Abstract

Inhaltsverzeichnis

1 Verlustbehaftete Kompression von volumetrischen Punkten **1**

2 State of the Art **3**
2.1 JPEG/JFIF Bildkompression . . . 3
2.2 Point Cloud Kompression . . . 4
2.3 Curve Fitting Transformation . . . 4
2.4 Compressive Sensing . . . 4
2.5 Entropie Kodierung . . . 4

3 Kompressionsverfahren der Feldlinien **6**
3.1 Ist-Komprimierung . . . 6
3.2 Lösungsansatz: Adaptives Subsampling . . . 6
3.2.1 Adaptives Subsampling . . . 7
3.2.2 Entropie Kodierung mittels RAR . . . 7
3.3 Lösungsantz: Diskrete Kosinus Transformation . . . 8
3.3.1 Subsampling . . . 8
3.3.2 Ableitung . . . 8
3.3.3 Cosinus-Transformation . . . 9
3.3.4 Quantisierung . . . 9
3.3.5 Entropie Kodierung . . . 9
3.4 Lösungsansatz: Prediktive Kodierung . . . 11
3.4.1 Angle Subsampling . . . 11
3.4.2 Quantisierung . . . 11
3.4.3 Wavelet Prediktive Kodierung . . . 11
3.4.4 Quantisierung . . . 11
3.4.5 Entropie Kodierung . . . 11

4 Qualitätsmessung der dekomprimierten Daten **13**
4.1 Auswahl und Erhebung der Testdaten . . . 13
4.2 Berechnung der Standardabweichung . . . 13
4.2.1 Berechnung der Standardabweichung . . . 14
4.3 Berechnung der angepassten PSNR-HVS-M . . . 15
4.3.1 Contrast Masking . . . 15
4.3.2 Umsetzung und Anpassung der PSNR-HVS-M für diese Arbeit . . . 16

5 Implementation **18**
5.1 Software Architektur . . . 18
5.1.1 Vorladen und Caching von Komprimierten und Dekomprimierten Feldlinien . . . 18
5.1.2 Asynchrone Aufrufe mittels Executor Services . . . 19

6 Resultate **20**
6.1 Lösungsansatz: Adaptives Subsampling . . . 20
6.2 Lösungansatz: Diskrete Kosinus Transformation . . . 21
6.2.1 Variante: DCT . . . 21
6.2.2 Variante: Ableitung+DCT . . . 23
6.2.3 Variante: PCA+Ableitung+DCT . . . 24
6.2.4 Variante: Ableitung+DCT+Byte Kodierung . . . 24
6.2.5 Variante: Randbehandlung+DCT+Byte Kodierung . . . 25
6.2.6 Ringing Artefakte . . . 26
6.2.7 Behandlung der Ringing Artefakte . . . 27

6.2.8 Abschliessende Variante 28
6.3 Lösungsansatz: Prediktive Kodierung 31
6.3.1 Variante: einfaches Subsampling 31
6.3.2 Variante: Adaptiven Subsampling 32
6.3.3 Wavelet Prediktive Kodierung 32

7 Diskussion **33**
7.1 Diskussion Lösungsansatz Adaptives Subsampling 33
7.2 Diskussion Lösungsansatz DCT 33

8 Fazit **35**

9 Anhang **38**
9.1 Installationsanleitung 38
9.1.1 Installation des JHelioviewers 38
9.1.2 Installation der Kompression 38
9.2 Dateiformat und Dekompression 39
9.2.1 Encodings 39
9.2.2 Transformation to Euler Coordinates 40
9.3 Performance Tests 41

10 Ehrlichkeitserklärung **42**

1 Verlustbehaftete Kompression von volumetrischen Punkten

Moderne Simulationen sind in der Lage grosse Mengen an Daten zu produzieren. Die rohe Datenmenge ist oft zu gross um sie zu archivieren oder in vernünftiger Zeit über eine Internetverbindung zu übertragen. In wissenschaftlichen Simulationen werden natürliche Phänomene wie Schwingungen, Flugbahnen, Kraftfelder etc. als Menge von Kurven im dreidimensionalen Raum abgebildet. Die Kurven sind dargestellt als Folge von Punkten. Ziel dieser Arbeit ist es eine verlustbehaftete Kompression für die Punktfolgen zu entwickeln, welche die Übertragung über eine Internetverbindung ermöglicht.

Im Rahmen dieses Projekts sollen Daten von Magnetfeldlinien der Sonne komprimiert werden, welche über eine Internetverbindung zum JHelioviewer übertragen werden. Der JHelioviewer ist eine Applikation zur visualisierung von Satellitenmessdaten und Simulationen der Sonne. Die Applikation wird von der ESA und der FHNW entwickelt. Die Abbildung 1 zeigt eine Visualisierung des JHelioviewers von der Sonnenoberfläche und Feldlinien.
Auf der Visualisierung sind Feldlinien in drei unterschiedlichen Farben zu erkennen, welche drei unterschiedle

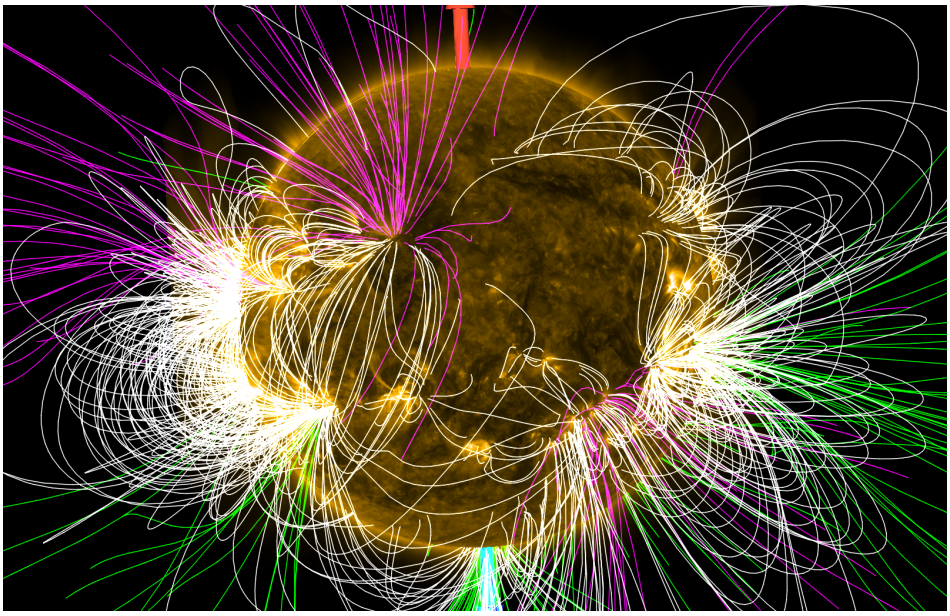

Abbildung 1: Visualisierung der Feldlinien im JHelioviewer

Typen darstellen: Linien, die auf der Sonne starten und wieder auf der Sonne landen, auf der Sonne starten und ins Weltall führen oder vom Weltall auf der Sonne landen. Die weissen Feldlinien repräsentieren ”Sonne zu Sonne´´, die Grünen ”Sonne zu Weltall´´ und die Violetten ”Weltall zu Sonne´´.

Im Vorfeld wurde bereits eine verlustbehaftete Kompression entwickelt, welche die Datenmenge auf durchschnittlich 1 Megabyte pro Simulation reduziert. Der JHelioviewer erstellt eine Animation der unterschiedlichen Daten, so werden nicht eine sondern mehrere Simulationen in Abfolge visualisiert. Es soll möglich sein, 1000 komprimierte Simulationen zwischenzuspeichern und möglichst wenig Arbeitsspeicher in Anspruch zu nehmen. In erster Linie wird erforscht, wie eine Kompression das Caching realisierbar macht. Ziel ist es eine Kompressionsrate von 8 bis 10 zu erreichen mit einer vergleichbaren Qualität zur Ist-Kompression.
Dass der Benutzer nicht immer warten muss, bis alle Simulationen heruntergeladen wurden, wird in dieser Arbeit nach Streamingmöglichkeiten gesucht.

Die Feldlinien werden mit einer Potential Field Source Surface (PFSS) Simulation berechnet, welche aus Messungen der Sonnenoberfläche die Feldlinien extrapoliert. Pro Simulation werden ungefähr 1.5 Mibyte an Daten produziert. Über eine Internetverbindung stellt ein Server die Simulationsdaten bereit.

Der JHelioviewer visualisiert eine Abfolge von Messungen und Simulationen, welche zur Laufzeit über eine Internetverbindung geladen werden. Für die Visualisierung der Feldlinien bedeutet das, dass mehrere Simulationen geladen werden müssen und dass die Bandbreite von anderen Daten bereits beansprucht wird. Deshalb soll eine verlustbehaftete Kompression umgesetzt werden, welche das Übertragen der Simulationen beschleunigt und weniger Bandbreitenintensiv gestaltet.

Im laufe des Projekts wurden zwei Kompressionen entwickelt. Die Kompressionsraten sind der Tabelle 1 ersichtlich.

Lösungsansatz	Kompressionsraten
Ist-Zustand	1
Adaptives Subsampling	11.6
Diskrete Kosinus Transformation	14.1

Tabelle 1: Tabelle der Lösungsansätzen und dessen Kompressionsraten.

2 State of the Art

Grob betrachtet können die Teilschritte einer verlustbehaftete Kompressionen in drei Verarbeitungsarten eingeteilt werden: Transformationen, Quantisierungen und Entropie Kodierungen. Die Abbildung 2 zeigt eine vereinfachte Abfolge. Die Inputdaten werden durch ein oder mehrere Verfahren transformiert. Die Transformationen haben das Ziel, dass die folgende Quantisierung unwichtige Informationen löschen kann. Die Transformationen sind meistens verlustfrei umkehrbar, sodass nur in den Quantisierungsschritten Informationen gelöscht werden. Die reduzierte Information wird Entropie Kodiert. Die Entropie Kodierung ist typischerweise für die Datenreduktion verantwortlich.

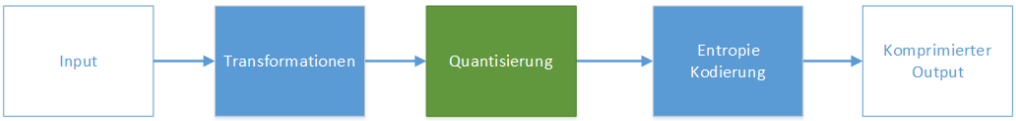

Abbildung 2: Vereinfachter Ablauf einer verlustbehafteten Kompression

2.1 JPEG/JFIF Bildkompression

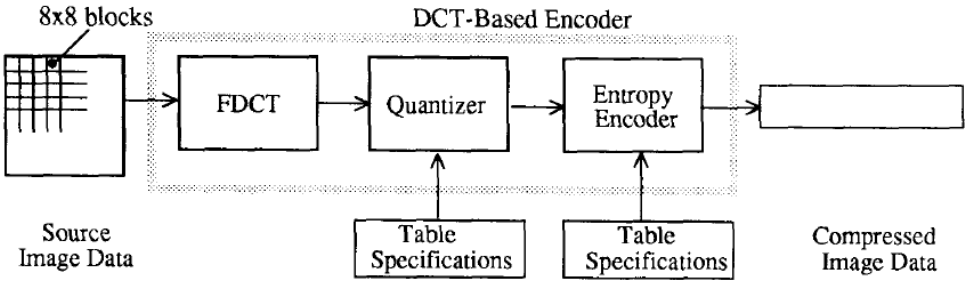

Abbildung 3: Aufbau der JPEG Kompression [1]

Das JPEG/JFIF Format ist eines der meist-verwendeten Bildkompressionsverfahren für natürliche Bilder. Das Diagramm der Abbildung 3 zeigt den Aufbau der Kompressionspipeline. JPEG/JFIF unterteilt das Eingabebild in $8 * 8$ Blöcke und führt auf ihnen eine Diskrete Kosinus Transformation (DCT) durch. Der Bildblock ist nun als Folge von Kosinus-Funktionen dargestellt.
Die Quantisierung versucht nun Frequenzen, welche das menschliche Auge schlecht erkennen kann weniger Präzise darzustellen. Wenn die Quantisierung gut gewählt wurde, kann der Mensch das dekomprimierte Bild nicht vom Original unterscheiden, verbraucht aber weniger Speicherplatz. JPEG/JFIF bietet vorgefertigte Quantisierungstabellen an. Der Benutzer kann aber eigene Tabellen für spezifizieren. Wie die Quantisierungstabelle optimal gewählt wird, ist ein aktives Forschungsfeld [2] [3] und kann von Anwendungsfall zu Anwendungsfall unterschiedlich sein.
Nach der Quantisierung werden die quantisierten Blöcke im Zick-Zack-Muster angeordnet, sodass die Entropie Kodierung eine bessere Kompression durchführen kann. JPEG/JFIF führt eine Run-Length und eine Huffman-Kodierung durch. JPEG bietet auch hier an, eine benutzerspezifizierte Huffman-Tabelle zu verwenden. Wenn keine Tabelle spezifiziert ist, wird

Wenn für die Kompression von wissenschaftlichen Daten eine Diskrete Kosinus Transformation eingesetzt werden soll, kann ein ähnlicher Aufbau verwendet werden wie die JPEG/JFIF Kompression. Für eine optimale Kompression wird die Umsetzung der einzelnen Schritte vermutlich von JPEG/JFIF abweichen.

2.2 Point Cloud Kompression

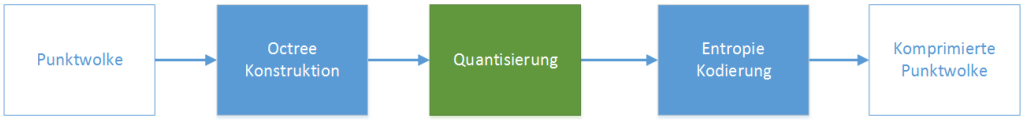

Abbildung 4: Aufbau einer Octree basierten Point Cloud Kompression.

3d Laser Sampling Geräte produzieren grosse Mengen an volumetrischen Punkten von alltäglichen Objekten. Die Kompression von solchen Punktwolken ist ein aktives Forschungsfeld. Eine vorgeschlagene Kompression von Schnabel und Klein [4] verwendet Octrees [5]. Das Diagramm der Abbildung 4 verdeutlicht den Ablauf.
Die dreidimensionalen Punkten werden in einem Octree mit einer begrenzten Anzahl an Levels abgelegt. Im Quantisierungsschritt werden die Punkte durch die Zellenmittelpunkte des Octrees ersetzt. Die Anzahl an Levels ist gleichzusetzen mit der Genauigkeit in Bits welche für jede Koordinatenachse zur Verfügung stehen. Wenn die Levels auf 8 begrenzt sind, steht für jede Achse 8 Bit Genauigkeit zur Verfügung.
In der Entropie Kodierung wird der Octree zuerst Binär abgebildet (0 für leere Knoten, 1 für befüllte Knoten) in Breadth-First Ordnung. Jedes Level im Octree stellt eine Approximation der Punktwolke dar. Durch die Breadth-First Ordnung werden die ungenaueren Approximationen zuerst abgelegt. Diese Eigenschaft wird mit einer Prediktiven Kodierung ausgenutzt: Aus den vorhergehenden Levels wird eine Vorhersage erstellt, wie die Punkte im nächsten Level zu liegen kommen. Die Kodierung kann die Information reduzieren, indem nur noch der Fehler der Kodierung abgespeichert wird. Je besser die Kodierung das Verhalten vorhersagen kann, desto besser ist die Kompression.

Für diesen Ansatz muss zu einem die Information, welcher Punkt zu welcher Linie gehört, gespeichert werden. Jedem Punkt kann eine Farbe zugewiesen werden. Die Farbe könnte man brauchen um die Punkte den Feldlinien zuzuordnen. Weiter müsste vermutlich die Prediktive Kodierung angepasst werden: Schnabel und Klein nehmen an, dass die Punktwolke eine Oberfläche darstellen. Die Feldlinien jedoch bilden im Allgemeinen keine Oberfläche und benötigen deshalb eine andere Prediktive Kodierung.

2.3 Curve Fitting Transformation

Im Curve Fitting wird ein Signal durch eine oder mehreren Funktionen dargestellt. Es ist dabei zwischen einem exakten Curve Fitting und einer Approximation zu unterscheiden. Die exakte Repräsentation wird typischerweise für die Signalinterpolation verwendet, wäherend eine Approximation in der Rauschunterdrückung zum Einsatz kommt.

Unser et al [6] zeigt ein Algorithmus, welche ähnlich wie eine Diskrete Kosinus Transformation ein Diskretes Signal als Folge von B-Splines darstellt. Es wird ebenfalls eine verlustbehaftete Bildkompression vorgestellt als Anwendungsfall.

Ein Curve-Fitting könnte die Daten approximieren und deutlich weniger Speicherplatz verbrauchen. Es existiert aber kein Datenformat, welche ein Curve-Fitting als Kompression verwendet. Hier liegt noch zusätzlichen Aufwand verborgen.

2.4 Compressive Sensing

2.5 Entropie Kodierung

Die Entropie Kodierung findet in allen Bereichen der Informatik Anwendungen. Ziel ist es die gleiche Information mit einem Minimum an Daten darzustellen. Verlustbehaftete Kompressionsverfahren reduzieren die

Information und verwenden im letzten Schritt eine Entropie Kodierung um die Datenmenge zu reduzieren.

Typische Verfahren sind Run Length, Huffman oder Aritmetische Kodierung. Die Information mit einem Minimum an Daten darzustellen hängt von der Information ab. Die typischen Verfahren sind weit verbreitet, da sie zwar nicht immer das Optimum finden, aber unabhängig von der Art der Information funktionieren. Es ist für Huffman nicht relevant, ob Floating-Point Zahlen oder Text. Speed

Es existieren auch Entropie Kodierer, welche auf eine Art von Information spezialisiert sind. So kann

Es ist ein verlustfreies Verfahren Information mit möglichst wenigen Bytes abzubilden. In verlustbehafteten Kompressionsverfahren ist die Entropie Kodierung meist der letzte Schritt. Deshalb ist ein wichtiger Teilschritt der Entropie Kodierung die Anordnung und Darstellung der Daten.
Je nach dem wie die Daten aussehen, ist ein Verfahren besser als das Andere. Die Kodierungen werden oft kombiniert für eine bessere Kompression. Die kombinationen sind in Paketen mitgeliefert und können auf beliebige Inputdaten angewendet werden.

Es existieren auch spezialisierte Entropie Kodierer beispielsweise für Floating Point Kodierer [7] effizient komprimieren können.
Frage Kompressionsrate und Laufzeit.

Werden oft in Kompbination verwendet, zb. Rle + huffman

Wird zuerst sicher ein allgemeines Verfahren.

3 Kompressionsverfahren der Feldlinien

In diesem Abschnitt werden die einzelnen Verfahren zu den Lösungsansätzen vorgestellt. Im ersten Abschnitt 3.1 wird die Ist-Kompression analysiert, während in den nächsten Abschnitten die Lösungsansätze vorgestellt werden und die Teilschritte erklärt.

3.1 Ist-Komprimierung

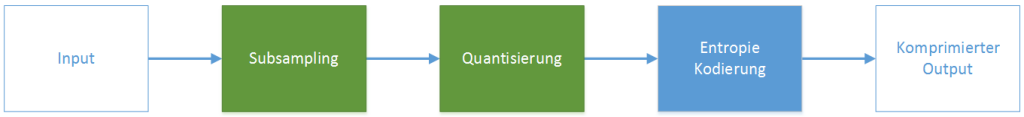

Abbildung 5: Aufbau der Ist-Kompression.

Das Ziel der Ist-Kompression ist es, die Datenmenge mit einfachen Mitteln drastisch zu reduzieren. Der Aufbau ist im Diagramm der Abbildung 5 dargestellt. Die Ist-Kompression führt zuerst ein Subsampling durch. Drei Viertel aller Punkte werden in diesem Schritt verworfen. Im nächsten Schritt werden die übrigen Punkte auf 16-Bit Integer diskretisiert. Das reduziert die Anzahl Bytes und verbessert die Kompression im Schritt Entropie Kodierung. Die Implementationen der Entropie Kodierer scheinen Integer-Werte einfacher komprimieren zu können. In der Entropie Kodierung werden die Daten geordnet wie in Tabelle 2 dargestellt. Als erstes werden

Anzahl Punkte der Feldlinien	X Kanal aller Punkte	Y Kanal aller Punkte	Z Kanal aller Punkte

Tabelle 2: Anordnung der Simulationsdaten der Ist-Kompression

die Längen aller Feldlinien abgelegt. Danach folgt der X, Y und der Z Kanal aller Punkte der Feldlinien. Diese Anordnung verbessert die Kompressionsrate der Entropie Kodierung. Je näher ähnliche Muster beieinander liegen, desto besser können sie Komprimiert werden. Für die eigentliche Entropie-Kodierung wird Gzip verwendet. Gzip basiert auf dem Deflate Algorithmus, welcher aus einer Kombination von LZ77 und Huffman Kodierung besteht [8].
Die Punktmenge ist für Low-End Grafikkarten zu gross. Um die Punktmenge für die Visualisierung zu verkleinern, führt der JHelioviewer ein weiteres Subsampling durch, welches im Abschnitt 3.2.1 beschrieben ist.

3.2 Lösungsansatz: Adaptives Subsampling

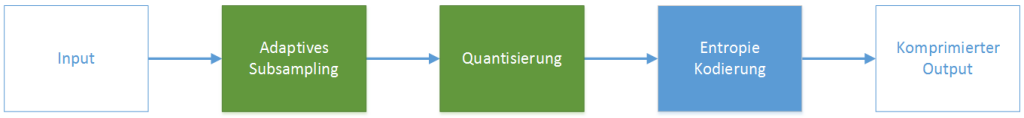

Abbildung 6: Aufbau des Lösungsansatzes: Adaptives Subsampling.

Dieser Lösungsansatz verwendet einen ähnlichen Aufbau wie die Ist-Kompression 3.1. Der Unterschied ist, dass das Subsampling Verfahren gewählt wurde, welches der JHelioviewer auf dem Client durchführt. Die Abbildung 6 zeigt den neuen Ablauf.
Dieser Ansatz überträgt nur die Information, die der JHelioviewer zur Visualisierung benötigt. Der Vorteil ist dass die Information, die der JHelioviewer visualisiert ein Bruchteil darstellt und somit eine hohe Kompressionsrate ereicht werden kann. Es ist möglich dass die Visualisierung zu einem späteren Zeitpunkt mehr Punkte darstellen soll. In diesem Fall müssten entweder mehr Punkte übertragen werden, was eine schlechtere Kompressionsrate zur Folge hat, oder der JHelioviewer muss eine Interpolation durchführen.

3.2.1 Adaptives Subsampling

Konzeptionell approximiert das adaptive Subsambling eine Kurve diruch eine Folge von Strecken. Je stärker die Krümmung der Kurve, desto mehr Strecken werden für die Approximation benötigt. Das Diagramm der

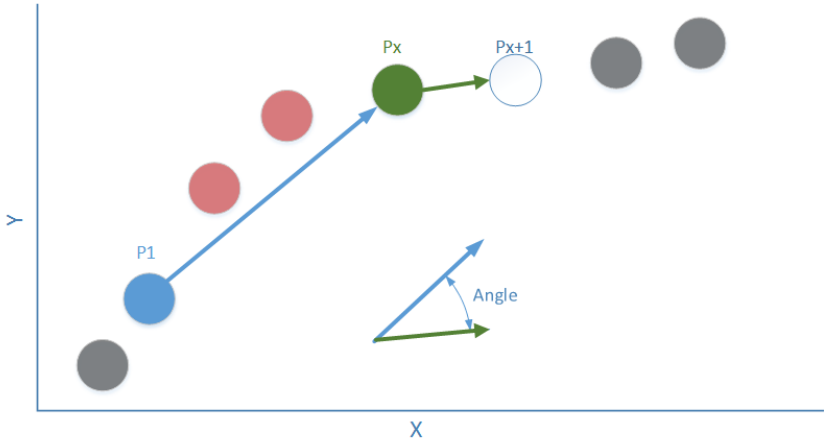

Abbildung 7: Darstellung des Adaptiven Subsapmlings im 2D Raum. Rot sind die Punkte, welche geprüft und gelöscht wurden. Grün ist der Punkt dargestellt, welcher geprüft wird.

Abbildung 7 stellt das Subsampling im zweidimensionalen Raum dar. Das Adaptive Subsampling wählt nun Punkte P aus der Feldlinie aus, welche Start- und Endpunkte der Strecken darstellen.
P_1 wurde bereits ausgewählt. Es wird nun ein Punkt P_x gesucht, der als Endpunkt einer Strecke von P_1 zu P_x die Feldlinie approximiert. Dazu wird der Winkel der Strecke P_1 zu P_x mit der Strecke P_x zu $P_x + 1$ verglichen. Wenn der Winkel kleiner ist, als ein Winkel α, wird der nächste Punkt $P_x + 1$ überprüft. Wenn der Winkel grösser ist, wird P_x ausgewählt. Danach wird der wird eine nächste Strecke startend von P_x gesucht.

3.2.2 Entropie Kodierung mittels RAR

Die Anordnung der Daten wurde aus der Ist-Kompression übernommen, jedoch wird Rar anstatt GZip verwendet. GZip konnte bei den Ist-Komprimierten Daten eine Kompressionsrate von 1.2 erreichen, während Rar bei selben Daten eine Rate von 3.7 erreicht.

3.3 Lösungsantz: Diskrete Kosinus Transformation

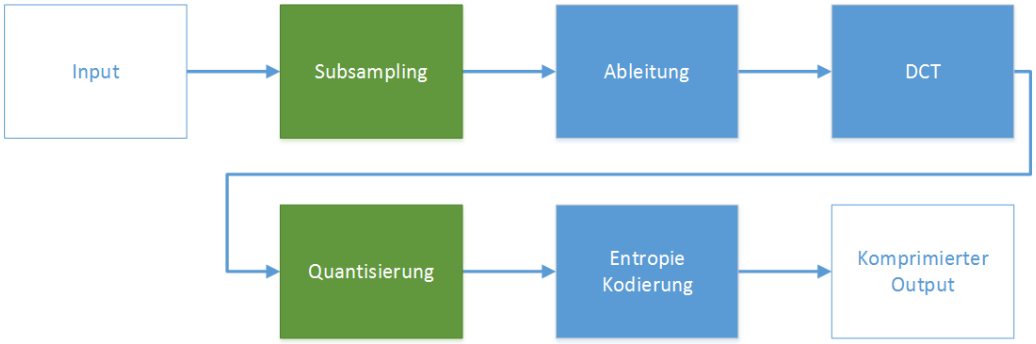

Abbildung 8: Aufbau des Lösungsansatzes: Adaptives Subsampling.

Die Kompression dieses Lösungsansatzes ist Dargestellt im Diagramm der Abbildung 8. Konzeptionell ähnelt dieser Ansatz der JPEG/JFIF Kompression (dargestellt in der Abbildung 3), die einzelnen Teilschritte können aber andere Algorithmen verwenden. Im Vergleich zum JPEG/JFIF Standard ist der grösste Unterschied, dass das Eingangssignal vor der Kosinus Transformation abgeleitet wird.

Die Feldlinien ähneln oft harmonischen Halbwellen, welche sich durch wenige Kosinusfunktionen approximieren lassen können. Um eine optimale Kompression mit dieser Variante zu erreichen, müssen Ringing Artefakte [9] behandelt werden. Sie äussern sich als Oszillieren im dekomprimierten Signal, was das menschliche Auge als störend empfindet. Beispiele für die Ringing Artefakten von Feldlinien sind im Abschnitt 6.2.6 zu finden.
Es gibt Möglichkeiten, die Ringing Artefakte zu Dämpfen oder gar zu beheben: Die simpelste Variante ist es, das dekomprimierte Signal zu glätten. Die Glättung kann das Signal verfälschen und ist deshalb nicht die optimale Lösung. In der Bild- und Audioverarbeitung wird aktiv nach Post-Processing Filter geforscht, welche die Ringing Artefakte in der Dekompression vermindern [11] [12]. Ein angepasstes Postprocessing wurde im Rahmen dieser Arbeit nicht implementiert.

3.3.1 Subsampling

Das Subsampling wurde aus der Ist-Kompression 3.1 übernommen und dient, die DCT zu beschleunigen. Da die DCT eine Komplexität von $O(n^2)$ aufweist, wird durch das Subsampling die Transformation wesentlich beschlenuitgt.
Falls die Laufzeit der Dekompression weiter verbessert werden soll, kann die Fast-Cosine-Transformation umgesetzt werden. Diese hat eine Komplexität von $O(nlog(n))$. Falls das nicht ausreicht, können die Linien in Blöcke unterteilt werden und die DCT pro Block ausführen. Dadurch wird die Komplexität auf $O(n)$ gesenkt. Jedoch ist es wahrscheinlich, dass durch die Unterteilung die Kompressionsrate leidet. Die Approximation mehrer Blöcke des Eingabesignals benötigt schlussendlich mehr Kosinusfunktionen, als die Approximation des gesamten Signals.

3.3.2 Ableitung

Das Eingabesignal wird abgeleitet und alle folgenden Transformationen werden auf den Steigungen des Signals ausgeführt. Damit die Transformation umkehrbar ist, muss der Startpunkt zusätzlich abgespeichert werden.
Die Artefakte, welche das abgeleitete dekomprimierte Signal beinhaltet, sind bei der Feldliniensimulation weniger störend für das menschliche Auge. Die Feldlinie bleibt tendenziell glatt und Artefakte äussern sich meist in veränderten Amplituden, welche erst erkennbar sind, wenn die Originalfeldlinie zum Vergleich bereit steht.

Diese Eigenschaft wirkt sich ebenfalls auf die Ringing-Artefakte aus. Die Ableitung hat einen dämpfenden Effekt auf die Ringing Artefakte.

3.3.3 Cosinus-Transformation

Die Diskrete Kosinus Transformation stellt eine endliche Menge von N Datenpunkten als N Kosinusfunktionen zu verschiedenen Frequenzen dar. Die Werte DCT-Koeffizienten stellt dar, wie hoch der Anteil einer bestimmten Frequenz ist im Originalsignal. Im optimalen Fall kann ein Signal durch niederfrequente Funktionen approximiert werden. Die hochfrequenten Anteile stellen Details dar, welche meist nicht relevant sind.
Es gibt verschiedene Möglichkeiten die Punkte zu transformieren. Hier wurde sich am JPEG/JFIF Standard orientiert, welche die DCT-II (3.1) als Forwärts und die DCT-III (3.2) als Rückwärtstransformation verwendet [1].

$$X_k = \sum_{n=0}^{N-1} x_n * cos[\frac{\pi}{N} k(n + \frac{1}{2})] \quad k = 0, 1, \ldots, N-1 \tag{3.1}$$

$$x_n = \frac{1}{2} X_0 + \sum_{k=1}^{N-1} X_k * cos[\frac{\pi}{N} k(n + \frac{1}{2})] \quad n = 0, 1, \ldots, N-1 \tag{3.2}$$

N bezeichnet die Länge des Eingabesisignals, x_n bezeichnet einen Wert im diskreten Signal und X_k ist der Anteil der Frequens k. Ein Eingabesignal der Länge N resultiert in N Kosinus-Funktionen.

Die DCT transformiert ein periodisches, unendliches Signal. Um ein endliches Signal zu transformieren, wird es konzeptionell wiederholt. Die gewählten Verfahren wiederholen das Signal jeweils in umgekehrter Reihenfolge.
Die Wiederholung beeinflusst die Transformation nicht, wenn das Signal an den Rändern abflacht. Falls das Signal nicht abflacht, können nach der Quantisierung an den Rändern markante Artefakte auftreten. Ein Beispiel für solche Artefakte ist im Diagramm der Abbildung 19 im Abschnitt 6.2.1 zu sehen.

3.3.4 Quantisierung

In der Visualisierung werden die Feldlinien in drei Typen unterschieden: ”Sonne zu Sonne”, ”Sonne ins Weltall” und ”Weltall zur Sonne”. Die ”Sonne zu Sonne” Feldlinien können besser durch Kosinusfunktionen approximiert werden und sind weniger Anfällig auf die Ringing Artefakte. Dieser Typ von Feldlinien wird deshalb stärker Quantisiert. Für die ”Sonne zu Sonne” Feldlinien werden maximal 35 DCT Koeffizienten abgelegt, wobei die letzten zehn Koeffizienten kaum noch Einfluss besitzen. Sie werden mit einem hohen Faktor quantisiert, sodass sie im Normalfall ebenfalls Null sind.
Die anderen Feldlinien benötigen mehr Koeffizienten für die Dämpfung der Ringing Artefakte. Bei ihnen liegt das Maximum bei 50 Koeffizienten.

3.3.5 Entropie Kodierung

Um die Entropie Kodierung zu verbessern wurden zwei Byte-Kodierungen hinzugefügt. Die Kanäle werden zuerst mit der Längenkodierung und darauf folgend mit der adaptive Genauigkeit kodiert.

Längenkodierung
Die Quantisierung erlaubt nur eine maximale Anzahl an Koeffizienten. Ab dem 50 Koeffizient, sind die Werte Null. Die Längenkodierung schneidet den Block von Null-Koeffizienten ab und fügt die Länge des Nicht-Null Blockes hinzu. Alle Längen und alle Blöcke werden zusammen abgespeichert, die Tabelle 3 verdeutlicht das Konzept. n_i ist die Länge der Feldlinie i und $x_{i,j}$ ist der j DCT-Koeffizient der Feldlinie i.

Um einen Kanal zu Dekodieren, muss die Anzahl an DCT-Koeffizienten n_i gelesen werden und danach

X Kanäle							
Block Feldlinie 0					...		
n_0	$x_{0,0}$	$x_{0,1}$	...	$x_{0,n-1}$	n_1	$x_{1,0}$	...

Tabelle 3: Beispiel eines abgespeicherten Kanals mit der Längenkodierung.

Nullen Anfügen, bis die ursprüngliche Länge des Kanals erreicht wurde. Die Anzahl Punkte jeder Feldlinie ist ebenfalls gespeichert.

Die Längenkodierung ist effizient, wenn die hochfrequenten DCT-Koeffizienten einen grossen, zusammenhängenden Null-Block bilden. Im schlechtesten Fall währe der letzte DCT-Koeffizient nicht Null. In diesem Fall würde die Längenkodierung nichts abschneiden.

Adaptive Genauigkeits-Kodierung
Die quantisierten Koeffizienten liegen meistens zwischen -50 und $+50$. Acht Bit Genauigkeit reichen im Allgemeinen aus um einen DCT-Koeffizienten abzuspeichern. Mit der adaptiven Genauigkeits-Kodierung sollen so wenige Bytes pro Koeffizient abgespeichert werden, wie benötigt werden. Dazu wird wie in Tabelle 4 eine neue Byte-Kodierung eingeführt. Das MSB wird als ”Continue Flag” verwendet. Wenn es gesetzt ist, gehört das folgende Byte ebenfalls zur Zahl.
Wenn ein Kanal hauptsächlich aus tiefen Zahlen besteht, können so Bytes gespart werden, ohne Genauigkeit

Byte							
Continue Flag	X	X	X	X	X	X	X

Tabelle 4: Aufteilung eines Bytes der adaptiven Genauigkeitskodierung. X sind Nutz-Bits.

zu verlieren. Wenn aber der Kanal aus hauptsächlich grossen Zahlen besteht, welche zwei oder mehr Bytes an Genauigkeit brauchen, wird weniger Speicherplatz gespart.

3.4 Lösungsansatz: Prediktive Kodierung

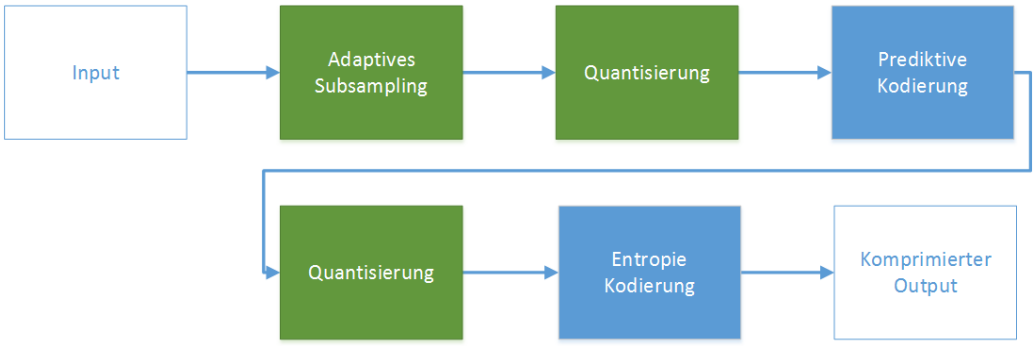

Abbildung 9: Aufbau des Lösungsansatzes: Adaptives Subsampling.

Warum Prediktive Kodierung, was ist das gute: weniger sichtbare Artefakte, sie äussern sich hauptsächlich durch Verschiebungen. Prediktive Kodierung stammt aus der verlustfreien Kompression, wie also Daten gelöscht werden ist eine zentrale Frage. Idee aus state of the art point cloud. Dort wird ein Octree verwendet, welche immer genauer die Daten approximiert.

3.4.1 Angle Subsampling

selbe subsampling wie lösung zuvor. es werden aber mehr punkte übertragen. Grund liegt darin dass das angle subsampling eine gute heuristik darstellt, unwichtige Informationen zu löschen.

3.4.2 Quantisierung

quantisierung in diskretisierten spärischen koordinaten. selbe wie in Lösung 0 aber angepasst?

3.4.3 Wavelet Prediktive Kodierung

Das selbe Prinzip wird nun auf die Kurven der Kanäle übertragen werden. Kurven dar (ref pca der resultate). Im Diagramm der Abbildung 10 ist der erste Schritt des Algorithmus dargestellt. Start und Endpunkt werden ohne Kodierung abgespeichert. Es wird der mittlere Punkt des Kanals kodiert. Es wird angenommen, dass der mittlere Punkt des Signals auf der Linie zwischen Start- und Endpunkt liegt. Algorithmus Schritt 1: Residual Fehler von der Vorhersage zum Mittelpunkt. Nächster Schritt ist nun das ganze rekursiv für die neuen Teilstrecken zu wiederholen. Der nächste Schritt ist im Diagramm 11 dargestellt. wiederholung, bis alles abgelegt ist. Residuals werden immer kleiner

3.4.4 Quantisierung

Eine weitere Quantisierung. darf nicht zu viel passieren, sonst geht alles hops. es wurde deshalb noch eine sehr schwache quantisierung eingeführt.

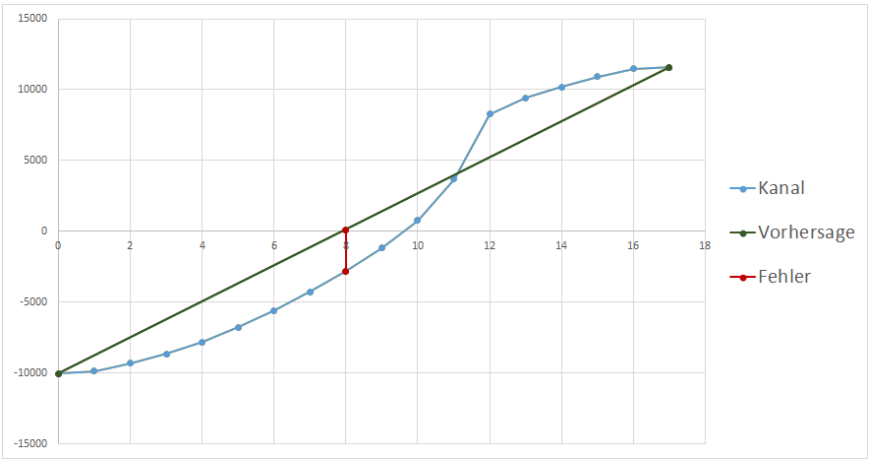

Abbildung 10: Erster Schritt der Wavelet Prediktiven Kodierung. Zu sehen sind das zu kodierende Signal, die Vorhersage und den Fehler der Vorhersage.

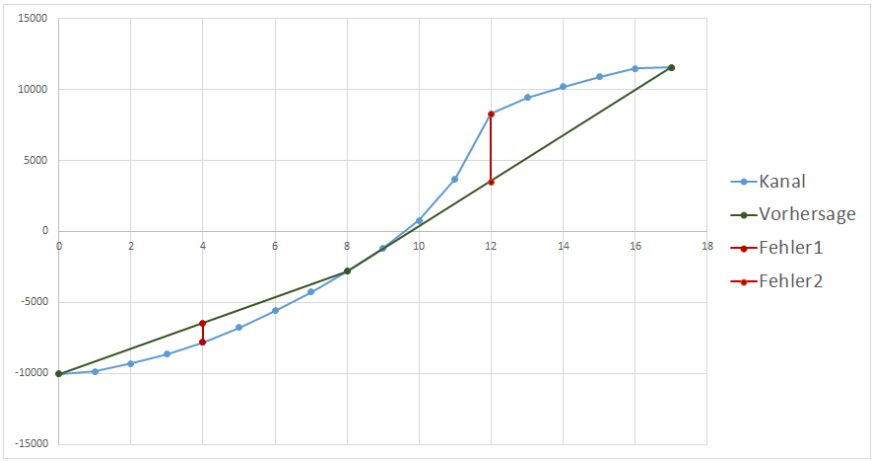

Abbildung 11: Zweiter Schritt der Wavelet Prediktiven Kodierung. Die Vorhersage beinhaltet nun zwei Strecken.

3.4.5 Entropie Kodierung

adaptive Kodierung aus DCT. die Längenkodierung wird nicht verwendet, da sie nicht viel auf 0 quantisieren wird.

4 Qualitätsmessung der dekomprimierten Daten

Bei verlustbehafteten Kompressionen muss die Qualität der dekomprimierten Daaten sichergestellt werden. Im optimalen Fall ähneln die dekomprimierten Daten ihren Originalen, sodass bei einer Visualisierung das menschliche Auge keinen Unterschied erkennen kann.
Im Verlauf der Arbeit wurden zwei Metriken verwendet: Die Standardabweichung und eine angepasste PSNR-HVS-M. Die Standardabweichung bewertet sichtbare Artefakte wie Oszillationen im dekomprimierten Signal nicht genügend stark. Der absolute Fehler bleibt klein, für das menschliche Auge jedoch sind solche Artefakte störend. Ein Beispiel wie solche Artefakte in sich in der Visualisierung bemerkbar machen ist in der Abbildung 25 im Abschnitt 6.2.6 zu sehen. Wie die Standardabweichung berechnet wird, ist im Abschnitt 4.2 beschrieben. Die PSNR-HVS-M stammt aus der Bildverarbeitung. Das Ziel des Fehlermasses ist es, eine hohe Korrelation zwischen der Metrik und dem menschlichen Augenmass zu erreichen. Wie die PSNR-HVS-M Metrik angepasst und umgesetzt wurde, ist im Abschnitt 4.3 beschrieben.
Für die Messungen wurden spezielle Aufnahmen der Feldlinien gewählt. Wie die Aufnahmen ausgewählt wurden, ist im Abschnitt 4.1 beschrieben.

4.1 Auswahl und Erhebung der Testdaten

Die Testdaten sollen zu einem alle Randfälle abdecken, als auch durchschnittliche Fälle enthalten. Aus diesem Grund wurden insgesamt zehn Datensätze ausgewählt: Vier Datensätze mit hoher Sonnenaktivität, zwei mit wenig und vier zufällig. Für die vier Datensätzen mit hoher Aktivität wurde in den Jahren 2014 und 2013 nach den grössten Solare Flares gesucht. Für die Datensätze mit wenig Aktivität wurde das Gegenteil gemacht, nach Zeiträumen mit möglichst kleinen Solar Flares gesucht.
Die Feldlinien werden aber nur alle sechs Stunden berechnet und Solar Flares sind sehr spontane Ereignisse. Auch eine grosse Flare kann während den sechs Stunden angefangen und wieder aufgehört haben. Für die grossen Solar Flares wurde deshalb beachtet, dass die Datensätze vor dem Ereignis verwendet wurden. Grosse Solar Flares entladen das Feld, vor dem Ereignis ist das Magnetfeld komplexer.

Wie im Abschnitt 3.1 beschrieben, wird bereits eine einfache verlustbehaftete Kompression durchgeführt. Für die Testdaten wurde diese entfernt, was die rohe Datenmenge entsprechend anwachsen liess auf etwa 10 MiBytes pro Aufnahme.

4.2 Berechnung der Standardabweichung

Die dekomprimierte Linie ähnelt dem Original, wenn die Abweichung konstant und klein bleiben. Eine seltene, dafür grosse Abweichung kann das Aussehen massgebend verändern. Für diesen Fall wurde Die Standardabweichung ausgewählt. Die originale und dekomprimierte Feldlinie können unterschiedliche Abtastraten aufweisen. Die Standardabweichung muss deshalb unabhängig von der Abtastrate berechnet werden.
Um die Abweichung zu berechnen wird konzeptionell zwischen den dekomprimierten Punkte eine Linie gezogen und den Abstand zwischen dieser Linie und den Originalpunkte berechnet. Der Vorgang ist dargestellt im Diagramm der Abbildung 12. Für jeden Punkt P'_i aus den dekomprimierten Punkten D, nehme P'_i, P'_{i-1},P'_{i+1} und P'_{i+2}. Ziehe drei Strecken, von $P'_{i-1}P'_i$, $P'_iP'_{i+1}$ und $P'_{i+1}P'_{i+2}$. Suche von P'_i den Originalpunkt P_i aus allen Originalpunkten O, berechne den minimalen Abstand zu den drei Strecken. Führe das für alle folgenden Originalpunkte durch, bis P_{i+1} erreicht wurde.

Die Abstandsberechnung von Strecke s zu einem Punkt P erfolgt in zwei Schritten: Zuerst wird mit der Formel (4.1) überprüft , ob eine Senkrechte durch P auf der Strecke s zu liegen kommt. Falls das der Fall ist, wird der Abstand von P zu s berechnet (4.2). Falls nicht, wird die kürzeste Distanz der Eckpunkte der Strecke zu

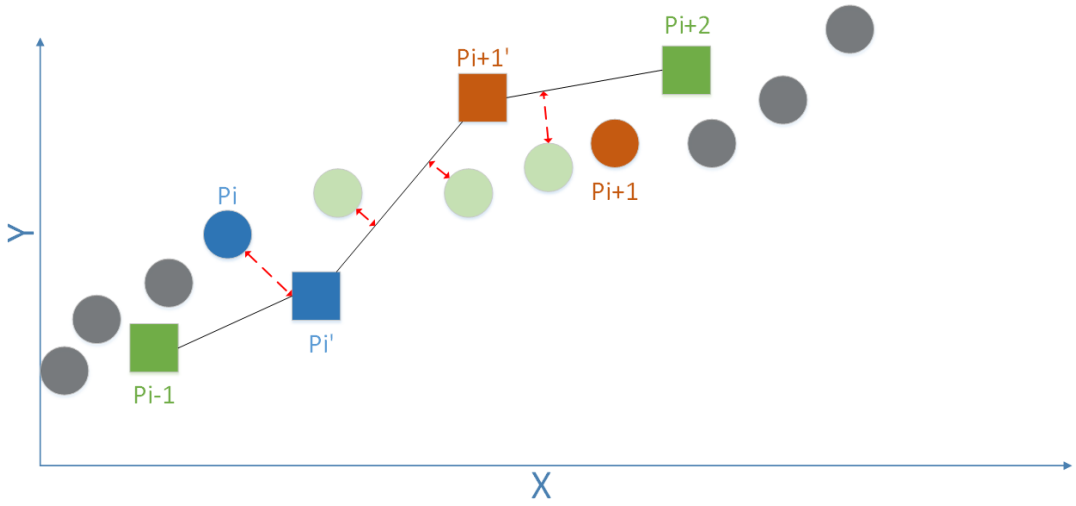

Abbildung 12: Darstellung der Fehlerberechnung. Die Punkte sind die Originaldaten, die Quadrate sind die Punkte nach der Kompression.

P berechnet.

$$t = \frac{\vec{AB} * \vec{AP}}{|\vec{AB}|^2} \quad 0 \leq t \leq 1 \tag{4.1}$$

$$distance = \frac{|\vec{BA} \times \vec{BP}|}{|\vec{BP}|} \tag{4.2}$$

A und B sind die Eckpunkte der Strecke. Falls $0 \leq t \leq 1$, existiert eine Senkrechte durch P mit Fusspunkt auf der Strecke s. Die Distanz von P zu s wird mit der Formel (4.2) berechnet.
Wenn das nicht möglich ist, wird der kürzere Distanz von P zu einem der Eckpunkte genommen.

4.2.1 Berechnung der Standardabweichung

$$\begin{aligned} \sigma(X) &= \sqrt{(variance(X))} \\ variance(X) &= \sum (x_i - E(x_i))^2 \end{aligned} \tag{4.3}$$

Die Standardabweichung σ einer Beobachtungsreihe X $(x_1, x_2, x_3, \ldots, x_n - 1)$ ergibt sich aus der Wurzel der Varianz von X. Die Varianz von X kann errechnet werden, wenn man den Distanz jeder Beobachtung x_i mit dem Erwartungswert $E(x_i)$ berechnet und quadriert. Die Beobachtung ist im diesen Fall ein Punkt der dekomprimierten Linie, während der Erwartungswert der Originalpunkt ist. Die Distanz wird mit dem besprochnen Verfahren 4.2 berechnet. Die Summe der quadratischen Abstände ergibt die Varianz. Die Varianz wird über alle Testdaten berechnet, somit erhält man für einen Test einen Wert für die Standardabweichung.

4.3 Berechnung der angepassten PSNR-HVS-M

Die Peak-Signal-Noise-Ratio (PSNR) Metrik ist ein weitverbreitetes Fehlermass in der Bildverarbeitung. Es wird für die Messung des Fehlers zwischen dekomprimierten Bild und dem Original eingesetzt werden.

$$\begin{aligned} PSNR &= 20 * log_{10}(MAX_I) - 10 * log_{10}(MSE) \\ MSE &= \frac{1}{n}\sum_{i=0}^{N-1}[E(i) - D(i)]^2 \end{aligned} \tag{4.4}$$

Die PSNR (4.4) setzt sich zusammen aus dem maximal möglichen Wert MAX_I und dem "Mean Squared Error´´ (MSE), der durchschnittliche Quadratische Fehler zwischen den Originaldaten $E()$ den dekomprimierten Daten $D()$. Das Problem der Metrik ist, dass sie nicht immer mit der menschlichen Wahrnehmung übereinstimmt. Ponomarenko et al. [13] haben eine modifizierte PSNR entwickelt; die PSNR Human Visual System Masking (HVS-M). In ihren Messungen erreichten sie eine hohe korrelation zwischen menschlicher Wahrnehmung von verrauschten Bildern und dem neuen Fehlermass.

Der Unterschied zwischen der PSNR und der PSNR-HVS-M liegt in der Berechnung des durchschnittlichen quadratischen Fehlers. Das Diagramm der Abbidlung 13 zeigt den Ablauf der neuen Berechnung.
PSNR-HVS-M berechnet die Differenz zwischen dem Originalbild E und dem verrauschtem Bild D und

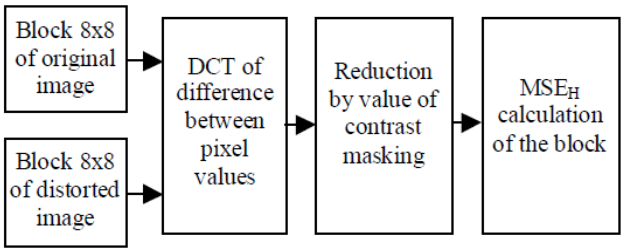

Abbildung 13: Flussdiagramm der PSNR-HVS-M Berechnung [13].

führt die Daten mittels einer DCT in den Frequenzraum. Der nächste Schritt Contrast Maskin reduziert die Differenz, wenn das menschliche Auge den Frequenzunterschied nicht erkennen kann. Die Berechnung des MSE_H Wertes erfolgt wieder gleich wie bei der PSNR.

4.3.1 Contrast Masking

Um das Contrast Masking zu berechnen, führt Ponomarenko et al. die gewichtete Energie der DCT Koeffizienten E_w (4.5)und den masking effect E_m (4.6) ein:

$$E_w(X) = \sum_{i=0}^{7}\sum_{j=0}^{7}[X_{ij}]^2 C_{ij} \tag{4.5}$$

$$E_m(X) = \frac{1}{f_m}E_w(X) \tag{4.6}$$

Wobei X die Kosinus-Koeffizienten eines Bildblocks sind und C die Gewichtungen der Frequenzen. Der Normalisierungsfaktor f_m wurde experimentell ermittelt und auf 16 festgelegt. Ponomarenko et al. argumentiert, dass der Unterschied zwischen einem Block X_e und einem verrauschten Block X_d unsichtbar sind, wenn die Formel (4.7) erfüllt ist.

$$E_w(X_e - X_d) < max[E_m(X_e), E_m(X_d)] \tag{4.7}$$

Das Contrast Masking fliesst mit folgender Formel in die Distanzberechnung mit ein (4.8).

$$X_{\Delta ij} = \begin{cases} X_{eij} - X_{dij}, & i = 0, j = 0 \\ 0, & \left|X_{eij} - X_{dij}\right| \le E_{norm} / C_{ij} \\ X_{eij} - X_{dij} - E_{norm} / C_{ij}, & X_{eij} - X_{dij} > E_{norm} / C_{ij} \\ X_{eij} - X_{dij} + E_{norm} / C_{ij}, & \text{otherwise} \end{cases} \tag{4.8}$$

Wobei $E_{norm} = \sqrt{max[E_m(X_e), E_m(X_d)]/64}$. X_{eij} ist der DCT Koeffizient des Originalblockes und X_{dij} der Koeffizient des verrauschten Blockes. Dabei ist es irrelevant ob die DCT Koeffizienten subtrahiert werden oder wie in der Abbildung 13 angedeutet, zuerst die Differenz der Pixelwerte berechnet und diese DC transformiert. Aus den Distanzen $X_{\Delta ij}$ wird die PSNR (4.4) berechnet.

Wie gut die PSNR-HVS-M Metrik mit dem menschlichen Auge übereinstimmt, hängt vom Normalisierungsfaktor f_m und von der Wahl der Gewichtungen C ab. Ponomarenko et al. verwendeten die normalisierten ($\frac{10}{x}$) und quadrierten Werte der JPEG/JFIF Quantisierungsmatrix [1]. Es ist zu beachten, dass der DC-Koeffizient nicht im Contrast Masking berücksichtigt wird, für den Wert wird die normale PSNR berechnet. Der DC-Koeffizient stellt die durchschnittliche Helligkeit in einem Block dar. Das menschliche Auge kann auch kleine Unterschiede in dieser Frequenz erkennen.

4.3.2 Umsetzung und Anpassung der PSNR-HVS-M für diese Arbeit

Der grösste Unterschied zur Arbeit von Ponomarenko et. al. ist der Einbezug des DC-Koeffizienten ins Contrast-Masking. Der DC-Koeffizient repräsentiert bei den Feldlinien eine Verschiebung. Das menschliche Auge kann leichte Verschiebungen der Feldlinien kaum unterscheiden. Des weiteren musste für die Feldlinie eine eigene Quantisierungsmatrix gefunden werden, ausdenen die Gewichtungen berechnet werden. Es wurde eine DCT-Kompression entwickelt, welche keine sichtbaren Artefakte aufweist. Die Quantisierungsmatrix wurde übernommen und auf die selbe Art normalisiert.
Für die PSNR wird der maximale Wert MAX_I benötigt. Hier wurde der maximal mögliche Wert der PFSS Simulation verwendet, den vierfachen Sonnenradius.

Der letzte Wert, den es zu setzen gibt, ist der Normalisierungsfaktor f_m. Ponomarenko et. al. schrieb keine zusätzliche Information oder Begründung zu diesem Wert. Dieser Wert stellt ein, wie stark das Contrast-Masking einfliessen soll. Je höher der Wert ist, desto ähnlicher ist die PSNR-HVS-M Metrik der normalen PSNR.
Die Tabelle 5 erforscht den Einfluss des f_m Faktors. Dazu wurde die PSNR-HVS-M zu drei dekomprimierten Simulationen berechnet, welche eine unterschiedliche Qualität aufweisen. Für diesen Anwendungsfall scheint

Qualität	f_m 8	f_m 16	f_m 32
Keine sichtbare Artefakte	96.8 dB	**95.6** dB	94.5 dB
Kaum sichtbare Artefakte	95.8 dB	**94.4** dB	93.3 dB
Sichtbare Artefakte	90.0 dB	**88.3** dB	87,0 dB

Tabelle 5: Einfluss des f_m Faktors auf die PSNR-HVS-M.

der Faktor f_m sich stabil zu verhalten: der Faktor kann verdoppelt oder halbiert werden und die resultierenden Distanzen bleiben in der selben Grössenordnung. Es wurde der Standardwert von 16 übernommen. Eine Anpassung von f_m scheint nicht die Metrik massgebend zu verändern und hat vermutlich weniger Einfluss als die Gewichtung der DCT Koeffizienten.

Die PSNR-HVS-M ist nicht unabhängig von der Abtastrate. Es erwartet, dass die zu vergleichenden Datenmengen gleich viele Punkte enthalten. Für diese Arbeit werden die Originaldaten auf die selbe Punktmenge reduziert. Wenn pro Feldlinie genau ein Punkt exakt abgespeichert wird, würde die PSNR-HVS-M trotzdem eine hohe Ähnlichkeit zwischen Original und dekomprimierten Feldlinien ergeben. Dies ist Vertretbar, da die Standardabweichung diesen Fall abdeckt und eine hohe Distanz ausgeben wird. Die PSNR-HVS-M soll hauptsächlich Artefakte aufdecken, welche in der Standardabweichung nicht ins Gewicht fallen wie die Ringing Artefakte vom Abschnitt 6.2.6.

5 Implementation

Der JHelioviewer lädt eine Folge von komprimierten Simulationen über eine Internetverbindung und muss diese vor der Visualisierung dekomprimieren. Das Herunterladen und die Dekomprimierung sind zeitaufwändige Operationen und kann zu Wartezeiten beim Benutzer führen. Um die Operationen zu beschleunigen werden im Ist-Zustand die Simulationen im Voraus asynchron heruntergeladen. Somit sind die Daten bereits im Arbeitsspeicher, bevor die Daten dekomprimiert und visualisert werden.
Die Dekompression wird direkt vor der Visualisierung durchgeführt. Bei einer Animation der Feldliniendaten führt das zu einer bemerkbaren Verzögerung bei jedem Wechsel. Um die Qualität der Animation zu verbessern und die Wartezeit weiter zu verkürzen wurden folgende Massnahmen umgesetzt:

1. Asynchrone Dekompression.
2. Vorladen der Dekomprimierten Feldlinien.
3. Caching von Komprimierten und Dekomprimierten Feldlinien.

5.1 Software Architektur

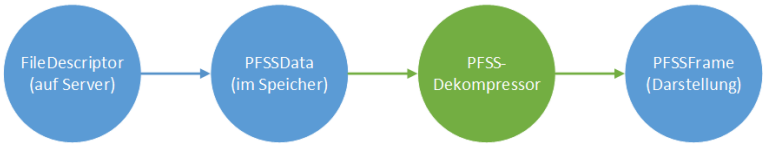

Abbildung 14: Zustandsdiagramm der Feldliniendaten

Die Daten der Feldlinien durchlaufen im JHelioviewer vier Zustände, welche durch vier Klassen abgebildet wurden. Die Klassen sowie die Zustandswechsel sind im Diagramm der Abbildung 14 dargestellt. Die Klasse ”FileDescriptor´´ repräsentiert eine Simulation von Feldlinien auf dem Server. In diesem Zustand sind die Daten bereit für das Herunterladen. Die folgende Klasse ”PFSSData´´ symbolisiert Feldlinien, welche in den lokalen Arbeitsspeicher geladen wurden. In diesem Zustand sind die Daten noch komprimiert und nicht bereit für eine Visualisierung. Die Klasse ”PFSSDekompressor´´ ist ein Zwischenzustand und stellt den Wechsel von komprimierten zu unkomprimierten Daten dar. Da der Zustandswechsel aufwändig ist, wird es durch eine eigene Klasse abgebildet. Die letzte Klasse ”PFSSFrame´´ repräsentiert die dekomprimierten Feldlinien. In diesem Zustand sind die Daten bereit für die Darstellung. Die Darstellung wird ebenfalls von der ”PFSSFrame´´ Klasse übernommen.

5.1.1 Vorladen und Caching von Komprimierten und Dekomprimierten Feldlinien

Um die Animation der Feldlinien möglichst Unterbrechungsfrei zu gestalten, werden die komprimierten und dekomprimierten Feldliniendaten vorgeladen und gecached. Mit dem Vorladen wird erreicht, dass der Wechsel von der Visualisierung einer Simulation zur anderen möglichst ohne Unterbrechung durchgeführt werden kann. Das Caching hilft, wenn der Benutzer einen anderen Zeitpunkt der Animation nochmals darstellen möchte. Aus dem Zustandsdiagramm der Abbildung 14 kann entnommen werden, dass die Komprimierten Feldlinien von den ”PFSSData” Objekten und die dekomprimierten Daten von den ”PFSSFrame” Objekten repräsentiert werden. Die Implementation ist im Diagramm der Abbildung 15 dargestellt. Die Klasse FrameManager ist zuständig für das Vorladen der dekomprimierten Daten, der PFSSFrame Objekte. Das Caching wird in der FrameCache Klasse umgesetzt. Die Klasse DataCache ist für das Vorladen und Caching der komprimierten Daten zuständig. Die FileDescriptor Objekte repräsentieren eine Feldliniensimulation auf dem Server, der FileDescriptorManager ist zuständig für das Auffinden der Simulationen.

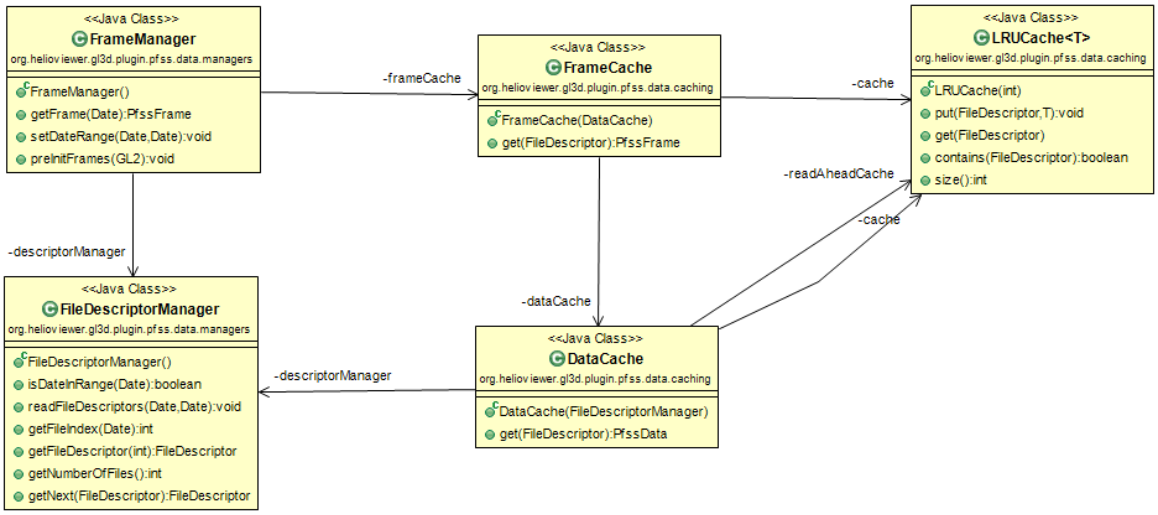

Abbildung 15: Diagramm der Implementation vom Vorladen und Caching

Die Klasse FrameManager repräsentiert die Facade der Vorladens- und Caching- Implementation. Sie abstrahiert das Zusammenspiel der verschiedenen Caches und den verschiedenen Zustände der Feldliniendaten und bietet eine vereinfachte Schnittstelle an.
Die Vorladen und Caching Implementation der PfssFrame Objekte wurde in zwei Klassen aufgeteilt: Die PfssFrame Objekte müssen vor der Visualisierung Resourcen der Grafikkarte allozieren. Nachdem ein Objekt visualisiert wurde, müssen die Resourcen freigegeben werden. Der FrameManager ist zuständig die Allozierung anzustossen und die Freigabe sicherzustellen.
Die PFSSData Objekte können vom Garbage-Collector verwaltet werden. Das Vorladen kann mit einer weiteren Instanz des LRU-Caches umgesetzt werden und wurde direkt in der DataCache Klasse implementiert.

In dieser Arbeit wurde ein Least-Recently-Used (LRU) Cache Algorithmus verwendet. Der LRU-Cache löscht das am längsten nicht verwendete Objekt, wenn der Cache gefüllt ist. Da der JHelioviewer im allgemeinen Fall sequenzell Objekte verlangt, kann der LRU Cache mit einer First-in-First-Out Queue implementiert werden. Das Objet, welches am längsten nicht verwendet wurde, ist das Letzte in der Queue. Ein LRU-Cache funktioniert in diesem Anwendungsfall optimal, wenn die Anzahl Objekte grösser ist, als der Cache. In einem Spezialfall ist der LRU-Algorithmus nicht optimal. Wenn der JHelioviewer zur letzten Simulation der Feldlinien angekommen ist, wird an Wrap-around durchgeführt und wieder die erste Simulation verlangt. Wenn der Cache $n-1$ von n Simulation abspeichern kann, so löscht der LRU-Algorithmus immer die Simulation, welches als übernächstes abgefragt wird. Das führt dazu, dass gleich viele Cache-Misses geschehen, als wenn der Cache wesentlich kleiner währe.

5.1.2 Asynchrone Aufrufe mittels Executor Services

Im Diagramm der Abbildung 15 zu sehen ist, wird das Erstellen von PFSSData und PFSSFrame Objekten jeweils von zwei Klassen übernommen werden, den Creators. Sie sind zuständig für das asynchrone Herunterladen und Dekomprimieren der Feldliniendaten. Die asynchrone Ausführung ist mit dem Java Executor Service umgesetzt. Der Executor Service verwaltet und begrenzt die Anzahl an Threads welche die Aufrufe bearbeiten, sodass auch bei hoher Auslastung ein möglichst hoher Durchsatz erreicht wird. Im Ist-Zustand wurden alle asynchrone Aufrufe jeweils in einem eigenen Thread ausgeführt. Bei hoher Auslastung steigt der Verwaltungsaufwand der Threads und bremst das System.

6 Resultate

In diesem Abschnitt werden die Ergebnisse der Tests vorgestellt. Aufgrund dieser Resultate wurden die Lösungsansätze vom Abschnitt 3 entwickelt. Für die PSNR-HVS-M Metrik existiert nicht immer ein Diagramm. Die Metrik wurde im Laufe der Arbeit entwickelt und wird ab Abschnitt 6.2.7 ebenfalls gemessen.

6.1 Lösungsansatz: Adaptives Subsampling

Im Ist-Zustand führt der JHelioviewer nach der Dekompression ein adaptives Subsampling durch. Dieser Lösungsansatz führt das adaptive Subsampling vor der Datenübertragung durch und Kodiert die Daten mit Rar anstatt mit Gzip. Eine genauere Beschreibung des Ansatzes ist im Abschnitt 3.2 zu finden. Wie im Dia-

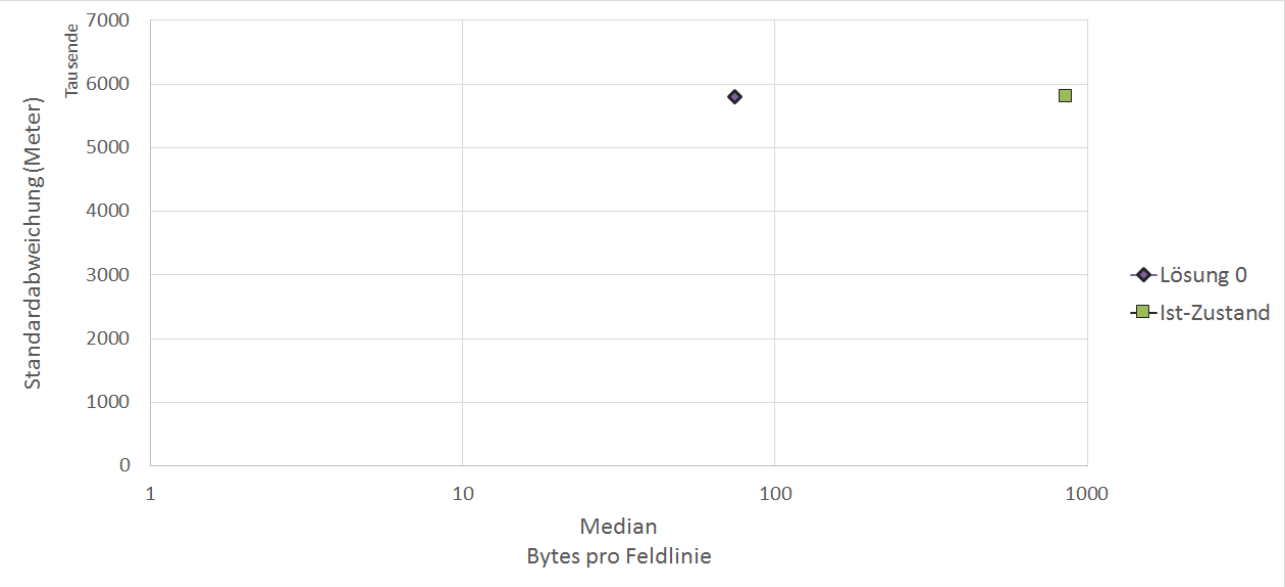

Abbildung 16: Vergleich des Lösungsansatzes: Adaptives Subsampling zur Ist-Kompression.

gramm 16 erkennbar ist, braucht dieser Lösungsansatz deutlich weniger Speicher als die Ist-Kompression. Das Adaptive Subsampling reduziert deutlich die Anzahl Punkte, während die Rar eine bessere Kompression erbringt. Dieser Ansatz kann die Daten um Faktor 11.6 besser komprimieren.
Die Abbildung 17 zeigt die Artefakte, die bei der Komprimierung der Lösung 0 entstehen. Es ist anzumerken, dass der Ist-Zustand die selben Artefakte aufweist.

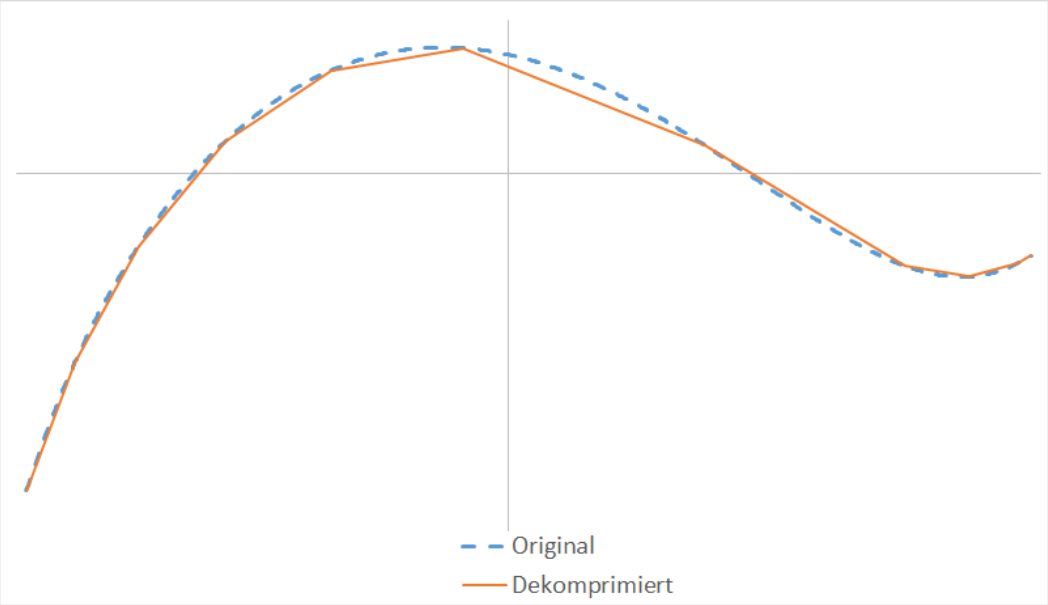

Abbildung 17: Artefakte der Lösung 0

6.2 Lösungansatz: Diskrete Kosinus Transformation

In diesem Abschnitt wird der Lösungsansatz mittels Diskreter Kosinus Transformation behandelt. Es wurden verschiedene Transformationen getestet, welche eine Approximation mittels Kosinus Funktionen vereinfachen könnten.
Um die Feldlinien optimal mit der DCT zu approximieren, müssen Ringing Artefakte, behandelt werden. Das Auftreten der Artefakte wird im Abschnitt 6.2.6 und die Behandlung im Abschnitt 6.2.7 besprochen .

Für alle Tests wurde eine lineare Quantisierung verwendet. Jeder DCT Koeffizient wird durch einen Faktor geteilt, der sich stetig ehöht. Zum Beispiel wird der erste Koeffizient durch zwei geteilt, der zweite durch Vier, der Dritte durch Sechse etc. Die Kompressionsrate kann durch einen höheren oder tieferen Faktor gesteuert werden. Diese Quantifizierung ist nicht das Optimum. Eine bessere Quantifizierung wird für die beste Variante ausgearbeitet. Wie die beste Variante im Detail umgesetzt wurde, wird im Abschnitt 3.2.2 behandelt.

6.2.1 Variante: DCT

Diese Variante verwendet einzig die Diskrete Kosinus Transformation und der linearen Quantisierung. Die Abbildung 18 zeigt den Vergleich der DCT Kompression mit dem Lösungsansatz des Adaptiven Subsamplings (siehe 6.1). Es ist deutlich zu erkennen, dass die Standardabweichung schnell steigt bei leicht sinkender Grösse. Der Grund dafür kann im Diagramm der Abbildung 19 entnommen werden. Un den meisten Fällen kann die DCT die Feldlinie gut approximieren. Bei dieser Feldlinie wird der Anfang der Kurve nicht richtig dargestellt. Das ist ein typisches Problem der DCT. Die Diskrete Kosinus Transformation nimmt an, dass das Signal sich periodisch wiederholt. Die implemementierte Transformation (siehe Abschnitt 3.3.3) nimmt an, dass am Anfang und am Ende das Signal in umgekehrter Reihenfolge wiederholt. Bei den Feldlinien kann das zu einer Diskontinuität im Signal führen, welches hochfrequente Kosinus Anteile. Wenn die Quantisierung Hochfrequente Schwingungen verschluckt, entstehen die Artefakte der Abbildung 19.
Eine Möglichkeit ist die Feldlinie um Punkte zu erweitern. Wenn die Feldlinie am Anfang und am Ende abflacht, sollte die resultierende Transformation weniger hochfrequente Schwingungen enthalten. Diese Variante wird im Abschnitt 6.2.5 behandelt und führt zu einer deutlich besseren Approximation. Durch eine andere Darstel-

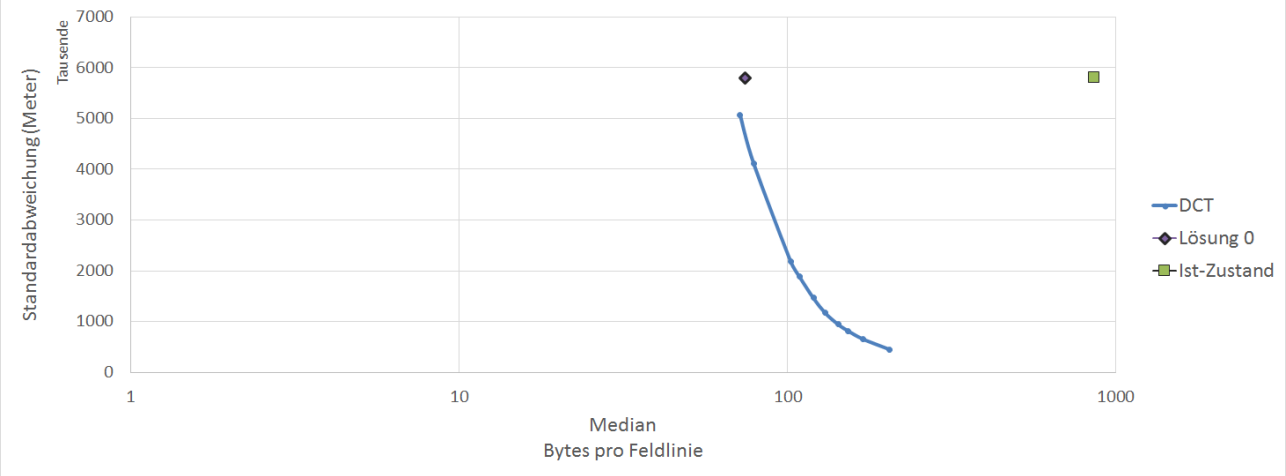

Abbildung 18: Vergleich der DCT Kompression mit der Lösung0

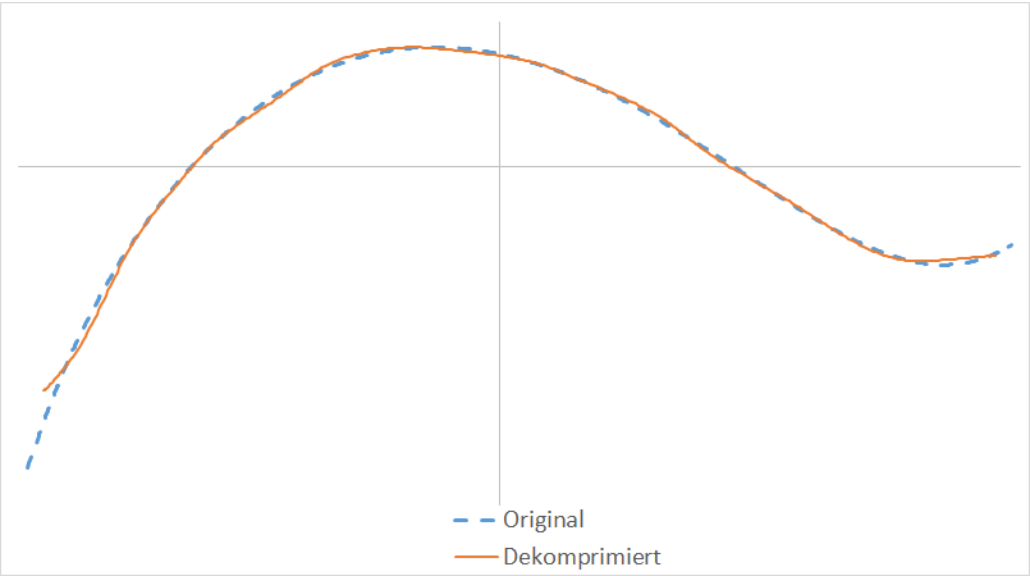

Abbildung 19: Artefakte der DCT Dekompression anhand Beispieldaten

lung der Daten kann das Problem ebenfalls gelöst werden.

6.2.2 Variante: Ableitung+DCT

Vor der DCT werden die Feldlinien abgeleitet. Mit der Ableitung soll das Randproblem dargestellt in 19 gelöst werden. Der Nachteil ist, dass Ungenauigkeiten sich durch die Kurve durchziehen und summieren. Am Ende kann die Approximation ungenauer sen, wie am Anfang der Feldlinie.
Das Diagramm der Abbildung 20 zeigt, dass die abgeleiteten Feldlinien besser approximiert werden können

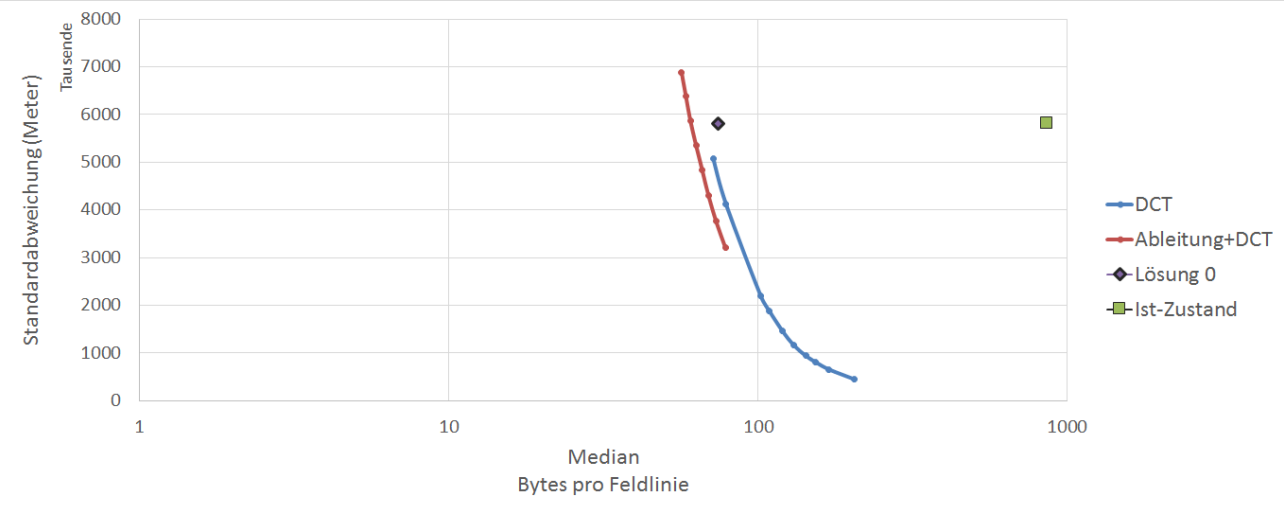

Abbildung 20: Vergleich der DCT Kompression der Ableitung mit der DCT Kompression

und erreichen dadurch eine Kompressionsrate von 14.3 mit einer vergleichbaren Genauigkeit. Das Randproblem wurde ebenfalls verbessert. Eine Darstellung der Artefakte ist im Diagramm der Abbildung21 zu finden.
Es ist deutlich zu sehen, dass die Kurve durch die Quantisierung gedämpft wird. das Maximum der Kur-

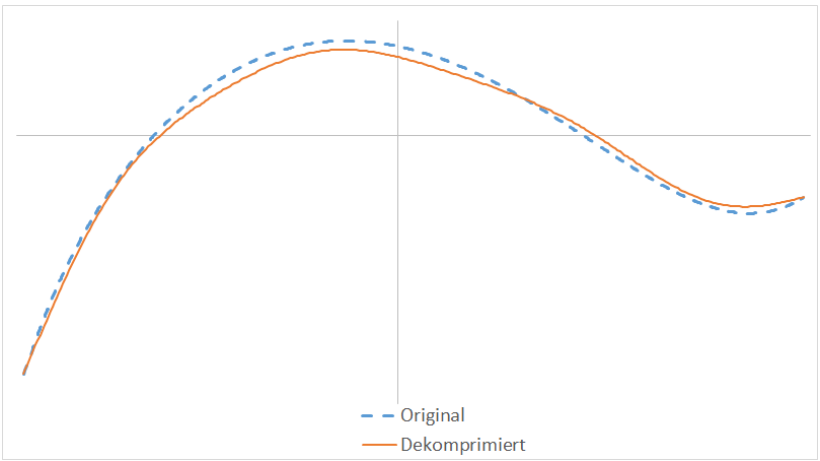

Abbildung 21: Artefakte der DCT Kompression der Ableitung

ve ist tiefer, sowie das lokale Minima der letzten Halbwelle höher. Der Vorteil dieser Variante ist, dass die resultierende Feldline sehr glatt verläuft. Ohne die Originalkurve währen die Artefakte nicht zu identifizieren.

6.2.3 Variante: PCA+Ableitung+DCT

Die Feldlinien liegen meist auf einer Ebene im dreidimensionalen Raum. Wenn die X,Y und Z Kanäle Kosinus-Transformiert werden, ist die Information etwa gleichmässig auf den Kanälen verteilt. Eine Linie könnte sich durch weniger Kosinus-Funktionen approximieren lassen, wenn die Linie zuerst in ein lokales Koordinatensystem transformiert wird.
Die Principal Component Analysis (PCA)[14] ist ein Verfahren aus der Statistik, welches Daten in ein neues koordinatensystem Transformiert. Dabei werden die Achsen so gelegt, dass die Daten entlang der ersten Achse die grösste Varianz aufweisen, entlang der zweiten Achse, welche orthogonal zur ersten liegt, die zweithöchste Varianz etc. Wenn das Vefahren auf die Feldlinien angewandt wird, werden die Feldlinien in ein lokales System transformiert indem der Z-Kanal 0 ist, wenn die Feldlinie in einer Ebene liegt..
Um die PCA wieder rückgängig zu machen, müssen pro Feldlinie sechs Parameter für die neuen Koordinatenachsen und 3 Parameter für die Verschiebung abgespeichert werden. Die PCA scheint die Kompression nicht

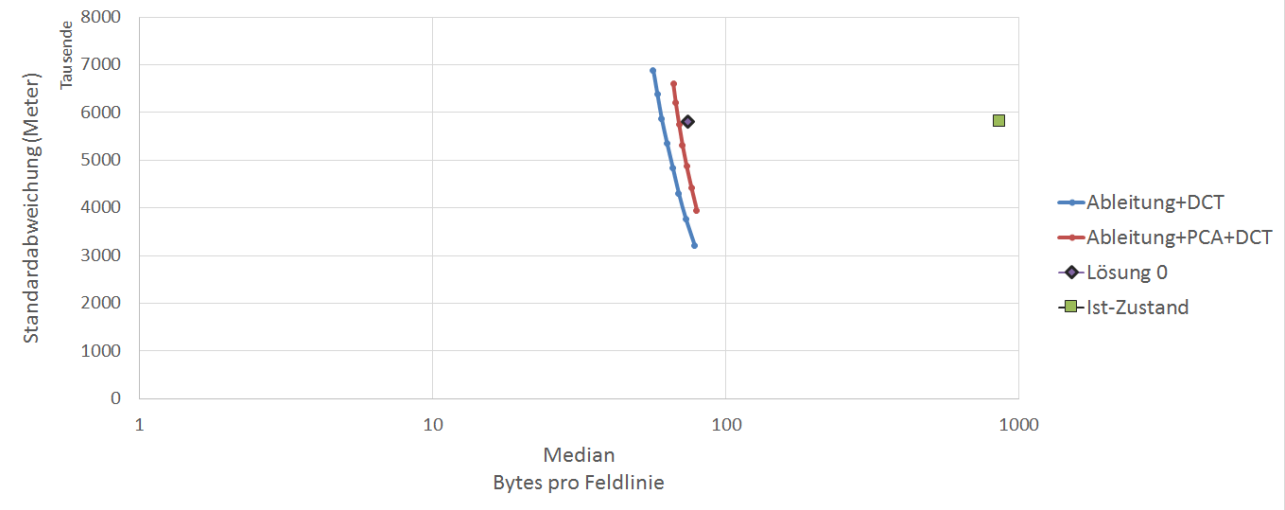

Abbildung 22: Vergleich der PCA DCT Kompression der Ableitung mit der DCT Kompression der Ableitung

verbessern zu können. Der Grund ist, dass die Variante ohne PCA zwischen 5 und 20 Kosinus-Funktionen braucht, um einen Kanal einer Feldlinie zu approximieren. Durch die PCA können 2-5 Parameter gespart werden, jedoch verbrauchen die zusätzlichen Koordinatenachsen und Verschiebungen mehr Speicher, als die Transformation gewinnt.

6.2.4 Variante: Ableitung+DCT+Byte Kodierung

Ein Kanal einer Feldlinie kann mit 5-20 DCT-Koeffizienten ausreichend approximiert werden. Die quantisieren Koeffizienten kommen meistens zwischen -50 und +50 zu liegen. 8 Bit Genauigkeit reichen im Allgemeinen aus, um einen quantisierten Koeffizienten abzuspeichern. Um die Kompressionsrate zu verbessern, werden zwei Byte-Kodierungen eingeführt: die Längenkodierung und die Byte-Kodierung (beschrieben im Abschnitt 3.3.5).
Das Diagramm der Abbildung 23 zeigt den Einfluss der Byte-Kodierung auf die Kompressionsrate. Es bewirkt eine deutliche Verbesserung, ohne die Qualität der Kompression negativ zu beeinflussen. Bei einer Vergleichbaren Genauigkeit wie die Ist-Lösung weist diese Variante eine Kompressionsrate von 24.5 auf.

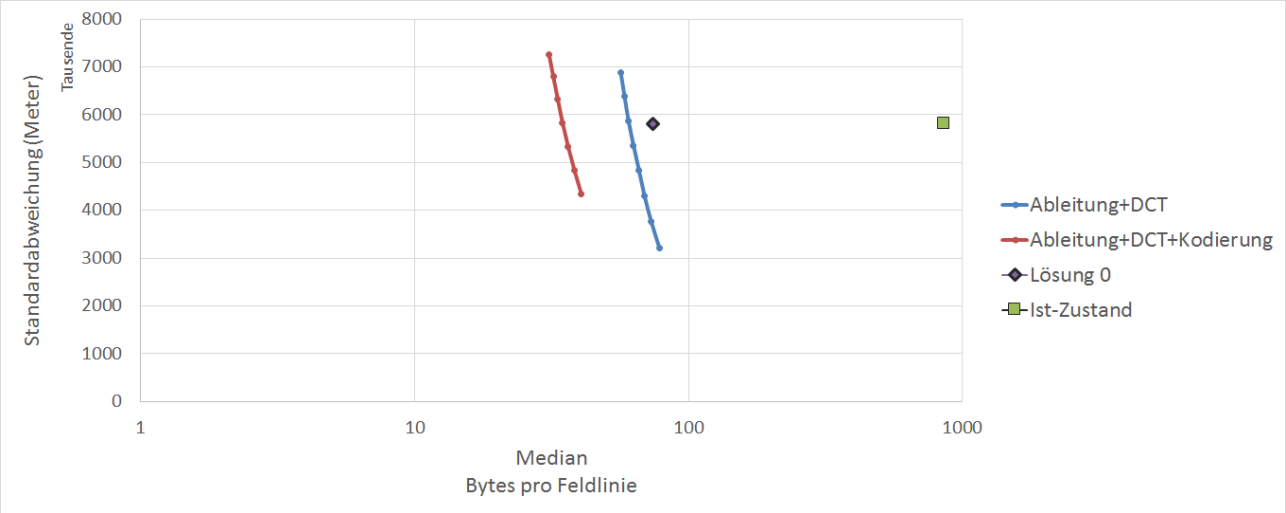

Abbildung 23: Vergleich der Kompression mit und ohne Byte-Kodierung

6.2.5 Variante: Randbehandlung+DCT+Byte Kodierung

Wenn die Artefakte 21 und 17 vergleicht, fällt auf, dass die Variante 6.2.1 die Feldlinie genauer approximiert, falls die Ränder der Feldlinie besser dargestellt werden könnte. Wenn die Feldlinie an den Rändern mit Punkten erweitert wird, welche die Transformation vereinfachen, könnte eine bessere Kompression erreicht werden.
Jeder Kanal einer Feldlinie wird so erweitert, dass der Anfang und das Ende abflacht. Die Byte-Kodierung wurde aus der vorhergehenden Variante übernommen. Das Diagramm der Abbildung 24 zeigt den Vergleich

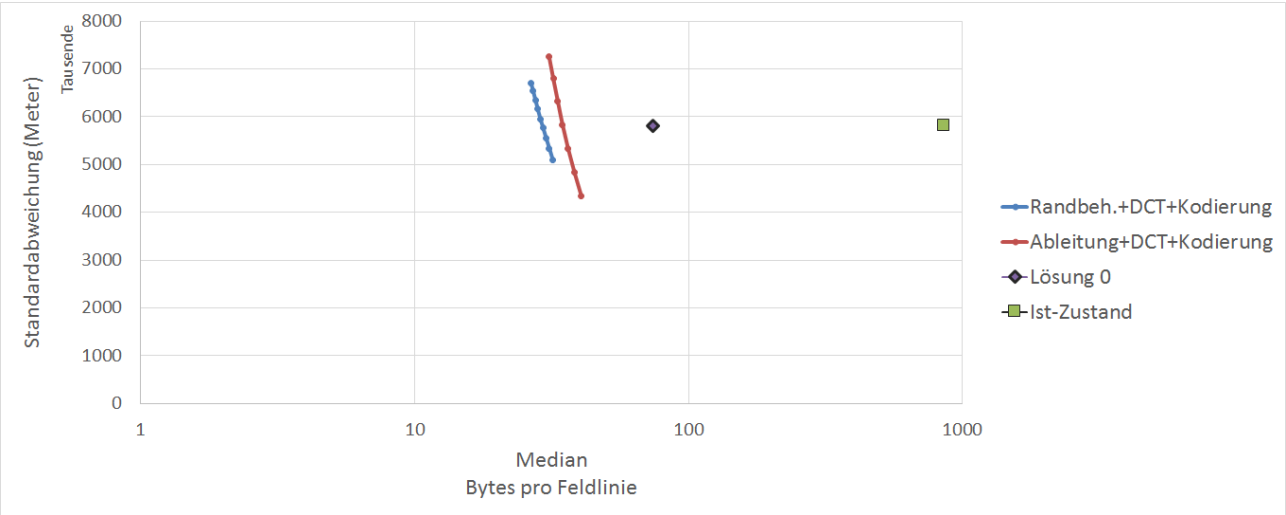

Abbildung 24: Vergleich des Einflusses der Randbehandlung

der Variante mit Randbehandlung und der Variante der abgeleiteten Feldlinie (beschrieben im Abschnitt 6.2.4. Es ist zu erkennen, dass dank der Randbehandlung die Feldlinien mit weniger Bytes ähnlich genau approximiert werden können.
Diese Variante führt aber Artefakte ein, welche die Standardabweichung nicht erkennen kann. Im Folgenden Abschnitt 6.2.6 werden diese besprochen.

6.2.6 Ringing Artefakte

Obwhol die Variante **??** eine vergleichbare Genauigkeit aufweist, wie die Ist-Lösung, sind auf der JHelioviewer Visualisierung deutliche Artefakte zu sehen. Die Abbildung 25 vergleicht die originalen mit dekomprimierten Feldlinien. Die Artefakte zeigen sich als Oszillationen. In diesem Fall scheint die Standardabweichung als

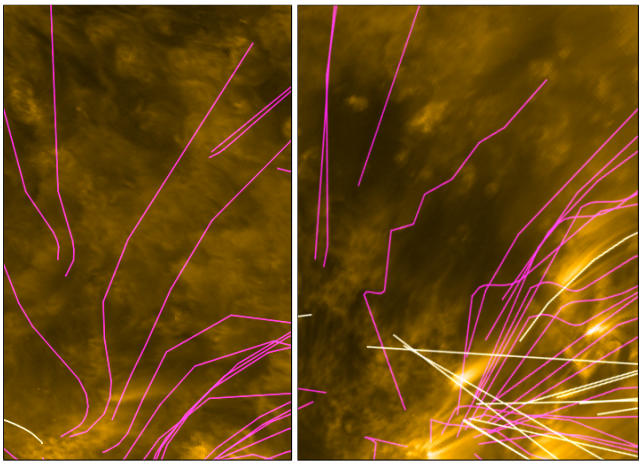

Abbildung 25: Artefakte der Kompression, links sind die originalen Feldlinien, rechts die Dekomprimierten.

Fehlermass zu versagen: Da die Oszillationen nahe an der originalen Feldlinie liegen, bleiben die Abstände klein. Jedoch sind die Artefakte für das menschliche Auge inakzeptabel.
Interessant ist, dass die die Variante der Ableitung (Abschnitt 6.2.4) ähnliche Artefakte aufweist. Im Diagramm der Abbildung 21, welches die Artefakte anhand einer Beispiellinie zeigt, sind keine Oszillationen zu entdecken. In der JHelioviewer Visualisierung jedoch, sind auch bei dieser Variante deutliche Oszillationen zu erkennen. Die Abbildung 26 zeigt die Artefakte. Es ist anzumerken, dass die Artefakte weniger ausgeprägt sind, aber dennoch störend für das menschliche Auge.
Die oszillierenden Artefakte sind typisch für eine DCT-Kompression und sind im Allgemeinen als Ringing

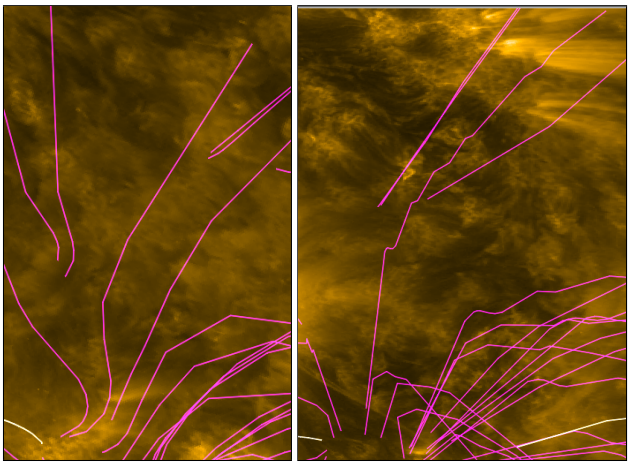

Abbildung 26: Artefakte der Kompression, links sind die originalen Feldlinien, rechts die Dekomprimierten der Variante 6.2.4.

Artefakte [9] bekannt: Abrupte Steigungen im Inputsignal werden in der DCT durch hochfrequente Anteile dargestellt. Durch die Quantisierung der hochfrequenten Anteile werden oszillierende Artefakte eingefügt. Die Ringing Artefakte typische Kompressionsartefakte von DCT-basierten Verfahren wie JPEG/JFIF oder MP3.
Die Feldlinien, welche am stärksten von den Artefakten betroffen sind, sind die ”Weltall zur Sonne” oder ”Sonne ins Weltall” Feldlinien. Sie verhalten sich nicht wie harmonische Halbwellen sondern steigen oft monoton,

mit teils abrupten Richtungswechseln in der Nähe der Sonnenoberfläche. Die Abbildung 27 zeigt ein Beispiel solcher Feldlinien. Die abrupten Wechsel führen zu abrupten Steigungen in den einzelnen Kanälen.

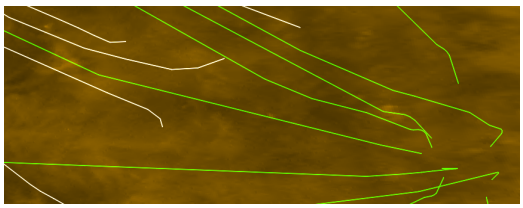

Abbildung 27: Abrupte Steigungen bei Feldlinien, welche von der Sonne ins Weltall führen.

6.2.7 Behandlung der Ringing Artefakte

Um eine optimale Kompression der Feldlinien zu erreichen, müssen die Ringing Artefakte behandelt werden. Abschnitt 6.2.6 wurde erwähnt, dass nicht alle Varianten gleich starke Artefakte verursachen. Es wurde ebenfalls besprochen, dass die Feldlinien, welche von der Sonnenoberfläche zur Oberfläche führen, kaum von den Artefakten betroffen sind. In diesem Abschnitt wird deshalb erforscht, welche Variante die ”Sonne zu Sonne” und welche die ”Weltall zur Sonne” oder ”Sonne ins Weltall” Feldlinien optimal approximieren kann. Für die Messung der Artefakte wird die PSNR-HVS-M Metrik aus Abschnitt 4.3 verwendet.
Für den Test werden insgesamt vier Varianten verglichen. Die Varianten 6.2.4 und und 6.2.5 jeweils mit und ohne PCA Transformation. Die Transformation wird hier nochmals geprüft. Die Transformation könnte die Rining Artefakte jeweils auf einen Kanal beschränken.
Das Diagramm der Abbildung 28 zeigt, wie gut die Varianten die Feldlinien approximieren können, welche

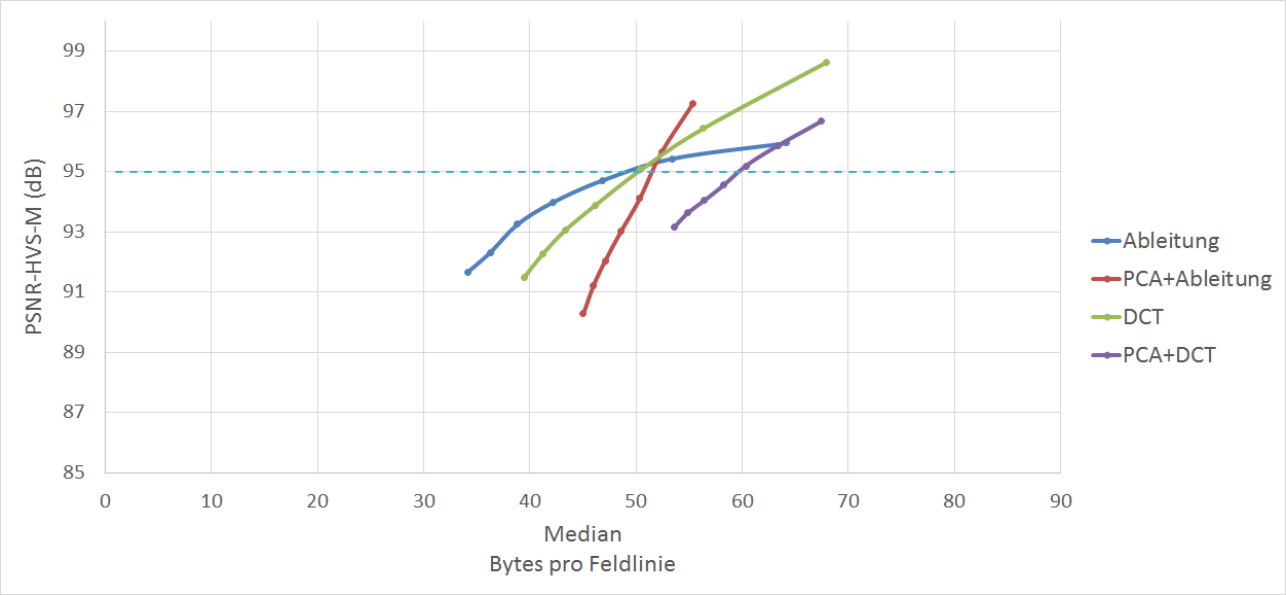

Abbildung 28: Approximation der Feldlinien ”Sonnenoberfläche zu Sonnenoberfläche”. Je höher die PSNR-HVS-M, desto besser ist die Approximation.

von der Sonnenoberfläche zur Sonnenoberfläche führen. Es ist anzumerken, dass ein PSNR-HVS-M Wert von etwa $95dB$ bedeutet, dass die Feldlinien kaum sichtbare Artefakte enthalten. Je höher der PSNR-HVS-M Wert, desto genauer ist die Approximation. Interessant ist, dass drei Varianten ähnlich viel Speicherplatz benötigen, für eine beinahe artefaktfreie Approximation. Bei stärkerer Quantisierung treten beiden abgeleiteten Feldlinien deutlich weniger Artefakte auf. Dies deckt sich mit den Beobachtung aus dem Abschnitt 6.2.6.

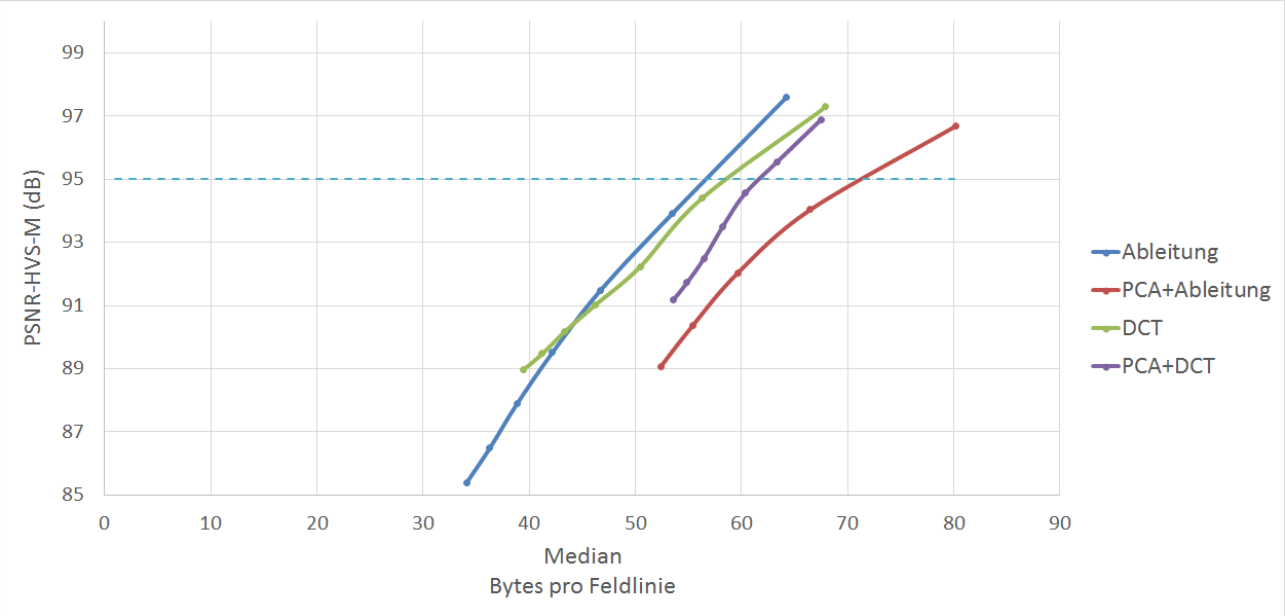

Abbildung 29: Approximation der der Feldlinien ”Sonnenoberfläche ins Weltall” oder ”Weltall zur Sonnenoberfläche”. Je höher die PSNR-HVS-M, desto besser ist die Approximation.

Ebenfalls interessant ist die Variante der PCA Transformation zusammen mit der Ableitung: Diese benötigt für eine beinahe artefaktfreie Approximation am wenigsten Speicherplatz, führt aber bei stärkerer Quantisierung viele Artefakte ein.

Das Diagramm der Abbildung 29 zeigt, wie gut die Varianten die die andern Typen von Feldlinien approximieren können. Hier ist zu sehen, dass die abgeleiteten Feldlinien weniger Artefakte hinzufügen, jedoch weniger deutlich als in der Abbildung 28. Interessant ist, dass die PCA keinen messbaren Vorteil erbrachte. Die zusätzlichen Parameter, die für die Umkehrung der PCA abgespeichert werden, verbrauchen mehr Speicherplatz als man durch die Transformation gewinnt.
Die DCT Variante aus Abschnitt 6.2.5 fällt ab einer PSNR-HVS-M von $90dB$ weniger schnell ab, als die abgeleiteten Feldlinien. Bei diesem PSNR-HVS-M Wert sind aber die Artefakte bereits zu deutlich und nicht mehr akzeptabel. Aufgrund dieser Daten wurde die Variante unter Abschnitt 6.2.4 ausgewählt.

6.2.8 Abschliessende Variante

Für den abschliessenden Test wurde die Variante unter Abschnitt 6.2.4 ausgewählt. Diese verursacht weniger starke Ringing Artefakte als die anderen Varianten. Dies liegt an der Artefaktdämpfenden Eigenschaft der Ableitung und an der auf die auf den Typ angepasste Quantisierung. Das Diagramm der Abbildung 30 zeigt die Standardabweichung der abschliessenden DCT Variante. Zu einer besseren Kompression kann sie deutlich genauer Punkte ablegen und erreicht eine Kompressionsrate von 14.1. Die Artefakte sind mit einer PSNR-HVS-M von 94.0 ebenfalls in Grenzen gehalten. Jedoch sind wegen dem Zoom-Feature des JHelioviewers immer noch Oszillationen zu erkennen. Das Linke Bild der Abbildung 31 zeigt die Artefakte der Dekompression. Die Artefakte sind aus der Simulation, welche diese Variante am schlechtesten approximieren konnte. Es ist ebenfalls anzumerken, dass die Artefakte bei weniger hohen Zoomstufen nicht mehr zu erkennen sind.
Die Verbesserung in der Genauigkeit wurde erreicht, indem die Feldlinientypen unterschiedlich quantisiert wurden. Die ”Sonne zu Sonne” können mit wenigeren Koeffizienten approximiert werden als die anderen Typen. Würde die Simulation nur aus diesen Feldlinien bestehen, könnte eine Kompressionsrate von $18 - 20$ erreicht werden zu einer ähnlichen Genauigkeit. Die Schwierigkeit liegt darin die anderen Feldlinien mit möglichst wenigen Artefakten zu approximieren. Eine Kompression mit der Diskreten Kosinus Transforma-

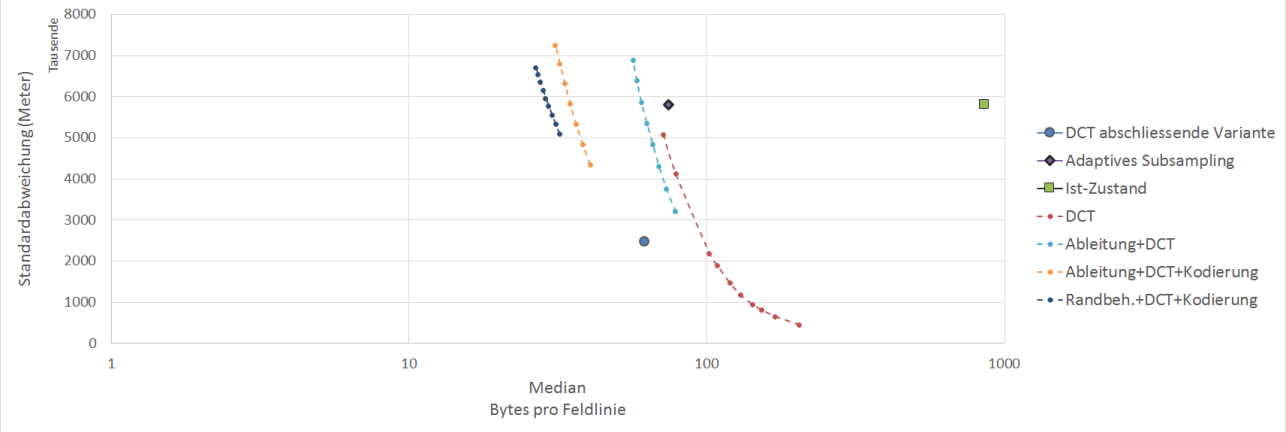

Abbildung 30: Standardabweichung der abschliessenden DCT Variante.

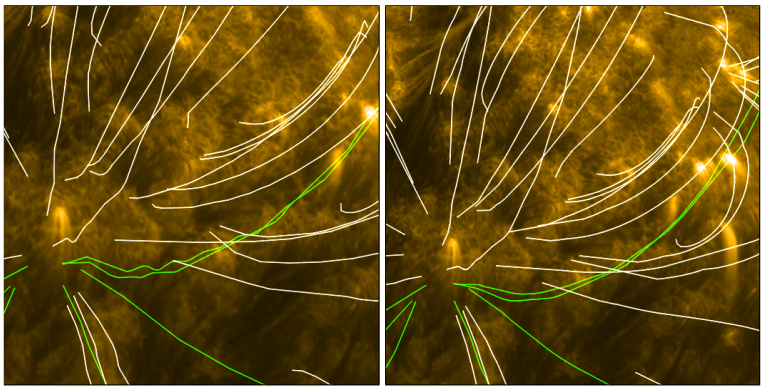

Abbildung 31: Artefakte der abschliessenden Variante. Links ohne Glättung, rechts mit Glättung

tion wird ab einer gewissen Zoom-Stufe immer Artefakte mit sich bringen, welche das menschliche Auge erkennen kann.
Der JHelioviewer kann die Artefakte verschleiern mit einer Kurvenglättung: Die komprimierten Daten enthalten mehr Punkte, als der JHelioviewer darstellen könnte. Die zusätzlichen Punkte können für die Glättung der Feldlinie verwendet werden. Den Effekt der Glättung ist in der Abbildung 31 verdeutlicht. Dadurch können die Artefakte auch bei höheren Zoomstufen verschleiert werden. Im Abschnitt 3.3 wurde erwähnt, dass reduktion von Ringing Artefakte ein aktives Forschungsfeld der Bildverarbeitung ist. Es existieren Post-Processing Filter, welche Ringing Artefakte vermindern. Eine Möglichkeit die Kompression zu verbessern ist es, einen Post-Processing Filter für wissenschaftliche Daten zu entwickeln.

6.3 Lösungsansatz: Prediktive Kodierung

Wie daten verloren gehen, warum

6.3.1 Variante: einfaches Subsampling

PCA + PCA erlaubt es, 16 Bit zu speichern. Für den Test dieser Variante wurde den Einfluss von vier Prediktoren überprüft:

- Konstanter Prediktor: Nimmt an, dass der nächste Wert im Signal gleich dem letzten Wert ist.
- Linearer Prediktor: Nimmt an, dass der nächste Wert auf der Gerade z
- Linearer Prediktor mit Moving Average:
- Adaptiver Linearer Prediktor mit Moving Average:

Wer die beste vorhersage machen kann Im Diagramm der Abbildung 32 sind die Kompressionsraten der

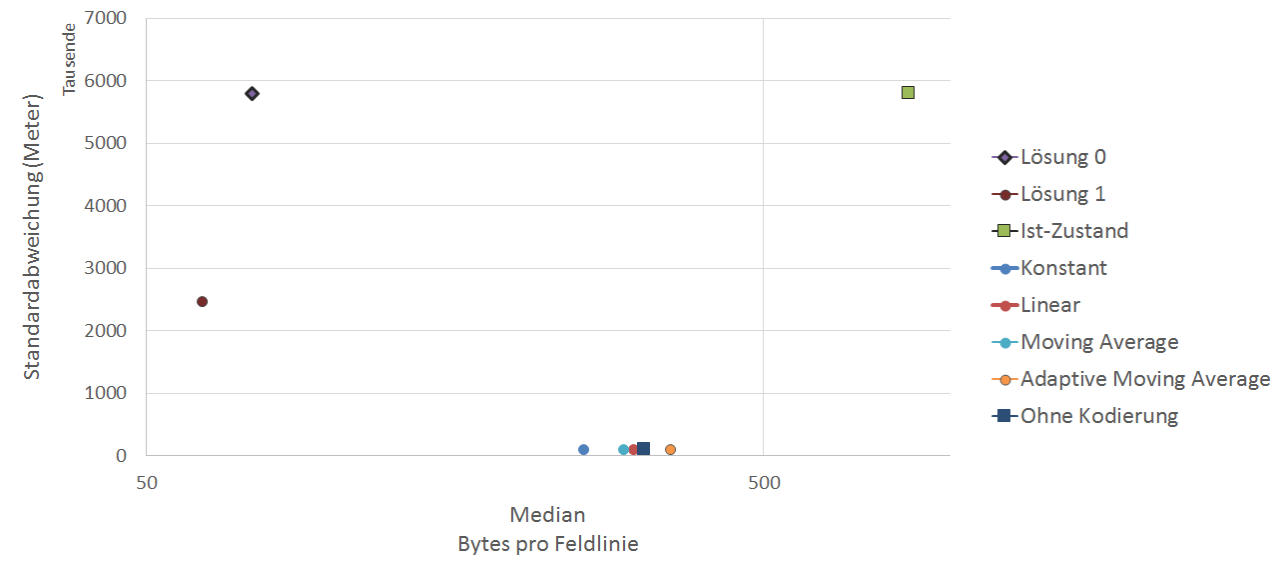

Abbildung 32: Kompressionsraten der vier Prediktoren im Vergleich zum Ist-Zustand

jeweiligen Prediktoren dargestellt. Ein Diagramm mit der PSNR-HVS-M wurde nicht erstellt. Sie ist für alle Prediktoren gleich und liegt bei 140.7 dB. Unerwartet ist, dass der Konstante Prediktor mit 255 Bytes pro Feldlinie die beste Kompression erreichte, obwohl die Daten nicht zuverlässig vorhersagen kann. Im Vergleich mit dem Moving Average Prediktor sind die Fehler der Vorhersagen bis zu 5 Mal grösser, verbrauchen aber 40 Bytes weniger um eine Feldlinie abzuspeichern. Der Fehler bleibt jedoch Konstant. Eine Möglichkeit ist, dass die Rar Kodierung sich wiederholende Muster findet.
Eine mögliche Optimierung ist die Adaptive Byte Kodierung der DCT-Variante, beschrieben im Abschnitt 3.3.5. Das Diagramm der Abbildung 33 zeigt die Resultate mit der Byte Kodierung. Der Konstante Prediktor verbraucht mit der Adaptiven Kodierung mehr Speicherplatz. Die Kompressionsrate der anderen Prediktoren wird durch die Adaptive Kodierung deutlich verbessert. Der Lineare Prediktor erreicht mit 214 Bytes pro Feldlinie die beste Kompression. Das bedeutet, dass die Fehler des Konstanten Prediktors grösser sind, als die der anderen Prediktoren. Es bestätigt die Vermutung, dass die anderen Prediktoren die Daten besser vorhersagen können. Die Kompressionsrate des Konstanten Prediktors ist auf die Rar Kodierung zurückzuführen, welche in den Prediktor-Fehler Muster erkennen und effizient kodieren kann.

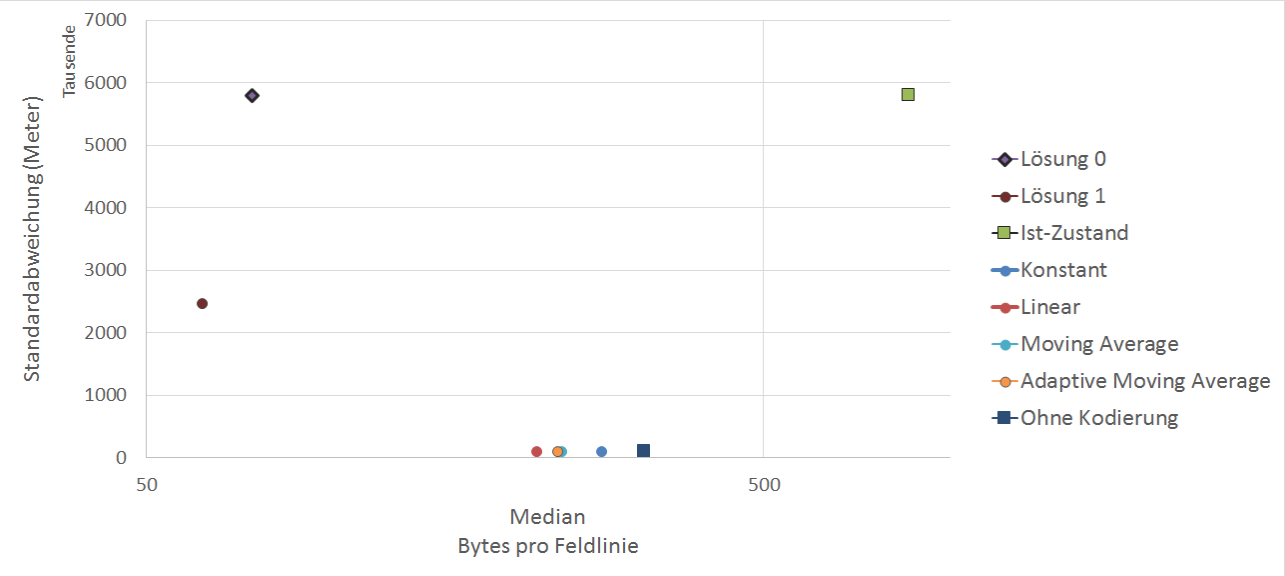

Abbildung 33: Artefakte der abschliessenden Variante. Links ohne Glättung, rechts mit Glättung

Der Linare Prediktor kann eine Feldinie mit etwa 214 Bytes darstellen, was eine Kompressionsrate von 4 ergibt. Auf kosten der Qualität wird versucht eine bessere kompressionsrate zu erreichen.

6.3.2 Variante: Adaptiven Subsampling

Gute Möglichkeit viele informationen zu verlieren. Figure schwieriger zu quantisieren, da weniger Stetig. Kurven der PCA. Möglichkeit eine Variante Prediktive kodierung für Kurven, wird aber einiges komplexer. Einfacher wenn Kodierung für Figure Sphärisches Koordinatensystem.Einfacher abzuspeichern POW Aber Ebenfalls schwer vorherzusagen

6.3.3 Wavelet Prediktive Kodierung

beschrieben im Abschnitt 3.4 figure figure psnr-hvs-m

7 Diskussion

7.1 Diskussion Lösungsansatz Adaptives Subsampling

Der Lösungsansatz des adaptiven Subsamplings erreicht eine Kompressionsrate von 11.6 gegenüber dem Ist-Zustand, indem es nur die Punkte überträgt, welche der JHelioviewers darstellt. Dadurch sind die visualisierten Feldlinie der Ist-Kompression mit diesem Lösungsansatz identisch. In Zukunft ist es aber denkbar, dass im JHelioviewer deutlich mehr Punkte dargestellt werden sollen. In diesem Fall müssen entweder mehr Punkte übertragen werden oder der JHelioviewer mit einer Interpolation erweitert werden.

Der JHelioviewer muss in der Lage sein 1000 komprimierte Simulationen im Arbeitsspeicher abzulegen. Mit dieser Kompression werden durchschnittlich 85 Megabyte an Arbeitsspeicher benötigt. Wenn von einer 10 Megabit Internetverbindung ausgegangen wird, werden für das Herunterladen von 1000 Simulationen 70 Sekunden benötigt. Das ist eine deutliche Verbesserung gegenüber zum Ist-Zustand, welcher unter den selben Bedingungen 790 Sekunden (13 Minuten) benötigt. Mit dieser Kompression können etwa 14 komprimierte Simulationen pro Sekunde übertragen werden. Der Benutzer erhält eine flüssige Animation der Feldlinien, wenn pro Sekunde weniger als 14 unterschiedliche Simulationen angezeigt werden müssen.

Ein Vorteil dieses Lösungsansatzes ist die Laufzeit Dekompression: Da keine rechenaufwändige Transformationen verwendet werden braucht dieser Lösungsansatz 19 Millisekunden für eine Dekompression (Siehe Abschnitt 9.3). Mit einem Thread ist die Testmaschine in der Lage 50 Simulationen pro Sekunde zu dekomprimieren. Anders als der DCT Lösungsansatz wird für die Dekompression keine Parameter zwischengespeichert. Dieser Lösungsansatz verbraucht für die Dekompression am wenigsten Ressourcen, vorausgesetzt dass keine Interpolation benötigt wird.

7.2 Diskussion Lösungsansatz DCT

Die DCT Kompression kann eine Kompressionsrate von 14.1 erreicht werden. Im Vergleich mit dem Lösungsansatz des adaptiven Subsampling wird eine höhere Kompressionsrate erreicht, obwohl eine höhere Anzahl an Punkte übertragen werden. Die Problematik dieses Ansatzes liegt darin, dass die DCT Kompression Ringing Artefakte hinzufügt (siehe Abschnitt 6.2.6). Die Artefakte können bei allen Kompressionsverfahren mit einer Kosinus Transformation auftreten, wie bei JPEG/JFIF Bilder und MP3 Audiodateien. Der JHelioviewer bietet die Möglichkeit, an die Feldlinien heranzuzoomen. Sobald Ringing Artefakte existieren ist der Benutzer in der Lage sie zu finden. Bei der Entwicklung der Kompression wurde nach Möglichkeiten gesucht, die Ringing Artefakte zu dämpfen. Die Daten sind Artefaktfrei bei etwa der Hälfte des Zoombereichs. Mit einer Glättung der Feldlinien konnten die Ringing Artefakte für alle Zoomstufen behoben werden. In der Bildverarbeitung wird nach weiteren Post-Processing Methoden geforscht, welche Ringing Artefakte vermindern. Die Qualität oder die Kompressionsrate könnte durch ein angepasstes Post-Processing weiter verbessert werden. Wavelet Transformation

Beim Caching von 1000 Simulationen benötigt dieser Ansatz 70 Megabyte Arbeitsspeicher, etwa 15 Megabyte weniger als der Lösungsansatz des adaptiven Subsamplings. Wenn von der selben 10 Megabit Internenetverbindung ausgegangen wird, werden 56 Sekunden benötigt um 1000 Simulationen herunterzuladen. Pro Sekunde werden 17 anstatt 14 Simulationen übertragen. Wenn für die Movies auf einen artefaktfreien Zoom verzichtet wird, ist eine höhere Kompressionsrate möglich.

Die bessere Kompressionsrate kommt auf Kosten der Komplexität der Dekompression. Die Laufzeit der Dekompression hängt im Wesentlichen von der Implementation der inversen DCT ab. Eine naive Implementation braucht für eine Dekompression etwa drei Sekunden. Die grösste Zeit wird in der Berechnung der Kosinus-Werte verbraucht. Bei der Implementation in dieser Arbeit werden die Kosinus-Werte über SoftReferences gecached. Der Cache passt sich somit an den Arbeitsspeicher an. Wie schnell die Dekompression ist, hängt

im Wesentlichen vom verfügbaren Arbeitsspeicher ab. Im besten Fall ergibt das eine Laufzeit von 65 Millisekunden und im schlechtesten Fall 350, was zu einem Durchsatz zwischen 3 und 15 Simulationen pro Sekunde pro Thread führt.

Falls 60 Frames pro sekunde erreicht werden sollen.

8 Fazit

Literatur

[1] Gregory K Wallace. The jpeg still picture compression standard. Consumer Electronics, IEEE Transactions on, 38(1):xviii–xxxiv, 1992.

[2] Siu-Wai Wu and Allen Gersho. Rate-constrained picture-adaptive quantization for jpeg baseline coders. In Acoustics, Speech, and Signal Processing, 1993. ICASSP-93., 1993 IEEE International Conference on, volume 5, pages 389–392. IEEE, 1993.

[3] Ching-Yang Wang, Shiuh-Ming Lee, and Long-Wen Chang. Designing jpeg quantization tables based on human visual system. Signal Processing: Image Communication, 16(5):501–506, 2001.

[4] Ruwen Schnabel and Reinhard Klein. Octree-based point-cloud compression. In SPBG, pages 111–120, 2006.

[5] Wikipedia. Octree — wikipedia, the free encyclopedia, 2014. [Online; accessed 4-January-2015].

[6] Michael Unser, Akram Aldroubi, and Murray Eden. B-spline signal processing. i. theory. Signal Processing, IEEE Transactions on, 41(2):821–833, 1993.

[7] Paruj Ratanaworabhan, Jian Ke, and Martin Burtscher. Fast lossless compression of scientific floating-point data. In Data Compression Conference, 2006. DCC 2006. Proceedings, pages 133–142. IEEE, 2006.

[8] Wikipedia. Gzip — wikipedia, the free encyclopedia, 2014. [Online; accessed 17-December-2014].

[9] Wikipedia. Ringing artifacts — wikipedia, the free encyclopedia, 2014. [Online; accessed 23-December-2014].

[10] Wikipedia. Gaussian filter — wikipedia, the free encyclopedia, 2014. [Online; accessed 2-January-2015].

[11] André Kaup. Reduction of ringing noise in transform image coding using simple adaptive filter. Electronics Letters, 34(22):2110–2112, 1998.

[12] HyunWook Park and Yung Lyul Lee. A postprocessing method for reducing quantization effects in low bit-rate moving picture coding. Circuits and Systems for Video Technology, IEEE Transactions on, 9(1):161–171, 1999.

[13] Nikolay Ponomarenko, Flavia Silvestri, Karen Egiazarian, Marco Carli, Jaakko Astola, and Vladimir Lukin. On between-coefficient contrast masking of dct basis functions. In Proceedings of the Third International Workshop on Video Processing and Quality Metrics, volume 4, 2007.

[14] Hervé Abdi and Lynne J Williams. Principal component analysis. Wiley Interdisciplinary Reviews: Computational Statistics, 2(4):433–459, 2010.

Abbildungsverzeichnis

1 Visualisierung der Feldlinien im JHelioviewer . . . 1
2 Vereinfachter Ablauf einer verlustbehafteten Kompression . . . 3
3 Aufbau der JPEG Kompression [1] . . . 3
4 Aufbau einer Octree basierten Point Cloud Kompression. . . . 4
5 Aufbau der Ist-Kompression. . . . 6
6 Aufbau des Lösungsansatzes: Adaptives Subsampling. . . . 6
7 Darstellung des Adaptiven Subsapmlings im 2D Raum. Rot sind die Punkte, welche geprüft und gelöscht wurden. Grün ist der Punkt dargestellt, welcher geprüft wird. . . . 7
8 Aufbau des Lösungsansatzes: Adaptives Subsampling. . . . 8
9 Aufbau des Lösungsansatzes: Adaptives Subsampling. . . . 11
10 Erster Schritt der Wavelet Prediktiven Kodierung. Zu sehen sind das zu kodierende Signal, die Vorhersage und den Fehler der Vorhersage. . . . 12
11 Zweiter Schritt der Wavelet Prediktiven Kodierung. Die Vorhersage beinhaltet nun zwei Strecken. 12
12 Darstellung der Fehlerberechnung. Die Punkte sind die Originaldaten, die Quadrate sind die Punkte nach der Kompression. . . . 14
13 Flussdiagramm der PSNR-HVS-M Berechnung [13]. . . . 15
14 Zustandsdiagramm der Feldliniendaten . . . 18
15 Diagramm der Implementation vom Vorladen und Caching . . . 19
16 Vergleich des Lösungsansatzes: Adaptives Subsampling zur Ist-Kompression. . . . 20
17 Artefakte der Lösung 0 . . . 21
18 Vergleich der DCT Kompression mit der Lösung0 . . . 22
19 Artefakte der DCT Dekompression anhand Beispieldaten . . . 22
20 Vergleich der DCT Kompression der Ableitung mit der DCT Kompression . . . 23
21 Artefakte der DCT Kompression der Ableitung . . . 23
22 Vergleich der PCA DCT Kompression der Ableitung mit der DCT Kompression der Ableitung . 24
23 Vergleich der Kompression mit und ohne Byte-Kodierung . . . 25
24 Vergleich des Einflusses der Randbehandlung . . . 25
25 Artefakte der Kompression, links sind die originalen Feldlinien, rechts die Dekomprimierten. . . 26
26 Artefakte der Kompression, links sind die originalen Feldlinien, rechts die Dekomprimierten der Variante 6.2.4. . . . 26
27 Abrupte Steigungen bei Feldlinien, welche von der Sonne ins Weltall führen. . . . 27
28 Approximation der Feldlinien ”Sonnenoberfläche zu Sonnenoberfläche”. Je höher die PSNR-HVS-M, desto besser ist die Approximation. . . . 27
29 Approximation der der Feldlinien ”Sonnenoberfläche ins Weltall” oder ”Weltall zur Sonnenoberfläche”. Je höher die PSNR-HVS-M, desto besser ist die Approximation. . . . 28
30 Standardabweichung der abschliessenden DCT Variante. . . . 29
31 Artefakte der abschliessenden Variante. Links ohne Glättung, rechts mit Glättung . . . 29
32 Kompressionsraten der vier Prediktoren im Vergleich zum Ist-Zustand . . . 31
33 Artefakte der abschliessenden Variante. Links ohne Glättung, rechts mit Glättung . . . 32

Tabellenverzeichnis

1 Tabelle der Lösungsansätzen und dessen Kompressionsraten. . . . 2
2 Anordnung der Simulationsdaten der Ist-Kompression . . . 6
3 Beispiel eines abgespeicherten Kanals mit der Längenkodierung. . . . 10
4 Aufteilung eines Bytes der adaptiven Genauigkeitskodierung. X sind Nutz-Bits. . . . 10
5 Einfluss des f_m Faktors auf die PSNR-HVS-M. . . . 16

6 Content of the FITS File 39
7 Beispiel eines abgespeicherten Kanals mit der Längenkodierung. 39
8 Aufteilung eines Bytes der adaptiven Genauigkeitskodierung. X sind Nutz-Bits. 40
9 Aufteilung eines Bytes der adaptiven Genauigkeitskodierung. X sind Nutz-Bits. 40

9 Anhang

9.1 Installationsanleitung

9.1.1 Installation des JHelioviewers

9.1.2 Installation der Kompression

9.2 Dateiformat und Dekompression

This section describes the encoding and decompression algorithm used for compressing fieldline data. The data is written in the FITS Format [**?**] and compressed with Rar. The content of the FITS file is described in Table 6.
After decompressing the FITS file with Rar apply the decodings described in section 9.2.1. The next step is

Name	Datatype	Encoding	Content Description
B0	Double	None	Latitude to Earth
L0	Double	None	Longitude to Earth
TYPE	Byte Array	None	Type of quantization used for each Fieldline
LINE_LENGTH	Byte Array	Adaptive Precision Unsigned	Length of each Fieldline
StartPointR	Byte Array	Length + Adaptive Precision	Radius of the Startpoint for each Fieldline
StartPointPhi	Byte Array	Length + Adaptive Precision	Latitude of the Startpoint for each Fieldline
StartPointTheta	Byte Array	Length + Adaptive Precision	Longitude of the Startpoint for each Fieldline
X	Byte Array	Length + Adaptive Precision	X Channel DCT Coefficients of each Fieldline
Y	Byte Array	Length + Adaptive Precision	Y Channel DCT Coefficients of each Fieldline
Z	Byte Array	Length + Adaptive Precision	Z Channel DCT Coefficients of each Fieldline

Tabelle 6: Content of the FITS File

to split up the data for each fieldline. Each fieldline contains exactly one Type, one Length and one Startpoint. The Length describes the number of DCT Coefficients of the fieldline. The first batch of coefficients belong to the first fieldline, and the second batch to the second line etc.
After splitting the data there are four steps left to do:

1. Transform the Startpoint to Euler coordinates
2. Multiply the DCT Coefficients according to the Type
3. Apply the Inverse Cosine Transform (Formula (3.2) in section 3.3.3)
4. Integrate the channels

How to Transform the Point to Euler coordinates is described in section 9.2.2. The multiplication factors can be gathered from the JHelioviewer implementation. Take note that the decompression time will depend on your implementation of the Inverse Cosine Transform, and a naive implementation can take three seconds or more for one FITS file.
The last step ”Integrate the channels” needs further clarification: Each channel is derived before the Discrete Cosine Transform is applied. After the Inverse Transform, each channel contains the rises and not the actual points.

9.2.1 Encodings

Length Encoding

Encoded Channel							
Block of Fieldline 0					...		
n_0	$x_{0,0}$	$x_{0,1}$	...	$x_{0,n-1}$	n_1	$x_{1,0}$	...

Tabelle 7: Beispiel eines abgespeicherten Kanals mit der Längenkodierung.

Unsigned Adaptive Precision Encoding

Byte							
Continue Flag	X	X	X	X	X	X	X

Tabelle 8: Aufteilung eines Bytes der adaptiven Genauigkeitskodierung. X sind Nutz-Bits.

Adaptive Precision Encoding

Byte							
Continue Flag	Sign Flag	X	X	X	X	X	X

Tabelle 9: Aufteilung eines Bytes der adaptiven Genauigkeitskodierung. X sind Nutz-Bits.

9.2.2 Transformation to Euler Coordinates

9.3 Performance Tests

Lösunsansatz	Durchschnittliche Dekompressionszeit (ms)
Adaptives Subsampling	19 ms
DCT - Naive Inverse DCT	3100 ms
DCT - Mit Kosinus Caching	$65 - 350$ ms

Testgerät

10 Ehrlichkeitserklärung