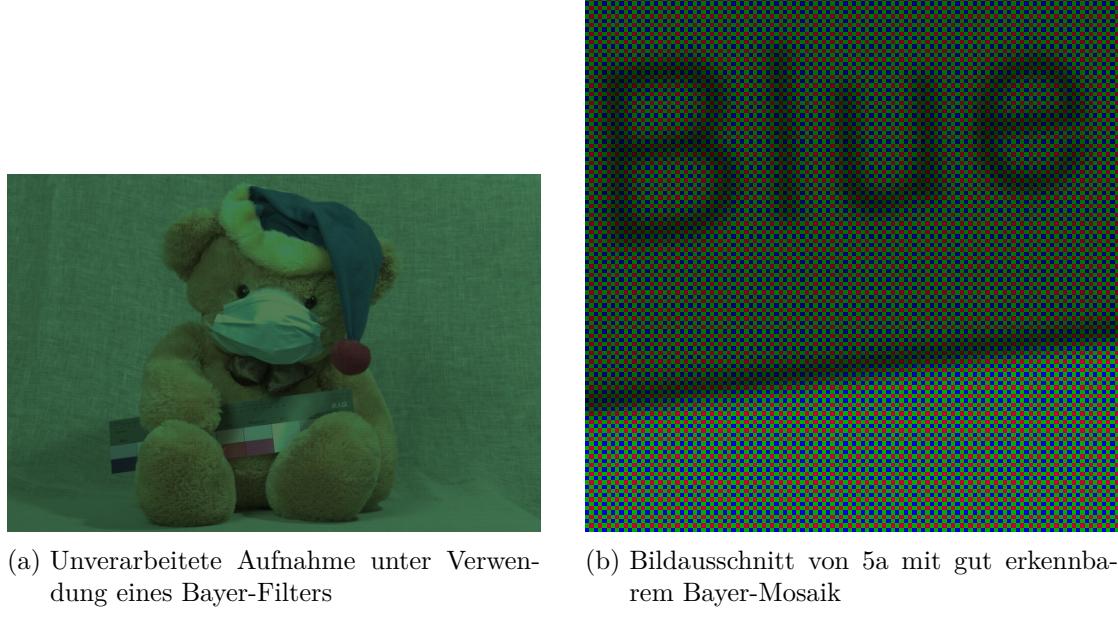


Abbildung 5: Unverarbeitete Aufnahme unter Verwendung eines Bayer-Filters

In Abbildung 5 kann man sehen, wie ein Bild ohne weitere Verarbeitung aussieht, wenn es mit einem Bayer-Filter aufgenommen wurde. In dem Bildausschnitt in Abbildung 5b, lässt sich das Bayer-Mosaik deutlich erkennen.

### 2.3. Demosaicing

Unter der Verwendung eines Sensors mit Bayer-Filter erhält man in jedem Pixel nur einen Wert für eine der drei Farben rot, grün oder blau. Für ein gewöhnliches Farbbild benötigen wir in jedem Bildpunkt einen Wert für jede der drei Farben. Die fehlenden Farbinformationen müssen unter Zuhilfenahme der Werte der anderen beiden Farbkanäle, geschätzt werden. Dieser Prozess der Interpolation wird als Demosaicing bezeichnet [15, S. 306]. Demosaicing ist ein notwendiger Schritt in der Bildverarbeitungs-Pipeline, bei Aufnahmen mit CFA.

Neben der Qualität des Resultatbildes können auch Speichernutzung, CPU-Nutzung und Energiebedarf des verwendeten Demosaicing-Algorithmus relevant sein [16, S. 153]. Somit sind viele verschiedene Algorithmen mit unterschiedlichen Anforderungen entstanden. Demosaicing-Algorithmen lassen sich in verschiedene Kategorien unterteilen [17, 16, 18]. Diese unterscheiden sich in verschiedenen Aspekten. Eine nähere Übersicht zu den Kategorien folgt in Kapitel 2.3.1.

Der Prozess des Demosaicings ist relativ anspruchsvoll, da bereits geringe Fehler in der Farbwertbestimmung zu unerwünschten Artefakten führen können, die für das menschliche Auge gut erkennbar sind [14, S. 440]. Näher wird in Kapitel 2.3.2 auf Artefakte eingegangen, die durch den Demosaicing-Vorgang entstanden sind.

Der simpelste Demosaicing-Algorithmus ist die Nearest Neighbour Interpolation [19, S. 2]. Dieser Algorithmus wird auch als Pixelwiederholung bezeichnet [8, S. 97]. Bei diesem Verfahren werden die fehlenden Farbwerte ermittelt, indem der entsprechende Farbwert von dem nächstgelegenen Pixel übernommen wird. Eine Variation des Algorithmus wäre eine 2x2 Pixelgruppe zu nehmen. Anschließend wird den jeweils anderen drei Pixeln der rote und blaue Farbwert zugewiesen. Die beiden Grünwerte können entweder horizontal oder vertikal weitergegeben werden. Diese Methode erzeugt signifikante Artefakte, besonders entlang von Kanten. Jedoch müssen keine Berechnungen durchgeführt werden, wodurch dieser Algorithmus in zeitkritischen Anwendungen nützlich sein kann [19, S. 2].

### 2.3.1. Kategorien von Demosaicing-Algorithmen

In den vergangenen Jahrzehnten wurden viele neue Demosaicing-Algorithmen entworfen, zum Teil wegen des stark wachsenden Marktes von digitalen Kameras. Demosaicing-Algorithmen lassen sich nach ihrer Funktionsweise grob in verschiedene Kategorien einteilen. Eine mögliche Einteilung ist in adaptive- und nicht-adaptive Algorithmen [17, S. 3].

- **Nicht-adaptive Algorithmen** zeichnen sich durch ein festgelegtes Muster aus, in welchem sie die Farbwerte für die Bestimmung der fehlenden Farbwerte nutzen. Es werden keinerlei dynamische Anpassungen anhand der Eigenschaften in dem vorliegenden Bild gemacht. Die Implementation dieser Algorithmen ist meist sehr simpel und sie benötigen nur wenig Rechenleistung. Beispiele für nicht-adaptive Algorithmen sind Nearest Neighbour, bilineare Interpolation, bilineare Median-Interpolation, bikubische Interpolation und Malvar-He-Cutler lineares Demosa-

cing und Directional Linear Minimum Mean-Square-Error (DLMMSE) Demosaicing [17, 18, S. 3–4, 505].

- **Adaptive Algorithmen** zeichnen sich durch Anpassung ihrer Rechenregeln für die Berechnung der fehlenden Farbwerte in jedem Pixel aus. Diese Anpassung kann zum Beispiel anhand von Schwellenwerten getroffen werden. Diese Kategorie von Algorithmen ist größtenteils komplexer und benötigt mehr Rechenleistung, erzielt aber in der Regel bessere Resultate bezüglich der Bildqualität. Beispiele für adaptive Algorithmen sind Hamilton-Adams Interpolation, Patterned Pixel Grouping (PPG) und Ratio Corrected Demosaicing (RCD) [17, 18, S. 3–4, 505].

Eine Erweiterung davon ist die Unterteilung in Farb-Differenz Demosaicing und residuale Demosaicing [18, S. 504–505]. Farb-Differenz Demosaicing beschreibt dabei die Methode, Unterschiede in direktionalen Schätzungen zu nutzen. Diese Kategorie lässt sich weiter in adaptive- und nicht-adaptive Algorithmen unterteilen, nach den oben genannten Definitionen. Dem gegenüber steht die neuere Kategorie an residualen Demosaicing-Algorithmen. Der erste residuale Demosaicing-Algorithmus wurde 2013 veröffentlicht [20]. Seitdem sind weitere residuale Algorithmen erschienen, die zusätzlich einen minimierten Laplacien nutzen [21, 22], iterative Schritte [23, 24], oder die bisherigen Aspekte adaptiv kombiniert [25]. Bei residualen Algorithmen wird durch guided Upsampling vorläufige Schätzwerte berechnet. Diese vorläufigen Schätzwerte werden mit den tatsächlich gemessenen Werten verglichen, um residuale Werte zu bestimmen. Die residuellen Werte werden anschließend interpoliert und zu den vorläufigen Schätzwerten addiert, um das interpolierte Farbband zu erhalten. Optional kann dieses in weiteren Iterationen genutzt werden, um es zu verfeinern, wie in Abbildung 6 dargestellt [25, S. 5].

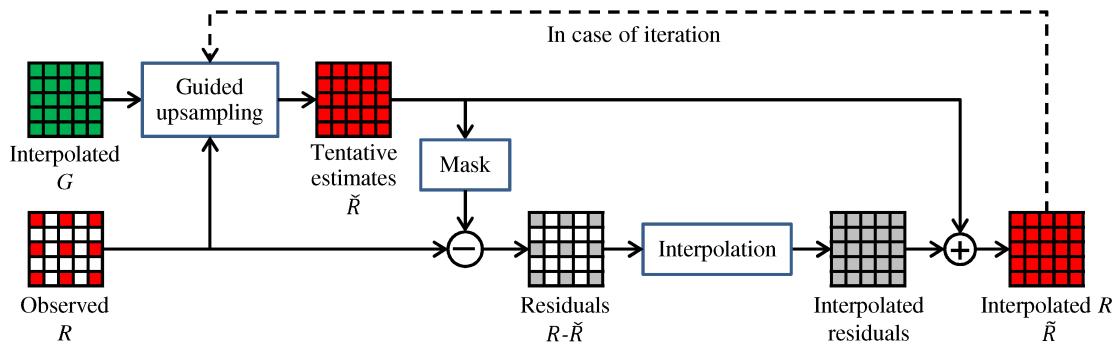


Abbildung 6: Framework der residualen Interpolation [25, S. 5]

Eine weitere mögliche Kategorisierung ist in Algorithmen in der räumlichen Domäne und Algorithmen in der Frequenz-Domäne. Algorithmen, die in der räumlichen Domäne operieren, nutzen zum Beispiel räumliche Korrelationen oder spektrale Korrelationen. Bei räumlicher Korrelation wird die Eigenschaft genutzt, dass benachbarte Pixel in einer homogenen Bildregion ähnliche Farbwerte haben. Bei spektraler Korrelation wird die Eigenschaft genutzt, dass die Werte der drei Farbkanäle in einer begrenzten Region ähnliche oder sich proportional verhaltende Werte haben. Algorithmen, die in der Frequenz-Domäne operieren, nutzen die a priori Annahme, dass das Bildsignal, wegen der Abtastrate der Farbkanäle, bandbegrenzt ist [16, S. 153–166]. Diese Algorithmen transformieren das Bild zunächst in die Frequenz-Domäne, in welcher das Bild als eine Kombination von Luminanz- und Chrominanzkomponenten dargestellt wird. Diese Komponente werden durch Filterung getrennt und zum Schluss wieder in die räumliche Domäne konvertiert. Vertreter des Frequenz-Demosaicing sind in folgenden Quellen zu finden: [26, 27, 28, 29].

Wavelet-basierte Algorithmen bilden eine weitere Kategorie. Diese nutzen eine sub-Band-Analyse. Subbänder werden erzeugt, indem das Ausgangssignal, durch einen Hochpass- und Tiefpassfilter in zwei Signale zerlegt wird. Die daraus resultierenden Signale können bis zur gewünschten Tiefe auf dieselbe Weise zerlegt werden. Dabei können bei einem Zerlegungsschritt mehr als zwei Subbänder entstehen, wenn entsprechende Filter gewählt werden. Diese Analyse kann auf initial interpolierte Bilder angewendet werden, um iterativ die hochfrequenten Komponenten der roten und blauen sub-Bänder in Abhängigkeit des grünen sub-Bands anzupassen [30, 31]. Alternativ können die iterativen Schritte ausgelassen werden, um die Ausführungszeit zu verbessern [32]. Die Wavelet-Analyse kann auch genutzt werden, um Gewichte für die horizontale und vertikale Interpolation zu bestimmen [33].

### 2.3.2. Demosaicing-Artefakte

Da nur ein Drittel der Farbwerte der aufgenommenen Szene gemessen wurden, fehlen strukturelle und spektrale Informationen. Deshalb kann es vorkommen, dass es bei der Rekonstruktion durch Demosaicing zu sichtbaren Fehlern im resultierenden Bild kommt. Solche Fehler werden als Artefakte bezeichnet. Die zwei häufigsten Formen von Artefakten, die durch den Demosaicing-Prozess entstehen, sind False-Color-Artefakte und Zipper-Artefakte. Sie treten häufig entlang scharfer Kanten, oder Regionen mit hohen Frequenzen auf. Im Gegensatz zu Aliasing- und Moiré-Artefakten, die aus anderen Gründen entstehen, treten Demosaicing-Artefakte meist nur in lokalen Regionen auf und sind

deshalb nicht so gut sichtbar, wenn das Bild in seiner natürlichen Auflösung dargestellt wird [19, S. 11] [18, S. 508] [14, S. 440–442] [4, S. 13–14] [34, S. 1–9].

- **False-Color-Artefakte** bilden plötzliche Änderungen der Farbtöne in einzelnen Pixeln, die in dem Originalbild nicht vorhanden sind. Diese Artefakte treten wegen Inkonsistenzen zwischen den drei Farbkanälen auf. Sie sind oft in Regionen mit feinen Bilddetails und entlang von Kanten erkennbar. False-Color-Artefakte können durch eine verbesserte Nutzung von spektralen Korrelationen zwischen Pixelwerten minimiert werden.
- **Zipper-Artefakte** bezeichnen abrupte oder unnatürliche Änderungen der Intensität, über mehrere benachbarte Pixel hinweg. Diese Artefakte werden hauptsächlich durch das inkorrekte Bilden von Mittelwerten mit benachbarten Farbwerten erzeugt. Sie treten meist als "an-aus"-Muster entlang von Kanten auf. Zipper-Artefakte können durch eine verbesserte Nutzung von räumlichen Korrelationen minimiert werden.



Abbildung 7: Beispiele von Demosaicing-Artefakten. Im linken Bild treten Zipper-Artefakte entlang der Motorhaube auf. Im rechten Bild treten False-Color-Artefakte am Außenspiegel und entlang der Windschutzscheibe auf. [19, S. 12]

## 2.4. RAW-Format

In häufig verwendeten Bildformaten wie Joint Photographic Experts Group (JPEG) oder Tagged Image File Format (TIFF) wurden verschiedene Verarbeitungsschritte und Komprimierungen vorgenommen. Meist werden diese Verarbeitungsschritte direkt beim

Schießen des Bildes von der Kamera vorgenommen. Teilweise kann man bei Kameras einstellen, welche Parameter für die Verarbeitung genutzt werden sollen. Trotzdem ist man nun an die Interpretation der Kamera von der jeweiligen Szene gebunden [13, S. 3]. Diese Verarbeitungen sind auch nicht reversibel. Die so entstandenen Bilder haben den Nachteil eines Qualitätsverlusts und einer eingeschränkten Weiterbearbeitung [8, S. 115]. Dieses Problem soll das RAW-Format lösen.

Eine RAW-Datei enthält unkomprimierte und unverarbeitete bis minimal verarbeitete Bilddaten, die durch den Kamerasensor aufgezeichnet wurden [35, S. 10]. Im Gegensatz zu den 8 Bit pro Komponente bei verlustbehafteten, regulären JPEG-Dateien, wird bei RAW-Dateien jeder Farbkanal mit 12 Bit oder mehr gespeichert [4, S. 369]. RAW-Dateien sind eine Art von Rasterdatei, aber selbst keine Bilddatei. Das bedeutet, dass sie erst mit einem RAW-Converter wie JENIFFER2 weiterverarbeitet werden müssen, bevor sie betrachtet und in ein anderes Bildformat wie JPEG exportiert werden kann [36]. Ein damit zwangsläufig immer notwendiger Verarbeitungsschritt ist das Demosaicing. Eine RAW-Datei beinhaltet nicht nur die aufgezeichneten Sensordaten. In der Regel besteht eine RAW-Datei aus den folgenden drei Komponenten [37, S. 7]:

- **Kameradaten**

Diese Metadaten sind üblicherweise als Exif Tags [38] hinterlegt. Diese beinhalten Informationen wie Kameramodell, Belichtungszeit oder Blendendetails. Die meisten dieser Einträge können nicht verändert werden.

- **Bilddaten**

Diese Daten beinhalten zum Beispiel Informationen über den Farbraum, Weißabgleich, Sättigung, Kontrast und angewandte Schärfung. Sie beeinflussen, wie das Bild in einem RAW-Converter verarbeitet wird. Diese Einträge können verändert werden.

- **Rohdaten**

Die nicht-interpolierten Daten selbst, die direkt vom Sensor bereitgestellt wurden.

Die wesentlichen Vorteile von RAW-Dateien gegenüber anderen Bildformaten werden im Folgenden aufgelistet [37, S. 11].

- **künstlerische Kontrolle**

RAW-Dateien bieten den Vorteil größtmöglicher künstlerischer Kontrolle über die Anpassung und Weiterverarbeitung der Bilder. Beispielhaft können Anpassungen wie Weißabgleich, Ton-Mapping, Rauschreduzierung, Schärfung oder Gammakorrektur erfolgen [35, 13, S. 10, 3].

- **verlustfreie und reversible Verarbeitung**

Durch Prozesse wie verlustbehaftete Komprimierung und einer geringeren Farbtiefe gehen irreversibel Informationen verloren. Damit kann die komplette Palette an Farben genutzt werden. Auch nachträgliche Optimierungen, wie die im vorherigen Punkt aufgeführten, lassen sich oft nicht ohne Verluste ungeschehen machen.

- **Beibehalt der Originaldatei**

Bei Anpassungen einer RAW-Datei, mit Optionen aus dem ersten Punkt, wird diese selbst nicht verändert. Anpassungen werden nur in den Metadaten festgehalten.

RAW-Dateien bringen allerdings auch mehrere Nachteile mit sich, die im Folgenden aufgelistet werden.

- **Dateigröße**

Da an RAW-Dateien keine Reduzierungen, wie Kompression oder Zuschneiden, vorgenommen wurde und jeder Farbpunkt mit einer hohen Anzahl an Bit kodiert wurde, sind RAW-Dateien wesentlich größer als JPEG- oder TIFF-Dateien. Damit kann nur ein Bruchteil an Dateien im RAW-Format auf einem Speichermedium gespeichert werden.

- **nicht standardisiertes Format**

Das RAW-Format ist kein einheitlicher Standard. Es existieren über 500 verschiedene Arten von RAW-Formaten [39]. Diese können sich alle in ihrem Aufbau und notwendigen Verarbeitungsschritten unterscheiden. Die meisten Formate sind auch proprietär. Das bedeutet, dass die Spezifizierung der Formate nicht öffentlich zugänglich ist. Hersteller eines Formates haben deswegen auch häufig ihre eigene Bildverarbeitungs-Software, die sie Nutzern bereitstellen. Ist diese Software nicht mehr zugänglich, lässt sich ihr proprietäres Format nicht mehr verarbeiten [4, 35, S. 371, 10]. Dies macht proprietäre RAW-Dateien auch ungeeignet für eine Archivierung. Ein solches proprietäres RAW-Format ist das NEF von Nikon.

- **aufwendiger Workflow**

Da zusätzliche Verarbeitungsschritte manuell durchgeführt werden müssen und nicht direkt fertig vorliegen, ist damit auch ein höherer Zeitaufwand verbunden. Zusätzlich ist auch ein größeres Verständnis der möglichen Verarbeitungsoptionen notwendig, um eine zielführende Optimierung durchzuführen [37, 8, S. 12, 120].

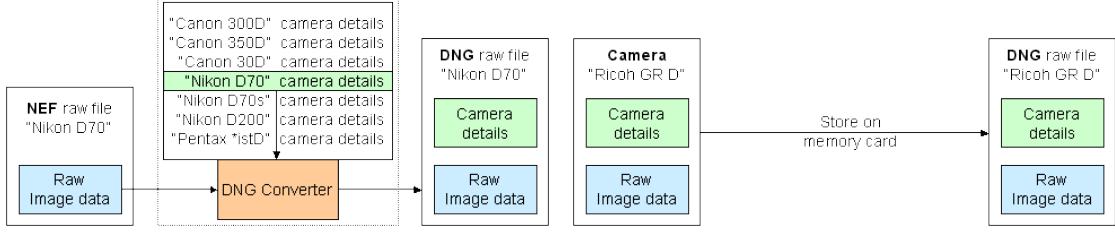
## 2.5. DNG-Format

Das DNG-Format ist ein RAW-Format, welches 2004 von Adobe veröffentlicht wurde [39]. DNG ist ein nicht-proprietaryer Standard, der das Problem der fehlenden Standardisierung bei RAW-Formaten (vgl. Kapitel 2.4) lösen soll. Zum Zeitpunkt dieser Arbeit ist die aktuelle Version der DNG-Spezifikation 1.6.0.0 [35].

Der herstellerübergreifende Umgang mit RAW-Dateien wurde durch die Unterschiede im Aufbau stark erschwert. Das DNG-Format soll hier verschiedene Vorteile mit sich bringen, wie eine herstellerübergreifende Archivierung, Verarbeitung und Support [40]. Im Folgenden werden die wesentlichen Eigenschaften aufgelistet, die das DNG-Format auszeichnen und damit die genannten Probleme lösen soll.

- **In sich geschlossen**

In sich geschlossen bedeutet in diesem Zusammenhang, dass ein Minimum an zusätzlichen Informationen, außer der DNG-Datei selbst, benötigt wird, um das Bild auszulesen. Damit Programme die proprietäre RAW-Formate verarbeiten können, benötigen diese Programme Informationen über das spezifische Kameramodell, welches die entsprechende Datei erstellt hat. Dies hat auch zur Folge, dass diese Programme Updates erhalten müssen, wenn neue Kameramodelle veröffentlicht werden [35, S. 10]. Dieses Problem wird von dem DNG-Standard gelöst, indem diese kameraspezifischen Informationen in die DNG-Datei eingebettet. Diese kameraspezifischen Informationen werden in Form von öffentlich dokumentierten Tags bei der Erstellung einer DNG-Datei hinterlegt. Neben den üblichen Bestandteilen Kamera-daten, Bilddaten und Rohdaten, die eine RAW-Datei beinhaltet (vgl. Kapitel 2.4), hat eine DNG-Datei also auch die zusätzliche Komponente der kameraspezifischen Informationen [41]. Dadurch kann ein RAW-Converter eine DNG-Datei auslesen und verarbeiten, ohne zusätzliche Informationen über die Kamera zu benötigen, mit der die Aufnahme gemacht wurde.



(a) Einbettung von Kameradetails durch einen DNG-Converter      (b) Einbettung von Kameradetails direkt in der Kamera

Abbildung 8: Methoden zum Erstellen von DNG-Dateien und Einbettung von Kameradetails [41]

- **Archivierbar**

Kamerahersteller beenden manchmal den Support eines proprietären RAW-Formats, nachdem die Produktion eines Kameramodells eingestellt wird. Ohne den kontinuierlichen Support des Herstellers kann es dazu kommen, dass der Zugriff auf Bilder in einem proprietären RAW-Format nicht mehr möglich ist und es somit unwiderruflich verloren ist. Dieses Problem soll durch die öffentliche Dokumentation des DNG-Formats vermieden werden [35, S. 11]. Durch seine Eigenschaft viele verschiedene Kameramodelle zu unterstützen erleichtert das DNG-Format die Archivierung, da nur ein Format benötigt wird. Dass das Format in sich geschlossen ist, wie im vorherigen Punkt erläutert, ist ein weiterer Faktor, der die Archivierung erleichtert. Ein weiterer Vorteil ist die Integration von Metadaten, in Form von eXtensible Metadata Platform (XMP), einem ebenfalls öffentlich dokumentierten Format. Diese Metadaten erlauben Einträge wie den Ersteller des Bildes, Ort der Erstellung oder das Einbetten von Editierungsschritten, wie zum Beispiel dem Zuschneiden des Bildes [40]. Zusätzlich ist es auch möglich, falls die DNG-Datei durch Konvertierung aus einer RAW-Datei in einem anderen Format erstellt wurde, diese originale RAW-Datei selbst in die DNG-Datei einzubetten [35, S. 40–41]. Die originale RAW-Datei kann so wieder aus der DNG-Datei extrahiert werden.

- **TIFF kompatibel**

DNG ist eine Erweiterung des TIFF 6.0 Formats und ist kompatibel mit dem TIFF-EP Standard. Es ist möglich, dass eine DNG-Datei gleichzeitig die Spezifikation des DNG-Formats und des TIFF-EP Standards erfüllt [35, S. 11]. In dem TIFF-EP Standard wurden zum Beispiel verschiedene Tags, Möglichkeiten zur Integration von Vorschaubildern und zur Kompression definiert [42].

### **2.5.1. Adobe DNG Converter**

Eine DNG-Datei kann direkt als natives Format kameraintern bei der Aufnahme erstellt werden, oder durch einen DNG Converter [41]. Adobe stellt dafür den kostenlosen Adobe DNG Converter bereit [43]. Dieser kann verschiedene proprietäre RAW-Formate in das DNG-Format überführen und läuft auf Windows und macOS Systemen. Für Linux-Systeme gibt es aktuell keine Version. Außerdem kann man auch DNG-Dateien mit diesem Converter konfigurieren, um eine neue DNG-Datei mit anderer Kompression oder einem Vorschaubild zu erzeugen. In Abbildung 9c sind die verschiedenen Dialogfenster des Adobe DNG Converters zu sehen. Von besonderer Relevanz sind die Voreinstellungen in Abbildung 9b. Dort können folgende Anpassungen vorgenommen werden:

- **Kompatibilität**

Hier kann die früheste Camera RAW-Version spezifiziert werden, mit der die erstellte DNG-Datei kompatibel sein soll. Zusätzliche Einstellungen lassen sich in dem Dialog für benutzerdefinierte Kompatibilität in Abbildung 9c vornehmen. Diese zusätzlichen Optionen beinhalten die konkrete Version der DNG-Spezifikation, eine Ausgabe in linear RAW (und damit unwiderrufliche Interpolation des Bayer-Mosaiks) und eine verlustfreie Kompression.

- **Vorschau/Schnell ladende Dateien**

Durch diese Einstellung kann ein JPEG-Bild in gewünschter Größe eingebettet werden. Bei zusätzlicher Auswahl der schnell ladenden Dateien werden weitere Vorschaubilder in verschiedenen Auflösungen eingebettet. Diese Optionen erhöhen die Gesamtgröße der erstellten DNG-Datei geringfügig.

- **Komprimierung/Bildgröße**

Mit dieser Option kann eine verlustbehaftete Komprimierung der Rohdaten vorgenommen werden. Optional kann auch die Anzahl der Pixel reduziert werden. Bei diesem Menüpunkt ist zu beachten, dass dadurch nicht nur die Qualität sinkt, sondern auch das Bayer-Mosaik interpoliert wird.

- **Raw-Originaldatei einbetten**

Hier kann die zu konvertierende originale RAW-Datei in die erstellte DNG-Datei eingebettet werden. Bei Bedarf kann die Originaldatei, über die Funktion Extrahieren in Abbildung 9a wieder gewonnen werden. Durch das Einbetten der originalen RAW-Datei ist die resultierende DNG-Datei entsprechend größer.



Abbildung 9: Adobe DNG Converter und seine Einstellungsmasken [43]

### 2.5.2. DNG Konvertierungspipeline

Wie bereits in Kapitel 1 erwähnt, müssen die vom Sensor gemessenen Rohdaten erst verschiedene Verarbeitungsschritte durchlaufen, bevor man eine betrachtbare Bilddatei erhält. Es gibt zwar einige notwendige Verarbeitungsschritte, je nach Hersteller können diese verschieden angeordnet oder zusammengefasst werden und beliebige weitere Verarbeitungsschritte hinzugefügt werden [44, S. 35]. In der DNG-Spezifikation sind mehrere Verarbeitungsschritte detailliert beschrieben [35]. Eine allgemeingültige Pipeline wird aber auch hier nicht definiert. Eine mögliche Pipeline zur Verarbeitung einer DNG-Datei, wurde bei der Erstellung von JENIFFER2 konzipiert [3, S. 31]. Dabei hängt von den Tags der DNG-Datei ab, welche Verarbeitungsschritte tatsächlich durchlaufen werden. In Abbildung 10 wurden Verarbeitungsschritte, die teilweise optional sind, rot markiert [3, S. 44]. Im Folgenden werden die Verarbeitungsschritte kurz erläutert.

- **Rohdatenzuordnung**

Zur einheitlichen Verarbeitung der Pixelinformationen, muss jeder Pixelwert zwischen 0,0 und 1,0 liegen. Der Wert 0,0 steht dabei für kein Licht und 1,0 für die maximale Sättigung [35, S. 84]. Die Rohdaten liegen in der Regel aber nicht in diesem Wertebereich und sind oft auch nicht linear transformiert (zum Beispiel zur effizienten Speicherung) [45, S. 7]. Bei dem DNG-Format sind die vier Schritte Linearisierung, Dunkelstromkompensation, Skalierung und Clipping definiert, um sogenannte lineare Referenzwerte zu erhalten [35, S. 84].

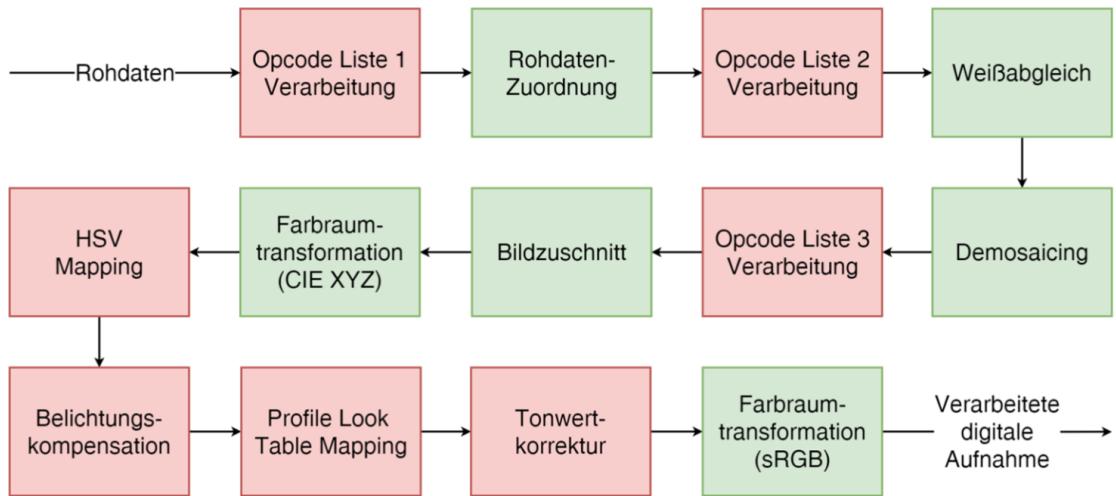


Abbildung 10: Verarbeitungsschritte einer DNG-Datei in JENIFFER2 [3, S. 31]

### – Linearisierung

Dieser Schritt muss nur durchgeführt werden, falls eine Linearisierungstabelle hinterlegt ist. Diese Tabelle ist eine Lookup-Tabelle, mit der die gespeicherten Werte auf lineare Werte abgebildet werden können. Die Linearisierungstabelle bietet den Vorteil, die Rohdaten mit einer höheren Kompressionsrate zu speichern [35, S. 25].

### – Dunkelstromkompensation

Selbst wenn kein Licht auf den Kamerasensor trifft, zeichnet dieser ein Signal auf, das als Dunkelstrom bezeichnet wird. Dieser kann durch Wärme erzeugte Elektronen im Sensorsubstrat entstehen. Dieser Dunkelstrom muss von den Rohdaten subtrahiert werden. Eine mögliche Strategie, zur Bestimmung des Wertes des Dunkelstroms, ist die Abdunklung der Randpixel des Sensors. Diese abgedunkelten Pixel messen theoretisch nur den Dunkelstrom [44, S. 38].

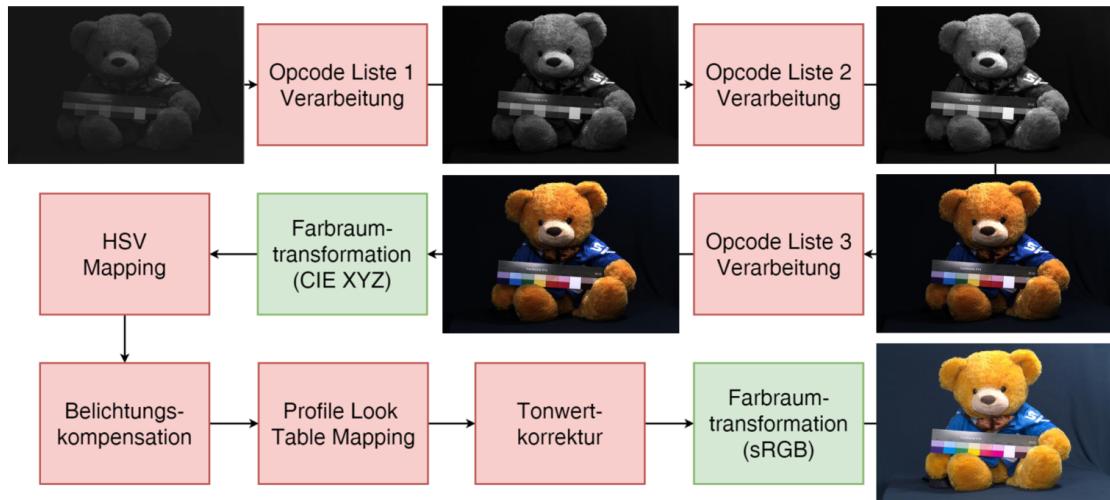


Abbildung 11: Verarbeitungsschritte einer DNG-Datei in JENIFFER2 mit Beispielbildern nach jedem Schritt [3, S. 31]

### – Skalierung

Bei der Skalierung werden die Rohdaten auf den Wertebereich zwischen 0 und 1 skaliert. Dazu wird der DNG-Tag WhiteLevel genutzt, der den maximalen Wert der Sättigung angibt [35, S. 84].

### – Clipping

Wegen Rauschen kann es vorkommen, dass der Sensor Werte liefert, die über dem theoretischen Maximalwert, oder unter dem Minimalwert liegen. Diese Werte werden in diesem Schritt auf den Wertebereich zwischen 0 und 1 geclipped.

### • Weißabgleich

Ein Objekt kann vermeintlich verschiedene Farben haben, wenn es mit verschiedenen Lichtquellen beleuchtet wird. Dies liegt an der Farbtemperatur der verschiedenen Lichtquellen. Eine Kamera zeichnet diese von der Lichtquelle beeinflusste Szene auf. Um diese Farbdifferenz zu kompensieren, wird der Weißabgleich durchgeführt [46, S. 460]. Oft ermittelt die Kamera automatisch die notwendigen Werte zum Weißabgleich. Üblicherweise werden dabei die roten und blauen Farbkanäle mit jeweils einem Wert multipliziert.

- **Demosaicing**

Das Demosaicing wird dem ausgewählten Algorithmus entsprechend wie in Kapitel 2.3 durchgeführt. Bei einer DNG-Datei ist zu beachten, dass der Tag Photometric-Interpretation den Wert 32803 haben muss. Nur dann enthält die Datei Rohdaten, die auf einem CFA basieren [35, S. 20–21]. Hat dieser Tag den Wert 34892, handelt es sich um eine LinearRAW-Datei und alle Farbkomponente sind bereits in jedem Pixel vorhanden. Dies kann zum Beispiel passieren, wenn das Bild direkt kameraintern demosaiced wurde, oder bei der Aufnahme direkt alle drei Farben aufgezeichnet wurden. Das AppleProRAW ist ein Beispiel dafür. Dieses Format kann von iPhones, mit einer iOS-Version von 14.3 oder neuer, erzeugt. Tatsächlich handelt es sich um DNG-Dateien die linearRAW verwenden. Der Demosaicing-Vorgang wurde bereits kameraintern durchgeführt [47]. Das DNG-Format erlaubt es auch andere Mosaiken als das Bayer Mosaik zu definieren. Es können andere Farben als die üblichen drei Rot, Grün und Blau definiert werden oder eine andere Anzahl an verschiedenen Farben oder ein anderes Muster, in welchem die Farben angeordnet sind. Bei einem anderen Muster oder einer anderen Anzahl an Farben, kann es sein, dass speziell dafür andere Demosaicing-Algorithmen genutzt werden müssen, wie bei dem X-Trans Sensor von Fujifilm [48, 49].

- **Bildzuschnitt**

Wie im Abschnitt zur Dunkelstromkompensation beschrieben, kann es vorkommen, dass Randpixel manchmal abgedunkelt werden. Diese sollen auf dem Resultatbild nicht zu sehen sein. Mit dem Tag ActiveArea werden die nicht maskierten Pixel angegeben [35, S. 12]. Es kann auch vorkommen, dass innerhalb des aktiven Bereichs zusätzliche Pixelwerte entlang der Ränder gespeichert werden. Diese werden genutzt, um Artefakte zu verhindern, die durch einen Mangel an Werten bei dem Demosaicing entstehen könnten. Diese zusätzlichen Pixel werden ebenfalls weggescchnitten [35, S. 29].

- **Farbraumtransformation**

Nach dem Demosaicing-Schritt erhält man zwar ein herkömmlich betrachtbares Bild, aber dessen Farben sind nicht die, die ein Monitor erwartet. Um dies zu korrigieren, müssen lineare Transformationen auf jeden Pixelwert angewendet werden [45, S. 3]. Zu Beginn befinden sich die Rohdaten in einem kameraspezifischen Farbraum. Kameraspezifische Farbräume sind meist für eine effiziente Speicherung, oder

ein Berechnungsmedium entworfen [44, S. 39]. Die DNG-Spezifikation definiert die notwendigen Tags, um die Werte von dem kameraspezifischen Farbraum in den CIE XYZ Farbraum zu überführen [35, S. 85–88]. Der CIE XYZ Farbraum kann zwar sämtliche Farben der menschlichen Wahrnehmung darstellen, eignet sich aber nicht für die Darstellung auf Monitoren. Für die Darstellung auf Monitoren eignet sich der sRGB Farbraum. Dieser Farbraum hat zwar einen wesentlich geringeren Umfang an Farben [8, S. 168], ist aber der am weitesten verbreitete Farbraum und wird von den meisten Computersystemen unterstützt [44, S. 40]. Abschließend muss noch eine Gammakorrektur vorgenommen werden. Dies ist notwendig, um die Bilddaten an die Nicht-Linearität des Monitors anzupassen [4, S. 78]. Der korrigierte Wert lässt sich für den sRGB Farbraum durch die Potenzierung mit dem Kehrwert von 2,2 berechnen [45, S. 11].

### 3. JENIFFER2

JENIFFER sollte, im Gegensatz zu kommerziellen Programmen, dem Nutzer offenlegen, welchen Demosaicing-Algorithmus er benutzt und auch einen transparenten Einblick in die sonstigen Verarbeitungsschritte ermöglichen [2]. Diese Iteration von JENIFFER konnte allerdings nur RAW-Dateien im NEF-Format verarbeiten. Diesen Nachteil umgeht JENIFFER2, indem es Dateien im DNG-Format einliest. Damit kann JENIFFER2 theoretisch auch alle RAW-Formate einlesen, die der kostenlose Adobe DNG Converter umwandeln kann. JENIFFER2 wurde mit Java erstellt und benötigt aktuell eine Laufzeitumgebung der Version 17 oder höher. Die erste Version von JENIFFER2 wurde von Eugen Ljavin im Jahr 2020 erstellt [3].

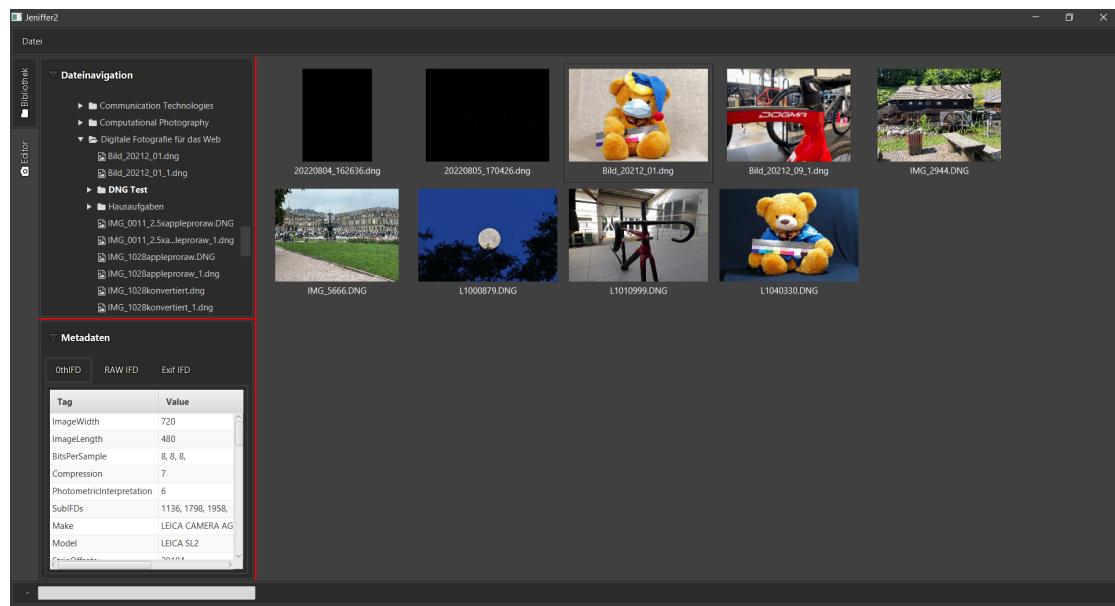


Abbildung 12: Bibliothek Tab von JENIFFER2

Die grafische Benutzeroberfläche wurde mit JavaFX realisiert. Die Benutzeroberfläche wurde modular aufgebaut und besteht aus zwei Tabs. Der erste Tab heißt Bibliothek und wird direkt beim Start von JENIFFER2 angezeigt, wie in Abbildung 12 dargestellt. Über das Fenster Dateinavigation kann im Dateisystem navigiert werden. Wenn hier ein Ordner ausgewählt wird, dann werden im zentralen rechten Fenster alle DNG-Dateien mit ihren Vorschaubildern und Dateinamen dargestellt. Hier kann man durch einen einzelnen Klick auf das gewünschte Bild, dessen Metadaten in dem Fenster unten rechts anzeigen oder durch Doppelklick den Dialog zur Auswahl des Demosaicing-Algorithmus aufrufen. Nach der Verarbeitung des Bildes wird dieses in dem Editor Tab zentral an-

gezeigt, wie in Abbildung 13 dargestellt. Durch das Gedrückthalten der Maustaste und gleichzeitiges Bewegen der Maus, kann das Bild verschoben werden, um bestimmte Bildausschnitte zu betrachten. Über die Schaltfläche darüber kann man in das Bild hinein- und herauszoomen. Alternativ kann man auch das Mausrad dafür verwenden. Wird der Mauszeiger über dem Bild im Hauptfenster gehalten, dann werden im Fenster Bilddaten oben links die entsprechend x und y Koordinaten, sowie die auf den Bereich 0–1 normierten RGB-Werte angezeigt.

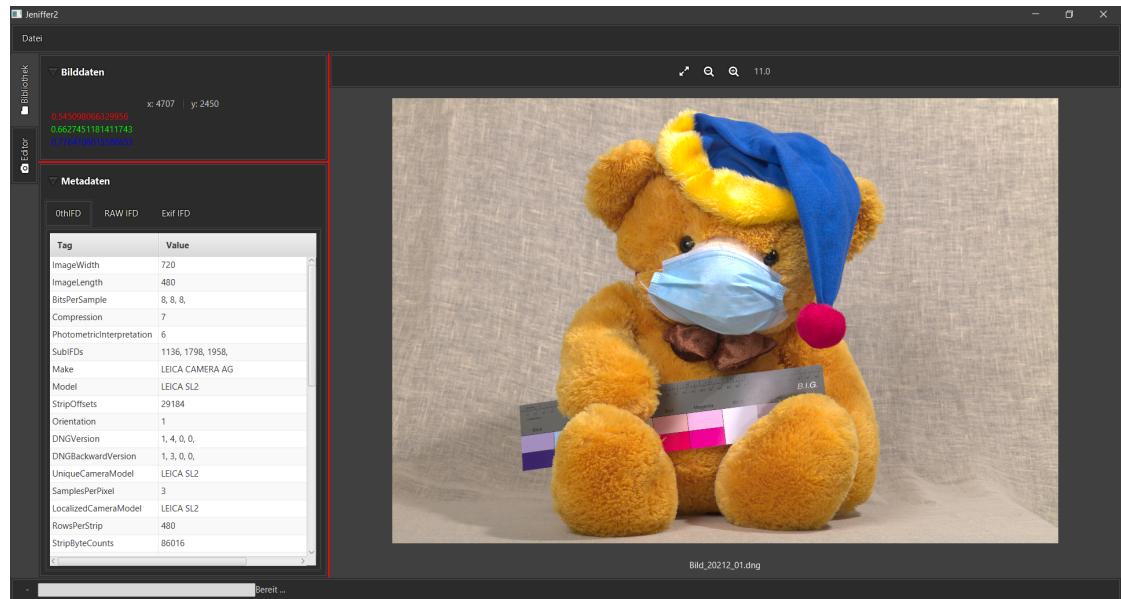


Abbildung 13: Editor Tab von JENIFFER2

Die Softwarestruktur von JENIFFER2 ist in drei Komponenten unterteilt. Die erste Komponente ist der DNG-Reader. Dieser ist für das Auslesen und Bereitstellen der Rohdaten, DNG Tags und des Vorschaubildes zuständig. Der DNG-Processor ist die zweite Komponente. Er beinhaltet die Bestandteile der Verarbeitungspipeline, wie Weißabgleich, Demosaicing oder Farbraumtransformation, die in Abbildung 10 dargestellt werden. Die dritte Komponente ist die Benutzeroberfläche von JENIFFER2. Diese stellt die Steuerung, wie sie eben beschrieben wurde, für den Nutzer bereit.

### 3.1. Verarbeitungsschritte und bisherige Ausführungszeit

Der DNG-Prozessor von JENIFFER2 bildet die Verarbeitungspipeline aus Abbildung 10. Diese ist wiederum in die vier Komponenten Preprocessor, Demosaicingprocessor, Otherprocessor und Postprocessor unterteilt. Der Preprocessor führt die Schritte Rohdatenzu-

ordnung und Weißabgleich durch. Der Demosaicingprocessor führt den entsprechenden Demosaicing-Algorithmus durch. Der Otherprocessor beinhaltet den Bildzuschnitt. Der Postprocessor führt die Schritte Farbraumtransformation, vom Kamerafarbraum in den CIE XYZ Farbraum, HSV-Mapping, Profile Look Table Mapping, Tonwertkorrektur und die Farbraumtransformation nach sRGB aus, wobei die Farbraumtransformation nach sRGB auch die entsprechende Gammakorrektur beinhaltet. Außerdem werden die Verarbeitungsschritte HSV-Mapping, Profile Look Table Mapping und Tonwertkorrektur nur bei Bedarf ausgeführt.

	Rohdaten-zuordnung [ms]	Weiß-abgleich [ms]	Demo-saicing [ms]	Farbraum-transformation (davon Gamma-korrektur) [ms]	Summe [ms]
<b>Pixel-wiederholung</b>	1375,3	525,3	761,7	4876,1 (4238,7)	7538,4
<b>Bilineare Interpolation</b>	1407,1	525	1117,7	4755,4 (4154,6)	7805,2
<b>Median Interpolation</b>	1373	521,8	2309,6	5153,2 (4549,8)	9357,6
<b>Bikubische Interpolation</b>	1361,9	520,4	3288,5	5137 (4525,9)	10307,8
<b>Mittelwert</b>	1381,1	523,1	-	4980,4 (4367,3)	-

Abbildung 14: Ausführungszeiten der einzelnen Verarbeitungsschritte von JENIFFER2 vor dieser Masterarbeit [3, S. 74]

In der Masterarbeit von Eugen Ljavin wurde die Ausführungszeit der wesentlichen Verarbeitungsschritte gemessen, wie in Abbildung 14 dargestellt [3, S. 74]. Für jeden Demosaicing-Algorithmus wurden zehn Durchläufe gemessen und davon der Mittelwert angegeben. Die dafür verwendete RAW-Datei hatte eine Auflösung von 8424 x 5632 Pixeln, was einer Gesamtzahl an 47.443.968 Pixeln entspricht. Der Bildzuschnitt wurde bewusst nicht angegeben, da dessen Methode nur einmal aufgerufen wird. Zu diesem Zeitpunkt hat die Farbraumtransformation, inklusive Gammakorrektur, die meiste Zeit in der Verarbeitungs-Pipeline beansprucht. Während die Pixelwiederholung (Nearest Neighbour) nur 761,7 Millisekunden und damit rund zehn Prozent der gesamten Verarbeitungsdauer beansprucht, benötigt die bikubische Interpolation bereits viermal länger, mit einer Verarbeitungszeit von 3288,5 Millisekunden und einer Beanspruchung von rund 32 Prozent der gesamten Verarbeitungsdauer. Dieser Trend könnte ohne weitere Optimierungen schnell zu sehr hohen Rechenzeiten führen, wenn komplexere Demosaicing-Algorithmen implementiert werden.

### **3.2. Implementierte Demosaicing-Algorithmen**

In den folgenden Unterkapiteln wird auf die verschiedenen Demosaicing-Algorithmen eingegangen, die nach dieser Masterarbeit alle in JENIFFER2 implementiert sind. Es wird auf bereits bekannte Stärken, Schwächen oder Besonderheiten der Algorithmen eingegangen. Die Algorithmen "keine Interpolation", Nearest Neighbour, bilineare Interpolation mit Durchschnitt, bilineare Interpolation mit Median und bikubische Interpolation, wurden bereits in der Masterarbeit von Eugen Ljavin [3] implementiert. In JENIFFER2 wird die Nearest Border Strategie angewendet, falls ein Algorithmus Pixelwerte anfordert, die außerhalb des Bildes liegen. Diese Strategie ist sehr schnell, erzeugt aber oft schlechte Ergebnisse. Diese sind aber in der Regel nicht sichtbar, da die äußeren Pixel nach dem Bildzuschnitt entfernt werden. Alle implementierten Algorithmen fallen in die Kategorien des Farb-Differenz Demosaicing und in die Kategorie des Demosaicing in der räumlichen Domäne (vgl. Kapitel 2.3.1).

Ein weiterer wichtiger Punkt ist die Ausführungsreihenfolge in welcher ein Demosaicing-Algorithmus durchgeführt wird. Vor dieser Masterarbeit wurde jeder Pixel der Reihe nach betrachtet und für diesen Pixel der Demosaicing-Algorithmus durchgeführt. Somit wurden an jedem Pixel immer direkt beide fehlende Farbwerte berechnet und es wurden immer nur Farbwerte für die Berechnung genutzt, die schon in den Rohdaten vorhanden waren. Viele Demosaicing-Algorithmen berechnen die fehlenden Werte aber nicht in dieser Reihenfolge. Oft werden zunächst die fehlenden grünen Farbwerte an roten und blauen Pixeln berechnet. Grund dafür ist, dass doppelt so viele grüne Farbwerte gemessen wurden wie rote oder blaue Werte. Die so berechneten grünen Farbwerte fließen auch oft in die Berechnung der fehlenden roten und blauen Farbwerte ein. Ohne entsprechende Zwischenspeicherung müssten die notwendigen grünen Farbwerte so für jedes Pixel neu berechnet werden. In dieser Masterarbeit wurden zwei Anpassungen vorgenommen, um die beschriebenen Probleme zu beheben. Der Loop, in welchem über das Raster an Pixelwerten iteriert wird, wurde aufgeteilt, sodass in dem ersten Loop nur fehlende grüne Farbwerte berechnet werden und in dem zweiten Loop nur die fehlenden roten und blauen Farbwerte. Für den Algorithmus RCD wurde der zweite Loop nochmal unterteilt, da dieser Algorithmus zunächst die fehlenden roten und blauen Farbwerte an roten und blauen Pixeln berechnet und diese berechneten Werte nutzt, um die fehlenden roten und blauen Farbwerte an grünen Pixeln zu berechnen. Außerdem wurden entsprechende Funktionen implementiert, um die berechneten Werte zu speichern, die in einem späteren Loop zur Berechnung benötigt werden.

### **3.2.1. Keine Interpolation**

Bei dieser Option wird kein Demosaicing durchgeführt. Die Pixelwerte werden nur durchgereicht. Diese Option kann genutzt werden, wenn der Demosaicing-Schritt übersprungen werden soll. Dies kann bei der Fehlerdiagnose oder Analyse des Bildes sinnvoll sein. Theoretisch könnte diese Option genutzt werden, um ein CFA-Muster auf das Ausgabebild anzuwenden, wenn eine zukünftige Version von JENIFFER2 lineare RAW-Dateien einlesen kann.

### **3.2.2. Nearest Neighbour**

Der Nearest Neighbour (NN) Algorithmus übernimmt die nahegelegenen Farbwerte, die dem aktuellen Pixel fehlen. Es handelt sich hier um einen nicht-adaptiven Algorithmus. Dies ist der theoretisch schnellste Algorithmus, da er keine Berechnungen durchführt. Es sind aber starke Artefakte zu erwarten. Zusätzlich hat dieser Algorithmus den Vorteil, dass er sich sehr einfach implementieren lässt [8, S. 97].

### **3.2.3. Bilineare Interpolation mit Mittelwert**

Die bilineare Interpolation ist einer der bekanntesten Demosaicing-Algorithmen. Er ist ein nicht-adaptiver Algorithmus. An roten und blauen Pixeln werden die vier angrenzenden Farbwerte der fehlenden Farben genommen und daraus der Mittelwert berechnet. An grünen Pixeln wird der Mittelwert aus den jeweils zwei angrenzenden roten und blauen Pixeln gebildet. Dieses Verfahren ist auch recht simpel, leicht zu implementieren und schnell. Jedoch sind die Resultate in der Regel auch nicht sehr gut. Oft führt es zu verschwommenen Kanten.

### **3.2.4. Bilineare Interpolation mit Median**

Dies ist eine Variation der bilinearen Interpolation. Es handelt sich hier ebenfalls um einen nicht-adaptiven Algorithmus. Anstatt den Mittelwert aus den vier angrenzenden Farbwerten an roten und blauen Pixeln zu berechnen, wird hier der Medianwert übernommen. Dieser Algorithmus hat dieselben Vorteile wie die bilineare Interpolation mit Mittelwert, sollte aber zusätzlich den Vorteil von Medianwerten haben: Robustheit gegenüber ausreißenden Werten. Die Sortierung von Werten, die notwendig ist, um den Median zu bestimmen, kann allerdings zu einer erhöhten Laufzeit führen.

### **3.2.5. Bikubische Interpolation**

Die bikubische Interpolation war das komplexeste Verfahren, das vor dieser Masterarbeit in JENIFFER2 implementiert war. Die bikubische Interpolation ist ein nicht-adaptiver Algorithmus. An grünen Pixeln werden für jeden fehlenden Farbwert jeweils acht umliegende Pixel betrachtet, um die zwei fehlenden Farbwerte zu berechnen. An roten und blauen Pixeln werden jeweils 16 Pixel betrachtet, um die zwei fehlenden Farbwerte zu berechnen. Sie ist in der Regel akkurater als der Nearest Neighbour Algorithmus oder die bilineare Interpolation, aber auch rechenintensiver [50].

### **3.2.6. Hamilton-Adams Interpolation**

Die Hamilton-Adams Interpolation (HA) wurde, von John Hamilton und James Adams, für die Eastman Kodak Company, als Patent im Jahr 1997 angemeldet [51]. Der Algorithmus hat hauptsächlich den Nachteil von False-Color-Artefakten, jedoch nicht so sehr wie zum Beispiel die bilineare Interpolation.

Die fehlenden grünen Farbwerte an roten und blauen Pixeln werden in Abhängigkeit der horizontalen und vertikalen Gradienten horizontal, vertikal oder aus einer Kombination davon interpoliert. Nachdem die fehlenden grünen Farbwerte berechnet wurden, werden diese zusätzlich zu den gemessenen roten und blauen Farbwerten verwendet, um die fehlenden roten und blauen Farbwerte zu berechnen.

Es handelt sich hierbei um einen adaptiven Algorithmus. Im Gegensatz zu den vorherigen Algorithmen in dieser Liste, nutzt die Hamilton-Adams Interpolation zusätzlich zur räumlichen Korrelation auch die spektrale Korrelation benachbarter Farbkanäle. Dieser Algorithmus wird in vielen anderen Arbeiten als Referenz verwendet, um die Qualität des Resultatbildes zu vergleichen [52, 53]. Er ist relativ schnell und einfach zu implementieren. Dieser Algorithmus wird auch als Teil anderer Demosaicing-Algorithmen verwendet, um eine Initialisierung für weitere Verarbeitungsschritte zu berechnen [54, 55, S. 91, 1197]. Der Algorithmus wurde entsprechend der Implementierung auf der folgenden Seite, für JENIFFER2 adaptiert: [https://www.ipol.im/pub/art/2011/g\\_gapd/](https://www.ipol.im/pub/art/2011/g_gapd/).

### **3.2.7. Malvar-He-Cutler lineare Interpolation**

Die Malvar-He-Cutler (MHC) Interpolation wurde, von Henrique Malvar, Li-wei He und Ross Cutler, im Jahr 2004 veröffentlicht [56]. Die Intention der Autoren war, einen linearen Algorithmus zu entwerfen, der eine bestmögliche Performance erbringt. Dieser

Algorithmus erzeugt in der Regel weniger False-Color-Artefakte als die bilineare Interpolation.

MHC baut auf der bilinearen Interpolation auf und erweitert diese auf einen 5x5 Filter. Der Farbwert an der aktuellen Position wird mit denselben Farbwerten innerhalb des 5x5 Filters verglichen, um scharfe Änderungen der Luminanz zu erkennen, wie es bei Kanten der Fall ist und das Resultat der bilinearen Interpolation mit dieser Luminanzänderung zu korrigieren.

Dieser Algorithmus nutzt also zusätzlich zur bilinearen Interpolation spektrale Korrelation. Dies ist ein nicht-adaptiver Algorithmus. Der Algorithmus ist relativ schnell und erzeugt meist bessere Ergebnisse als die bilineare Interpolation. Dieser Algorithmus wird von MATLAB als Demosaicing-Algorithmus genutzt [57]. MHC wird auch als Standardalgorithmus in manchen Anwendungen von NASA verwendet, wie dem Curiosity Rover [58, S. 433]. Der Algorithmus wurde entsprechend der Implementierung auf der folgenden Seite, für JENIFFER2 adaptiert: [https://www.ipol.im/pub/art/2011/g\\_mhcd/](https://www.ipol.im/pub/art/2011/g_mhcd/).

### 3.2.8. Chuan-Kai Lin Patterned Pixel Grouping

Der Patterned Pixel Grouping (PPG) Algorithmus wurde 2003 von Chuan-Kai Lin entwickelt und veröffentlicht [59]. Bei dem Entwurf hat sich der Autor an anderen Algorithmen wie Variable Number of Gradients (VNG) und Linear Color Correction I orientiert. Sein Ziel war es, einen Algorithmus zu entwerfen, der künstliche Szenen mit großen Flächen an einheitlichen Farben, die von klaren Grenzen unterteilt werden, gut verarbeiten kann. PPG funktioniert dafür nicht gut bei detaillierten Texturen, oder starken Änderungen der Luminanz in Bereichen, die kleiner als 3x3 Pixel groß sind.

Der Algorithmus berechnet zunächst die fehlenden grünen Farbwerte an roten und blauen Pixeln. Dafür werden Gradienten in vier Richtungen berechnet und entlang des kleinsten Gradienten interpoliert. Die so berechneten grünen Farbwerte werden für die folgenden beiden Schritte verwendet. Im zweiten Schritt werden die roten und blauen Farbwerte an grünen Pixeln berechnet. Dafür wird die Funktion hue\_transit verwendet. Diese soll erkennen, ob in den betrachteten drei Pixeln ein Farbverlauf, oder eine Kante vorliegt. Dementsprechend wird interpoliert. Im letzten Schritt werden die fehlenden roten und blauen Farbwerte an roten und blauen Pixeln berechnet. Dazu werden die beiden Gradienten entlang der zwei diagonalen Richtungen berechnet und anschließend mit der hue\_transit Funktion entlang des kleineren Gradienten interpoliert.

Diese Anpassungen der Rechenregeln machen PPG zu einem adaptiven Algorithmus. Er ist zwar etwas komplexer als die bisher gelisteten Algorithmen, dafür aber noch recht schnell. Er wurde in anderen Open-Source Programmen wie DCRAW [60] genutzt und war der default Algorithmus in Darktable, bis zur Version 3.6, die im Juli 2021 veröffentlicht wurde. Der Algorithmus wurde entsprechend der Implementierungen auf den folgenden Seiten für JENIFFER2 adaptiert:

<https://github.com/Beep6581/RawTherapee/blob/3ad786745cf11fade334ff41aad0573dfcd7c04e/rtengine/dcraw.c>,  
[https://github.com/mashaka/Image-Recognition-and-Processing/blob/master/task1/Task\\_1.ipynb](https://github.com/mashaka/Image-Recognition-and-Processing/blob/master/task1/Task_1.ipynb),  
[https://www.informatik.hu-berlin.de/de/forschung/gebiete/viscom/teaching/media/cphoto10/cphoto10\\_03.pdf](https://www.informatik.hu-berlin.de/de/forschung/gebiete/viscom/teaching/media/cphoto10/cphoto10_03.pdf).

### 3.2.9. Zhang-Wu Directional Linear Minimum Mean-Square Error

Der Directional Linear Minimum Mean-Square-Error (DLMMSE) Algorithmus, der von Lei Zhang und Xiaolin Wu entwickelt wurde, wurde 2005 veröffentlicht [61]. Die Autoren haben versucht, einen Algorithmus zu entwerfen, der die Korrelation der Differenzsignale zwischen dem grünen und roten oder blauen Farbkanal nutzt. Dieses Differenzsignal korreliert nach ihrer Observation nicht mit den Fehlern, die durch gradientengeleitete Verfahren entstehen. Diese beiden Beobachtungen sollen genutzt werden, um in Kombination mit einer linearen minimum mean square-error estimation die Beschränkung der zu niedrigen Abtastrate zu überwinden. Der Algorithmus liefert gute Ergebnisse bei Bildern mit hohem Rauschen oder hohen ISO-Werten. Er kann ebenfalls nützlich sein, um Aliasing-Artefakte zu minimieren, die bei anderen Algorithmen verstärkt auftreten können. Dieser Algorithmus erzeugt dafür häufiger Artefakte entlang von Kanten mit großen Chrominanzunterschieden.

Der Algorithmus besteht im Wesentlichen aus drei Schritten. In den ersten beiden Schritten werden die fehlenden grünen Farbwerte an roten und blauen Stellen berechnet. Im dritten Schritt werden die gemessenen und berechneten Werte genutzt, um die fehlenden roten und blauen Farbwerte zu berechnen. Im ersten Schritt werden die primären Differenzsignale mittels Laplace-Interpolation ermittelt. Das Differenzsignal wird anschließend mit einem Gauß-Filter durch LMMSE entrauscht. Im nächsten Schritt werden das horizontale und vertikale Differenzsignal aus dem ersten Schritt fusioniert. Dabei wird erneut eine Schätzung mit LMMSE genutzt. Nun sind alle grünen Farbwerte bekannt. Im letzten Schritt werden die primären Differenzsignale entlang der beiden Diagonalen

berechnet und anschließend von dem grünen Farbwert des Pixels subtrahiert, um den roten oder blauen Farbwert zu erhalten.

Dieser Algorithmus ist nicht-adaptiv, da er immer dieselben Berechnungen anwendet, um einen bestimmten Farbwert zu bestimmen. DLMMSE ist wesentlich komplexer als PPG und benötigt wesentlich mehr Rechenzeit. Dieser Algorithmus wird in Open-Source Programmen wie RawTherapee [62] und Darktable [48] genutzt. In beiden Programmen wird der Algorithmus als LMMSE bezeichnet. Zusätzlich wird dieser Algorithmus auch in vielen anderen Arbeiten als Referenz genutzt [63, 25]. In JENIFFER2 wurden zwei Variationen implementiert. Die beiden Varianten Directional Linear Minimum Mean-Square-Error mit Code Estimator (DLMMSE-CodeEst) und Directional Linear Minimum Mean-Square-Error mit Paper Estimator (DLMMSE-PaperEst) unterscheiden sich darin, wie die lokalen Signalmittelwerte berechnet werden. Grund dafür ist, dass dieser Wert in dem Paper [61] und der originalen Implementierung der Autoren (<http://www4.comp.polyu.edu.hk/~cslzhang/code/dlmmse.m>) unterschiedlich berechnet wird. Eine weitere Implementierung des Algorithmus ist auf folgender Seite zu finden: [https://www.ipol.im/pub/art/2011/g\\_zwld/](https://www.ipol.im/pub/art/2011/g_zwld/).

### 3.2.10. Luis Sanz Rodríguez Ratio Corrected Demosaicing

Der Ratio Corrected Demosaicing (RCD) Algorithmus wurde von Luis Sanz Rodriguez entwickelt und 2017 veröffentlicht. Der Autor hat versucht einen Algorithmus zu entwerfen, der Fehler der Farbkorrektur glätten soll, die in anderen Algorithmen auftreten. Der Algorithmus liefert gute Resultate bei der Rekonstruktion von hochfrequenten Bereichen, wie feinen Details, Kanten, runden Kanten oder Sternen. Bei Aufnahmen mit niedrigen ISO-Werten und Astro-Photographie erhält man außerdem glattere Resultate und Rauschen ist weniger körnig. Er ist allerdings nicht so gut in Bereichen mit niedrigen Frequenzen mit Rauschen.

Der Algorithmus besteht aus vier wesentlichen Schritten. Auch in diesem Algorithmus werden zunächst die fehlenden grünen Farbwerte berechnet, um diese später für die Berechnung der fehlenden roten und blauen Farbwerte zu nutzen. Zusätzlich nutzt dieser Algorithmus die fehlenden roten und blauen Farbwerte an roten und blauen Pixeln, um die fehlenden roten und blauen Farbwerte an grünen Pixeln zu berechnen. Diese Eigenschaft wurde bei der Implementierung in JENIFFER2 genutzt, damit eine bessere Ausführungszeit erreicht wird und Werte nicht mehrfach berechnet werden müssen. Im ersten Schritt werden der horizontale und vertikale Gradient berechnet und zu ei-

nem Wert kombiniert. Dasselbe wird für die vier angrenzenden Pixel gemacht, wobei die vier resultierenden Werte zu einem Nachbarschaftswert kombiniert werden. Anschließend wird aus dem zentralen Wert und dem Nachbarschaftswert der passendere ausgewählt. Im zweiten Schritt wird ein Gaußfilter verwendet, um die Unterschiede in der Farbin-tensität, zwischen dem roten und blauen Kanal zu glätten. Im dritten Schritt werden die fehlenden grünen Farbwerte an roten und blauen Pixeln berechnet. Dafür wird ei-ne neue ratio-correted Interpolation vorgeschlagen, die die gefilterten Werte aus dem vorherigen Schritt nutzt. Diese neue Methode soll verhindern, dass an harten Kanten False-Color-Artefakte auftreten. Diese Werte werden anschließend mit den Ergebnissen aus dem ersten Schritt kombiniert, was als Resultat den grünen Farbwert hat. Der vierte Schritt verläuft zunächst identisch wie der erste Schritt, mit dem Unterschied, dass die Gradienten entlang der beiden Diagonalen berechnet werden. Danach werden ähnlich wie in Schritt drei die fehlenden roten und blauen Farbwerte an roten und blauen Pixeln berechnet. Dafür werden auch die grünen Farbwerte aus dem dritten Schritt genutzt. Zuletzt werden wieder ähnlich wie im dritten Schritt die roten und blauen Farbwerte an grünen Pixeln berechnet, wobei hier noch die eben berechneten roten und blauen Farbwerte genutzt werden.

Dieser Algorithmus wählt adaptiv die besseren Gradientenkombination zur weiteren Berechnung aus. Dies ist ebenfalls wesentlich komplexer als die anderen Demosaicing-Algorithmen in JENIFFER2 und hat auch eine sehr hohe Ausführungszeit. Der Algo-rithmus wird in den Open-Source Programmen RawTherapee [62] und Darktable [48] genutzt. Nachdem die Implementierung in Darktable soweit optimiert wurde, dass sie ähnliche Ausführungszeiten wie PPG hat, ist RCD seit Version 3.6 im Juli 2021 der default Algorithmus. Der Algorithmus wurde entsprechend der Implementierung auf der folgenden Seite für JENIFFER2 adaptiert: [https://github.com/LuisSR/RCD-Demosaicng/blob/master/rcd\\_demosaicing.c](https://github.com/LuisSR/RCD-Demosaicng/blob/master/rcd_demosaicing.c).

### **3.2.11. DLMMSE Grünkanalinterpolation mit RCD Rot- und Blaukanalinterpolation**

Für JENIFFER2 wurde ein neuer Algorithmus erstellt und getestet. Dieser nutzt für die Bestimmung des fehlenden Grünwertes dasselbe Verfahren wie DLMMSE und für die Bestimmung der fehlenden Rot- und Blauwerte dasselbe Verfahren wie RCD. Die Idee dahinter war, das im Vergleich zu DLMMSE komplexere Verfahren für die Rot- und Blauwertinterpolation zu testen. Dadurch könnte die Anfälligkeit von fehlerhaften

Interpolationen entlang von Kanten minimiert werden. Falls dieses Verfahren bessere Ergebnisse als das Verfahren von DLMMSE erzielt, könnte dieser Algorithmus so verbessert werden.

### 3.3. Stand der Demosaicing-Algorithmen

Die aktuell kompetitivsten Vertreter der interpolationsbasierten Demosaicing-Algorithmen kommen aus der Kategorie der residualen Interpolation (vgl. Kapitel 2.3.1) [64, 53, S. 221, 234] und werden nur von Convolutional Neural Network (CNN) Algorithmen übertroffen.

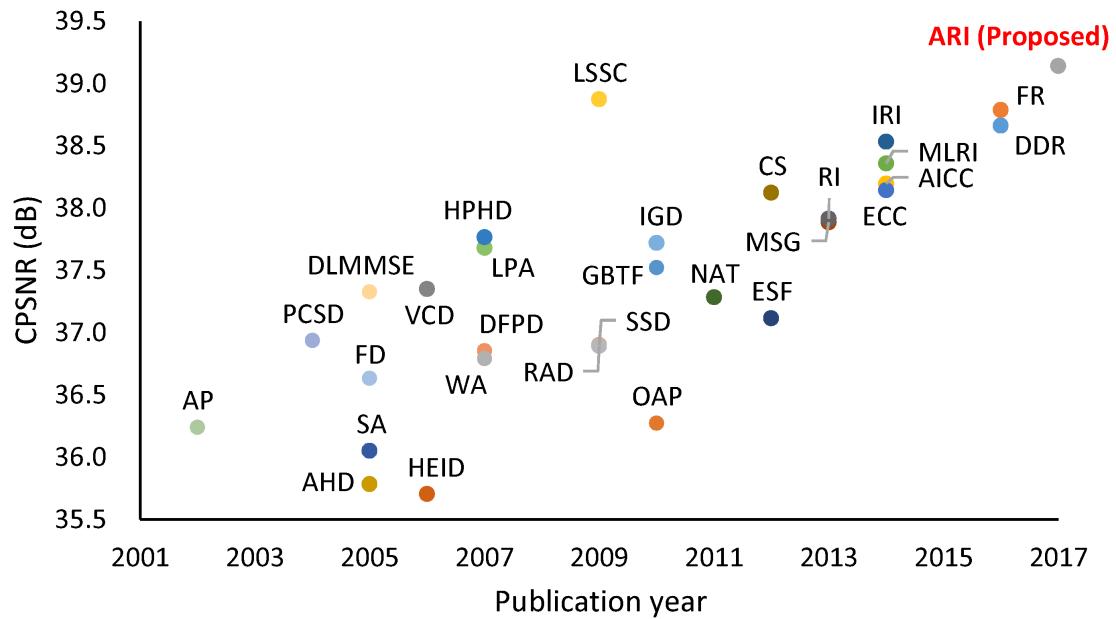


Abbildung 15: Color peak signal-to-noise ratio Performanz verschiedener representativer Demosaicing-Algorithmen auf dem IMAX und Kodak Dataset (vgl. Kapitel 4.2) [25, S. 2]

Der Vertreter der residualen Interpolation, der die besten Ergebnisse erzielt, ist Adaptive Residual Interpolation (ARI). Dieser Algorithmus verbessert vorherige residuale Interpolationsalgorithmen, indem er adaptiv zwei Algorithmen kombiniert, die ebenfalls residual operieren und für jedes Pixel eine geeignete Iterationsanzahl wählt [25, S. 1]. In Abbildung 15 ist ein Vergleich zu anderen repräsentativen Demosaicing-Algorithmen zu sehen, darunter auch DLMMSE. In visuellen Analyse erzeugt ARI wenige Zipper-Artefakte und kann Rauschen effektiv entfernen. Jedoch sind auch bei diesem Algorithmus Artefakte

in hochfrequenten Bereichen zu beobachten. Da ARI außerordentlich gute Ergebnisse erzielt und der Source-Code offen zugänglich ist, wäre dieser Algorithmus ein Kandidat für eine Implementierung in JENIFFER2. Diese ist aber aktuell aus zwei Gründen problematisch. Der erste Grund ist die Komplexität des Algorithmus. Das Demosaicing-Modul von JENIFFER2 müsste umfangreich angepasst werden, um die notwendigen Funktionalitäten bereitzustellen. Der zweite und gravierendere Punkt ist die Laufzeit. Während der Implementierung der neuen Demosaicing-Algorithmen im Rahmen dieser Masterarbeit benötigte RCD etwa 30 Sekunden für ein Bild mit fast 47 Millionen Pixel, allerdings mit Parallelisierung. ARI benötigte in der Implementierung seiner Autoren etwa 31 Sekunden ohne Parallelisierung, jedoch für Bilder mit circa 400.000 Pixel [25, S. 14–15]. Hauptgrund für die lange Laufzeit ist die iterative Komponente des Algorithmus. Die Autoren geben jedoch an, dass es ihr Ziel ist, eine bezüglich Laufzeit optimierte Variante zu erstellen [25, S. 14].

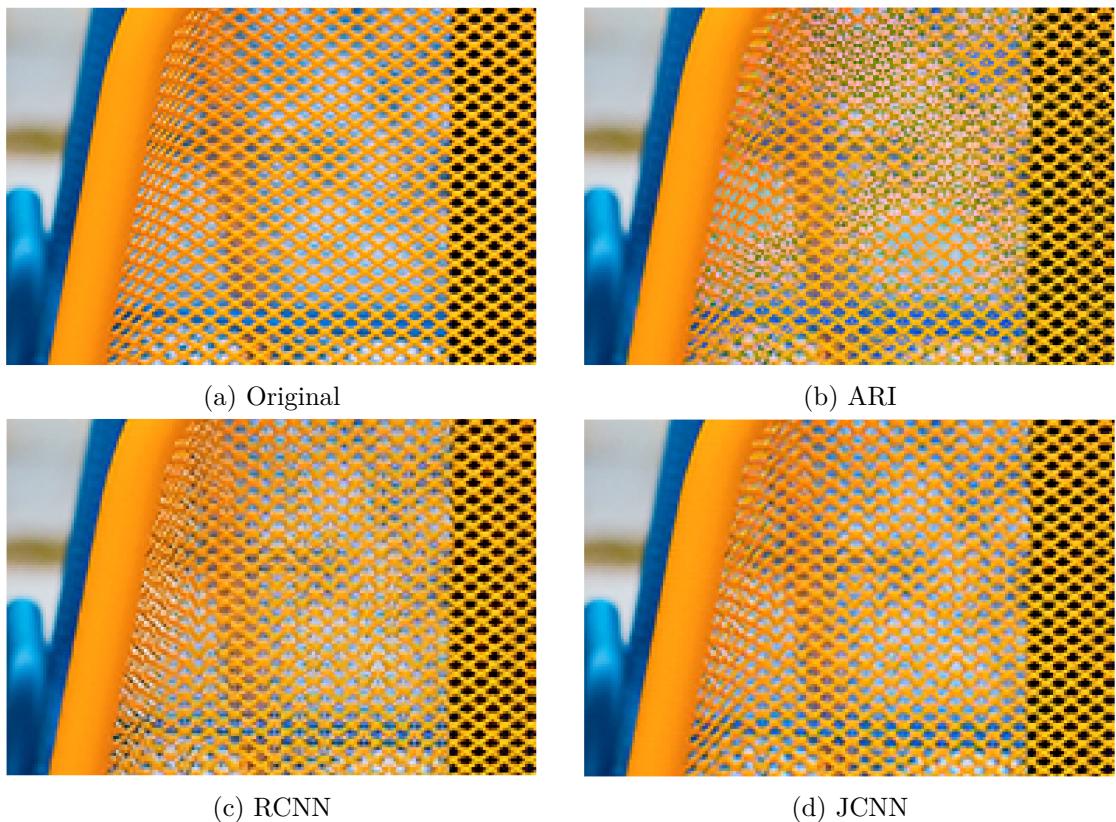


Abbildung 16: Visueller Vergleich verschiedener state-of-the-art Algorithmen [64, S. 227]

CNN basierte Algorithmen erzielen aktuell die besten Ergebnisse, im Vergleich zu anderen Demosaicing-Algorithmen [64, 53, 63]. Zwei kompetitive Vertreter davon sind JCNN

[65] (teilweise auch als DEMONET bezeichnet) und RCNN [66] (teilweise auch als DRL bezeichnet). Die Bildqualität beider Algorithmen ist vergleichbar, jedoch erzielt JCNN bei stärkerem Rauschen deutlich bessere Ergebnisse und ist auch effektiver darin Artefakte zu vermeiden, was in Abbildung 16 verdeutlicht wird. Die beiden Algorithmen benötigen zwar nur etwa ein Viertel der Ausführungszeit von ARI [64, S. 228], jedoch haben auch diese Algorithmen recht hohe Ausführungszeiten für Bilder mit Pixel in zweistelliger Millionenhöhe. Bezuglich einer Implementierung in JENIFFER2 wären diese Algorithmen mit einem höheren Aufwand als ARI verbunden.

### **3.4. Weitere Änderungen an JENIFFER2**

Im Rahmen dieser Masterarbeit lag der Fokus bezüglich JENIFFER2 darauf, neue Demosaicing-Algorithmen zu implementieren. Es wurden aber auch andere Änderungen an JENIFFER2 vorgenommen. Diese sollten die Performance und Usability von JENIFFER2 verbessern. Auf diese Änderungen wird im Folgenden kurz eingegangen.

- **Parallelisierung der Demosaicing-Algorithmen**

Da die Berechnung eines Farbwertes von jedem Pixel unabhängig voneinander ist, konnte dieser Vorgang parallelisiert werden. Dazu wurde das `java.util.stream` Paket genutzt. Dadurch werden während des Demosaicing-Vorgangs alle CPU-Kerne genutzt, was die Gesamtrechendauer erheblich reduziert. Diese Anpassung funktioniert rückwirkend für bereits implementierte Algorithmen und kann auch für zukünftige Algorithmen genutzt werden. Eine derartige Parallelisierung könnte auch auf andere Verarbeitungsschritte angewendet werden.

- **Automatisches Speichern und Wiederherstellen der Dateinavigation**

Direkt nach dem Starten von JENIFFER2 wird im Tab Bibliothek links das Navigationsmodul angezeigt. Dort kann man durch die Ordner des eigenen Systems navigieren. Die Neuerung hier ist, dass der zuletzt geöffnete Ordner vor dem Schließen von JENIFFER2 gespeichert wird und dieser Ordner nach einem Neustart wieder geöffnet wird. Dazu wird eine versteckte Datei im selben Ordner wie die JAR-Datei erstellt. Diese Funktion soll die Usability verbessern.

- **veränderbare Fenstergröße der Module**

Die verschiedenen Module lassen sich nun, durch Ziehen der roten Trennbalken, die in Abbildung 12 und 13 zu sehen sind, vergrößern und verkleinern. Dadurch wird die Benutzerfreundlichkeit erhöht.

- **Optimierung der Zoom-Funktion**

Im Editor Tab stehen dem Nutzer Zoom-Funktionen zur Verfügung. Diese hatten teilweise keinen konsistenten Zoom-Faktor. Es war auch möglich, aus einem Zoom-Level nicht mehr herauszukommen. Dies wurde entsprechend angepasst.

- **Exposure Correction**

Das Modul der Exposure Correction wurde in der vorherigen Version von JENIFFER2 nicht genutzt. Dieses Modul wurde nun aktiviert. Der Tag Baseline-Exposure oder BaselineExposureOffset müssen in der DNG-Datei vorhanden sein, damit das Modul genutzt wird.

## 4. Vergleich der Demosaicing-Algorithmen

Die in JENIFFER2 implementierten Demosaicing-Algorithmen haben unterschiedliche Auswirkungen auf die Bildqualität. Diese können objektiv bewertet werden. Dazu wurden verschiedene Performance-Parameter gemessen. Mehrere Bildersammlungen wurden genutzt, um Referenzbilder mit den in JENIFFER2 verarbeiteten Bildern zu vergleichen. Zusätzlich wurden die Ausführungszeiten der verschiedenen Algorithmen gemessen. Diese Messungen werden in Kombination mit den Parametern der Bildqualität genutzt, um mögliche Zusammenhänge zu bestimmen. In einem subjektiven Vergleich werden die deutlich sichtbaren Stärken und Schwächen der Algorithmen hervorgehoben. Außerdem wird ein Vergleich zwischen den implementierten Demosaicing-Algorithmen und den Demosaicing-Algorithmen der kommerziellen Bildverarbeitungsprogrammen Adobe Photoshop und Capture One durchgeführt.

### 4.1. Teststrategien

Die Qualität eines Bildes kann mit subjektiven und objektiven Methoden bestimmt werden. Durch die subjektive Bewertung eines Menschen können deutliche Artefakte erkannt und verglichen werden. Diese Methode eignet sich aber nicht, um große Datensätze auszuwerten und feinere Unterschiede zu quantifizieren. Dazu können verschiedene objektive Methoden genutzt werden. Diese objektiven Methoden lassen sich unterteilen in Methoden, die ein Referenzbild benötigen, und solchen, die kein Referenzbild benötigen.

Eine mögliche Methode, die Qualität eines Bildes ohne Referenzbild zu bewerten, kann durch eine Unschärfemetrik erfolgen [67]. Bei dieser Methode werden zunächst alle Kantenpixel durch die Verwendung eines Filters, wie dem Sobel- oder Prewitt-Filter identifiziert. Anschließend wird der Mittelwert der Breite aller Kanten im Bild bestimmt. Auf dieser Unschärfemetrik baut eine weitere referenzlose Methode auf, die zusätzlich einen Edge-Slope-Measure bestimmt [19, S. 17–18]. Mit diesem soll gemessen werden, wie scharf eine rekonstruierte Kante ist. Eine weitere referenzlose Methode, soll die Quantifizierung von False-Color-Artefakten ermöglichen [19, S. 18–19]. Ähnlich wie bei dem Edge-Slope-Measure werden hier abrupte Änderungen entlang von Kanten bestimmt.

Methoden mit Referenzbild werden am häufigsten verwendet, um Demosaicing-Algorithmen zu bewerten. Wie in Abbildung 17 wird dafür das Bayermosaik auf ein Bild mit allen Farbinformationen angewendet. Dieses Bild wird mit einem Demosaicing-Algorithmus verarbeitet und das rekonstruierte Bild wird mit dem Originalbild verglichen. Ähnlich wie

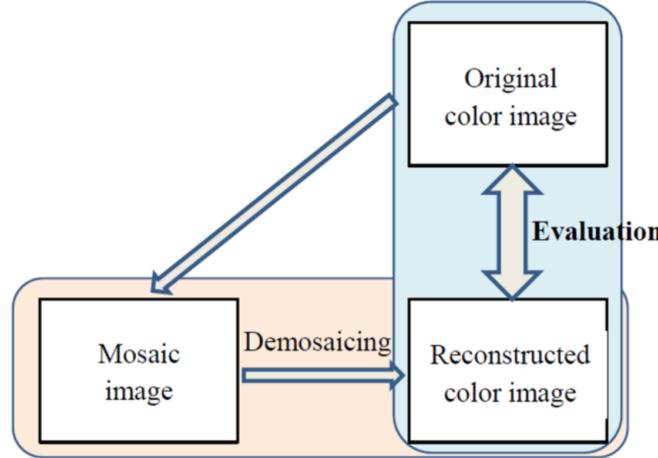


Abbildung 17: Evaluationsprozedur von Demosaicing-Algorithmen mit Referenzbildern [68, S. 104]

bei den referenzlosen Methoden, gibt es auch Methoden mit Referenzbild, um einen False-Color-Parameter zu bestimmen [68, S. 105–106]. Zipper-Artefakte lassen sich ebenfalls mit Referenzbildern quantifizieren [68, S. 106]. Die Parameter, die am häufigsten bei der Evaluation von Demosaicing-Algorithmen genutzt werden, sind Mean Square Error (MSE), Peak Signal to Noise Ratio (PSNR) und Structural Similarity Index Measure (SSIM). Diese drei Parameter benötigen alle ein Referenzbild.

- **MSE** misst den Mittelwert der quadrierten Differenzen zwischen den originalen Pixeln und den errechneten Pixeln [69]. Diese Metrik lässt sich leicht errechnen. Sie hat allerdings zwei Nachteile. Einerseits stimmt dieser Wert nicht mit der menschlichen Wahrnehmung überein. Eine kleinere Änderung an einem Bild kann zu einer großen Änderung des MSE führen, lässt sich von einem Menschen aber kaum wahrnehmen [4, S. 289]. Ein weiterer Nachteil ist, dass der MSE vom Wertebereich der Pixel abhängt. Um MSE-Werte verschiedener Bilder vergleichen zu können, müssen diese im selben Wertebereich sein. Ein möglichst geringer MSE-Wert entspricht einer größeren Ähnlichkeit der zwei Bilder. In Formel 1 ist die Formel des MSE angegeben. Dabei entsprechen  $I_1$  und  $I_2$  den beiden Eingabebildern.  $c$  entspricht der Anzahl der Farbkanäle, was bei einem RGB-Bild 3 wäre.  $H$  und  $W$  entsprechen der Höhe und Breite des Bildes, wobei  $h$  und  $w$  der entsprechende Index ist.

$$MSE(I_1, I_2) = \frac{1}{cHW} \sum_{h,w}^c [I_1^c(h, w) - I_2^c(h, w)]^2 \quad (1)$$

- **PSNR** wird vom MSE abgeleitet und gibt das Verhältnis zwischen der maximalen Pixelintensität und des Störsignals wieder [69]. PSNR ist die am häufigsten verwendete Metrik zur Evaluation von Bildqualität [68, S. 105]. Diese Metrik lässt sich ebenfalls leicht errechnen. Sie hat wie MSE den Nachteil, dass sie nicht mit der menschlichen Wahrnehmung übereinstimmt. Allerdings ist PSNR nicht abhängig vom Wertebereich der Pixel und wird in Dezibel gemessen. Ein höherer PSNR-Wert entspricht einer besseren Bildqualität. In Formel 2 ist die Formel des PSNR angegeben. *peakval* entspricht dem Wertebereich der Pixel. Bei einem vorzeichenlosen 8-Bit Integer gilt beispielhaft *peakval* = 255.

$$PSNR(I_1, I_2) = 10 \log_{10} \left( \frac{peakval^2}{MSE(I_1, I_2)} \right) \quad (2)$$

- **SSIM** kombiniert die Werte der Luminanz, der Kontrasts und der Bildstruktur in einem einzigen Wert. Bildstruktur bedeutet hier Muster von Pixelintensitäten, besonders zwischen benachbarten Pixeln [69]. Im Gegensatz zu MSE und PSNR ist die Berechnung von SSIM etwas komplexer. Dafür entspricht der resultierende Wert eher der menschlichen Wahrnehmung. Der resultierende Wert von SSIM befindet sich typischerweise im Bereich von 0 bis 1, wobei 1 der höchstmöglichen Qualität entspricht. Die allgemeine Formel für SSIM ist in Formel 3 angegeben. Sie besteht aus drei Termen, die in Formel 4, 5 und 6 angegeben sind. Formel 4 entspricht dem Term für Luminanz. Formel 5 entspricht dem Term für Kontrast. Formel 6 entspricht dem Term für Struktur. Dabei entspricht  $\mu$  dem Mittelwert des jeweiligen Bildes.  $\sigma_{I_1}^2$  und  $\sigma_{I_2}^2$  entsprechen der Standardabweichung des jeweiligen Bildes.  $\sigma_{I_1 I_2}$  entspricht der Kreuzkovarianz der beiden Bilder.  $C_1$ ,  $C_2$  und  $C_3$  sind kleine Konstanten, um eine Division durch null zu vermeiden. Wenn für die Gewichte  $\alpha = \beta = \gamma = 1$  und für die Gewichte  $C_3 = C_2/2$  gilt, dann lässt sich der Term in Formel 3 zu dem Term in Formel 7 vereinfachen. Bei SSIM ist zu beachten, dass sich die in dieser Arbeit verwendete Implementierung der Formel nur auf einen Farbkanal anwenden lässt. Um den SSIM für ein RGB-Bild zu erhalten, wurde der Mittelwert der drei SSIM-Werte genutzt.

$$SSIM(I_1, I_2) = [l(I_1, I_2)]^\alpha \cdot [c(I_1, I_2)]^\beta \cdot [s(I_1, I_2)]^\gamma \quad (3)$$

$$l(I_1, I_2) = \frac{2\mu_{I_1}\mu_{I_2} + C_1}{\mu_{I_1}^2 + \mu_{I_2}^2 + C_1} \quad (4)$$

$$c(I_1, I_2) = \frac{2\sigma_{I_1}\sigma_{I_2} + C_2}{\sigma_{I_1}^2 + \sigma_{I_2}^2 + C_2} \quad (5)$$

$$s(I_1, I_2) = \frac{\sigma_{I_1 I_2} + C_3}{\sigma_{I_1}\sigma_{I_2} + C_3} \quad (6)$$

$$SSIM(I_1, I_2) = \frac{(2\mu_{I_1}\mu_{I_2} + C_1)(2\sigma_{I_1 I_2} + C_2)}{(\mu_{I_1}^2 + \mu_{I_2}^2 + C_1)(\sigma_{I_1}^2 + \sigma_{I_2}^2 + C_2)} \quad (7)$$

## 4.2. Image Dataset

Um die im vorherigen Kapitel vorgestellten drei Teststrategien zu verwenden, benötigt man das verarbeitete Bild und ein Referenzbild. Das Referenzbild muss, wie das verarbeitete Bild, in jedem Pixel die Information von allen Farbwerten besitzen. Deswegen ist es nicht möglich, unverarbeitete RAW-Bilder als Referenzbilder zu nehmen, die als Input für einen Demosaicing-Algorithmus verwendet werden. Um solche Referenzbilder zu erzeugen, die auch als Ground Truth Data bezeichnet wird, wurden verschiedene Techniken angewendet. Im Folgenden werden drei verschiedene Bildersammlungen vorgestellt. Diese gehören zu den am häufigsten verwendeten Bildersammlungen in wissenschaftlichen Arbeiten zu Demosaicing-Algorithmen. Diese Bildersammlungen wurden auch als Referenzbilder, für die Bewertung der Demosaicing-Algorithmen in dieser Masterarbeit genutzt.

- **Kodak Lossless True Color Image Suite**

Diese Bildersammlung besteht aus 24 Bildern mit den Maßen 768x512. Eine Zusammenstellung der Bilder ist in Abbildung 18 dargestellt. Die Bildersammlung wurde 1991 von Kodak veröffentlicht [70, S. 2289]. Es wird davon ausgegangen, dass diese Bilder mit einem Farbfilm aufgezeichnet und anschließend mit einem Scanner digitalisiert wurden [71, S. 7]. Das Kodak Set wird in den meisten wissenschaftlichen Arbeiten zu Demosaicing verwendet. Jedoch wird die Repräsentativität dieser Bildersammlung, für digitale Farbbilder, in manchen Arbeiten kritisiert [70, 71]. Speziell haben die Bilder des Kodak Sets sehr hohe spektrale Korrelation, glatte chromatische Gradienten und eine niedrige Sättigung [71, S. 6–7]. Ein Link zu dem Kodak Set ist in Quelle [72] zu finden.



Abbildung 18: Ausschnitte von Bildern der Kodak Lossless True Color Image Suite

- **McMaster Dataset**

Diese Bildersammlung besteht aus 18 Bildern mit den Maßen 500x500. Eine Zusammenstellung der Bilder ist in Abbildung 19 dargestellt. Das McMaster Dataset wird manchmal auch als IMAX Dataset bezeichnet. Die Bildersammlung wurde an der McMaster Universität in Kanada für die Erforschung neuer Demosaicing-Algorithmen erstellt[71, S. 7]. Die Bilder wurden mit einem Kodak Farbfilm aufgezeichnet und anschließend digitalisiert. Im Gegensatz zu dem Kodak Set, ist die spektrale Korrelation in den Bildern geringer. Die McMaster Bilder haben eine höhere Sättigung und viele scharfe Strukturen mit abrupten Farbtransitionen [71, S. 9–10]. Ein Link zu dem McMaster Set ist in Quelle [73] zu finden.



Abbildung 19: Ausschnitte von Bildern des McMaster Dataset

- **Microsoft Research Cambridge Demosaicing Dataset**

Diese Bildersammlung besteht aus Aufnahmen mit zwei verschiedenen Kameras. Mit einer Canon EOS 550D wurden 57 Bilder erstellt. Mit einer Panasonic Lumix DMC-LX3 wurden 500 Bilder erstellt. Zu jedem Ground-Truth-Bild liegt zusätzlich eine Version in Graustufen vor, die mit dem RGGB-Bayer Pattern erstellt wurde. Von diesen Bildern liegt zusätzlich eine Version mit zusätzlichem Rauschen vor. Von den 500 Bildern, die mit der Panasonic Kamera erstellt wurden, liegt außerdem eine Version vor, bei der die Graustufenbilder mit dem Fujifilm X-Trans Muster erstellt wurden. Von den X-Trans Bildern liegt jedoch keine Version mit Rauschen vor. Die Bildersammlung enthält auch die Resultatbilder der Bilder mit Rauschen, die mit verschiedenen Demosaicing-Algorithmen erstellt wurden. Die Bilder, die mit der Canon Kamera erstellt wurden, haben die Maße 318x210. Die Bilder, die mit der Panasonic Kamera erstellt wurden, haben die Maße 220x132. Die Bilder wurden im Rahmen einer wissenschaftlichen Arbeit zu einem Demosaicing-Algorithmus mit zusätzlicher Denoising-Funktion veröffentlicht [74]. Für diese Masterarbeit wurden die Bilder 1-24 und 500, die mit der Panasonic Kamera aufgenommen wurden, verwendet sowie die verrauschten Versionen dieser Bilder. Eine Zusammenstellung der Bilder ist in Abbildung 20 dargestellt. Um die Referenzbilder aus den RAW-Bildern der Kamera zu erzeugen, beschreiben die Autoren eine Maximum-Entropy-Strategie zum Downsampling. Diese Strategie nutzt für eine gegebene Fenstergröße bestimmte Gewichte für verschiedene Pixelwerte, bevor diese zu einem Pixel mit voller Farbinformation zusammengefasst werden. Die einfachere Strategie, immer vier Pixel zu einem Pixel mit kompletten Farbinformationen zusammenzufassen, erzeugt wesentlich mehr Artefakte als die beschriebene Maximum-Entropy-Strategie. Um die verrauschten Bilder zu erstellen, wurde eine Methode genutzt, die ein Profil anhand von RAW-Bildern erstellt und dazu die Poisson- und Gaußverteilung nutzt [75]. Über die Charakteristik der Bilder liegen keine konkreten Untersuchungen vor wie zu dem Kodak und McMaster Set. Nach einer subjektiven Einschätzung ähneln die 25 gewählten Bilder für diese Arbeit eher den Bildern des Kodak Sets. Ein Link zu dem MicroSoft Research (MSR) Set ist in Quelle [76] zu finden.

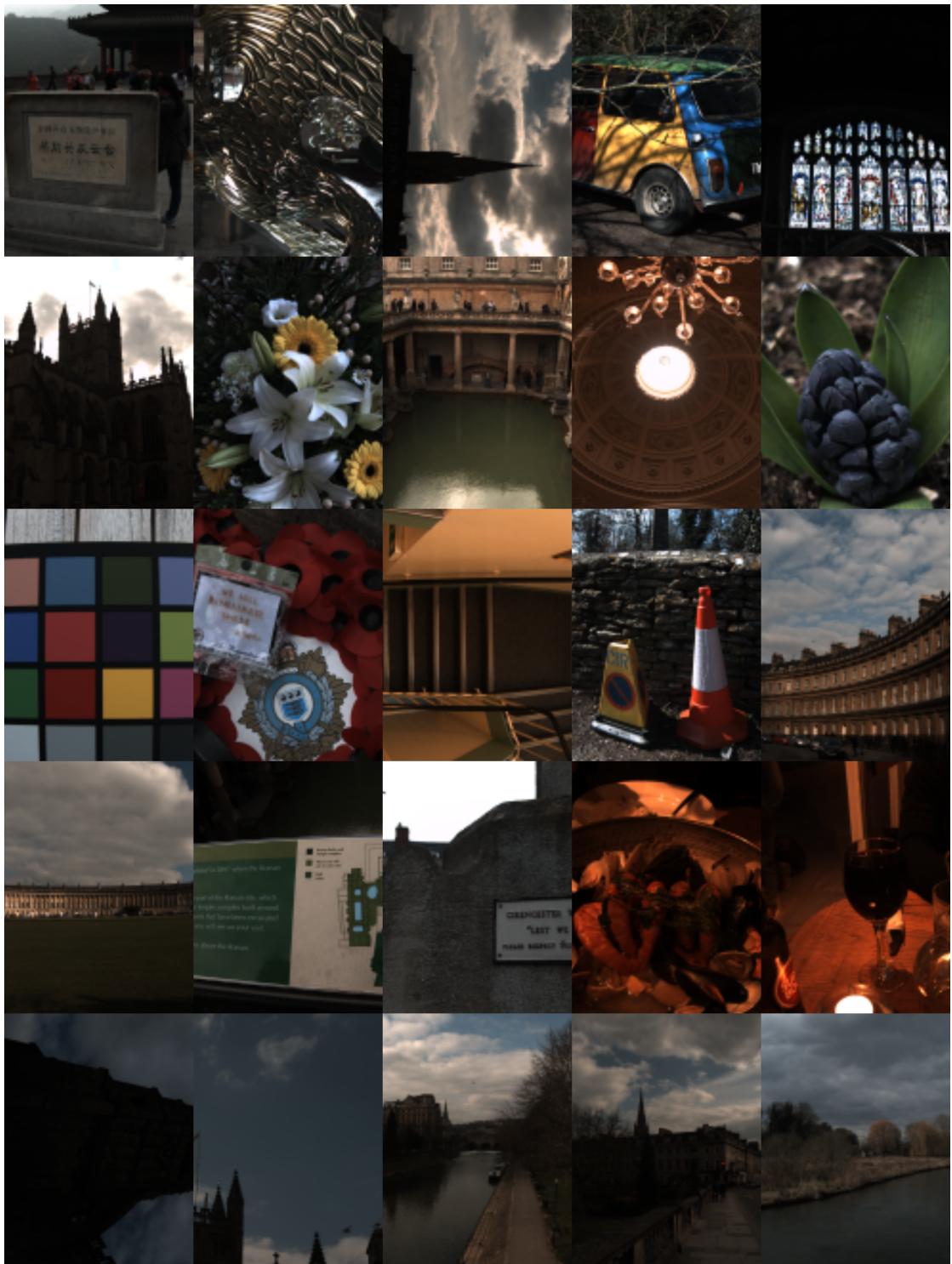


Abbildung 20: Ausschnitte von Bildern des Microsoft Research Cambridge Demosaicing Dataset

JENIFFER2 kann normalerweise nur Bilder im DNG-Format einlesen. Damit Bilder aus den drei vorgestellten Bildersammlungen eingelesen werden können, wurde eine angepasste Version von JENIFFER2 erstellt. Diese Version liest unmittelbar vor dem Demosaicing die gewünschte Portable Network Graphics (PNG)-Datei ein. Diese Datei wird mit einem manuell vorgegebenen Bayer-Raster verarbeitet und anschließend in eine neue PNG-Datei geschrieben. JENIFFER2 wird nach dem Schreiben der Datei vorzeitig beendet, da ansonsten Fehler bei der folgenden Verarbeitungspipeline auftreten. Auf die ursprüngliche PNG-Dateien musste das Bayer-Mosaik angewendet werden. Dieser Vorgang ist in Abbildung 21 abgebildet. Im ersten Schritt wird das Bayer-Mosaik in der Variante RGGB auf das Bild angewendet. Bei dieser Variante ist der Pixel in der oberen linken Ecke rot. Anschließend ist in jedem Pixel nur die Information von einem der drei Farbkanäle vorhanden. Dafür wurde der Code genutzt, der in Kapitel 3.2.7 verlinkt ist. Im zweiten Schritt wird das Bild in ein Graustufenbild konvertiert. Das Graustufenbild verfügt über nur einen Farbkanal. Für die Konvertierung zum Graustufenbild ist zu beachten, dass für jeden Farbkanal das Gewicht eins gewählt werden muss.

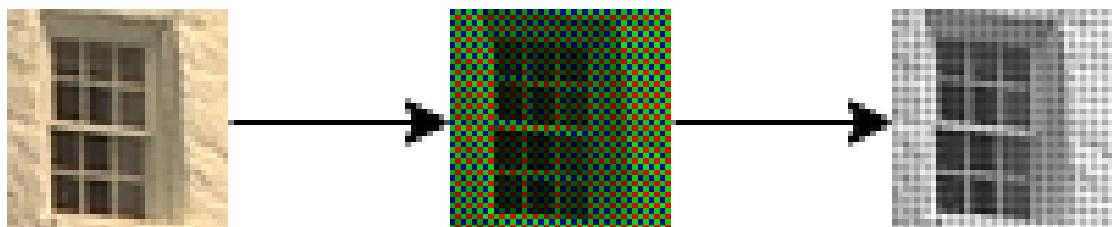


Abbildung 21: Verarbeitungsschritte zur Vorbereitung einer PNG-Datei für das Demosaicing

#### 4.3. Subjektiver Vergleich

Im Folgenden werden verschiedene Bilder, aus den drei Bildersammlungen von Kodak, McMaster und MSR, mit den verarbeiteten Bildern der in JENIFFER2 implementierten Demosaicing-Algorithmen verglichen. Dazu wurden nur Ausschnitte aus diesen Bildern gewählt. Zusätzlich werden Differenzbilder aus dem jeweiligen Originalbild und den durch Demosaicing-Algorithmen erzeugten Bildern abgebildet. Diese sollen Unterschiede verdeutlichen.

Abbildung 22 zeigt einen Ausschnitt aus Bild 13 aus dem Kodak Set. Im Original sind feine Strukturen am Flussufer abgebildet. Diese Details sind wegen ihrer geringen Größe und den hohen farblichen Unterschieden schwer zu rekonstruieren. Nach der Messung

in Kapitel 4.4, erzielen die Demosaicing-Algorithmen bei diesem Bild die schlechtesten Resultate innerhalb des Kodak Sets. Bei fast allen Algorithmen sind deutliche False-Color-Artefakte am Flussufer und in den schäumenden Teilen des Flusses zu finden. Die meisten False-Color-Artefakte erzeugt der NN Algorithmus. Bei der bikubischen Interpolation sind fast genauso viele False-Color-Artefakte erkennbar sowie eine Unschärfe des Bildes. Bei den beiden Varianten der bilinearen Interpolation sind etwas weniger False-Color-Artefakte als bei der bikubischen Interpolation erkennbar, dafür aber eine stärkere Unschärfe. Bei PPG sind etwas weniger False-Color-Artefakte vorhanden. Bei RCD, HA und MHC sind noch weniger False-Color-Artefakte als bei PPG zu erkennen. Besonders in den grünen Pflanzen sind kaum Artefakte zu sehen. Bei den verschiedenen DLMMSE Varianten sind fast keine Artefakte erkennbar. Zwischen den verschiedenen DLMMSE Varianten sind keine Unterschiede erkennbar.

Abbildung 23 zeigt einen Ausschnitt aus Bild 19 des Kodak Sets. Im Original sind senkrechte Zaunlatten, mit zunehmender Dichte zum rechten Bildrand abgebildet. Diese Strukturen sind wegen der zu niedrigen Abtastrate des simulierten CFA Sensors von besonderem Interesse. Dieses Bild wurde deswegen häufig in anderen Arbeiten zu Demosaicing-Algorithmen visuell ausgewertet. Bei allen Algorithmen sind an Teilen des Zauns False-Color-Artefakte zu erkennen. Bei dem NN Algorithmus sind die meisten False-Color-Artefakte, sowie starke Zipper-Artefakte erkennbar. Bei den beiden Varianten der bilinearen Interpolation und der bikubischen Interpolation sind etwas weniger False-Color-Artefakte sowie eine Unschärfe des Bildes zu erkennen. Zusätzlich sind bei der bilinearen Interpolation mit Mittelwert Zipper-Artefakte erkennbar, zum Beispiel entlang der weißen Fassade. MHC und PPG erzeugen weniger False-Color-Artefakte. Besonders in allen Bereichen außer des Zauns sind wesentlich weniger False-Color-Artefakte vorhanden. PPG erzeugt zudem wegen seiner adaptiven Komponente unregelmäßige Strukturen in den schwierigen Bereichen des Zauns. Bei HA sind wieder etwas weniger False-Color-Artefakte erkennbar. Ähnlich wie bei PPG sind unregelmäßige Strukturen wegen der adaptiven Komponente von HA in Bereichen des Zauns erkennbar. RCD erzeugt weniger erkennbare False-Color-Artefakte als HA. Die DLMMSE Varianten erzeugen am wenigsten erkennbare False-Color-Artefakte. Zwischen den verschiedenen DLMMSE Varianten sind keine Unterschiede erkennbar.

Abbildung 24 zeigt einen Ausschnitt aus Bild 5 aus dem McMaster Set. Im Original sind stark gesättigte Farben, die teilweise nur wenige Pixel breit sind, auf einem weißen Hintergrund abgebildet. NN erzeugt hier starke False-Color-Artefakte und Zipper-Artefakte entlang der Farbübergänge. Bei der bikubischen Interpolation sind stärkere False-Color-

Artefakte als bei den bilinearen Varianten entlang der Farbübergänge erkennbar. Dafür ist bei den bilinearen Varianten ein größerer Verlust an Bildschärfe zu erkennen, besonders bei kleinen Farbpunkten auf weißem Hintergrund. Die bilineare Interpolation mit Mittelwert erzeugt zudem Zipper-Artefakte, die entlang von Farbübergängen erkennbar sind. PPG erzeugt leichte False-Color-Artefakte an Farbübergängen. Die restlichen Algorithmen erzeugen keine erkennbaren False-Color-Artefakte. HA, MHC und die DLMMSE Varianten erzeugen alle Zipper-Artefakte entlang von Farbübergängen. Zwischen diesen drei erzeugt HA die wenigsten erkennbaren Fehler und MHC die Meisten. Zwischen den DLMMSE Varianten erzeugen die Varianten, die RCD für die Rot- und Blau-Kanal Berechnung nutzen, weniger erkennbare Fehler als die Standardvarianten. Dies ist auch auf den Differenzbildern erkennbar. RCD erzeugt bei diesem Bild die wenigsten erkennbaren Fehler. Hier sind nur teilweise dunkle Verfärbungen gegenüber dem Original in feinen Strukturen erkennbar.

Abbildung 25 zeigt einen Ausschnitt aus Bild 18 aus dem MSR Set. Im Original sind feine Strukturen mit unterschiedlicher Helligkeit an einer metallischen Oberfläche abgebildet. Nach der Messung in Kapitel 4.4 erzielen die Demosaicing-Algorithmen bei diesem Bild die schlechtesten Resultate innerhalb des MSR Sets. Die meisten False-Color- und Zipper-Artefakte erzeugt der NN Algorithmus. Bei der bikubischen und bilinearen Interpolation werden etwas weniger False-Color-Artefakte erzeugt. Die Helligkeitsunterschiede sind jedoch besser als bei der bilinearen Interpolation erkennbar. Die bilineare Interpolation erzeugt etwas weniger False-Color-Artefakte als die bikubische Interpolation. Jedoch sind die feinen Helligkeitsunterschiede schlecht erkennbar. HA, PPG und MHC erzeugen weniger erkennbare False-Color-Artefakte als die bikubische Interpolation, dafür aber leichte Zipper-Artefakte. Zwischen diesen drei Algorithmen erzeugt PPG die meisten erkennbaren False-Color- und Zipper-Artefakte und HA die Wenigsten. RCD erzeugt weniger erkennbare False-Color-Artefakte als die drei eben benannten Algorithmen. Auf dem Differenzbild von RCD sind die stärksten Unterschiede entlang von schmalen Kanten mit verschiedenen Helligkeiten erkennbar. Die wenigsten erkennbaren False-Color-Artefakte werden von den DLMMSE Varianten erzeugt. Zwischen den verschiedenen DLMMSE Varianten sind keine Unterschiede erkennbar.

Abbildung 26 zeigt einen Ausschnitt aus Bild 500 aus dem MSR Set. Im Original ist ein Colorchecker abgebildet. Die einheitlich gefärbten Quadrate und schwarzen, geraden Kanten sind künstliche Strukturen. Diese können für manche Algorithmen herausfordernd sein. Bei dem NN Algorithmus sind deutliche Zipper-Artefakte entlang der senkrechten Kante erkennbar sowie zusätzliche Farbbalken entlang der horizontalen Kante.

Bei der bilinearen Interpolation mit Mittelwert sind deutliche Zipper-Artefakte entlang der Kanten erkennbar. Bei der bilinearen Interpolation mit Median ist erkennbar, wie entlang der Kanten Farbübergänge entstehen. Außerdem sind Unregelmäßigkeiten in den Ecken erkennbar. Bei der bikubischen Interpolation sind teilweise Zipper-artige teilweise gleichmäßige Helligkeitsunterschiede entlang der Kanten erkennbar. MHC und PPG erzeugen beide Zipper-Artefakte entlang der Kanten, wobei das Muster bei PPG unregelmäßig ist. Die DLMMSE Varianten erzeugen etwas weniger Zipper-Artefakte entlang der Kanten. Hier ist deutlich erkennbar, dass die DLMMSE Varianten, die RCD für die Rot- und Blau-Kanal Berechnung nutzen, weniger Artefakte erzeugen. Bei HA sind nur einzelne fehlerhafte Pixel in den Ecken sowie ein schmaler Übergang entlang der Kanten erkennbar. Dies ist in den Differenzbildern gut erkennbar. RCD erzeugt hier keine erkennbaren Artefakte. Lediglich im direkten Vergleich mit dem Original ist ein schmaler Übergang entlang der Kanten erkennbar. In den Differenzbildern ist erkennbar, dass die DLMMSE Varianten und MHC in den gleichmäßigen Flächen unterschiedliche Farbtöne erzeugen. Bei der bikubischen Interpolation ist dasselbe in abgeschwächter Form zu sehen.

Abbildung 27 und 28 zeigen Ausschnitte der Bilder 18 und 500 aus dem MSR Set mit Rauschen. Das Rauschen ist besonders gut auf den einheitlichen Farbflächen in Abbildung 28 erkennbar. Am wenigsten Unregelmäßigkeiten durch Rauschen sind bei der bilinearen Interpolation und der bikubischen Interpolation erkennbar. Bei NN sind auch fast keine Unregelmäßigkeiten erkennbar. Die Unregelmäßigkeiten, die erkennbar sind, stechen dafür stark hervor. Anschließend erzeugen RCD und PPG die wenigsten erkennbaren Unregelmäßigkeiten, wobei RCD etwas besser ist. Die restlichen Algorithmen erzeugen kaum erkennbare Unterschiede an Unregelmäßigkeiten.

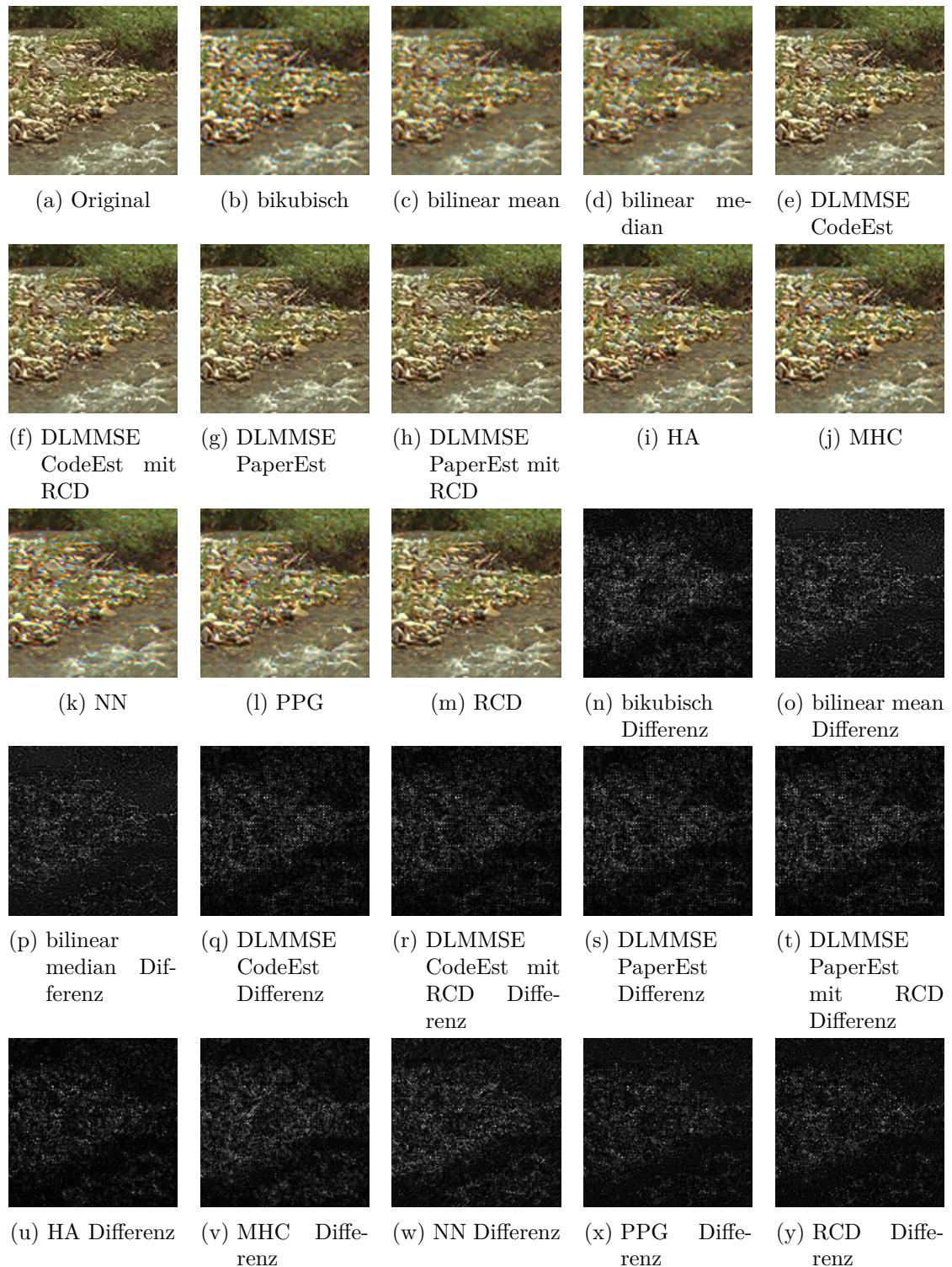


Abbildung 22: Ausschnitte von Bild 13 aus dem Kodak Set von verschiedenen Algorithmen und Differenzbilder

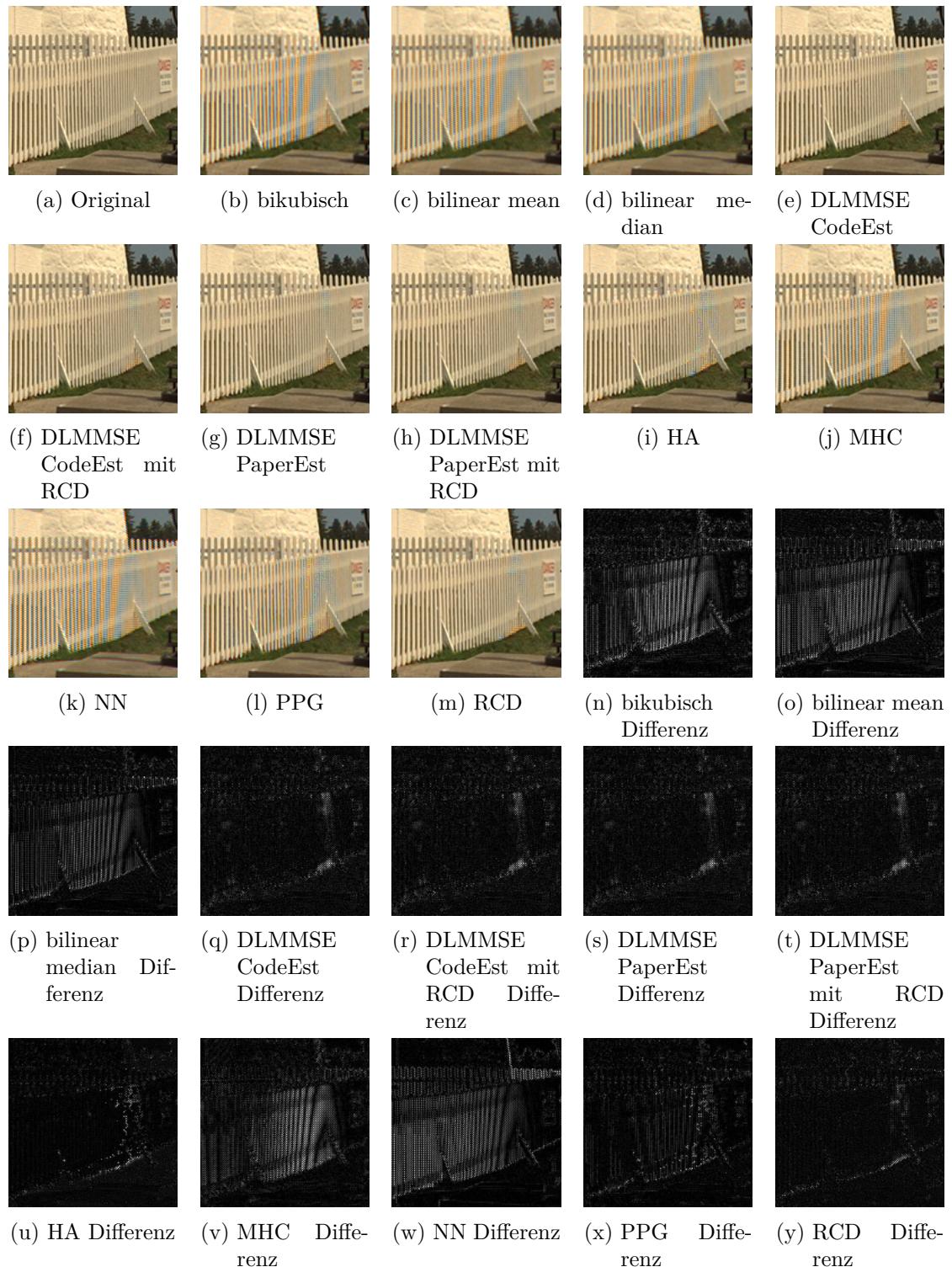


Abbildung 23: Ausschnitte von Bild 19 aus dem Kodak Set von verschiedenen Algorithmen und Differenzbilder

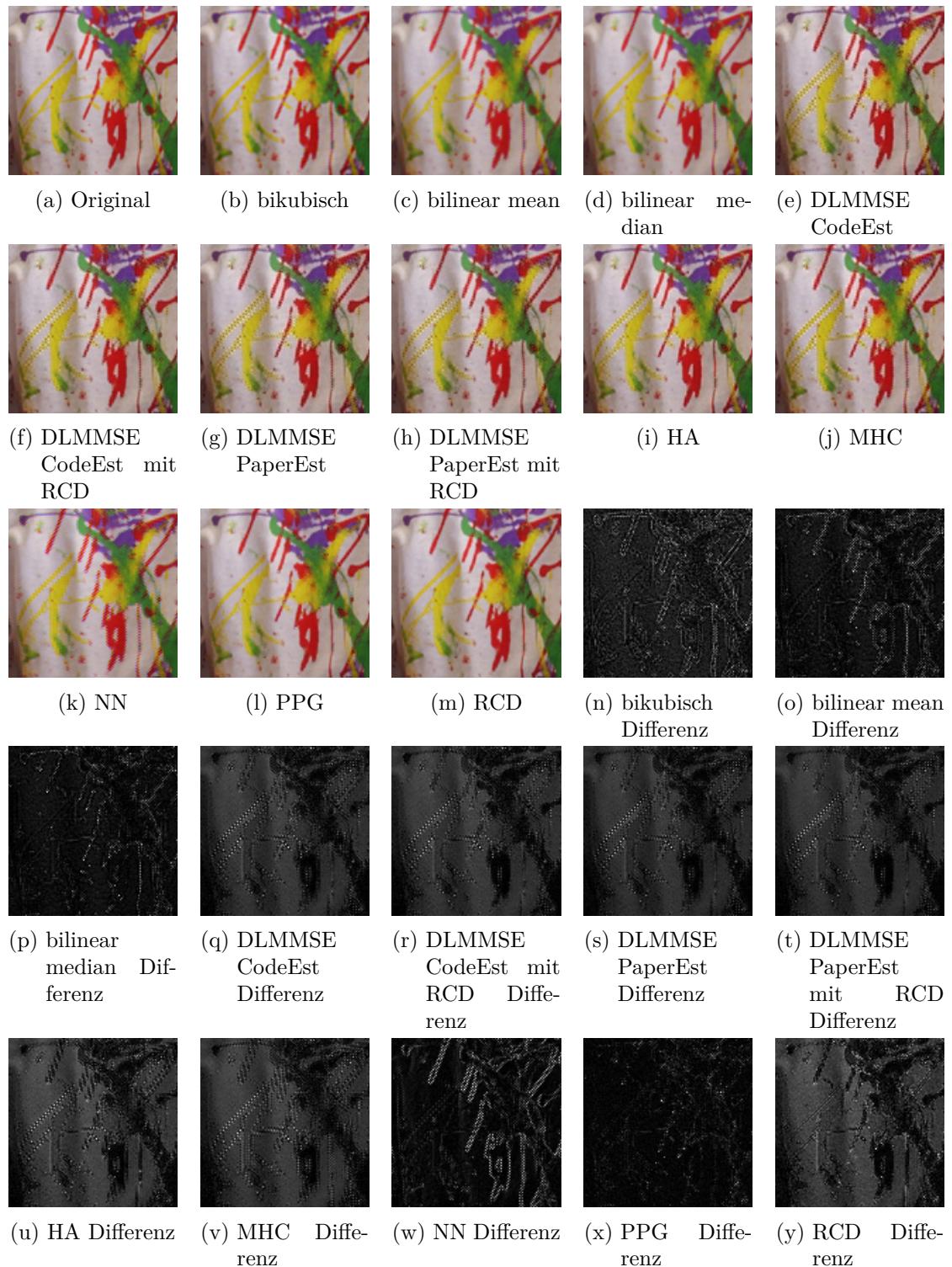


Abbildung 24: Ausschnitte von Bild 5 aus dem McMaster Set von verschiedenen Algorithmen und Differenzbilder

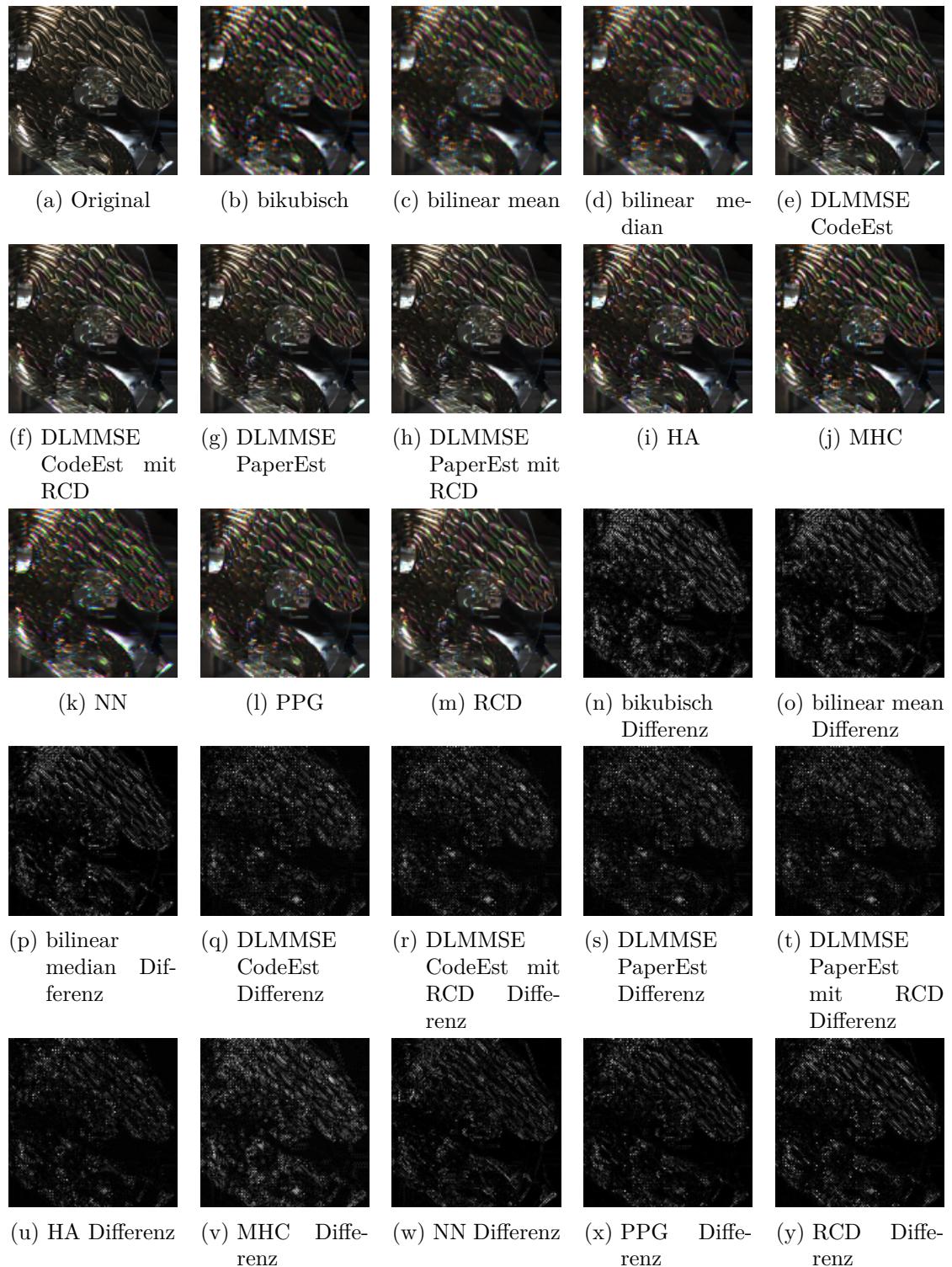


Abbildung 25: Ausschnitte von Bild 18 aus dem MSR Set von verschiedenen Algorithmen und Differenzbilder

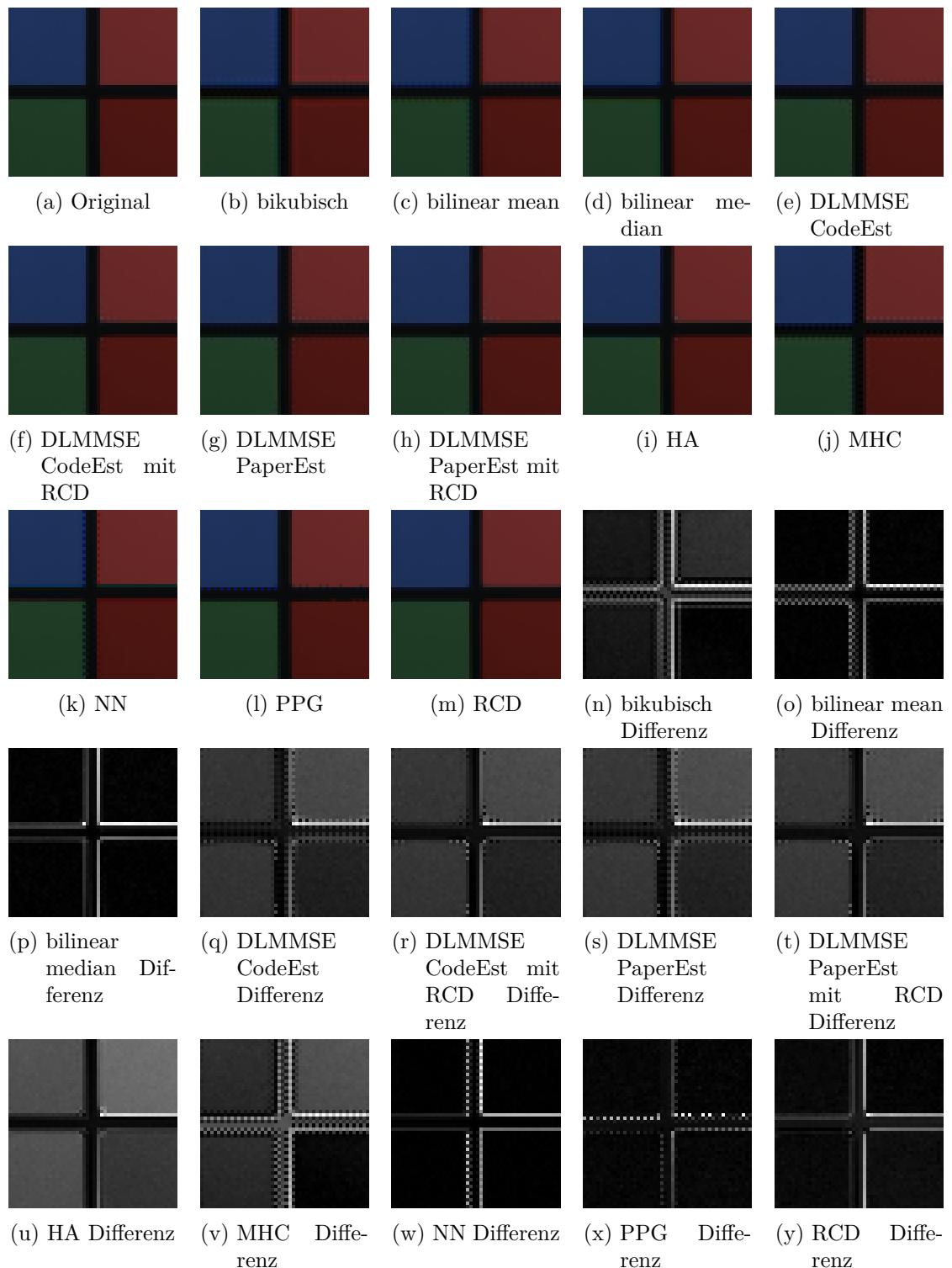


Abbildung 26: Ausschnitte von Bild 500 aus dem MSR Set von verschiedenen Algorithmen und Differenzbilder



Abbildung 27: Ausschnitte von Bild 18 aus dem MSR Set mit Rauschen von verschiedenen Algorithmen und Differenzbilder

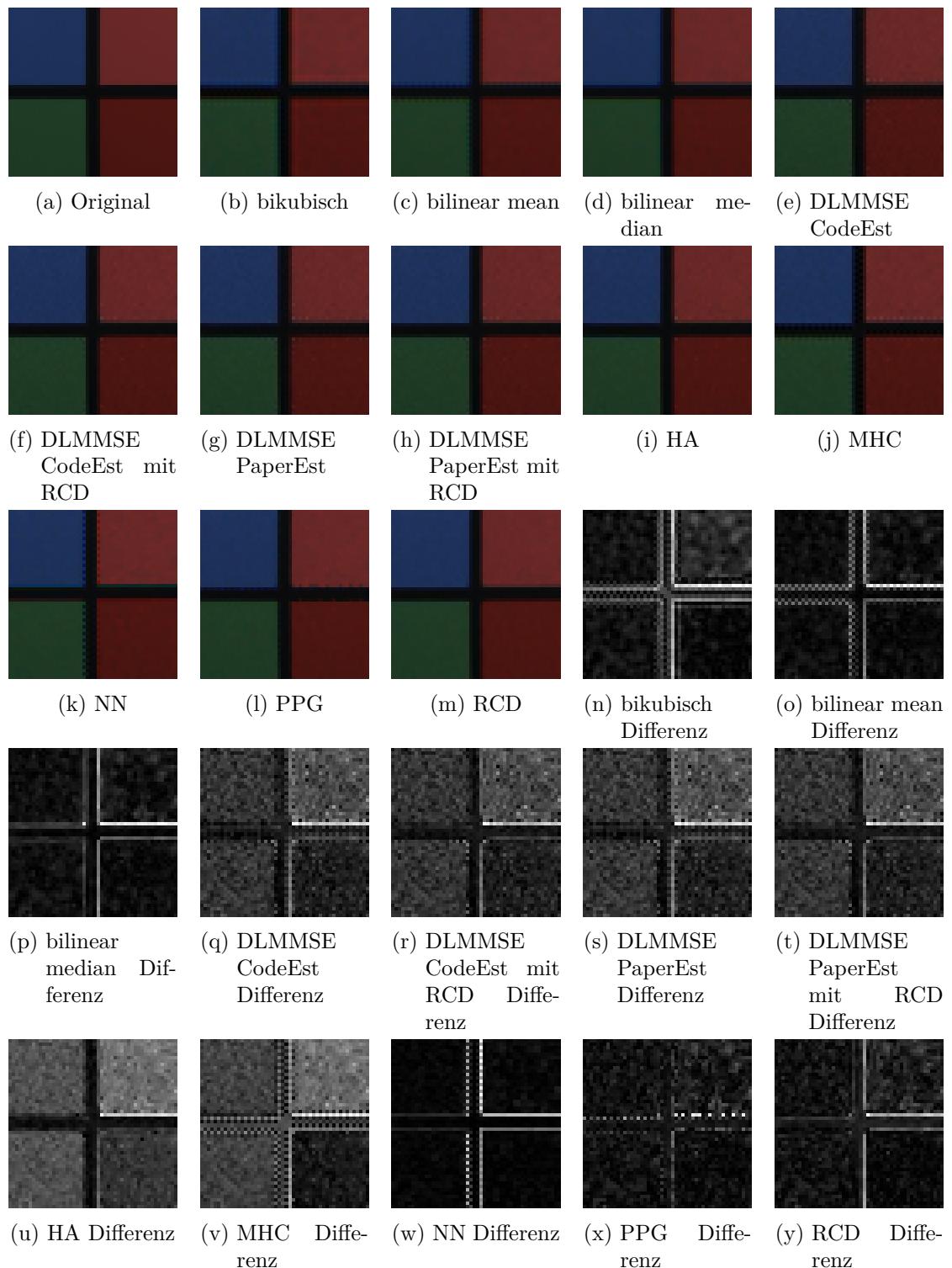


Abbildung 28: Ausschnitte von Bild 500 aus dem MSR Set mit Rauschen von verschiedenen Algorithmen und Differenzbilder

#### **4.4. Messung**

In Tabelle 1 sind die durchschnittlichen MSE-, PSNR und SSIM-Werte, die im Kapitel 4.1 vorgestellt wurden, innerhalb der drei Bildersammlungen, die im Kapitel 4.2 vorgestellt wurden, von jedem in JENIFFER2 implementierten Demosaicing-Algorithmus dargestellt. Der jeweils beste Wert innerhalb einer Spalte ist fett und unterstrichen hervorgehoben. Zur Berechnung der Messwerte wurden die Implementierungen aus Matlab genutzt, die in Quelle [69] zu finden sind. Die Bilder wurden vor den Berechnungen alle in den Wertebereich von vorzeichenlosen 8-Bit-Integer konvertiert. Bei den Bildern des McMaster Sets und des MSR Sets wurden vor der Berechnung Ränder mit einer Breite von fünf Pixeln entfernt. Dies soll eine Verfälschung der Messwerte wegen fehlerhaft berechneten Randpixeln verhindern. Bei den Bildern des Kodak Sets wurden 10 Pixel an den Rändern entfernt, da diese Bilder teilweise zusätzliche graue Ränder haben. Im Anhang, unter Kapitel A, sind die vollständigen Messdaten zu finden.

	MSE				PSNR				SSIM			
	Kodak	McM	MSR	Mean	Kodak	McM	MSR	Mean	Kodak	McM	MSR	Mean
bilkubisch	88,33	45,7	88	76,76	30,02	32,81	30,88	31,09	0,8825	0,9289	0,9192	0,9087
bilinear mean	82,63	50,32	84,6	74,68	30,23	32,32	30,91	31,05	0,8888	0,9278	0,92	0,9109
bilinear median	80,42	47,02	82,25	72,13	30,39	32,61	31,12	31,26	0,8892	0,9294	0,9224	0,9124
DLMmse code	8,2	29,86	16,33	17,05	39,76	34,7	38,32	37,87	<b>0,985</b>	0,9367	0,9808	0,9704
DLMmse code RCD	8,02	24,2	15,11	<b>15,01</b>	39,9	35,52	38,7	38,28	0,9846	0,9456	<b>0,9824</b>	0,9733
DLMmse paper	<b>7,75</b>	31,06	15,87	17,04	39,94	34,56	38,46	37,94	0,9846	0,9367	0,9812	0,9704
DLMmse paper RCD	7,79	25,24	<b>14,66</b>	15,04	<b>40,1</b>	35,38	<b>38,84</b>	<b>38,36</b>	<b>0,985</b>	0,9456	<b>0,9824</b>	<b>0,9734</b>
HA	16,88	28,07	25,77	23,2	37,02	34,77	36,44	36,2	0,9725	0,9417	0,9724	0,9642
MHC	23,03	29,92	32,64	28,47	35,69	34,4	35,06	35,11	0,9688	0,9367	0,9668	0,9594
NN	187,78	161,13	199,29	184,92	26,5	27,11	26,86	26,8	0,8108	0,8467	0,854	0,8366
PPG	44,72	33,18	46,66	42,34	32,88	33,97	33,71	33,48	0,9346	0,9433	0,954	0,9442
RCD	17,73	<b>20,69</b>	22,75	20,4	36,86	<b>36,04</b>	37,18	36,76	0,9713	<b>0,9539</b>	0,9768	0,9687
Set Mean	47,77	43,86	53,66	48,92	34,94	33,68	34,71	34,52	0,9381	0,9311	0,951	0,941

Tabelle 1: MSE, PSNR und SSIM Mittelwerte in den drei Bildersammlung Kodak, McMster und MSR. Das jeweils beste Ergebnis in einer Spalte ist hervorgehoben.

In Tabelle 2 sind die durchschnittlichen MSE-, PSNR und SSIM-Werte, die im Kapitel 4.1 vorgestellt wurden, innerhalb des MSR Sets mit Rauschen, das im Kapitel 4.2 vorgestellt wurde, von jedem in JENIFFER2 implementierten Demosaicing-Algorithmus dargestellt. Im rechten Abschnitt der Tabelle, sind die Differenzen der jeweiligen Werte zwischen dem MSR Set mit Rauschen und ohne Rauschen abgebildet. Diese sollen den Qualitätsverlust durch Rauschen einfacher erkennbar machen. Alle Werte wurden identisch zu den Werten in Tabelle 1 berechnet. Hier ist zu beachten, dass die Differenz des PSNR Werts nicht als eine Robustheit gegenüber Rauschen zu deuten ist.

	noisy MSR			difference noisy MSR - MSR		
	MSE	PSNR	SSIM	MSE	PSNR	SSIM
bikubisch	94,07	30,46	0,9012	6,06	-0,42	-0,018
bilinear mean	88,96	30,58	0,908	<b>4,36</b>	-,033	<b>-0,012</b>
bilinear median	86,79	30,75	0,9088	4,54	-0,37	-0,0136
DLMMSE code	21,8	36,56	0,9584	5,47	-1,76	-0,0224
DLMMSE code RCD	20,67	36,81	0,96	5,56	-1,9	-0,0224
DLMMSE paper	21,39	36,64	0,9584	5,53	-1,83	-0,0228
DLMMSE paper RCD	<b>20,27</b>	<b>36,88</b>	<b>0,9612</b>	5,61	-1,96	-0,0212
HA	31,7	35,12	0,9488	5,93	-1,32	-0,0236
MHC	38,89	33,99	0,9412	6,26	-1,07	-0,0256
NN	206,47	26,64	0,8288	7,17	<b>-0,22</b>	-0,0252
PPG	52,3	32,93	0,9324	5,64	-0,78	-0,216
RCD	28,03	35,79	0,956	5,28	-1,39	-0,0208
Set Mean	59,28	33,6	0,9303	5,62	-1,11	-0,0208

Tabelle 2: MSE, PSNR und SSIM Mittlwerte in den MSR mit Rauschen und die Differenz zu dem MSR Set ohne Rauschen. Das jeweils beste Ergebnis in einer Spalte ist hervorgehoben.

In Tabelle 3 sind die durchschnittlichen MSE-, PSNR und SSIM-Werte, die im Kapitel 4.1 vorgestellt wurden, mit einem in Adobe Photoshop und Capture One verarbeiteten Referenzbild von jedem in JENIFFER2 implementierten Demosaicing-Algorithmus dargestellt. Das verwendete Bild hat eine Auflösung von 8368x5584 Pixel. Die Bilder befanden sich vor der Berechnung im Wertebereich von vorzeichenlosen 16-Bit-Integer. Dieser Wertebereich, in Kombination mit der Größe des Bildes, führt zu Unterschieden zwischen den Algorithmen, die erst nach mehreren Nachkommastellen sichtbar werden. Die Werte des MSE wurden mit dem Wert  $10^{-7}$  multipliziert.

In Tabelle 4 sind die durchschnittlichen Ausführungszeiten der verschiedenen Pipelineschritte von JENIFFER2 bei der Wahl verschiedener Demosaicing-Algorithmen dar-

	Photoshop			Capture One		
	MSE	PSNR	SSIM	MSE	PSNR	SSIM
bikubisch	5,36623	19,03277	0,89719	2,53989	22,28133	0,92289
bilinear mean	5,35627	19,04084	0,89649	2,52629	22,30464	0,92616
bilinear median	5,36198	19,03622	0,89504	2,53151	22,29566	0,92483
DLMMSE code	5,34697	19,04839	0,91824	2,51701	22,32061	0,94459
DLMMSE code RCD	<b>5,33986</b>	<b>19,05417</b>	<b>0,91887</b>	<b>2,50973</b>	<b>22,3332</b>	0,94596
DLMMSE paper	5,347	19,04836	9,1826	2,51704	22,32057	0,9446
DLMMSE paper RCD	5,33987	19,05416	0,91875	2,50974	22,3318	<b>0,94599</b>
HA	5,34813	19,04745	0,91562	2,51879	22,31754	0,94151
MHC	5,35427	19,04246	0,91313	2,52755	22,30246	0,93651
NN	5,45186	18,96402	0,85429	2,61518	22,15445	0,88241
PPG	5,351	19,04511	0,90731	2,52259	22,311	0,93421
RCD	5,34146	19,05287	0,91664	2,51144	22,33024	0,94399
Set Mean	5,35874	19,0389	0,90582	2,5289	22,30041	0,9328

Tabelle 3: MSE, PSNR und SSIM Werte mit einem verarbeiteten Bild aus Adobe Photoshop und Capture One als Referenz. MSE-Werte wurden mit dem Faktor  $10^{-7}$  herunterskaliert.

gestellt. Für alle Zeitmessungen wurde dasselbe Eingabebild mit einer Auflösung von 8424x5632 Pixel verwendet. Für jeden Algorithmus wurden zehn Messungen durchgeführt. Die Mittelwerte der jeweiligen zehn Messungen sind in Tabelle 4 dargestellt. Diese Messungen wurden analog zu den Messungen in der Quelle [3, S. 73–74] durchgeführt. Das System, auf dem JENIFFER2 ausgeführt wurde, hat folgende Eigenschaften:

- **Betriebssystem** Windwos 10 Home - 64-Bit - Version 21H2
- **Prozessor** 11th Gen Intel Core i7-11700 @ 2,5 GHz mit 8 Rechenkernen
- **Arbeitsspeicher** 32,0 GB, davon 31,9 GB verwendbar

In Abbildung 29 sind die durchschnittlichen Ausführungszeiten der verschiedenen Demosaicing-Algorithmen sortiert aufgelistet. In Abbildung 30 sind die verschiedenen Demosaicing-Algorithmen bezüglich ihrer Ausführungszeit und PSNR abgebildet. Die Algorithmen wurden nach ihrer Ausführungszeit sortiert.

	Rohdaten-zuordnung	Weiß-abgleich	Demosaicing	Farbraum-transformation	Gamma-korrektur	Sum
bikubisch	802,4	211,9	506,6	382,5	3154,8	5058,2
bilinear mean	832,3	212,6	489	345,6	3110,9	4990,4
bilinear median	835,7	212,9	769,4	333,2	3084,9	5236,1
DLMMSE code	800,7	212,6	7927,7	318,6	3041,3	12300,9
DLMMSE code RCD	781,6	210,2	28982,8	317,9	3015,4	33307,9
DLMMSE paper	838,8	218,9	9428,7	347,7	3121,9	13955,4
DLMMSE paper RCD	823,6	218	31639,9	336	3112,3	36129,8
HA	850,3	222	720,5	376,8	3151,9	5321,5
MHC	832	221,7	511,4	382,1	3220	5167,2
NN	843,3	212,3	485,2	330,2	3175,7	5046,7
PPG	819,6	220,8	960,7	347,2	3092,8	5441,1
RCD	832,9	216,4	29355,7	335,9	3133,3	33874,2
Mean	832,4	215,9	9314,8	346,1	3117,9	13819,1

Tabelle 4: Ausführungszeiten in ms der einzelnen Verarbeitungsschritte für jeden Demosaicing-Algorithmus analog zu [3, S. 74]

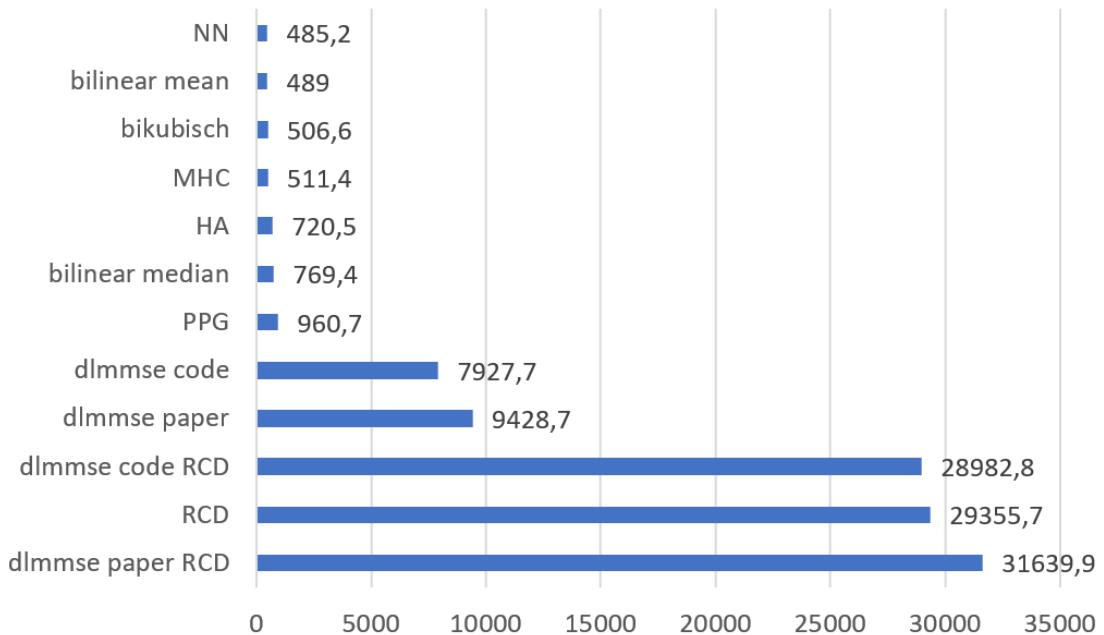
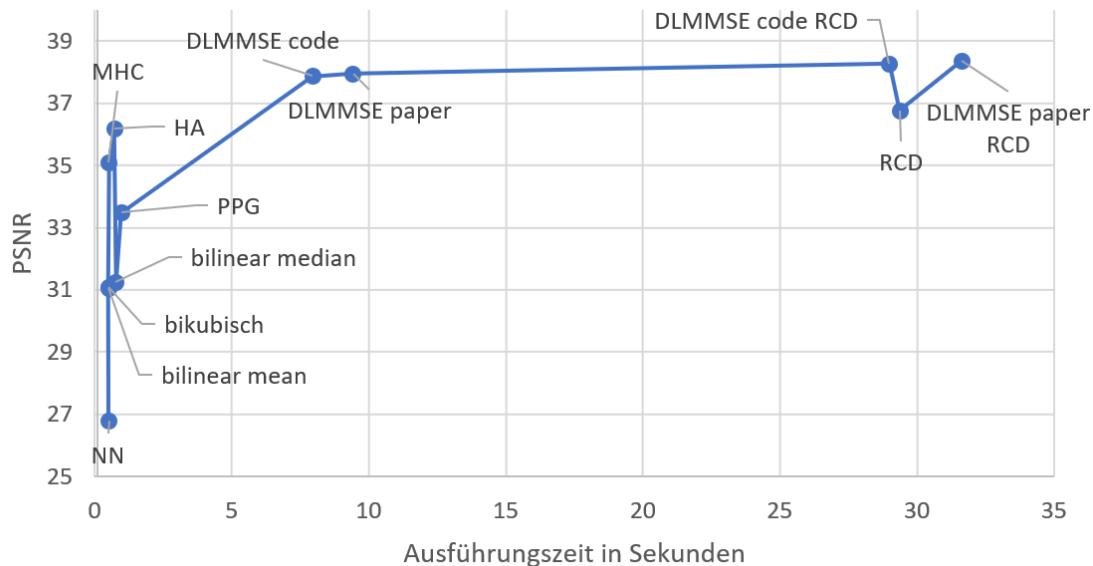
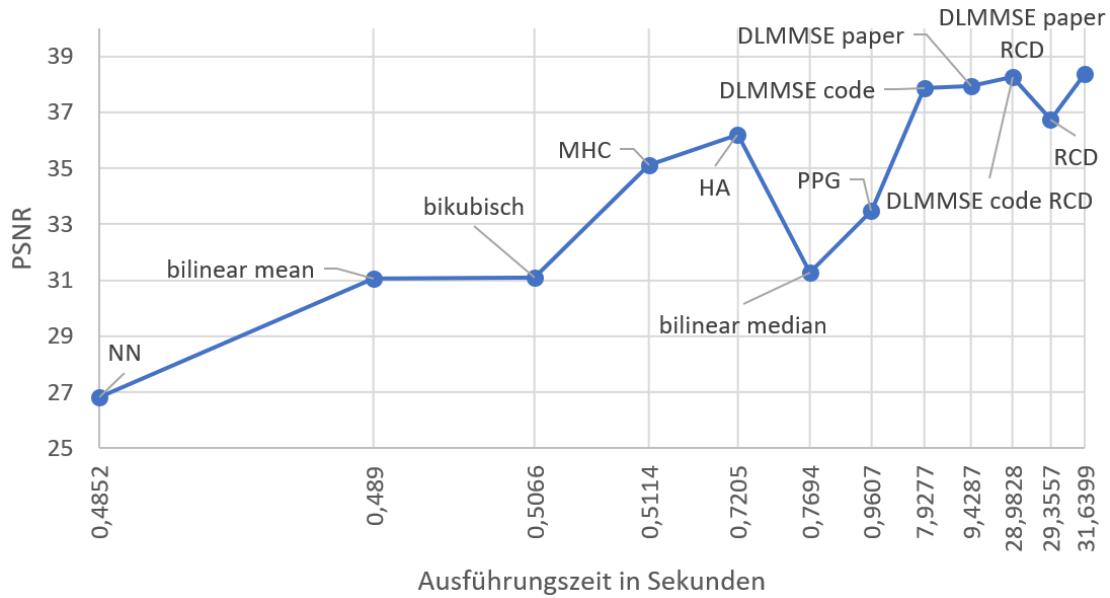


Abbildung 29: Ausführungszeiten der Demosaicing-Algorithmen analog zu [3, S. 75]



(a) Ausführungszeiten gegenüber PSNR



(b) Ausführungszeit logarithmisch skaliert gegenüber PSNR

Abbildung 30: Ausführungszeiten der in JENIFFER2 implementierten Demosaicing-Algorithmen gegenüber ihrer durchschnittlichen PSNR-Werte

## 4.5. Diskussion

### 4.5.1. Auswertung der Bildqualität der implementierten Demosaicing-Algorithmen

- **Nearest Neighbour**

NN erzeugt bei jedem Bild bezüglich der Parameter MSE, PSNR und SSIM die schlechtesten Ergebnisse. Eine Ausnahme ist die Verschlechterung in PSNR, zwischen dem MSR Set mit Rauschen und ohne Rauschen in Tabelle 2. Dies liegt aber an der Eigenschaft des PSNR Werts. Dieser war bereits in dem MSR Set ohne Rauschen sehr niedrig. In Bildausschnitten mit vielen kleinen Details erzeugt dieser Algorithmus erhebliche False-Color-Artefakte. Entlang von Kanten werden starke Zipper-Artefakte erzeugt. Ausschließlich uniforme Flächen werden originalgetreu rekonstruiert. Dieser Algorithmus erzeugt die subjektiv schlechtesten Bilder.

- **Bilineare Interpolation mit Mittelwert**

Dieser Algorithmus bewegt sich hinsichtlich MSE, PSNR und SSIM im unteren Drittel der untersuchten Algorithmen. Am besten schneidet dieser Algorithmus im Vergleich zu den anderen Algorithmen im Kodak Set und am schlechtesten im McMaster Set ab. Im Vergleich zwischen dem MSR Sets mit und ohne Rauschen erzeugt dieser Algorithmus die wenigsten Verluste. Dieser Algorithmus erzeugt erhebliche False-Color-Artefakte in nicht uniformen Bereichen und Zipper-Artefakte entlang von Kanten. Details werden verwischt und verlieren an Auflösung.

- **Bilineare Interpolation mit Median**

Dieser Algorithmus bewegt sich bezüglich MSE, PSNR und SSIM im unteren Drittel der untersuchten Algorithmen, ähnlich wie die bilineare Interpolation mit Mittelwert. Am schlechtesten schneidet dieser Algorithmus im McMaster Set ab. Bei fast jedem Bild erzeugt dieser Algorithmus geringfügig bessere Ergebnisse als die bilineare Interpolation mit Mittelwert. Lediglich im Vergleich mit den Bildern aus Photoshop und Capture One in Tabelle 3 verliert dieser Algorithmus im direkten Vergleich mit der anderen bilinearen Variante. Dieser Algorithmus erzeugt erhebliche False-Color-Artefakte in nicht uniformen Bereichen. Details werden verwischt, insbesondere Kanten. Dafür werden an Kanten keine Zipper-Artefakte erzeugt wie bei der bilinearen Interpolation mit Mittelwert. Ähnlich wie die bilineare Interpolation mit Mittelwert weist dieser Algorithmus geringe Verluste im Vergleich zu den Werten des MSR Sets mit Rauschen und ohne Rauschen auf.

- **Bikubische Interpolation**

Dieser Algorithmus erzielt in fast jeder Bildersammlung die schlechtesten Ergebnisse hinsichtlich MSE, PSNR und SSIM und wird nur von NN unterboten. Die Ausnahme sind hier die Ergebnisse im McMaster Set. Hier erzielt die bikubische Interpolation die besten Ergebnisse im Vergleich zu den anderen Algorithmen. Dieser Algorithmus erzeugt False-Color-Artefakte in ähnlicher Quantität wie die Varianten der bilinearen Interpolation. Starke Zipper-Artefakte werden nicht erzeugt. Dennoch sind Unregelmäßigkeiten entlang von Kanten erkennbar. Trotz der erhöhten Komplexität gegenüber den Varianten der bilinearen Interpolation, kann dieser Algorithmus nur bessere Ergebnisse bei Bildern mit den Eigenschaften des McMaster Sets erzielen.

- **Malvar-He-Cutler Interpolation**

MHC bewegt sich hinsichtlich MSE, PSNR und SSIM im Vergleich zu den anderen Algorithmen in der Mitte. In manchen Bildern des McMaster Sets kann MHC bessere Ergebnisse erzielen als DLMMSE. MHC hat zwar eine ähnliche Komplexität wie die bilineare Interpolation und die bikubische Interpolation, erzielt aber wesentlich bessere Ergebnisse. Dieser Algorithmus erzeugt False-Color-Artefakte in detailreichen Regionen, allerdings nicht so sehr wie die bilineare und bikubische Interpolation. Entlang von Kanten erzeugt dieser Algorithmus Unregelmäßigkeiten. Diese sind besonders auffällig in Bereichen mit stark gesättigten Farben, wie in Abbildung 24j.

- **Patterned Pixel Grouping**

PPG erzielt hinsichtlich MSE, PSNR und SSIM die schlechtesten Ergebnisse im Vergleich zu den anderen Algorithmen, die in dieser Masterarbeit neu implementiert wurden. Lediglich in Bildern mit einheitlichen Flächen und weichen Kanten kann PPG bessere Ergebnisse erzielen. In detailreichen Flächen erzeugt der Algorithmus False-Color-Artefakte. Obwohl dieser Algorithmus laut dem Autor für künstliche Szenen mit klaren Grenzen konzipiert wurde (vgl. Kapitel 3.2.8), erzeugt der Algorithmus entlang solcher Kanten unregelmäßige Strukturen, wie in Abbildung 26l. In Bereichen in welchen die Abtastrate zu niedrig ist, können unregelmäßige Muster erzeugt werden, wie in Abbildung 23l.

- **Hamilton-Adams Interpolation**

HA erzielt hinsichtlich MSE, PSNR und SSIM Ergebnisse, die diesen Algorithmus im oberen Mittelfeld der untersuchten Algorithmen platzieren. Im Kodak Set erzielt HA im Durchschnitt bessere Ergebnisse als RCD und im McMaster Set erzielt HA im Durchschnitt bessere Ergebnisse als DLMMSE. In schwierigen Bereichen wie in Abbildung 23i kann HA gute Ergebnisse erzielen und wird hier nur von den DLMMSE Varianten übertroffen. Ähnlich wie PPG erzeugt HA dort aber unregelmäßige Muster wegen seiner adaptiven Eigenschaft, jedoch in einem geringeren Maß. Entlang von Kanten mit stark gesättigten Farben erzeugt HA auch ähnliche Artefakte wie MHC in Abbildung 24i. Deutlich bessere Ergebnisse als PPG erzielt HA aber entlang der Kanten in Abbildung 26i. Dort sind keine unregelmäßigen Strukturen erkennbar. HA erzeugt auch False-Color-Artefakte in detailreichen Regionen, allerdings weniger als die bisher aufgelisteten Algorithmen.

- **Directional Linear Minimum Mean-Square Error Interpolation**

DLMMSE erzielt hinsichtlich MSE, PSNR und SSIM im Durchschnitt die zweitbesten Ergebnisse. Innerhalb des Kodak Sets kann DLMMSE teilweise die besten Ergebnisse erzielen. Am schlechtesten ist die Performance von DLMMSE im McMaster Set. Dort liegt dieser Algorithmus sogar hinter HA und teilweise hinter PPG und MHC. In detaillierten Bereichen erzeugt DLMMSE fast keine erkennbaren Artefakte. In den schwierigen Strukturen von Abbildung 23 erzeugt dieser Algorithmus, neben der Kombination von DLMMSE und RCD, die wenigsten Aliasing-Effekte. Dafür hat DLMMSE Probleme entlang von Kanten. Besonders bei stark gesättigten Farben treten Artefakte auf wie in Abbildung 24, ähnlich zu HA und MHC. Die beiden Varianten des DLMMSE Algorithmus erzielen im Wesentlichen ähnliche Ergebnisse. Unterschiede sind mit dem menschlichen Auge fast nicht zu erkennen. Der Code Estimator erzielt im McMaster Set bessere Ergebnisse, der Paper Estimator in den anderen beiden Sets.

- **Ratio Corrected Demosaicing**

RCD erzielt hinsichtlich MSE, PSNR und SSIM im Durchschnitt nach den verschiedenen DLMMSE Varianten die besten Ergebnisse. Innerhalb des McMaster Sets erzielt RCD die besten Resultate. Im Vergleich zu den Bildern aus Photoshop und Capture One, erzielt RCD nach dem Kombinationsalgorithmus aus DLMMSE und RCD die besten Ergebnisse. Innerhalb des Kodak Sets sind die Werte von

RCD am schlechtesten. Hier wird RCD sogar von HA übertroffen. Im Vergleich zwischen den MSR Sets mit und ohne Rauschen erzielt RCD, hinsichtlich MSE und SSIM gute Ergebnisse. Nur die beiden bilinearen Varianten können RCD hier deutlich übertreffen. In Bereichen mit detaillierten Strukturen, wie in Abbildung 22m, erzeugt RCD nur wenige False-Color-Artefakte. Aliasing-Effekte treten auch verminderd in Abbildung 23m auf. DLMMSE ist in diesen beiden Beispielen jedoch besser. RCD erzeugt die optisch besten Ergebnisse in Bildern mit stark gesättigten Farben und entlang von Kanten.

- **DLMMSE Grünkanalinterpolation mit RCD Rot- und Blaukanalinterpolation**

Dieser Algorithmus erzielt hinsichtlich MSE, PSNR und SSIM im Durchschnitt die besten Ergebnisse. Lediglich im McMaster Set erzielt RCD bessere Ergebnisse. Auch im Vergleich zu den Bildern aus Photoshop und Capture One kann dieser Kombinationsalgorithmus die besten Werte erzielen. Im Wesentlichen ähneln die optisch erkennbaren Eigenschaften dieses Algorithmus denen von DLMMSE. In detaillierten Bereichen werden fast keine erkennbaren Artefakte erzeugt. Aliasing-Effekte werden gut unterdrückt. Allerdings entstehen auch dieselben Fehler wie bei DLMMSE in Bildern mit stark gesättigten Farben und entlang von Kanten. Dieser Algorithmus unterdrückt diese Fehler jedoch besser als DLMMSE. Dies ist in den Abbildungen der Differenzbilder 26r und 26t am besten erkennbar. DLMMSE erzeugt entlang der kompletten Kante Artefakte. Bei dem Kombinationsalgorithmus treten diese nur in den Ecken auf. Die Unterschiede zwischen dem Code Estimator und Paper Estimator verhalten sich bei diesem Algorithmus wie bei DLMMSE.

Zusammenfassend lässt sich feststellen, dass man in JENIFFER2 wahrscheinlich die besten Bilder erhält, wenn man bei Bildern mit vielen Details den Kombinationsalgorithmus aus DLMMSE und RCD wählt. Bei Bildern mit vielen klaren Kanten oder stark gesättigten Farben sollte RCD gewählt werden. Ein eindeutig bester Algorithmus lässt sich aber nicht bestimmen. Hier wurde gezeigt, dass der beste Algorithmus bezüglich der drei berechneten Messwerte, von den Eigenschaften des Bilds abhängt. Hätte man beispielsweise nur Bilder aus dem McMaster Set untersucht, dann wäre RCD der beste Algorithmus. Hätte man nur Bilder aus dem Kodak Set untersucht, dann wäre RCD hinter dem Kombinationsalgorithmus, DLMMSE und HA gelandet. Der ideale Algorithmus muss also in Abhängigkeit des Bildes und der Präferenzen des Nutzers gewählt werden.

#### 4.5.2. Auswertung der Laufzeiten der Demosaicing-Algorithmen

Um die Ausführungszeit der Demosaicing-Algorithmen zu reduzieren, wurde deren Ausführung parallelisiert. Die Verkürzung der Ausführungszeit lässt sich durch den Vergleich der Tabelle in Abbildung 14 und der Tabelle 4 erkennen. Vor dieser Optimierung benötigte der Demosaicing-Schritt bei NN 10,1%, bei bilinearer Interpolation mit Mittelwert 14,3%, bei bilinearer Interpolation mit Median 24,7% und bei der bikubischen Interpolation 31,9% der gesamten Verarbeitungsdauer. Nach der Optimierung benötigt der Demosaicing-Schritt bei NN 9,6%, bei bilinearer Interpolation mit Mittelwert 9,8%, bei bilinearer Interpolation mit Median 14,7% und bei der bikubischen Interpolation 10% der gesamten Verarbeitungsdauer. Die geringste Beschleunigung erhält der schnellste Algorithmus NN durch die Parallelisierung. Die größte Verbesserung erhält der zuvor längste Algorithmus, die bikubische Interpolation. Mit zunehmender Laufzeit scheint der Gewinn durch die Parallelisierung größer zu werden. Die Ausnahme ist hier die bilineare Interpolation mit Median. Dieser Algorithmus hat nun die längste Laufzeit im Vergleich von diesen vier Algorithmen. Die separate Funktion, die den Medianwert durch Sortieren einer Liste bestimmt, benötigt vermutlich sehr lange. Der Algorithmus mit der längsten Laufzeit, DLMMSE-PaperEst + RCD benötigt 87,6% des gesamten Bildverarbeitungsvorgangs. Hier wird deutlich, dass der Demosaicing-Schritt den Großteil der Verarbeitungszeit beansprucht. Dies verdeutlicht die Notwendigkeit von Optimierungen der Demosaicing-Algorithmen. Sollten noch rechenintensivere Algorithmen implementiert werden, wie ARI aus Kapitel 3.3, sind außerdem weitere Verbesserungen wie die Implementierung einer Hardwarebeschleunigung notwendig.

Abbildung 29 zeigt die Laufzeiten der verschiedenen Demosaicing-Algorithmen. Die meisten Algorithmen benötigen weniger als eine Sekunde. DLMMSE benötigt deutlich länger. Die Faltungen mit verschiedenen Filtern und eine hohe Anzahl an Operationen führt zu dieser Dauer. Außerdem ist erkennbar, dass der Code Estimator schneller ist als der Paper Estimator. Zwischen RCD und DLMMSE ist ein weiterer großer Sprung bei der Rechenzeit. Da die Kombinationsalgorithmen aus DLMMSE und RCD eine ähnliche Rechenzeit wie RCD haben, lässt sich dieser große Sprung auf die Berechnung des roten und blauen Kanals zurückführen. Dieser benötigt bei RCD wesentlich länger, da die fehlenden Rot- und Blauwerte an roten und blauen Stellen in einer separaten Schleife von den grünen Stellen berechnet werden.

Mit Abbildung 30 sollen die Ausführungszeiten der Algorithmen in ein Verhältnis zu ihrer Bildqualität gesetzt werden. Dies wurde auch in anderen Arbeiten getan [25, S. 14–15][53, S. 275]. Aus Abbildung 30 lässt sich ein grober Trend erkennen, dass mit zuneh-

mender Rechenzeit, ein höherer PSNR Wert erzielt wird. Die drei Algorithmen bilineare Interpolation mit Median, PPG und RCD folgen diesem Trend nicht eindeutig. Die Messungen aus der Quelle [25, S. 14–15] weisen denselben Trend auf, dass mit zunehmender Rechenzeit eine höhere Bildqualität erzielt werden kann. Aus Abbildung 30 und der Quelle [25, S. 14–15] lässt sich außerdem der Trend ablesen, dass der Zugewinn an Bildqualität mit zunehmender Rechenzeit kleiner wird. Eindeutig lassen sich diese beiden Trends aber nicht nachweisen. Die Laufzeit der Algorithmen hängt nicht nur von der Anzahl ihrer Operationen, sondern auch maßgeblich von ihrer Implementierung ab. In der Quelle [25, S. 14–15] wurden die Algorithmen auf verschiedener Software ausgeführt und nur teilweise mit paralleler Berechnung implementiert. In der Quelle [53, S. 275] lässt sich zudem erkennen, dass Algorithmen, die lernbasierte Verfahren nutzen, dem Trend von herkömmlichen Algorithmen nicht folgen. Lernbasierte Algorithmen können eine sehr hohe Bildqualität erzielen und benötigen dafür relativ kurze Ausführungszeiten.

#### 4.5.3. Vorteile des kombinierten DLMMSE + RCD Algorithmus

- Mit den Messungen aus Tabelle 1 und 2 konnte gezeigt werden, dass der Kombinationsalgorithmus aus DLMMSE und RCD im Durchschnitt bessere Ergebnisse erzielen kann als die Algorithmen, aus denen er kombiniert wurde. DLMMSE wurde in den meisten Bildern übertroffen oder es wurden sehr ähnliche Werte erzielt. Besonders im McMaster Set kann der Kombinationsalgorithmus wesentlich bessere Ergebnisse als DLMMSE erzielen.
- Die größte sichtbare Schwäche von DLMMSE wurde entlang von Kanten sichtbar. Besonders bei stark gesättigten Farben kann ein sehr auffälliges Fehlermuster entstehen. Durch die Berechnung des roten und blauen Kanals von RCD konnte die Intensität dieses Fehlermusters sichtbar reduziert werden.
- Mit dem hier vorgestellten Kombinationsalgorithmus konnte gezeigt werden, dass verschiedene Komponenten eines Algorithmus an verschiedene Anforderungen angepasst werden können. Damit können maßgeschneiderte Lösungen, ohne die aufwendige Entwicklung von komplett neuen Algorithmen gefunden werden. Algorithmen können an die Eigenschaften von Bildern, oder eine gewünschte Laufzeit angepasst werden. Jedoch ist ein tieferes Wissen über die zur Verfügung stehenden Komponenten der Algorithmen notwendig.

## 5. Fazit und Ausblick

In dieser Arbeit wurden Demosaicing-Algorithmen, ihre Relevanz und ihre Eigenschaften erläutert. Ausgewählte Demosaicing-Algorithmen wurden in JENIFFER2 implementiert und detailliert ausgewertet. Diese Erkenntnisse sowie potenzielle weitere Verbesserungen von JENIFFER2 werden im Folgenden zusammengefasst.

### 5.1. Fazit

In dieser Arbeit wurde der RAW-Konverter JENIFFER2 mit neuen Demosaicing-Algorithmen erweitert und untersucht, welche Bildqualität die verschiedenen Algorithmen erzielen können.

Zunächst wurde erläutert, wieso man Demosaicing benötigt. Diese werden wegen der Eigenschaften von Bildsensoren benötigt, da diese nur Graustufenbilder erzeugen können und deswegen um ein CFA erweitert werden müssen. Die verschiedenen Demosaicing-Algorithmen können anhand ihrer Eigenschaften in verschiedene Kategorien eingeteilt werden. Dazu gehören adaptive- und nicht-adaptive-Algorithmen, Farb-Differenz-Demosaicing und residuales Demosaicing, Demosaicing in der räumlichen- und Frequenz-Domäne und Wavelet-basierte Algorithmen. Außerdem entstehen bei Bildern, die durch Demosaicing-Algorithmen erzeugt werden, typische Artefakte. Diese wurden erläutert und die zwei relevantesten Artefakte, die Zipper-Artefakte und False-Color-Artefakte, erklärt. Das RAW-Format kann verwendet werden, um unverarbeitete Bilder zu speichern, welche mit dem gewünschten Demosaicing-Algorithmus verarbeitet werden können. Allerdings gibt es viele verschiedene proprietäre RAW-Formate. Das DNG-Format soll dieses Problem lösen. Es wurde gezeigt, wie verschiedene RAW-Formate mit dem Adobe DNG Converter in das DNG-Format überführt werden können. Die Verarbeitungspipeline von JENIFFER2 wurde vorgestellt. Die wichtigsten Verarbeitungsschritte wurden erläutert und es wurde gezeigt, dass manche Verarbeitungsschritte nur ausgeführt werden, wenn die dafür notwendigen Tags in der DNG-Datei vorhanden sind.

Anschließend wurde das Programm JENIFFER2 selbst näher beschrieben. Die grafische Benutzeroberfläche von JENIFFER2 wurde präsentiert und erklärt, zusammen mit den drei Komponenten DNG-Reader, DNG-Prozessor und Benutzeroberfläche. Die Komponente DNG-Prozessor wurde näher erläutert und wie sie die verschiedenen Pipelineschritte kombiniert. In JENIFFER2 sind nun zehn verschiedene Demosaicing-Algorithmen implementiert, wobei zwei dieser Algorithmen zwei Variationen haben. Die neu imple-

mentierten Algorithmen wurden aus Algorithmen ausgewählt, die in wissenschaftlicher Literatur oft referenziert werden oder in anderen RAW-Konvertern genutzt werden. Die implementierten Algorithmen wurden in ihren Eigenschaften wie Bildqualität und Rechenzeit sowie in ihrer Funktionsweise und Aufbau beschrieben. Zusätzlich wurde ein neuer Demosaicing-Algorithmus, aus Bestandteilen von RCD und DLMMSE entworfen. Es wurden weitere Demosaicing-Algorithmen präsentiert, die aktuell die besten Resultate bezüglich der Bildqualität erzielen. Dazu gehört der stärkste Vertreter der residualen Interpolation ARI und Vertreter von lernbasierten Algorithmen wie JCNN und RCNN. Diese Algorithmen konnten allerdings wegen ihrer hohen Komplexität, Programmiersprache und hohen Rechenzeit nicht in JENIFFER2 implementiert werden. Neben den neuen Demosaicing-Algorithmen wurden weitere kleine Veränderungen an JENIFFER2 vorgenommen. Dazu gehören die Parallelisierung der Demosaicing-Algorithmen, eine automatische Speicherung und Wiederherstellung der Dateinavigation und die Aktivierung des Exposure-Moduls. Außerdem wurde die Ausführung von Demosaicing-Algorithmen optimiert, um die Ausführungszeit zu optimieren. Dafür werden zuerst alle fehlenden Grünwerte berechnet und zwischengespeichert und anschließend die fehlenden Rot- und Blauwerte.

Abschließend wurde die Bildqualität und Laufzeit der in JENIFFER2 implementierten Demosaicing-Algorithmen untersucht und ausgewertet. Dafür wurden die Parameter MSE, PSNR und SSIM, sowie die Image Datasets von Kodak, McMaster und MSR vorgestellt und genutzt. Ergänzend wurden die Bilder visuell verglichen. Dabei konnte festgestellt werden, dass NN, die bilineare Interpolation mit Mittelwert und Median und die bikubische Interpolation die schlechteste Bildqualität erzeugen. Die beste Bildqualität konnte von DLMMSE und RCD sowie dem Kombinationsalgorithmus erzeugt werden. Dabei ist der Kombinationsalgorithmus bei fast jedem untersuchten Bild besser als DLMMSE. DLMMSE und der Kombinationsalgorithmus erzielen die besseren Ergebnisse bei Bildern mit vielen kleinen Details und bei hoher spektraler Korrelation. Dafür entstehen bei Bildern mit hoher Farbsättigung und scharfen Strukturen mit abrupten Farbtransitionen deutliche Artefakte. Hier kann dafür RCD die besten Ergebnisse erzielen. RCD ist auch robuster gegenüber Rauschen als DLMMSE. Es wurde auch festgestellt, dass der Kombinationsalgorithmus den Ergebnissen der Demosaicing-Algorithmen von Photoshop und Capture One am meisten ähnelt. Im Vergleich der Laufzeiten der Demosaicing-Algorithmen konnte die Beschleunigung durch die parallele Ausführung gezeigt werden. Des Weiteren liefert HA die beste Qualität der Algorithmen, die weniger als eine Sekunde benötigen und MHC hat das beste Verhältnis von Bildqua-

lität zu Laufzeit. RCD und der Kombinationsalgorithmus benötigen mit Abstand am längsten, gefolgt von DLMMSE. Es konnte auch ein Trend identifiziert werden, dass die erzielte Bildqualität mit höherer Rechenzeit zunimmt. Außerdem scheint der Zugewinn an Bildqualität mit zunehmender Rechenzeit zu sinken. Diese zwei Eigenschaften sind allerdings nur Faustregeln. Es wurde gezeigt, dass diese Trends auch gebrochen werden und von der Implementation und Optimierung der Algorithmen abhängt. Es wurde gezeigt, dass der Kombinationsalgorithmus aus DLMMSE und RCD verschiedene Vorteile hat. Daraus kann abgeleitet werden, dass verschiedene Algorithmenkomponente genutzt werden können, um ohne den großen Aufwand der Entwicklung eines neuen Algorithmus, für bestimmte Situationen angepasste Algorithmen zu erstellen.

## 5.2. Ausblick

In der Auswahl, Implementierung und Analyse der Demosaicing-Algorithmen wurde klar, dass die Laufzeit der Hauptfaktor ist, der die Implementierung komplexerer Algorithmen beschränkt. Hier ist entweder eine massive Optimierung des jeweiligen Algorithmus notwendig oder eine Integration von Hardwarebeschleunigung. Hardwarebeschleunigung wird auch von anderen RAW-Konvertern verwendet. Interessant wäre auch die Implementierung von lernbasierten Algorithmen. Die im Rahmen dieser Arbeit untersuchten CNN Algorithmen sind allerdings alle in Python implementiert und benötigen auch Python-Bibliotheken. Hier muss noch untersucht werden, ob und wie sich diese Algorithmen in eine Java-Anwendung übertragen lassen. Ein weiterer potenzieller Algorithmus ist Amaze. Dieser wird in den Raw-Konvertern RawTherapee und Darktable verwendet und wird dort auch teilweise als Standardalgorithmus genutzt. Dieser Algorithmus ist allerdings sehr komplex. Zum Vergleich: die Implementierung von RCD in RawTherapee benötigt 348 Zeilen und die Implementierung von Amaze 1610 Zeilen.

Eine weitere mögliche Erweiterung von JENIFFER2 wäre die Ausweitung von lesbaren DNG-Dateien. Hier können aktuell nicht alle Dateien von JENIFFER2 eingelesen werden, die sich im DNG-Format befinden. Ein Beispiel dafür sind die DNG-Dateien, deren CFA-Mosaik bereits aufgelöst wurde. Solche Dateien haben bei dem Tag PhotometricInterpretation den Wert 34892 LinearRaw und werden zum Beispiel von iPhones generiert. Denkbar ist es auch DNG-Dateien einzulesen, die nicht das Bayer-Mosaik verwenden. Manche CFAs können von den bereits in JENIFFER2 implementierten Demosaicing-Algorithmen aufgelöst werden. Bei CFAs wie dem X-Trans Sensor von Fujifilm sind jedoch andere Demosaicing-Algorithmen notwendig.

RAW-Konverter bieten auch meist weitere Bearbeitungsfunktionen des Bildes an, die über die Benutzeroberfläche gesteuert werden können. Diese könnten auf das Bild angewendet werden, nachdem es die Verarbeitungspipeline durchlaufen hat. Solche Bearbeitungsfunktionen könnten das Zuschneiden und Rotieren oder das nachträgliche Anpassen von Farbwerten, Sättigung, Belichtung, Schärfung und vielen weiteren Parametern ermöglichen. Eine einzelne Operation wie das Ändern der Sättigung ist vermutlich nicht sehr aufwendig, benötigt aber auch eine nutzerfreundliche Oberfläche. Hier müsste zusätzlich neben den zugrundeliegenden Algorithmen untersucht werden, wie nutzerfreundliche Interaktionen ermöglicht werden.

## Literaturverzeichnis

- [1] Digital Preservation at the Library of Congress. *Camera Raw Formats (Group Description)*. 2016. URL: <https://www.loc.gov/preservation/digital/formats/fdd/fdd000241.shtml>. (Zugriff: 01.12.2022).
- [2] Thomas Walter, Claudia Grosch, Joachim Groß und Michael Kessler. „Mit JENIFFER zum digitalen Highend-Bild“. In: *Infromatik Spektrum* 28.5 (2005), S. 437–438.
- [3] Eugen Ljavin. „JENIFFER2: Ein RAW-Processor mit wählbaren Demosaicing-Algorithmen für das universelle RAW-Format DNG“. Masterarbeit. Universität Tübingen, 2020.
- [4] Rastislav Lukac. *Single-sensor imaging: methods and applications for digital cameras*. Bd. 9. Image processing series. CRC Press, 2009.
- [5] Adam Coupe. *The Benefits of RAW Format*. 2017. URL: <https://adamcoupe.com/benefits-raw-architectural-photographer-london/>. (Zugriff: 01.12.2022).
- [6] Thomas Maschke. *Digitale kameratechnik: technik digitaler kameras in theorie und praxis*. Springer-Verlag, 2004.
- [7] Abbas El Gamal und Helmy Eltoukhy. „CMOS image sensors“. In: *IEEE Circuits and Devices Magazine* 21.3 (2005), S. 6–20.
- [8] Thomas Walter. *MediaFotografie—analog und digital: Begriffe, Techniken, Web*. Springer, 2005.
- [9] Dave Litwiller. „Ccd vs. cmos“. In: *Photonics spectra* 35.1 (2001), S. 154–158.
- [10] RawPedia. *Demosaicing*. 2019. URL: <https://rawpedia.rawtherapee.com/Demosaicing>. (Zugriff: 01.12.2022).
- [11] Junichi Nakamura. *Image sensors and signal processing for digital still cameras*. CRC press, 2006.
- [12] Bryce E Bayer. *Color imaging array*. US Patent 3,971,065. Juli 1976.
- [13] Bruce Fraser. *Understanding Digital Raw Capture*. Techn. Ber. Adobe Systems Incorporated, 2004. (Zugriff: 01.12.2022).
- [14] Richard Szeliski. *Computer Vision: Algorithms and Applications*. Springer Nature, 2011.
- [15] Rajeev Ramanath, Wesley E Snyder, Griff L Bilbro und William A Sander. „Demosaicking methods for Bayer color arrays“. In: *Journal of Electronic imaging* 11.3 (2002), S. 306–315.
- [16] M Guarnera, G Messina und V Tomaselli. „Demosaicing and Aliasing Correction“. In: *Red* 33.R53 (), S. 149–190.

- [17] Ping-Sing Tsai, Tinku Acharya und Ajay K Ray. „Adaptive fuzzy color interpolation“. In: *Journal of Electronic Imaging* 11.3 (2002), S. 293–305.
- [18] MS Safna Asiq und WR Sam Emmanuel. „Colour filter array demosaicking: a brief survey“. In: *The Imaging Science Journal* 66.8 (2018), S. 502–512.
- [19] Robert A Maschal Jr, S Susan Young, Joe Reynolds, Keith Krapels, Jonathan Fanning und Ted Corbin. *Review of Bayer pattern color filter array (CFA) demosaicing with new quality assessment algorithms*. Techn. Ber. Army Research Lab Adelphi Md Sensors und Electron Devices Directorate, 2010.
- [20] Daisuke Kiku, Yusuke Monno, Masayuki Tanaka und Masatoshi Okutomi. „Residual interpolation for color image demosaicking“. In: *2013 IEEE international conference on image processing*. IEEE. 2013, S. 2304–2308.
- [21] Daisuke Kiku, Yusuke Monno, Masayuki Tanaka und Masatoshi Okutomi. „Minimized-Laplacian residual interpolation for color image demosaicking“. In: *Digital Photography X*. Bd. 9023. SPIE. 2014, S. 197–204.
- [22] Daisuke Kiku, Yusuke Monno, Masayuki Tanaka und Masatoshi Okutomi. „Beyond color difference: Residual interpolation for color image demosaicking“. In: *IEEE Transactions on Image Processing* 25.3 (2016), S. 1288–1300.
- [23] Wei Ye und Kai-Kuang Ma. „Image demosaicing by using iterative residual interpolation“. In: *2014 IEEE International Conference on Image Processing (ICIP)*. IEEE. 2014, S. 1862–1866.
- [24] Wei Ye und Kai-Kuang Ma. „Color image demosaicing using iterative residual interpolation“. In: *IEEE Transactions on Image Processing* 24.12 (2015), S. 5879–5891.
- [25] Yusuke Monno, Daisuke Kiku, Masayuki Tanaka und Masatoshi Okutomi. „Adaptive residual interpolation for color and multispectral image demosaicking“. In: *Sensors* 17.12 (2017), S. 2787.
- [26] Eric Dubois. „Frequency-domain methods for demosaicking of Bayer-sampled color images“. In: *IEEE Signal Processing Letters* 12.12 (2005), S. 847–850.
- [27] David Alleysson, Sabine Susstrunk und Jeanny Héault. „Linear demosaicing inspired by the human visual system“. In: *IEEE Transactions on Image Processing* 14.4 (2005), S. 439–449.
- [28] Nai-Xiang Lian, Lanlan Chang, Yap-Peng Tan und Vitali Zagorodnov. „Adaptive filtering for color filter array demosaicking“. In: *IEEE Transactions on Image Processing* 16.10 (2007), S. 2515–2525.

- [29] Brian Leung, Gwanggil Jeon und Eric Dubois. „Least-squares luma–chroma demultiplexing algorithm for Bayer demosaicking“. In: *IEEE Transactions on Image Processing* 20.7 (2011), S. 1885–1894.
- [30] Bahadir K Gunturk, Yucel Altunbasak und Russell M Mersereau. „Color plane interpolation using alternating projections“. In: *IEEE transactions on image processing* 11.9 (2002), S. 997–1013.
- [31] Da-Cheng Sung und Hen-Wai Tsao. „Color filter array demosaicing by using subband synthesis scheme“. In: *IEEE Sensors Journal* 15.11 (2015), S. 6164–6172.
- [32] Yue M Lu, Mina Karzand und Martin Vetterli. „Demosaicing by alternating projections: Theory and fast one-step implementation“. In: *IEEE Transactions on Image Processing* 19.8 (2010), S. 2085–2098.
- [33] Daniele Menon und Giancarlo Calvagno. „Demosaicing based onwavelet analysis of the luminance component“. In: *2007 IEEE International conference on image processing*. Bd. 2. IEEE. 2007, S. II–181.
- [34] Lanlan Chang und Yap-Peng Tan. „Hybrid color filter array demosaicing for effective artifact suppression“. In: *Journal of Electronic Imaging* 15.1 (2006), S. 013003.
- [35] Adobe Inc. *Digital Negative (DNG) Specification*. Techn. Ber. 2021. (Zugriff: 01.12.2022).
- [36] Adobe Inc. *RAW files*. URL: <https://www.adobe.com/creativecloud/file-types/image/raw.html>. (Zugriff: 01.12.2022).
- [37] Philip Andrews, Yvonne Butler, Yvonne J Butler und Joe Farace. *Raw workflow from capture to archives: a complete digital photographer’s guide to raw imaging*. Focal Press, 2006.
- [38] Camera und Imaging Products Association. *CIPA DC-010-2020 Exif 2.32 meta-data for XMP*. Techn. Ber. 2020. (Zugriff: 01.12.2022).
- [39] Adobe Inc. *DNG files*. URL: <https://www.adobe.com/creativecloud/file-types/image/raw/dng-file.html>. (Zugriff: 01.12.2022).
- [40] Barry Pearson. *Benefits of DNG*. 2011. URL: <http://www.barrypearson.co.uk/articles/dng/benefits.htm#industry>. (Zugriff: 01.12.2022).
- [41] Barry Pearson. *Camera details embedded in DNG*. 2011. URL: <http://www.barrypearson.co.uk/articles/dng/profiles.htm>. (Zugriff: 01.12.2022).
- [42] ISO/TC 42. *Photography — Electronic still picture imaging — Removable memory — Part 2: Image data format — TIFF/EP*. Techn. Ber. 1998. (Zugriff: 01.12.2022).
- [43] Adobe Inc. *Adobe Digital Negative Converter*. 2022. URL: <https://helpx.adobe.com/de/camera-raw/using/adobe-dng-converter.html>. (Zugriff: 01.12.2022).

- [44] Rajeev Ramanath, Wesley E Snyder, Youngjun Yoo und Mark S Drew. „Color image processing pipeline“. In: *IEEE Signal Processing Magazine* 22.1 (2005), S. 34–43.
- [45] Rob Sumner. „Processing raw images in matlab“. In: *Department of Electrical Engineering, University of California Santa Cruz* (2014).
- [46] Yung-Cheng Liu, Wen-Hsin Chan und Ye-Quang Chen. „Automatic white balance for digital still camera“. In: *IEEE Transactions on Consumer Electronics* 41.3 (1995), S. 460–466.
- [47] Apple Inc. *Informationen zu Apple ProRAW*. 2022. URL: <https://support.apple.com/de-de/HT211965>. (Zugriff: 01.12.2022).
- [48] darktable. *demosaic*. URL: <https://docs.darktable.org/usermanual/4.0/en/module-reference/processing-modules/demosaic/>. (Zugriff: 01.12.2022).
- [49] FUJIFILM Corp. *X-Trans CMOS*. 2022. URL: <https://fujifilm-x.com/global/products/x-trans-cmos/>. (Zugriff: 01.12.2022).
- [50] Robert Keys. „Cubic convolution interpolation for digital image processing“. In: *IEEE transactions on acoustics, speech, and signal processing* 29.6 (1981), S. 1153–1160.
- [51] James E Adams Jr und John F Hamilton Jr. *Adaptive color plane interpolation in single sensor color electronic camera*. US Patent 5,652,621. Juli 1997.
- [52] Joan Duran und Antoni Buades. „A demosaicking algorithm with adaptive inter-channel correlation“. In: *Image Processing On Line* 5 (2015), S. 311–327.
- [53] Qiyu Jin, Yu Guo, Jean-Michel Morel und Gabriele Facciolo. „A mathematical analysis and implementation of residual interpolation demosaicking algorithms“. In: *Image Processing On Line* 11 (2021), S. 234–283.
- [54] Pascal Getreuer. „Gunturk-altunbasak-mersereau alternating projections image demosaicking“. In: *Image Processing on Line* 1 (2011), S. 90–97.
- [55] Antoni Buades, Bartomeu Coll, Jean-Michel Morel und Catalina Sbert. „Self-similarity driven color demosaicking“. In: *IEEE Transactions on Image Processing* 18.6 (2009), S. 1192–1202.
- [56] Henrique S Malvar, Li-wei He und Ross Cutler. „High-quality linear interpolation for demosaicing of Bayer-patterned color images“. In: *2004 IEEE International Conference on Acoustics, Speech, and Signal Processing*. Bd. 3. IEEE. 2004, S. 485–488.
- [57] Inc. The MathWorks. *demosaic - Convert Bayer pattern encoded image to truecolor image*. URL: <https://de.mathworks.com/help/images/ref/demosaic.html>. (Zugriff: 01.12.2022).

- [58] James F Bell III, A Godber, S McNair, MA Caplinger, JN Maki, MT Lemmon, J Van Beek, MC Malin, D Wellington, KM Kinch u. a. „The Mars Science Laboratory Curiosity rover Mastcam instruments: Preflight and in-flight calibration, validation, and data archiving“. In: *Earth and Space Science* 4.7 (2017), S. 396–452.
- [59] Chuan-Kai Lin. *Demosaic*. 2010. URL: <https://web.archive.org/web/20160923211135/https://sites.google.com/site/chklin/demosaic/>. (Zugriff: 01.12.2022).
- [60] Dave Coffin. *Decoding raw digital photos in Linux*. 2018. URL: <https://www.dechifro.org/dcraw/>. (Zugriff: 01.12.2022).
- [61] Lei Zhang und Xiaolin Wu. „Color demosaicking via directional linear minimum mean square-error estimation“. In: *IEEE Transactions on Image Processing* 14.12 (2005), S. 2167–2178.
- [62] RawTherapee. *Demosaicing*. 2019. URL: <https://rawpedia.rawtherapee.com/Demosaicing>. (Zugriff: 01.12.2022).
- [63] Chiman Kwan. „A brief review of some interesting Mars Rover image enhancement projects“. In: *Computers* 10.9 (2021), S. 111.
- [64] Thibaud Ehret und Gabriele Facciolo. „A study of two CNN demosaicking algorithms“. In: *Image Processing On Line* 9 (2019), S. 220–230.
- [65] Michaël Gharbi, Gaurav Chaurasia, Sylvain Paris und Frédo Durand. „Deep joint demosaicking and denoising“. In: *ACM Transactions on Graphics (ToG)* 35.6 (2016), S. 1–12.
- [66] Runjie Tan, Kai Zhang, Wangmeng Zuo und Lei Zhang. „Color image demosaicking via deep residual learning“. In: *Proc. IEEE Int. Conf. Multimedia Expo (ICME)*. 2017, S. 793–798.
- [67] Pina Marziliano, Frederic Dufaux, Stefan Winkler und Touradj Ebrahimi. „Perceptual blur and ringing metrics: application to JPEG2000“. In: *Signal processing: Image communication* 19.2 (2004), S. 163–172.
- [68] Bui Duc Tho, Ho PhuocTien und Nguyen Tan Khoi. „DEMOSAICING ALGORITHM-HOW TO EVALUATE IT?“ In: (2018), S. 103–109.
- [69] Inc. The MathWorks. *Image Quality Metrics*. URL: <https://de.mathworks.com/help/images/image-quality-metrics.html>. (Zugriff: 01.12.2022).
- [70] Stefano Andriani, Harald Brendel, Tamara Seybold und Joseph Goldstone. „Beyond the Kodak image set: A new reference set of color image sequences“. In: *2013 IEEE International Conference on Image Processing*. IEEE. 2013, S. 2289–2293.

- [71] Lei Zhang, Xiaolin Wu, Antoni Buades und Xin Li. „Color demosaicking by local directional interpolation and nonlocal adaptive thresholding“. In: *Journal of Electronic imaging* 20.2 (2011), S. 023016.
- [72] *Kodak Lossless True Color Image Suite*. URL: <http://r0k.us/graphics/kodak/>. (Zugriff: 01.12.2022).
- [73] Lei Zhang. *McMaster Dataset*. URL: [https://www4.comp.polyu.edu.hk/~cslzhang/CDM\\_Dataset.htm](https://www4.comp.polyu.edu.hk/~cslzhang/CDM_Dataset.htm). (Zugriff: 01.12.2022).
- [74] Daniel Khashabi, Sebastian Nowozin, Jeremy Jancsary und Andrew W Fitzgibbon. „Joint demosaicing and denoising via learned nonparametric random fields“. In: *IEEE Transactions on Image Processing* 23.12 (2014), S. 4968–4981.
- [75] Alessandro Foi, Mejd Trimeche, Vladimir Katkovnik und Karen Egiazarian. „Practical Poissonian-Gaussian noise modeling and fitting for single-image raw-data“. In: *IEEE Transactions on Image Processing* 17.10 (2008), S. 1737–1754.
- [76] Daniel Khashabi, Sebastian Nowozin, Jeremy Jancsary und Andrew W Fitzgibbon. *Microsoft Research Cambridge Demosaicing Dataset*. URL: <https://www.microsoft.com/en-us/download/details.aspx?id=52535>. (Zugriff: 01.12.2022).

# Anhang

## A. Vollständige Messdaten

In Kapitel 4.4 wurden nur die Mittelwerte der verschiedenen Bildersammlungen bezüglich des jeweiligen Parameters angegeben. Im Folgenden werden alle gemessenen Werte aufgelistet.

Image Number	bikubisch	bilinear mean	bilinear median	DLM/MSE code	DLM/MSE code RCD	DLM/MSE paper	DLM/MSE paper RCD	HA	MHC	NN	PPG	RCD	Image Mean
1	162,71	151,08	149,12	10,74	<b>8,68</b>	9,09	26,47	40,59	308,36	84,95	30,77	82,74	
2	33,55	29,9	29,36	4,81	<b>4,74</b>	4,75	6,94	10,06	66,74	17,24	7,47	18,37	
3	24,31	22,74	21,54	3,79	3,17	<b>3,15</b>	5,85	7,23	49,13	13,66	5,42	13,65	
4	27,57	27,7	27,11	5,4	5,08	<b>5,01</b>	8,3	8,3	69,46	17,3	8,77	17,95	
5	136,67	138,68	128,46	10,35	9,4	<b>10,42</b>	<b>9,37</b>	21,54	30	351,87	64,6	23,24	77,88
6	120,56	107,12	105,74	7,42	7,44	6,86	<b>6,85</b>	20,72	30,63	212,93	61,85	12,9	58,42
7	26,34	28,99	26,23	4,04	<b>3,64</b>	4,09	3,67	5,51	7,66	93,4	14,21	5,22	18,58
8	302,51	282,39	271,7	15,87	15,84	14,1	<b>14</b>	38,08	78,5	674,36	121,23	35,65	155,35
9	37,56	37,77	35,56	4,05	<b>3,9</b>	3,97	9,79	6,72	10,86	104,18	18,07	6,87	23,28
10	38,5	37,07	35,41	4,11	3,96	4,07	<b>3,91</b>	6,95	9,31	97,98	17,48	6,98	22,14
11	86,11	78,22	76,17	7,21	7,22	6,78	<b>6,76</b>	15,17	22,14	171,76	42,46	16,02	44,67
12	31,57	29,46	28,42	3,36	3,33	3,32	<b>3,27</b>	5,64	8,81	68,49	15,81	5,78	17,27
13	299,61	265,77	266,59	23,98	24,34	<b>20,69</b>	20,98	66,68	72,3	520,32	165,55	77,16	152
14	80,1	75,9	72,99	12,81	<b>11,13</b>	13,07	11,28	18,07	23,71	186,73	43,06	18,01	47,24
15	33,39	31,53	30,42	5,88	5,83	5,81	<b>5,74</b>	10,19	10,04	87,42	19,96	11,44	21,47
16	54,58	48,18	47,5	3,26	3,23	3,06	<b>3,01</b>	9,06	14,84	86,42	28,61	7,83	25,8
17	42,13	40,18	38,66	4,81	4,84	<b>4,62</b>	4,63	9,29	11,04	95,94	22,09	10,34	24,05
18	105,16	99,04	99	12,36	12,39	11,8	<b>11,77</b>	24,56	26,7	228,23	58,76	29,34	59,93
19	105,59	101,61	96,64	6,35	6,37	<b>6,06</b>	6,06	12,22	28,48	237,88	42,84	12,77	55,24
20	46,73	44,31	40,83	5,47	5,19	5,29	<b>4,98</b>	9,34	12,62	114,1	22,45	9,72	26,75
21	98,66	91,02	89,65	8,83	8,84	8,06	<b>8,04</b>	18,91	25,45	200,45	52,09	20,14	52,51
22	62,24	58,57	57,61	9,66	9,38	9,54	<b>9,24</b>	14,75	18,9	137,31	32,65	15,7	36,3
23	17,93	20,51	19,1	2,98	<b>2,7</b>	3,01	2,71	4,35	5,31	60,15	11,03	4,39	12,85
24	145,92	135,39	136,3	19,62	19,76	<b>18,9</b>	18,97	39,72	39,28	283,22	85,27	43,68	82,17
Algorithm Mean	88,33	82,63	80,42	8,2	8,02	<b>7,75</b>	7,79	16,88	23,03	187,78	44,72	17,73	

Tabelle 5: MSE des Kodak Sets

Image Number	bikubisch	bilinear median	DLMMSE code	DLMMSE code RCD	DLMMSE paper	DLMMSE paper RCD	HA	MHC	NN	PPG	RCD	Image Mean
1	26,02	26,34	26,4	38,02	37,82	<b>38,74</b>	38,55	33,9	32,05	23,24	28,84	33,25
2	32,87	33,37	33,45	41,31	41,29	<b>41,37</b>	41,36	39,72	38,1	29,89	35,76	39,4
3	34,27	34,56	34,8	42,35	43,13	<u>42,36</u>	<b>43,15</b>	40,46	39,54	31,22	36,78	40,79
4	33,73	33,71	33,8	40,81	41,07	40,84	<b>41,13</b>	38,94	38,94	29,71	35,75	38,7
5	26,77	26,71	27,04	37,98	38,4	37,95	<b>38,41</b>	34,8	33,36	22,67	30,03	34,47
6	27,32	27,83	27,89	39,43	39,42	<b>39,77</b>	<u>39,77</u>	34,97	33,27	24,85	30,22	35,37
7	33,92	33,51	33,94	42,07	<b>42,52</b>	<u>42,02</u>	42,48	40,72	39,29	28,43	36,61	40,96
8	23,32	23,62	23,79	36,13	36,16	36,64	<b>36,67</b>	32,32	29,18	19,84	27,29	32,61
9	32,38	32,36	32,62	42,05	42,21	42,14	<b>42,34</b>	39,86	37,77	27,95	35,56	39,76
10	32,28	32,44	32,64	42	42,16	42,04	<b>42,21</b>	39,71	38,44	28,22	35,7	39,69
11	28,78	29,2	29,31	39,55	39,54	39,82	<b>39,83</b>	36,32	34,68	25,78	31,85	36,09
12	33,14	33,39	33,59	42,87	42,9	42,92	<b>42,99</b>	40,61	38,68	29,77	36,14	40,51
13	23,37	23,89	23,87	34,33	34,27	<b>34,97</b>	<u>34,97</u>	34,91	29,89	29,54	20,97	25,94
14	29,09	29,33	29,5	37,05	<b>37,67</b>	36,97	37,61	35,56	34,38	25,42	31,79	35,58
15	32,9	33,14	33,3	40,44	40,47	40,49	<b>40,54</b>	38,05	38,11	28,71	35,13	37,54
16	30,76	31,3	31,36	43	43,04	43,28	<b>43,34</b>	38,56	36,42	28,76	33,57	39,19
17	31,88	32,09	32,26	41,31	41,28	<b>41,48</b>	<u>41,48</u>	41,47	38,45	37,7	28,31	34,69
18	27,91	28,17	28,17	37,21	37,2	37,41	<b>37,42</b>	34,23	33,87	24,55	30,44	33,46
19	27,89	28,06	28,28	40,1	40,09	<b>40,31</b>	<u>40,31</u>	37,26	33,58	24,37	31,81	37,07
20	31,44	31,67	32,02	40,75	40,98	40,9	<b>41,16</b>	38,43	37,12	27,56	34,62	38,25
21	28,19	28,54	28,61	38,67	38,66	39,07	<b>39,08</b>	35,36	34,07	25,11	30,96	35,09
22	30,19	30,45	30,53	38,28	38,41	38,34	<b>38,47</b>	36,44	35,37	26,75	32,99	36,17
23	35,59	35,01	35,32	43,39	<b>43,82</b>	43,35	43,81	41,75	40,88	30,34	37,71	41,7
24	26,49	26,81	26,79	35,2	35,17	<b>35,37</b>	35,35	32,14	32,19	23,61	28,82	31,73
Algorithm Mean	30,02	30,23	30,39	39,76	39,9	39,94	<b>40,1</b>	37,02	35,69	26,5	32,88	36,86

Tabelle 6: PSNR des Kodak Sets

Image Number	bikubisch	bilinear mean	bilinear median	DLMMSE code	DLMMSE code RCD	DLMMSE paper	DLMMSE paper RCD	HA	MHC	NN	PPG	RCD	Image Mean	
1	0,8	0,81	0,81	<b>0,99</b>	0,98	<b>0,99</b>	<b>0,99</b>	0,97	0,96	0,71	0,9	0,96	0,9058	
2	0,89	0,9	0,9	<b>0,98</b>	<b>0,98</b>	<b>0,98</b>	<b>0,98</b>	0,97	0,96	0,83	0,94	0,97	0,94	
3	0,93	0,93	0,93	<b>0,99</b>	<b>0,99</b>	<b>0,99</b>	<b>0,99</b>	0,98	0,98	0,9	0,96	0,98	0,9625	
4	0,91	0,91	0,91	<b>0,98</b>	<b>0,98</b>	<b>0,98</b>	<b>0,98</b>	0,97	0,97	0,84	0,94	0,97	0,945	
5	0,88	0,88	0,88	<b>0,99</b>	<b>0,99</b>	<b>0,99</b>	<b>0,99</b>	0,98	0,97	0,76	0,94	0,98	0,9358	
6	0,84	0,85	0,85	<b>0,99</b>	<b>0,99</b>	<b>0,99</b>	<b>0,99</b>	0,97	0,96	0,77	0,91	0,97	0,9233	
7	0,96	0,95	0,96	<b>0,99</b>	<b>0,99</b>	<b>0,99</b>	<b>0,99</b>	<b>0,99</b>	<b>0,99</b>	0,98	0,89	0,97	<b>0,99</b>	0,9708
8	0,82	0,83	0,83	<b>0,99</b>	<b>0,99</b>	<b>0,99</b>	<b>0,99</b>	<b>0,99</b>	<b>0,99</b>	0,97	0,96	0,7	0,91	0,97
9	0,91	0,92	0,92	<b>0,98</b>	<b>0,98</b>	<b>0,98</b>	<b>0,98</b>	<b>0,98</b>	<b>0,98</b>	0,97	0,97	0,86	0,95	0,97
10	0,91	0,92	0,92	<b>0,99</b>	<b>0,99</b>	<b>0,99</b>	<b>0,99</b>	<b>0,99</b>	<b>0,99</b>	0,98	0,97	0,85	0,95	0,9533
11	0,86	0,87	0,87	<b>0,99</b>	<b>0,99</b>	<b>0,99</b>	<b>0,99</b>	<b>0,99</b>	<b>0,99</b>	0,97	0,97	0,79	0,93	0,97
12	0,9	0,91	0,91	<b>0,99</b>	<b>0,99</b>	<b>0,99</b>	<b>0,99</b>	<b>0,99</b>	<b>0,99</b>	0,98	0,97	0,86	0,94	0,9508
13	0,76	0,77	0,77	<b>0,98</b>	<b>0,98</b>	<b>0,98</b>	<b>0,98</b>	<b>0,98</b>	<b>0,98</b>	0,95	0,95	0,65	0,87	0,94
14	0,87	0,87	0,87	<b>0,98</b>	<b>0,98</b>	<b>0,98</b>	<b>0,98</b>	<b>0,98</b>	<b>0,98</b>	0,97	0,97	0,78	0,93	0,97
15	0,91	0,92	0,92	<b>0,98</b>	<b>0,98</b>	<b>0,98</b>	<b>0,98</b>	<b>0,98</b>	<b>0,98</b>	0,97	0,97	0,85	0,94	0,97
16	0,87	0,88	0,88	<b>0,99</b>	<b>0,99</b>	<b>0,99</b>	<b>0,99</b>	<b>0,99</b>	<b>0,99</b>	0,98	0,97	0,82	0,93	0,98
17	0,92	0,93	0,93	<b>0,99</b>	<b>0,99</b>	<b>0,99</b>	<b>0,99</b>	<b>0,99</b>	<b>0,99</b>	0,98	0,98	0,86	0,96	0,9583
18	0,87	0,87	0,87	<b>0,98</b>	<b>0,98</b>	<b>0,98</b>	<b>0,98</b>	<b>0,98</b>	<b>0,98</b>	0,97	0,97	0,76	0,92	0,96
19	0,87	0,87	0,87	<b>0,98</b>	<b>0,98</b>	<b>0,98</b>	<b>0,98</b>	<b>0,98</b>	<b>0,98</b>	0,97	0,97	0,79	0,93	0,97
20	0,91	0,92	0,92	<b>0,98</b>	<b>0,98</b>	<b>0,98</b>	<b>0,98</b>	<b>0,98</b>	<b>0,98</b>	0,97	0,97	0,87	0,95	0,97
21	0,88	0,89	0,89	<b>0,98</b>	<b>0,98</b>	<b>0,98</b>	<b>0,98</b>	<b>0,98</b>	<b>0,98</b>	0,97	0,97	0,81	0,93	0,97
22	0,88	0,89	0,89	<b>0,98</b>	<b>0,98</b>	<b>0,98</b>	<b>0,98</b>	<b>0,98</b>	<b>0,98</b>	0,97	0,97	0,81	0,93	0,97
23	0,88	0,88	0,88	<b>0,98</b>	<b>0,98</b>	<b>0,98</b>	<b>0,98</b>	<b>0,98</b>	<b>0,98</b>	0,96	0,96	0,79	0,93	0,96
24	0,95	0,96	0,96	<b>0,99</b>	<b>0,99</b>	<b>0,99</b>	<b>0,99</b>	<b>0,99</b>	<b>0,99</b>	0,98	0,98	0,91	0,97	0,97
Algorithm Mean	0,8825	0,8888	0,8892	<b>0,985</b>	0,9846	0,9846	<b>0,985</b>	0,9725	0,9688	0,8108	0,9346	0,9713		

Tabelle 7: SSIM des Kodak Sets

Image Number	bikubisch	bilinear mean	DLM-MSE code	DLM-MSE code RCD	DLM-MSE paper	DLM-MSE paper RCD	HA	MHC	NN	PPG	RCD	Image Mean	
1	123,75	127,73	119,58	121,39	99,69	125,09	103,15	110,19	116,12	365,87	104,41	<b>80,16</b>	
2	50,19	51,21	48,02	26,81	23,82	27,61	24,58	26,27	31,16	154,87	33,76	<b>21,64</b>	
3	141,28	149,3	142,54	34,78	<b>31,49</b>	35,26	32	43,35	48,9	416,3	79,5	39,01	
4	76,22	98,99	84,91	23,11	17,36	24,96	19,25	21,45	26,65	394,19	37,6	<b>13,09</b>	
5	38,31	42,07	39,28	44,86	36,35	47,79	39,09	37,04	40,54	142,32	32,49	<b>25,16</b>	
6	12,3	16,38	15,33	26,03	18,89	28,61	21,3	19,54	19,95	74,83	14,74	<b>10,7</b>	
7	66,88	61,98	61,67	9,03	8,84	8,04	<b>7,83</b>	19,44	20,68	155,75	41,41	22,06	
8	47,03	51,95	50,27	10,62	<b>10,23</b>	10,77	10,35	13,64	16,36	155,92	27,87	13,69	
9	28,01	34,59	31,13	22,03	17,44	22,97	18,34	18,72	19,91	124,4	21,53	<b>12,14</b>	
10	16,76	19,4	18,05	14,19	10,83	14,53	11,14	12,65	13,27	72,13	14,62	<b>8,91</b>	
11	13,44	14,57	13,93	11,6	8,97	11,9	9,24	11,09	10,68	52,87	11,92	<b>8,02</b>	
12	32,42	34,59	32,28	13,09	10,73	13,23	10,89	12,72	15,93	107,6	16,72	<b>10</b>	
13	11,32	12,72	11,71	8,25	6,92	8,44	7,11	7,46	9,01	42,29	8,13	<b>5,64</b>	
14	16,11	16,51	15,42	11,58	10,49	11,81	10,7	11,1	12,62	47,65	12,38	<b>8,89</b>	
15	13,4	14,31	13,23	11,68	10,25	13,4	10,45	10,95	11,52	49,55	10,8	<b>8,55</b>	
16	41,36	50,85	48,54	51,87	38,67	53,51	40,15	48,71	45,9	170,83	39,33	<b>29,11</b>	
17	<b>28,93</b>	36,04	33,83	70,64	52,26	74,68	56,04	53,67	47,79	137,15	38,82	32,96	
18	<u>64,82</u>	72,48	66,61	<u>25,99</u>	<b>22,33</b>	<u>26,46</u>	<u>22,75</u>	<u>27,33</u>	<u>31,63</u>	<u>235,77</u>	<u>51,14</u>	<u>22,61</u>	<u>55,83</u>
Algorithm Mean	45,7	50,32	47,02	29,86	24,2	31,06	25,24	28,07	29,92	161,13	33,18	<b>20,69</b>	

Tabelle 8: MSE des McMaster Sets

Image Number	bikubisch	bilinear mean	DLM-MSE code	DLM-MSE code RCD	DLM-MSE paper	DLM-MSE paper RCD	HA	MHC	NN	PPG	RCD	Image Mean	
1	27,21	27,07	27,35	27,29	28,14	27,16	28	27,71	27,48	22,5	<b>29,09</b>	27,25	
2	31,12	31,04	31,323	33,85	34,36	33,72	34,23	33,94	33,19	26,23	32,85	<b>34,75</b>	32,55
3	26,63	26,39	26,59	32,72	<b>33,15</b>	32,66	33,08	31,76	31,24	21,94	29,13	<u>32,22</u>	29,79
4	29,31	28,18	28,84	34,49	35,73	34,16	35,29	34,82	33,87	22,17	32,38	<b>36,96</b>	32,18
5	32,3	31,89	32,19	31,61	32,53	31,34	32,21	32,44	32,05	26,6	33,01	<b>34,12</b>	31,86
6	37,23	35,99	36,27	33,98	35,37	33,57	34,85	35,22	35,13	29,39	36,44	<b>37,84</b>	35,11
7	29,88	30,21	30,23	38,58	38,67	39,08	<b>39,19</b>	35,24	34,97	26,21	31,96	<u>34,7</u>	34,08
8	31,41	30,98	31,12	37,87	<b>38,03</b>	37,81	37,98	36,78	35,99	26,2	33,68	36,77	34,55
9	33,66	32,74	33,2	34,7	35,72	34,52	35,5	35,41	35,14	27,18	34,8	<b>37,29</b>	34,16
10	35,89	35,25	35,57	36,61	37,79	36,51	37,66	37,11	36,9	29,55	36,48	<b>38,63</b>	36,16
11	36,85	36,5	36,69	37,49	38,61	37,38	38,48	37,68	37,84	30,9	37,37	<b>39,09</b>	37,07
12	33,02	32,74	33,04	36,96	37,82	36,91	37,76	37,09	36,11	27,81	35,9	<b>38,13</b>	35,27
13	37,59	37,09	37,45	38,97	39,73	38,87	39,61	39,4	38,58	31,87	39,03	<b>40,62</b>	38,23
14	36,06	35,95	36,25	37,49	37,92	37,41	37,84	37,68	37,12	31,35	37,2	<b>38,64</b>	36,74
15	36,86	36,57	36,91	37,45	38,02	36,86	37,94	37,74	37,52	31,18	37,79	<b>38,81</b>	36,97
16	31,97	31,07	31,27	30,98	32,26	30,85	32,09	31,25	31,51	25,81	32,18	<b>33,49</b>	31,23
17	<b>33,52</b>	32,56	32,84	29,64	30,95	29,4	30,65	30,83	31,34	26,76	32,24	<u>32,95</u>	31,14
18	30,01	29,53	29,9	33,98	<b>34,64</b>	33,9	34,56	33,76	33,13	24,41	31,04	34,59	31,95
Algorithm Mean	32,81	32,32	32,61	34,70	35,52	34,56	35,38	34,77	34,4	27,11	33,97	<b>36,04</b>	

Tabelle 9: PSNR des McMaster Sets

Image Number	bikubisch	bilinear median	DLM/MSE code	DLM/MSE code RCD	DLM/MSE paper	DLM/MSE paper RCD	HA	MHC	NN	PPG	RCD	Image Mean
1	0,87	0,87	0,87	0,87	0,89	0,87	0,89	0,88	0,87	0,75	0,89	<b>0,91</b>
2	0,9	0,91	0,91	0,93	0,93	0,93	0,93	0,92	0,82	0,93	<b>0,94</b>	0,911
3	0,9	0,89	0,9	<b>0,96</b>	<b>0,96</b>	<b>0,96</b>	<b>0,96</b>	<b>0,95</b>	0,79	0,94	0,96	0,923
4	0,96	0,95	0,96	0,98	<b>0,99</b>	0,98	<b>0,99</b>	0,98	0,86	0,98	<b>0,99</b>	0,963
5	0,93	0,93	0,93	0,92	0,93	0,92	0,93	0,92	0,84	0,94	<b>0,95</b>	0,918
6	<b>0,96</b>	<b>0,96</b>	<b>0,96</b>	0,93	0,94	0,92	0,94	0,94	0,94	0,87	0,95	<b>0,96</b>
7	0,87	0,87	0,87	0,97	0,97	<b>0,98</b>	<b>0,98</b>	0,96	0,96	0,77	0,92	0,95
8	0,94	0,94	0,94	<b>0,97</b>	<b>0,97</b>	<b>0,97</b>	<b>0,97</b>	<b>0,97</b>	0,96	0,88	0,96	<b>0,97</b>
9	0,94	0,94	0,95	0,94	0,95	0,94	0,95	0,95	0,94	0,87	0,95	<b>0,96</b>
10	0,95	0,95	0,95	0,95	0,96	0,95	0,96	0,96	0,95	0,88	0,96	<b>0,97</b>
11	<b>0,96</b>	0,95	0,95	0,94	0,95	0,94	0,95	0,95	0,95	0,89	<b>0,96</b>	<b>0,96</b>
12	0,94	0,94	0,94	0,95	<b>0,96</b>	0,95	<b>0,96</b>	0,95	0,95	0,87	0,95	<b>0,96</b>
13	0,94	0,95	0,95	0,95	0,95	0,95	0,95	0,95	0,94	0,9	0,95	<b>0,96</b>
14	0,94	0,94	0,94	0,95	0,95	0,95	0,95	0,95	0,94	0,89	0,95	<b>0,96</b>
15	0,94	<b>0,95</b>	<b>0,95</b>	0,94	0,94	0,94	0,94	0,94	0,94	0,89	<b>0,95</b>	<b>0,95</b>
16	<b>0,93</b>	0,92	0,92	0,89	0,92	0,89	0,92	0,9	0,91	0,82	<b>0,93</b>	0,937
17	<b>0,95</b>	0,94	0,94	0,88	0,91	0,88	0,9	0,91	0,9	0,84	0,94	0,902
18	0,9	0,9	0,9	0,94	<b>0,95</b>	0,94	<b>0,95</b>	0,94	0,94	0,81	0,93	<b>0,95</b>
Algorithm Mean	0,9289	0,9278	0,9294	0,9367	0,9456	0,9367	0,9456	0,9417	0,9367	0,8467	0,9433	<b>0,9539</b>

Tabelle 10: SSIM des McMaster Sets

Image Number	bikubisch	bilinear median	DLM/MSE code	DLM/MSE paper RCD	DLM/MSE paper	DLM/MSE paper RCD	HA	MHC	NN	PPG	RCD	Image Mean
1	32,67	29,82	29,77	5,14	4,97	4,73	<b>4,56</b>	10,55	11,37	68,39	19,39	19,23
2	18,22	21,14	20,34	4,13	<b>3,63</b>	4,39	3,88	4,88	7,19	83,9	10,44	3,81
3	8,01	8,3	7,64	1,13	1,01	1,1	<b>0,99</b>	1,98	2,64	27,01	3,02	1,61
4	12,82	13,05	12,59	2,45	2,26	2,44	<b>2,25</b>	3,33	4,68	38,66	7,85	2,88
5	142,52	147,06	136,81	12,66	12,18	11,24	<b>10,83</b>	29,33	39,41	422,12	58,74	25,76
6	110,36	116,5	106,12	22,79	<b>21,61</b>	22,79	21,69	31,58	37,81	304,42	56,08	29,29
7	155,01	143,42	139,12	28,82	<b>26,95</b>	28,37	<b>26,51</b>	42,22	54,44	311,98	79,57	34,35
8	43,29	42,41	40,16	4,6	4,1	4,56	<b>4,06</b>	7,16	15,63	96,71	19,84	5,24
9	49,17	45,83	45,45	9,95	9,43	9,66	<b>9,18</b>	15,84	19,64	104,95	26,87	14,05
10	68,36	68,89	64,08	18,2	15,25	18,33	15,41	15,63	36,76	169,69	30,12	<b>11,09</b>
11	39,84	38,18	36,91	23,55	20,74	24,22	21,4	20,65	27,14	100,87	23,21	<b>16,92</b>
12	46,49	48,6	44,93	35,83	29,62	37,42	31,11	28,9	36,46	136,3	29,75	<b>19,05</b>
13	60,32	50,19	50,45	11,11	10,89	10,56	<b>10,34</b>	22,36	24,95	85,63	41,63	18,44
14	2,31	2,45	2	0,3	0,27	0,29	<b>0,25</b>	0,48	0,73	6,8	1,07	0,38
15	37,41	50,21	49,84	8,99	<b>8,39</b>	9,81	9,24	10,83	13,56	180,56	24,56	8,79
16	43,54	45,03	39,99	3,46	3,47	<b>2,91</b>	2,92	9,51	13,17	127,18	20,66	9,08
17	23,59	23,62	21,45	2,54	2,52	2,35	<b>2,34</b>	4,37	7,92	60,54	11,47	4,21
18	339,93	307,1	307,36	35,23	34,58	30,76	<b>30,35</b>	87,89	103,41	558,11	179,98	84,93
19	60,26	56,21	52,24	16	<b>14,06</b>	16,26	14,25	19,71	24,93	140,58	30,76	16,51
20	273,69	259,64	263,58	39,17	39,75	<b>35,97</b>	36,42	84,7	93,66	583,54	148,86	79,1
21	111,93	110,92	106,76	19,59	<b>18,03</b>	19,52	18,04	31,03	37,9	283,87	54,67	27,87
22	190,05	177,2	175,99	49,3	45,59	48,78	<b>45,02</b>	68,42	79,98	375,36	112,64	59,06
23	226,54	202,59	202,43	38,13	37,32	35,65	<b>34,85</b>	71,31	79,72	405,73	131,22	70,33
24	32,63	29,91	38,88	3,72	3,56	<b>3,41</b>	8,39	10,55	65,71	16,82	7,93	18,04
500	71,2	76,8	70,24	11,27	7,52	10,98	<b>7,19</b>	13,23	32,29	243,72	27,37	8,64
Algorithm Mean	88,01	84,6	82,25	16,33	15,11	15,87	<b>14,66</b>	25,77	32,64	199,29	46,66	22,75

Tabelle 11: MSE des MSR Sets

Image Number	bikubisch	bilinear median	DLM-MSE code	DLM-MSE code RCD	DLM-MSE paper	DLM-MSE paper RCD	HA	MHC	NN	PPG	RCD	Image Mean
1	32,99	33,39	33,39	41,02	41,17	41,38	<b>41,54</b>	37,9	29,78	35,25	38,16	36,96
2	35,53	34,88	35,05	41,97	<b>42,53</b>	41,71	42,24	41,25	28,89	37,94	42,32	38,66
3	39,09	38,94	39,3	47,59	48,09	47,7	<b>48,18</b>	45,16	43,91	43,33	46,06	43,43
4	37,05	36,97	37,13	44,24	44,6	44,26	<b>44,61</b>	42,91	41,43	32,26	39,18	43,54
5	26,59	26,46	26,77	37,11	37,27	37,62	<b>37,78</b>	33,46	32,17	21,88	30,44	34,2
6	27,7	27,47	27,87	34,55	<b>34,78</b>	34,55	34,77	33,14	32,35	23,3	30,64	33,46
7	26,23	26,56	26,7	33,53	33,83	33,6	<b>33,9</b>	31,88	30,77	23,19	29,12	32,77
8	31,77	31,86	32,09	41,5	42	41,54	<b>42,04</b>	39,58	36,19	28,28	35,16	40,94
9	31,21	31,52	31,56	38,15	38,39	38,28	<b>38,5</b>	36,13	35,2	27,92	33,84	34,78
10	29,78	29,75	30,06	35,53	36,3	35,5	36,25	36,19	32,48	25,83	33,34	<b>37,68</b>
11	32,13	32,31	32,46	34,41	34,96	34,29	34,83	34,98	33,79	28,09	34,47	<b>35,85</b>
12	31,46	31,26	31,61	32,59	33,42	32,4	33,2	33,52	32,51	26,79	33,4	<b>35,33</b>
13	30,33	31,12	31,1	37,67	37,76	37,89	<b>37,99</b>	34,64	34,16	28,8	31,94	35,47
14	44,5	44,24	45,12	53,3	53,9	53,51	<b>54,11</b>	51,36	49,49	39,81	47,84	52,33
15	32,4	31,12	31,16	38,59	<b>38,89</b>	38,22	38,48	37,79	36,81	25,56	34,23	38,69
16	31,74	31,6	32,11	42,73	42,73	<b>43,48</b>	<b>43,48</b>	38,35	36,94	27,09	34,98	38,55
17	34,4	34,4	34,82	44,08	44,11	44,42	<b>44,44</b>	41,72	39,14	30,31	37,53	41,89
18	22,82	23,26	23,25	32,66	32,74	33,25	<b>33,31</b>	28,69	27,99	20,66	25,58	28,84
19	30,33	30,63	30,95	36,09	<b>36,65</b>	36,02	36,59	35,18	34,16	26,65	33,25	35,95
20	23,76	23,99	23,92	32,2	<b>32,14</b>	<b>32,57</b>	32,52	28,85	28,42	20,47	26,4	29,15
21	27,64	27,68	27,85	35,21	<b>35,57</b>	35,23	<b>35,57</b>	33,21	32,34	23,6	30,75	33,68
22	25,34	25,65	25,68	31,2	31,54	31,25	<b>31,6</b>	29,78	29,1	22,39	27,61	30,42
23	24,58	25,06	25,07	32,32	32,41	32,61	<b>32,71</b>	29,6	29,12	22,05	26,95	29,66
24	32,99	33,37	33,37	42,25	42,43	42,62	<b>42,8</b>	38,89	37,9	29,95	35,87	39,14
500	29,61	29,28	29,67	37,61	39,37	37,72	<b>39,56</b>	36,92	33,04	24,26	33,76	38,77
Algorithm Mean	30,88	30,91	31,12	38,32	38,7	38,46	<b>38,84</b>	36,44	35,06	26,86	33,71	37,18

Tabelle 12: PSNR des MSR Sets

Image Number	bikubisch	bilinear mean	bilinear median	DLM/MSE code	DLM/MSE code RCD	DLM/MSE paper	DLM/MSE paper RCD	HA	MHC	NN	PPG	RCD	Image Mean
1	0,91	0,91	0,91	<b>0,98</b>	<b>0,98</b>	<b>0,98</b>	<b>0,98</b>	0,97	0,97	0,85	0,94	0,97	0,9458
2	0,98	0,97	0,97	0,99	<b>1</b>	0,99	0,99	0,99	0,99	0,92	0,99	0,99	0,9808
3	0,98	0,98	0,98	<b>1</b>	<b>1</b>	<b>1</b>	<b>1</b>	0,99	0,99	0,96	0,99	<b>1</b>	0,9892
4	0,96	0,96	0,96	<b>0,99</b>	<b>0,99</b>	<b>0,99</b>	<b>0,99</b>	<b>0,99</b>	<b>0,99</b>	0,98	0,91	0,97	<b>0,99</b>
5	0,94	0,94	0,94	<b>0,99</b>	<b>0,99</b>	<b>0,99</b>	<b>0,99</b>	<b>0,99</b>	<b>0,99</b>	0,98	0,86	0,97	<b>0,99</b>
6	0,91	0,91	0,92	<b>0,98</b>	<b>0,98</b>	<b>0,98</b>	<b>0,98</b>	<b>0,98</b>	<b>0,98</b>	0,97	0,96	0,84	0,95
7	0,9	0,89	0,9	<b>0,98</b>	<b>0,98</b>	<b>0,98</b>	<b>0,98</b>	<b>0,98</b>	<b>0,98</b>	0,97	0,96	0,8	0,94
8	0,93	0,93	0,93	<b>0,99</b>	<b>0,99</b>	<b>0,99</b>	<b>0,99</b>	<b>0,99</b>	<b>0,99</b>	0,98	0,98	0,87	0,96
9	0,9	0,9	0,9	<b>0,98</b>	<b>0,98</b>	<b>0,98</b>	<b>0,98</b>	<b>0,98</b>	<b>0,98</b>	0,97	0,96	0,82	0,95
10	0,94	0,93	0,94	0,98	0,98	0,98	0,98	0,98	0,98	0,98	0,96	0,86	0,97
11	0,94	0,95	0,95	0,96	0,96	0,95	0,96	0,96	0,96	0,95	0,95	0,9	0,9508
12	0,95	0,95	0,95	0,94	0,95	0,94	0,95	0,95	0,95	0,94	0,88	0,96	<b>0,97</b>
13	0,94	0,95	0,95	<b>0,99</b>	<b>0,99</b>	<b>0,99</b>	<b>0,99</b>	<b>0,99</b>	<b>0,99</b>	0,98	0,98	0,92	0,96
14	0,99	0,99	0,99	<b>1</b>	<b>1</b>	<b>1</b>	<b>1</b>	<b>1</b>	<b>1</b>	0,98	0,98	0,99	<b>1</b>
15	0,95	0,94	0,94	<b>0,99</b>	<b>0,99</b>	<b>0,99</b>	<b>0,99</b>	<b>0,99</b>	<b>0,99</b>	0,98	0,98	0,84	0,97
16	0,88	0,89	0,89	0,98	0,98	0,98	0,98	0,98	0,98	0,96	0,98	0,92	0,9442
17	0,94	0,94	0,95	<b>0,99</b>	<b>0,99</b>	<b>0,99</b>	<b>0,99</b>	<b>0,99</b>	<b>0,99</b>	0,99	0,98	0,98	0,9575
18	0,8	0,81	0,81	<b>0,98</b>	<b>0,98</b>	<b>0,98</b>	<b>0,98</b>	<b>0,98</b>	<b>0,98</b>	0,95	0,94	0,73	0,89
19	0,92	0,93	0,93	<b>0,97</b>	<b>0,97</b>	<b>0,97</b>	<b>0,97</b>	<b>0,97</b>	<b>0,97</b>	<b>0,97</b>	0,96	0,86	0,96
20	0,92	0,92	0,93	<b>0,99</b>	<b>0,99</b>	<b>0,99</b>	<b>0,99</b>	<b>0,99</b>	<b>0,99</b>	0,98	0,98	0,86	0,96
21	0,88	0,88	0,88	<b>0,98</b>	<b>0,98</b>	<b>0,98</b>	<b>0,98</b>	<b>0,98</b>	<b>0,98</b>	0,96	0,96	0,77	0,94
22	0,84	0,84	0,85	<b>0,96</b>	<b>0,96</b>	<b>0,96</b>	<b>0,96</b>	<b>0,96</b>	<b>0,96</b>	0,94	0,94	0,75	0,91
23	0,8	0,81	0,81	0,96	0,97	0,97	0,97	0,97	0,94	0,93	0,72	0,88	0,8917
24	0,95	0,95	0,95	<b>0,99</b>	<b>0,99</b>	<b>0,99</b>	<b>0,99</b>	<b>0,99</b>	<b>0,99</b>	0,98	0,98	0,9	0,97
500	0,93	0,93	0,93	0,98	0,98	0,98	0,98	0,98	0,98	0,96	0,96	0,84	0,97
Algorithm Mean	0,9192	0,92	0,9224	0,9808	<b>0,9824</b>	0,9812	<b>0,9824</b>	0,9724	0,9668	0,854	0,954	0,9768	

Tabelle 13: SSIM des MSR Sets

Image Number	bikubisch	bilinear mean	bilinear median	DLMMSE code	DLMMSE paper	DLMMSE code RCD	DLMMSE paper RCD	HA	MHC	NN	PPG	RCD	Image Mean
1	35,93	32,09	32,18	8,2	7,79	<b>7,65</b>	8,05	13,81	14,83	72,41	22,93	13,03	22,41
2	20,66	22,86	22,14	6,22	6,52	6,04	<b>5,75</b>	7,15	9,72	86,72	12,88	5,83	17,71
3	10,14	9,82	9,2	2,8	2,78	<b>2,69</b>	2,72	3,79	4,83	29,69	4,82	3,2	7,21
4	15,16	14,65	14,3	4,4	4,43	4,27	<b>4,24</b>	5,43	7,11	41,73	9,8	4,73	10,85
5	146,98	150,56	140,3	17,09	15,75	<b>15,34</b>	16,64	34,04	44,07	426,3	64,76	30,11	91,83
6	112,63	118,39	108,07	24,71	24,75	23,69	<b>23,6</b>	33,76	40,16	306,8	58,05	31,24	75,49
7	157,18	145,2	141,04	30,93	30,5	<b>28,72</b>	29,11	44,48	56,7	314,84	81,28	36,33	91,36
8	45,04	43,58	41,42	6,21	6,17	<b>5,73</b>	5,77	8,82	17,5	98,78	22,09	6,78	25,66
9	51,75	47,69	47,33	12,23	11,97	<b>11,52</b>	11,76	18,35	22,34	107,97	29,36	16,24	32,38
10	83,11	79,17	75,21	31,57	31,71	29,05	28,89	28,53	51,95	187,87	42,82	<b>22,5</b>	57,7
11	56,55	49,67	48,7	35,72	36,49	34,29	33,56	33,9	43,13	121,23	37,75	<b>29,09</b>	46,67
12	53,57	53,59	50,4	41,15	42,81	36,72	35,2	34,51	43,35	145,63	36,51	<b>24,19</b>	49,8
13	63,08	52,13	52,53	13,37	12,84	<b>12,63</b>	13,16	24,62	27,75	88,95	44,69	20,85	35,55
14	3,17	3,07	2,65	1,03	1,02	<b>0,99</b>	1	1,26	1,63	7,83	1,83	1,06	2,21
15	40,24	52,44	52,26	11,57	12,4	11,88	<b>11,02</b>	13,67	16,515	183,74	27,48	11,37	37,05
16	53,72	52,29	47,56	13,72	13,29	<b>13,27</b>	13,69	20,63	24,13	139,85	30,23	18,7	36,76
17	25,65	25,05	23,02	4,4	<b>4,22</b>	4,23	4,41	6,39	10,96	62,92	13,54	5,96	15,82
18	343,6	309,81	310,12	37,76	33,34	<b>32,95</b>	37,11	90,4	106,6	561,47	182,07	87,12	177,7
19	63,19	58,24	54,54	18,75	19,02	17,1	<b>16,89</b>	22,84	28,12	144,1	33,8	19,41	41,33
20	281,93	265,74	269,21	44,56	<b>41,38</b>	41,89	45,17	92,3	100,77	593,06	154,14	84,31	167,87
21	117,97	115,23	111,11	24,47	24,41	23,01	<b>22,95</b>	35,56	44,2	290,3	59,61	32,2	75,09
22	192,92	179,25	178,06	51,53	51,04	<b>47,42</b>	47,95	71,64	82,65	379,08	114,1	61,04	121,39
23	268,21	232,95	233,84	82,47	80,64	<b>79,85</b>	81,67	119,04	125,58	454,14	173,52	113,88	170,48
24	35,59	32,08	32,19	6,71	6,4	<b>6,26</b>	6,55	11,98	13,76	69,21	19,93	10,74	20,95
500	73,8	78,57	72,3	13,44	13,17	<b>9,54</b>	9,85	15,64	34,89	247,05	29,63	10,73	50,72
Algorithm Mean	94,07	88,96	86,79	21,8	21,39	<b>20,27</b>	20,67	31,7	38,89	206,47	52,3	28,03	

Tabelle 14: MSE des MSR Sets mit Rauschen

Image Number	bikubisch	bilinear mean	bilinear median	DLMMSE code	DLMMSE code RCD	DLMMSE paper	DLMMSE paper RCD	HA	MHC	NN	PPG	RCD	Image Mean
1	32,58	33,07	33,05	38,99	39,07	<u><b>39,22</b></u>	39,29	36,73	36,42	29,53	34,53	36,98	35,78833333
2	34,98	34,54	34,68	40,19	40,53	<u><b>39,99</b></u>	<u><b>40,32</b></u>	39,59	38,25	28,75	37,03	40,47	37,44333333
3	38,07	38,21	38,49	43,65	43,79	<u><b>43,69</b></u>	<u><b>43,83</b></u>	42,34	41,29	33,41	41,3	43,07	40,92833333
4	36,32	36,47	36,58	41,7	41,85	41,67	<u><b>41,83</b></u>	40,79	39,61	31,93	38,22	41,38	39,0291667
5	26,46	26,35	26,66	35,8	35,92	<u><b>36,16</b></u>	36,27	32,81	31,69	21,83	30,02	33,34	31,1091667
6	27,61	27,4	27,79	34,2	34,4	34,2	<u><b>34,39</b></u>	32,85	32,09	23,26	30,49	33,18	30,98833333
7	26,17	26,51	26,64	33,23	33,49	<u><b>33,29</b></u>	33,55	31,65	30,6	23,15	29,03	32,53	29,9866667
8	31,59	31,74	31,96	40,2	40,52	<u><b>40,23</b></u>	40,55	38,68	35,7	28,18	34,69	39,82	36,155
9	30,99	31,35	31,38	37,26	37,43	<u><b>37,35</b></u>	37,52	35,5	34,64	27,8	33,45	36,02	34,2241667
10	29,93	29,14	29,37	33,14	33,52	33,12	33,5	33,58	30,98	25,39	31,81	<b>34,61</b>	31,5075
11	30,61	31,17	31,26	32,6	32,87	32,51	32,78	32,83	31,78	27,29	32,36	<b>33,49</b>	31,79583333
12	30,84	30,84	31,11	31,99	32,66	31,82	32,48	32,75	31,76	26,5	32,51	<b>34,3</b>	31,63
13	30,13	30,96	30,93	36,87	36,94	<u><b>37,05</b></u>	37,12	34,22	33,7	28,64	31,63	34,94	33,5941667
14	43,12	43,26	43,9	48	48,14	<u><b>48,04</b></u>	48,18	47,13	46,01	39,19	45,51	47,89	45,6975
15	32,08	30,93	30,95	37,5	37,71	37,2	<u><b>37,38</b></u>	36,77	35,95	25,49	33,74	37,57	34,4391667
16	30,83	30,95	31,36	36,76	36,77	<u><b>36,9</b></u>	36,9	34,99	34,31	26,67	33,33	35,41	33,765
17	34,04	34,14	34,51	41,7	<b>41,69</b>	41,88	41,87	40,07	38,1	30,14	36,81	40,38	37,9441667
18	22,77	23,22	23,22	32,36	32,44	<u><b>32,9</b></u>	32,95	28,57	27,85	20,64	25,53	28,73	27,59833333
19	30,12	30,48	30,76	35,4	35,85	35,34	<u><b>35,8</b></u>	34,54	33,64	26,54	32,84	35,25	33,0466667
20	23,63	23,89	23,83	31,64	<u><b>31,58</b></u>	31,96	31,91	28,48	28,1	20,4	26,25	28,87	27,545
21	27,41	27,52	27,67	34,24	34,52	34,25	<u><b>34,51</b></u>	32,62	31,68	23,5	30,38	33,05	30,94583333
22	25,28	25,6	25,63	31,01	31,32	<u><b>31,05</b></u>	31,37	29,58	28,96	22,34	27,56	30,27	28,33083333
23	23,85	24,46	24,44	28,97	29,01	<u><b>29,07</b></u>	29,11	27,37	27,14	21,56	25,74	27,57	26,5241667
24	32,62	33,07	33,05	39,86	39,97	<u><b>40,07</b></u>	40,17	37,34	36,75	29,73	35,13	37,82	36,29833333
500	29,45	29,18	29,54	36,85	38,2	<u><b>36,94</b></u>	38,33	36,19	32,7	24,2	33,41	37,82	33,5675
Algorithm Mean	30,4592	30,578	30,7504	36,5644	36,8076	<u><b>36,636</b></u>	36,8764	35,1188	33,988	26,6424	32,932	35,7904	

Tabelle 15: PSNR des MSR Sets mit Rauschen

Image Number	bikubisch	bilinear mean	bilinear median	DLM-MSE code	DLM-MSE code RCD	DLM-MSE paper	DLM-MSE paper RCD	HA	MHC	NN	PPG	RCD	Image Mean
1	0,89	0,9	0,9	<b>0,96</b>	<b>0,96</b>	<b>0,96</b>	<b>0,96</b>	0,95	0,95	0,93	0,95	0,95	0,9283
2	0,97	0,97	0,97	<b>0,99</b>	<b>0,99</b>	<b>0,99</b>	<b>0,99</b>	0,98	0,98	0,98	<b>0,99</b>	<b>0,99</b>	0,9758
3	0,96	0,97	0,97	<b>0,98</b>	<b>0,98</b>	<b>0,98</b>	<b>0,98</b>	<b>0,98</b>	<b>0,98</b>	0,97	0,97	<b>0,98</b>	0,9708
4	0,94	0,95	0,95	<b>0,98</b>	<b>0,98</b>	<b>0,98</b>	<b>0,98</b>	<b>0,98</b>	<b>0,98</b>	0,97	0,89	0,96	<b>0,98</b>
5	0,92	0,92	0,93	<b>0,97</b>	<b>0,97</b>	<b>0,97</b>	<b>0,97</b>	<b>0,97</b>	<b>0,97</b>	0,96	0,84	0,95	0,9433
6	0,9	0,91	0,91	<b>0,97</b>	<b>0,97</b>	<b>0,97</b>	<b>0,97</b>	<b>0,97</b>	<b>0,97</b>	0,96	0,95	0,83	0,94
7	0,89	0,89	0,89	0,97	0,97	0,97	0,97	0,97	0,97	0,96	0,96	0,8	0,94
8	0,91	0,92	0,92	0,98	0,98	<b>0,98</b>	<b>0,98</b>	0,98	0,98	0,97	0,96	0,86	0,95
9	0,89	0,89	0,89	0,96	0,96	<b>0,96</b>	<b>0,96</b>	<b>0,97</b>	<b>0,97</b>	0,95	0,95	0,81	0,93
10	0,87	0,89	0,89	0,91	0,92	0,91	0,92	0,92	0,92	0,89	0,79	0,91	<b>0,93</b>
11	0,87	0,89	0,89	0,88	0,89	0,88	0,89	0,89	0,89	0,86	0,81	0,89	<b>0,91</b>
12	0,92	0,93	0,93	0,92	0,93	0,92	0,93	0,93	0,93	0,91	0,86	0,94	<b>0,95</b>
13	0,92	0,93	0,93	<b>0,97</b>	<b>0,97</b>	<b>0,97</b>	<b>0,97</b>	<b>0,97</b>	<b>0,97</b>	0,96	0,95	0,89	0,94
14	0,98	0,98	0,98	<b>0,99</b>	<b>0,99</b>	<b>0,99</b>	<b>0,99</b>	<b>0,99</b>	<b>0,99</b>	0,98	0,98	0,96	0,98
15	0,94	0,93	0,93	<b>0,98</b>	<b>0,98</b>	<b>0,98</b>	<b>0,98</b>	<b>0,98</b>	<b>0,98</b>	0,97	0,97	0,83	0,96
16	0,93	<b>0,95</b>	<b>0,95</b>	0,92	0,92	0,92	0,92	0,92	0,92	0,89	0,89	0,76	0,87
17	0,93	0,93	0,94	<b>0,98</b>	<b>0,98</b>	<b>0,98</b>	<b>0,98</b>	<b>0,98</b>	<b>0,98</b>	0,97	0,96	0,87	0,96
18	0,79	0,8	0,8	<b>0,97</b>	<b>0,97</b>	<b>0,97</b>	<b>0,97</b>	<b>0,97</b>	<b>0,97</b>	0,94	0,93	0,72	0,89
19	0,91	0,92	0,92	<b>0,96</b>	<b>0,96</b>	<b>0,96</b>	<b>0,96</b>	<b>0,96</b>	<b>0,96</b>	0,95	0,95	0,85	0,94
20	0,9	0,91	0,91	<b>0,97</b>	<b>0,97</b>	<b>0,97</b>	<b>0,97</b>	<b>0,97</b>	<b>0,97</b>	0,96	0,95	0,83	0,94
21	0,87	0,87	0,87	<b>0,97</b>	<b>0,97</b>	<b>0,97</b>	<b>0,97</b>	<b>0,97</b>	<b>0,97</b>	0,95	0,95	0,76	0,92
22	0,84	0,84	0,84	<b>0,96</b>	<b>0,96</b>	<b>0,96</b>	<b>0,96</b>	<b>0,96</b>	<b>0,96</b>	0,94	0,93	0,74	0,9
23	0,74	0,76	0,76	0,88	0,88	0,88	0,88	0,88	0,88	0,85	0,85	0,65	0,81
24	0,93	0,93	0,93	<b>0,97</b>	<b>0,97</b>	<b>0,97</b>	<b>0,97</b>	<b>0,97</b>	<b>0,97</b>	0,96	0,96	0,87	0,95
500	0,92	0,92	0,92	0,97	0,97	<b>0,98</b>	<b>0,98</b>	<b>0,98</b>	<b>0,98</b>	0,97	0,95	0,83	0,96
Algorithm Mean	0,9012	0,908	0,908	0,9584	0,96	0,9584	0,9612	0,9488	0,9412	0,8288	0,9324	0,956	

Tabelle 16: SSIM des MSR Sets mit Rauschen

Image Number	bikubisch	bilinear mean	bilinear median	DLM/MSE code	DLM/MSE code RCD	DLM/MSE paper	DLM/MSE paper RCD	HA	MHC	NN	PPG	RCD	Image Mean
1	3,26	<b>2,27</b>	2,41	3,06	2,82	2,92	3,49	3,26	3,46	4,02	3,54	3,64	3,17916667
2	2,44	1,72	1,8	2,09	2,89	<b>1,65</b>	1,87	2,27	2,53	2,82	2,44	2,02	2,21166667
3	2,13	<b>1,52</b>	1,56	1,67	1,77	1,59	1,73	1,81	2,19	2,68	1,8	1,59	1,83666667
4	2,34	<b>1,6</b>	1,71	1,95	2,17	1,83	1,99	2,1	2,43	3,07	1,95	1,85	2,0825
5	4,46	3,5	<b>3,49</b>	4,43	3,57	4,1	5,81	4,71	4,66	4,18	6,02	4,35	4,44
6	2,27	1,89	1,95	1,92	3,14	<b>0,9</b>	1,91	2,18	2,35	2,38	1,97	1,95	2,0675
7	2,17	1,78	1,92	2,11	3,55	<b>0,35</b>	2,6	2,26	2,26	2,86	1,71	1,98	2,12916667
8	1,75	1,17	1,26	1,61	2,07	<b>1,17</b>	1,71	1,66	1,87	2,07	2,25	1,54	1,6775
9	2,58	<b>1,86</b>	1,88	2,28	2,54	<b>1,86</b>	2,58	2,51	2,7	3,02	2,49	2,19	2,37416667
10	14,75	<b>10,28</b>	11,13	13,37	16,46	10,72	13,48	12,9	15,19	18,18	12,7	11,41	13,3808333
11	16,71	11,49	11,79	12,17	15,75	<b>10,07</b>	12,16	13,25	15,99	20,36	14,54	12,17	13,8708333
12	7,08	4,99	5,47	5,32	13,19	<b>-0,7</b>	4,09	5,61	6,89	9,33	6,76	5,14	6,0975
13	2,76	<b>1,94</b>	2,08	2,26	1,95	2,07	<b>2,82</b>	2,26	2,8	3,32	3,06	2,41	2,4775
14	0,86	<b>0,62</b>	0,65	0,73	0,75	0,7	0,75	0,78	0,9	1,03	0,76	0,68	0,7675
15	2,83	2,23	2,42	2,58	4,01	2,07	<b>1,78</b>	2,84	2,955	3,18	2,92	2,58	2,69958333
16	10,18	<b>7,26</b>	7,57	10,26	9,82	10,36	10,77	11,12	10,96	12,67	9,57	9,62	10,0133333
17	2,06	<b>1,43</b>	1,57	1,86	1,7	1,88	2,07	2,02	2,14	2,38	2,07	1,75	1,91083333
18	3,67	2,71	2,76	2,53	<b>-1,24</b>	2,19	6,76	2,51	3,19	3,36	2,09	2,19	2,72666667
19	2,93	2,03	2,3	2,75	4,96	<b>0,84</b>	2,64	3,13	3,19	3,52	3,04	2,9	2,8525
20	8,24	6,1	5,63	5,39	<b>1,63</b>	5,92	8,75	7,6	7,11	9,52	5,28	5,21	6,365
21	6,04	4,31	4,35	4,88	6,38	<b>3,49</b>	4,91	4,53	6,3	6,43	4,94	4,33	5,07416667
22	2,87	2,05	2,07	2,23	5,45	<b>-1,36</b>	2,93	3,22	2,67	3,72	1,46	1,98	2,44083333
23	41,67	<b>30,36</b>	31,41	44,34	43,32	44,2	46,82	47,73	45,86	48,41	42,3	43,55	42,4975
24	2,96	<b>2,17</b>	2,28	2,83	2,68	2,7	3,14	3,59	3,21	3,5	3,11	2,81	2,915
500	2,6	1,77	2,06	2,17	5,65	<b>-1,44</b>	2,66	2,41	2,6	3,33	2,26	2,09	2,34666667
Algorithm	6,0644	<b>4,362</b>	4,5408	5,4716	6,2792	4,4032	6,0088	5,9304	6,2562	7,1736	5,6412	5,2772	
Mean													

Tabelle 17: MSE Differenz des MSR Sets mit Rauschen - MSR Set

Image Number	bikubisch	bilinear mean	bilinear median	DLMMSE code	DLMMSE code RCD	DLMMSE paper	DLMMSE paper RCD	HA	MHC	NN	PPG	RCD	Image Mean
1	-0,41	-0,32	-0,34	-2,03	-2,1	-2,16	-2,25	-1,17	-1,15	<b>-0,25</b>	-0,72	-1,18	-1,17
2	-0,55	-0,34	-0,37	-1,78	-2	-1,72	-1,92	-1,66	-1,31	<b>-0,14</b>	-0,91	-1,85	-1,21
3	-1,02	-0,73	-0,81	-3,94	-4,3	-4,01	-4,35	-2,82	-2,62	<b>-0,4</b>	-2,03	-2,99	-2,5
4	-0,73	-0,5	-0,55	-2,54	-2,75	-2,59	-2,78	-2,12	-1,82	<b>-0,33</b>	-0,96	-2,16	-1,65
5	-0,13	-0,11	-0,11	-1,31	-1,35	-1,46	-1,51	-0,65	-0,48	<b>-0,05</b>	-0,42	-0,86	-0,7
6	-0,09	-0,07	-0,08	-0,35	-0,38	-0,35	-0,38	-0,29	-0,26	<b>-0,04</b>	-0,15	-0,28	-0,23
7	-0,06	-0,05	-0,06	-0,3	-0,34	-0,31	-0,35	-0,23	-0,17	<b>-0,04</b>	-0,09	-0,24	-0,19
8	-0,18	-0,12	-0,13	-1,3	-1,48	-1,31	-1,49	-0,9	-0,49	<b>-0,1</b>	-0,47	-1,12	-0,76
9	-0,22	-0,17	-0,18	-0,89	-0,96	-0,93	-0,98	-0,63	-0,56	<b>-0,12</b>	-0,39	-0,64	-0,56
10	<b>0,15</b>	-0,61	-0,69	-2,39	-2,78	-2,38	-2,75	-2,61	-1,5	<b>-0,44</b>	-1,53	-3,07	-1,72
11	-1,52	-1,14	-1,2	-1,81	-2,09	-1,78	-2,05	-2,15	-2,01	<b>-0,8</b>	-2,11	-2,36	-1,75
12	-0,62	-0,42	-0,5	-0,6	-0,76	-0,58	-0,72	-0,77	-0,75	<b>-0,29</b>	-0,89	-1,03	-0,66
13	-0,2	<b>-0,16</b>	-0,17	-0,8	-0,82	-0,84	-0,87	-0,42	-0,46	<b>-0,16</b>	-0,31	-0,53	-0,48
14	-1,38	-0,98	-1,22	-5,3	-5,76	-5,47	-5,93	-4,23	-3,48	<b>-0,62</b>	-2,33	-4,44	-3,43
15	-0,32	-0,19	-0,21	-1,09	-1,18	-1,02	-1,1	-1,02	-0,86	<b>-0,07</b>	-0,49	-1,12	-0,72
16	-0,91	-0,65	-0,75	-5,97	-5,96	-6,58	-6,58	-3,36	-2,63	<b>-0,42</b>	-1,65	-3,14	-3,22
17	-0,36	-0,26	-0,31	-2,38	-2,42	-2,54	-2,57	-1,65	-1,04	<b>-0,17</b>	-0,72	-1,51	-1,33
18	-0,05	-0,04	-0,03	-0,3	-0,3	-0,35	-0,36	-0,12	-0,14	<b>-0,02</b>	-0,05	-0,11	-0,16
19	-0,21	-0,15	-0,19	-0,69	-0,8	-0,68	-0,79	-0,64	-0,52	<b>-0,11</b>	-0,41	-0,7	-0,49
20	-0,13	-0,1	-0,09	-0,56	-0,56	-0,61	-0,61	-0,37	-0,32	<b>-0,07</b>	-0,15	-0,28	-0,32
21	-0,23	-0,16	-0,18	-0,97	-1,05	-0,98	-1,06	-0,59	-0,66	<b>-0,1</b>	-0,37	-0,63	-0,58
22	-0,06	<b>-0,05</b>	-0,05	-0,19	-0,22	-0,2	-0,23	-0,2	-0,14	<b>-0,05</b>	-0,05	-0,15	-0,13
23	-0,73	-0,6	-0,63	-3,35	-3,4	-3,54	-3,6	-2,23	-1,98	<b>-0,49</b>	-1,21	-2,09	-1,99
24	-0,37	-0,3	-0,32	-2,39	-2,46	-2,55	-2,63	-1,55	-1,15	<b>-0,22</b>	-0,74	-1,32	-1,33
500	-0,16	-0,1	-0,13	-0,76	-1,17	-0,78	-1,23	-0,73	-0,34	<b>-0,06</b>	-0,35	-0,95	-0,56
Algorithm Mean	-0,42	-0,33	-0,37	-1,76	-1,9	-1,83	-1,96	-1,32	-1,07	<b>-0,22</b>	-0,78	-1,39	

Tabelle 18: PSNR Differenz des MSR Sets mit Rauschen - MSR Set

Image Number	bikubisch	bilinear mean	bilinear median	DLM/MSE code	DLM/MSE paper	DLM/MSE paper RCD	HA	MHC	NN	PPG	RCD	Image Mean
1	-0,02	-0,01	-0,01	-0,02	-0,02	-0,02	-0,02	-0,02	-0,02	<b>-0,01</b>	-0,02	-0,0175
2	-0,01	<b>0</b>	<b>0</b>	-0,01	<b>0</b>	-0,01	<b>0</b>	-0,01	-0,01	-0,01	<b>0</b>	-0,005
3	-0,02	<b>-0,01</b>	<b>-0,01</b>	-0,02	-0,02	-0,02	<b>-0,01</b>	<b>-0,01</b>	-0,03	-0,02	-0,02	-0,0183
4	-0,02	<b>-0,01</b>	<b>-0,01</b>	<b>-0,01</b>	<b>-0,01</b>	<b>-0,01</b>	<b>-0,01</b>	<b>-0,02</b>	<b>-0,01</b>	<b>-0,01</b>	<b>-0,01</b>	-0,0125
5	-0,02	<b>-0,01</b>	<b>-0,01</b>	-0,02	-0,02	-0,02	-0,02	-0,02	-0,02	-0,02	-0,03	-0,02
6	-0,01	<b>0</b>	-0,01	-0,01	-0,01	-0,01	-0,01	-0,01	-0,01	-0,01	<b>0</b>	-0,0083
7	-0,01	<b>0</b>	-0,01	-0,01	-0,01	<b>0</b>	-0,01	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>	-0,005
8	-0,02	<b>-0,01</b>	<b>-0,01</b>	<b>-0,01</b>	<b>-0,01</b>	<b>-0,01</b>	<b>-0,01</b>	<b>-0,01</b>	<b>-0,02</b>	<b>-0,01</b>	<b>-0,01</b>	-0,0117
9	<b>-0,01</b>	<b>-0,01</b>	<b>-0,01</b>	-0,02	-0,02	-0,02	-0,01	-0,02	<b>-0,01</b>	<b>-0,01</b>	<b>-0,01</b>	-0,0142
10	-0,07	<b>-0,04</b>	-0,05	-0,07	-0,06	-0,07	-0,06	-0,06	-0,07	-0,07	-0,06	-0,0617
11	-0,07	<b>-0,06</b>	<b>-0,06</b>	-0,08	-0,07	-0,07	-0,07	-0,07	-0,09	-0,09	-0,07	-0,0717
12	-0,03	<b>-0,02</b>	<b>-0,02</b>	<b>-0,02</b>	<b>-0,02</b>	<b>-0,02</b>	<b>-0,02</b>	<b>-0,02</b>	-0,03	<b>-0,02</b>	<b>-0,02</b>	-0,0217
13	-0,02	-0,02	-0,02	-0,02	-0,02	-0,02	-0,02	-0,02	-0,03	-0,03	-0,02	-0,0208
14	<b>-0,01</b>	<b>-0,01</b>	<b>-0,01</b>	<b>-0,01</b>	<b>-0,01</b>	<b>-0,01</b>	<b>-0,01</b>	<b>-0,01</b>	-0,02	<b>-0,01</b>	<b>-0,01</b>	-0,0117
15	<b>-0,01</b>	<b>-0,01</b>	<b>-0,01</b>	<b>-0,01</b>	<b>-0,01</b>	<b>-0,01</b>	<b>-0,01</b>	<b>-0,01</b>	<b>-0,01</b>	<b>-0,01</b>	<b>-0,01</b>	-0,01
16	0,05	<b>0,06</b>	<b>0,06</b>	-0,06	-0,06	-0,07	-0,07	-0,07	-0,07	-0,06	-0,06	-0,0342
17	<b>-0,01</b>	<b>-0,01</b>	<b>-0,01</b>	<b>-0,01</b>	<b>-0,01</b>	<b>-0,01</b>	<b>-0,01</b>	<b>-0,02</b>	<b>-0,02</b>	<b>-0,01</b>	<b>-0,01</b>	-0,0133
18	-0,01	-0,01	-0,01	-0,01	-0,01	-0,01	-0,01	-0,01	-0,01	<b>0</b>	-0,01	-0,0092
19	<b>-0,01</b>	<b>-0,01</b>	<b>-0,01</b>	<b>-0,01</b>	<b>-0,01</b>	<b>-0,01</b>	<b>-0,01</b>	<b>-0,02</b>	<b>-0,01</b>	<b>-0,01</b>	<b>-0,01</b>	-0,0117
20	-0,02	<b>-0,01</b>	<b>-0,02</b>	-0,02	-0,02	-0,02	-0,02	-0,02	-0,03	-0,03	-0,02	-0,0208
21	<b>-0,01</b>	<b>-0,01</b>	<b>-0,01</b>	<b>-0,01</b>	<b>-0,01</b>	<b>-0,01</b>	<b>-0,01</b>	<b>-0,01</b>	<b>-0,01</b>	<b>-0,01</b>	<b>-0,01</b>	-0,0117
22	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>	-0,0033
23	-0,06	<b>-0,05</b>	<b>-0,05</b>	-0,08	-0,09	-0,09	-0,08	-0,09	-0,08	-0,07	-0,07	-0,0742
24	<b>-0,02</b>	<b>-0,02</b>	<b>-0,02</b>	<b>-0,02</b>	<b>-0,02</b>	<b>-0,02</b>	<b>-0,02</b>	<b>-0,02</b>	-0,03	<b>-0,02</b>	<b>-0,02</b>	-0,0208
500	<b>-0,01</b>	<b>-0,01</b>	<b>-0,01</b>	<b>-0,01</b>	<b>-0,01</b>	<b>-0,01</b>	<b>-0,01</b>	<b>-0,01</b>	<b>-0,01</b>	<b>-0,01</b>	<b>-0,01</b>	-0,01
Algorithm	-0,018	<b>-0,012</b>	-0,0136	-0,0224	-0,0224	-0,0228	-0,0212	-0,0236	-0,0256	-0,0252	-0,0216	-0,0208
Mean												

Tabelle 19: SSIM Differenz des MSR Sets mit Rauschen - MSR Set

## B. Installationsanleitung/ Installation Guide

Die folgende Installationsanleitung wurde von Florian Kellner verfasst.

The following installation guide was written by Florian Kellner.

E-Mail: mr.florian.kellner@posteo.de

### B.1. Prerequisites

This java project uses Apache Maven as a build tool.

The language version specified in the project is 12, so you need JDK 12 or higher (check with

```
java -version
```

(note the single dash!) from the command line/shell of your convenience).

Tip: On Debian-based Linux systems, you can install a specific java version via

```
apt install openjdk-<version>-jdk
```

(at the time of writing, the latest stable version is 17).

The following instructions assume you use Maven from the command line. You can of course run the same Maven Actions from your IDE, if supported.

### B.2. Building the DNG reader

Switch to the dng folder/project, open a new console and run

```
mvn verify
```

to compile it and see if everything works correctly. Then run

```
mvn install (or mvn clean install)
```

to publish the package to your local repository, so that it can be imported and used by the UI.

### B.3. Building the UI

Switch to the ui folder/project, open a new console and run

```
mvn verify.
```

Then run

```
mvn install (or mvn clean install).
```

The compiled JAR can be found in the generated target folder (/JENIFFER2/ui/target).

Run it with

```
java -jar Jennifer2-<version>-jar-with-dependencies.jar.
```

## **C. Benutzerhandbuch**

Das folgende Benutzerhandbuch wurde von Florian Kellner verfasst.  
E-Mail: mr.florian.kellner@posteo.de

# JENIFFER2 Benutzerhandbuch



Figure 1: Jeniffer2 Logo

*English version is available [here](#).*

## Was ist Jeniffer2?

JENIFFER steht für **J**ava **E**nhan**C**hed **N**ef **I**mage **F**ile **F**ormat **E**dito**R**. Version 2 benutzt stattdessen das Adobe-lizenzierte DNG-Format mit offener Spezifikation als Input (NEF ist das proprietäre Format von Nikon).

Jeniffer2 is Open Source Software zur Entwicklung von DNG-Dateien zu TIFF, JPEG oder PNG und bietet eine große Auswahl an Demosaicing-Algorithmen.

## Systemvoraussetzungen

### JRE 17 or neuer

Jeniffer2 benötigt eine Java Laufzeitumgebung (JRE) der Version 17 oder höher. Sie können die Version ihrer aktuellen Java Laufzeitumgebung testen, indem sie ein Teminal (MacOS/Linux) oder eine Eingabeaufforderung (Windows) öffnen und `java -version` eingeben. Die meisten Linux-Distributionen sollten Java bereits installiert haben.

Wir empfehlen die Verwendung des OpenJDK JREs:

- Auf Linux sollten die meisten Distributionen das Paket `openjdk-17-jre` oder `openjdk-19-jre` in den Standard-Paketquellen haben, auf einem Debian-basierten Linux kann die Installation also über `sudo apt-get install openjdk-17-jre` erfolgen.
- Eclipse Temurin stellt [Ausführbare Installer](#) für alle Betriebssysteme zur Verfügung ([Anleitung Windows](#), [Anleitung MacOS](#), [Anleitung Linux](#))

Wenn Sie Windows verwenden, sollten Sie darauf achten, die Option **JAVA\_HOME Umgebungsvariable anpassen** auszuwählen. Nur so können Sie Java von der Eingabeaufforderung aus starten und dabei Konfigurationsoptionen wie den verfügbaren Arbeitsspeicher anpassen.

## **RAW-Dateien im DNG-Format**

Die meisten proprietären RAW-Formate können entweder mithilfe des [Adobe DNG Converters](#) (läuft nur auf Windows und MacOS) oder einem Online-Tool wie [diesem](#) in das DNG-Format konvertiert werden.

## **Jeniffer2 ausführen**

Jeniffer2 wird als Java-Archiv (JAR Datei) ausgeliefert. Um es auszuführen, müssen Sie ein Terminal/eine Eingabeaufforderung in dem Ordner, in dem Sie die Datei gespeichert haben, öffnen, und den folgenden Befehl ausführen:

```
java -jar Jeniffer2.1.jar
```

## **Mehr RAM einstellen**

Java-Programme haben eine festgelegt maximale RAM-Belegung (Heap Size), der Standardwert auf Ihrem System ist möglicherweise zu gering. Um Jeniffer2 die Belegung von mehr Arbeitsspeicher zu ermöglichen, können Sie beim Programmstart explizit ein Maximum setzen, z.B. so:

```
java -Xmx8192M -jar Jeniffer2.1.jar
```

für 8GB oder so:

```
java -Xmx4096M -jar Jeniffer2.1.jar
```

für 4GB. **Setzen Sie diesen Wert niemals auf die Menge des im System verbauten RAMs!** Andere Programme und vor allem Ihr Betriebssystem benötigen ebenfalls Arbeitsspeicher, um zu funktionieren.

## **Gespeicherter Zustand**

Jeniffer2 schreibt eine versteckte Datei namens `folderSave` in den Ordner, in dem es ausgeführt wird. Hier wird der zuletzt in der Baumansicht ausgeklappte Ordner gespeichert, um diesen beim nächsten Öffnen wieder auszuklappen.

## **Logs**

Jeniffer2 erstellt einen Ordner namens `jeniffer2-logs` in dem Ordner, in dem es aufgerufen wird. Hier werden Systeminformationen und Ausführungszeiten einzelner Verarbeitungsschritte im CSV-Format gespeichert. Sie können diese Dateien gerne mit einem Text-Editor, Python, R, Excel usw. öffnen, wenn Sie neugierig sind, aber wenn Sie zur Forschung zur Entwicklung von Jeniffer2 beitragen wollen, ändern Sie diese Dateien bitte nicht, bevor Sie sie abgeben.

## **Danksagung**

Jeniffer2 wird unter der Leitung von Prof. Thomas Walter an der Uni Tübingen entwickelt.

Credits gehen an:

- Eugen Ljavin
- Joachim Groß
- Michael Kessler
- Claudia Grosch
- Andreas Reiter
- Florian Kellner

## Quellcode

Wir arbeiten gerade daran, den Quellcode von Jeniffer2 zu veröffentlichen. Bis dahin können Sie [Florian Kellner](#) kontaktieren, um eine Kopie des Quellcodes zu erhalten oder Feedback zu geben.

## **D. User Manual**

The following user manual was written by Florian Kellner.

E-Mail: mr.florian.kellner@posteo.de

# JENIFFER2 User Manual



Figure 1: Jeniffer2 Logo

*Deutsche Version ist hier verfügbar.*

## About

JENIFFER stands for **J**ava **E**nhan**c**ed **N**ef **I**mage **F**ile **F**ormat **E**dito**R**. Version 2 is using the Adobe licensed open DNG format as input instead of the proprietary Nikon NEF format.

JENIFFER2 is Open Source Software for developing DNG files to TIFF, JPEG or PNG, offering a big choice of demosaicing algorithms.

## Prerequisites

### JRE 17 or higher

JENIFFER requires you to have a Java Runtime Environment (JRE) of Version 17 or higher installed.

You can check the version of your installed JRE by opening a command prompt (Windows) or terminal (MacOS/Linux) of your choice and running `java -version`. Modern Linux distributions should already have a version of Java installed.

We recommend using the OpenJDK JRE:

- On Linux, most distributions already provide the `openjdk-17-jre` or `openjdk-19-jre` package, which can e.g. on Debian-based systems installed via `sudo apt-get install openjdk-17-jre`.
- Eclipse Temurin provides [Downloadable Installers](#) for all operating systems as well as Installation Instructions ([Windows](#), [MacOS](#), [Linux](#))

Windows users should make sure to check the box to **update the JAVA\_HOME environment variable** during the installation process in order to be able to run Java from the command line and pass options such as allocating more memory.

### Raw images in DNG format

Most proprietary RAW image formats like .NEF can be converted to DNG either via the [Adobe DNG Converter](#) (runs on MacOS and Windows) or an online tool

like [this one](#).

## Running JENIFFER2

Jeniffer2 is distributed as a Java Archive (JAR file). To run it, open a terminal/command prompt in the folder where you saved it and run

```
java -jar Jennifer2.1.jar
```

### Adding more RAM

Java programs have a fixed maximum memory allocation (heap size), the default on your system may be too low. To allow Jeniffer2 to use more memory, you can explicitly set the maximum, e.g. like this:

```
java -Xmx8192M -jar Jeniffer2.1.jar
```

for 8GB of RAM or like this:

```
java -Xmx4096M -jar Jeniffer2.1.jar
```

for 4GB of RAM. **Never set this value to the amount of RAM available on your system!** Other programs as well as your operating system need some RAM, too.

### Saved GUI state

Jeniffer2 writes a hidden `folderSave` file to the folder it is run in, where it stores the last location you opened in the file tree view, so that it will be expanded when you open the program again.

### Logs

Jeniffer2 creates a `jeniffer2-logs` folder in the folder it is run in. It stores performance and system information in CSV format. You can open and inspect these files with a text editor, python, R, Excel... if you are curious, but if you want to contribute to the research concerning Jeniffer2 development, please do not edit them before handing them in.

### Credits

Jeniffer2 has been developed under the supervision of Prof. Thomas Walter at Tübingen University.

Credits go to:

- Eugen Ljavin
- Joachim Groß
- Michael Kessler
- Claudia Grosch

- Andreas Reiter
- Florian Kellner

## Source Code

We are currently working on making the source code of Jeniffer2 public. In the meantime, you can contact [Florian Kellner](#) for the source code or with any issues you might have.

## **E. Rechtlicher Hinweis**

- Dieses Produkt enthält die bei Adobe lizenzierte DNG-Technologie.
- This product includes DNG technology under license by Adobe.