

## **Abstract**

# Inhaltsverzeichnis

<b>1</b>	<b>Verlustbehaftete Kompression von volumetrischen Punkten</b>	<b>1</b>
<b>2</b>	<b>State of the Art</b>	<b>3</b>
2.1	JPEG/JFIF Bildkompression . . . . .	3
2.2	Point Cloud Kompression . . . . .	4
2.3	Curve Fitting . . . . .	4
2.4	Compressive Sensing . . . . .	4
2.5	Entropie Kodierung . . . . .	4
<b>3</b>	<b>Kompressionsverfahren der Feldlinien</b>	<b>6</b>
3.1	Ist-Komprimierung . . . . .	6
3.2	Lösungsansatz: Adaptives Subsampling . . . . .	6
3.2.1	Adaptives Subsampling . . . . .	6
3.2.2	Entropie Kodierung mittels RAR . . . . .	7
3.3	Lösungsansatz: 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
<b>4</b>	<b>Qualitätsmessung der dekomprimierten Daten</b>	<b>11</b>
4.1	Auswahl und Erhebung der Testdaten . . . . .	11
4.2	Berechnung der Standardabweichung . . . . .	11
4.2.1	Allgemeiner Fall . . . . .	11
4.2.2	Randbehandlung . . . . .	12
4.2.3	Berechnung der Standardabweichung . . . . .	12
4.3	Berechnung der angepassten PSNR-HVS-M . . . . .	13
4.3.1	Contrast Masking . . . . .	13
4.3.2	Umsetzung und Anpassung der PSNR-HVS-M für diese Arbeit . . . . .	14
<b>5</b>	<b>Implementation</b>	<b>16</b>
5.1	Software Architektur . . . . .	16
5.1.1	Mehrstufiges Vorladen und Caching . . . . .	16
5.1.2	Asynchrone Aufrufe mittels Executor Services . . . . .	18
<b>6</b>	<b>Resultate</b>	<b>19</b>
6.1	Lösungsansatz: Adaptives Subsampling . . . . .	19
6.2	Lösungsansatz: Diskrete Kosinus Transformation . . . . .	20
6.2.1	Variante: DCT . . . . .	20
6.2.2	Variante: Ableitung+DCT . . . . .	21
6.2.3	Variante: PCA+Ableitung+DCT . . . . .	22
6.2.4	Variante: Ableitung+DCT+Byte Kodierung . . . . .	23
6.2.5	Variante: Randbehandlung+DCT+Byte Kodierung . . . . .	23
6.2.6	Ringing Artefakte . . . . .	24
6.2.7	Behandlung der Ringing Artefakte . . . . .	24
6.2.8	Abschliessende Variante . . . . .	26
<b>7</b>	<b>Diskussion</b>	<b>29</b>
7.1	Diskussion Lösungsansatz Adaptives Subsampling . . . . .	29

7.2 Diskussion Lösungsansatz DCT . . . . .	29
<b>8 Fazit</b>	<b>30</b>
<b>9 Anhang</b>	<b>33</b>
<b>10 Ehrlichkeitserklärung</b>	<b>34</b>

## 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 Anwendungen können natürliche Phänomene wie Schwingungen, Flugbahnen, Kraftfelder etc. als Linien im dreidimensionalen Raum abgebildet werden. Eine wissenschaftliche Simulation erstellt eine Menge an volumetrischen Punkten, welche die Linien im Raum darstellen. Ziel dieser Arbeit ist es eine verlustbehaftete Kompression von wissenschaftlichen Daten 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

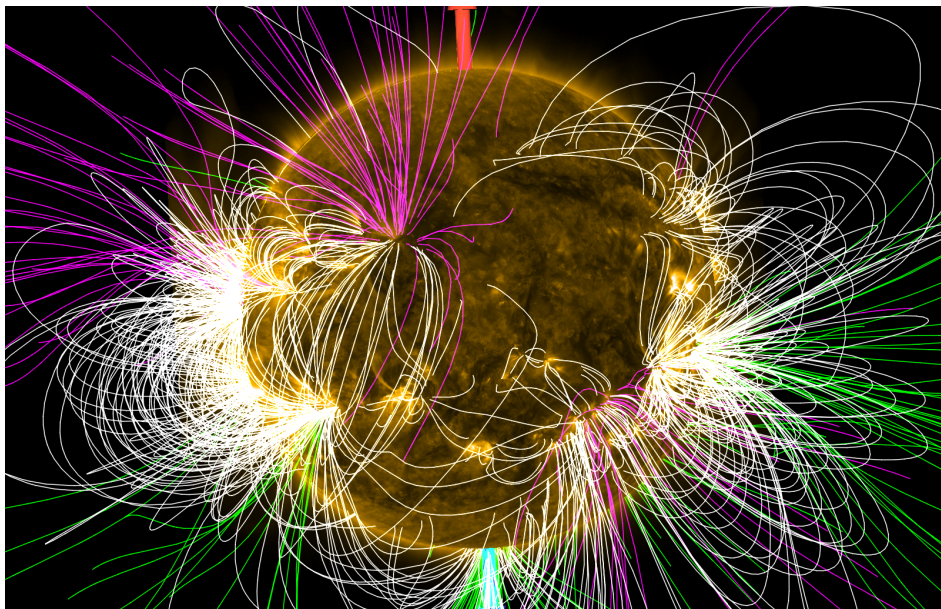


Abbildung 1: Visualisierung der Feldlinien im JHelioviewer

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".

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 er-

sichtlich.

Lösungsansatz	Kompressionsraten
Adaptives Subsampling	11.6
Diskrete Kosinus Transformation	x

Tabelle 1: Tabelle der Lösungsansätzen und dessen Kompressionsraten.

## 2 State of the Art

Grob betrachtet beinhalten verlustbehaftete Kompressionen drei Teilschritte: Transformation, Quantisierung und Entropie Kodierung. Die Abbildung 2 zeigt eine vereinfachte Abfolge. Die Inputdaten werden zuerst durch ein oder mehrere Verfahren transformiert. So transformieren, dass in der Quantisierung unwichtige Informationen entfernt werden können. Oft gehen nur im Quantisierungsschritt Daten verloren, während alle anderen Schritte verlustfrei umkehrbar sind. Danach werden die Daten Entropie Kodiert. Für jeden Teilschritt gibt es unterschiedliche Verfahren. So können auch mehrere Transformationen hintereinander durchgeführt werden, oder eine ganze Folge von Transformations- und Quantisierungsverfahren.

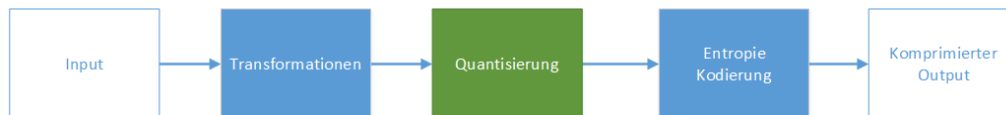


Abbildung 2: Typische Teilschritte einer Kompression

### 2.1 JPEG/JFIF Bildkompression

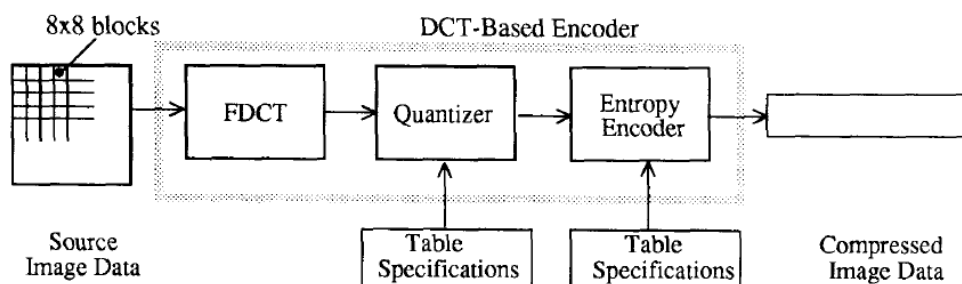


Abbildung 3: Aufbau der JPEG Kompression [1]

Das JPEG/JFIF Bildformat 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  $8 \times 8$  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 mit 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 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

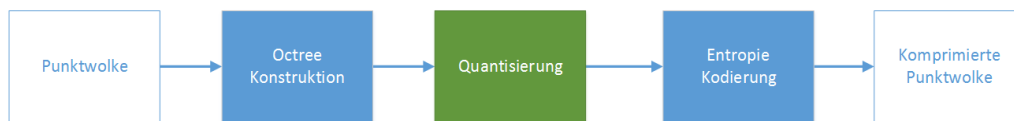


Abbildung 4: Aufbau einer Octree basierten Point Cloud Kompression.

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 Punkte 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 das nächste Level aussieht. Die Prediktive Kodierung muss also nur die Daten abspeichern, welche nicht mit der Vorhersage übereinstimmen.

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 benutzten eine Oberflächen-Approximation. Die Feldlinien jedoch bilden im Allgemeinen keine Oberfläche.

## 2.3 Curve Fitting

Curve Fitting ist ein Teilgebiet der Signalverarbeitung. Ein Signal wird aus einer oder mehreren Basisfunktionen ausgedrückt und findet typischerweise Anwendungen in den Bereichen Signalinterpolation, Differenzierung und Rauschunterdrückung. Aber auch eine verlustbehaftete Kompression ist Möglich.

Die B-Spline Transformation [6] ist in linearer Zeit. Möglichkeit, ein Signal mit B-Splines zu approximieren. Die Problematik liegt in den Details wie Anzahl Funktionen, Verteilung der Spline Knotenpunkten etc.

## 2.4 Compressive Sensing

## 2.5 Entropie Kodierung

Die Entropie Kodierung findet in allen Bereichen der Informatik Anwendungen. 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.

Typische Verfahren sind Run Length, Huffman oder Aritmetische Kodierung. 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 Kombination verwendet, zb. Rle + huffman

Wird zuerst sicher ein allgemeines Verfahren.



### 3 Kompressionsverfahren der Feldlinien

konzept der unterschiedlichen kompressionsverfahren. zwei lösungsansätze. bereits eine simple Kompression implementiert.

#### 3.1 Ist-Komprimierung

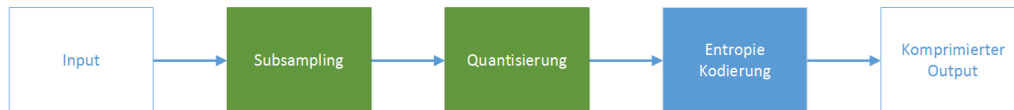


Abbildung 5: Aufbau der Ist-Kompression.

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 Implementierungen der Entropie Kodierer scheinen Integer-Werte einfacher komprimieren zu können. In der Entropie Kodierung werden die Daten geordnet wie in Tabelle 2 dargestellt. Alle X-Kanäle der Feldlinien werden hintereinander abgelegt, ge-

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

Tabelle 2: Anordnung der Simulationsdaten der Ist-Kompression

folgt von allen Y-Kanälen etc. Zuvor werden aber noch alle Längen der Feldlinien abgespeichert, sodass jeder Punkt einer Linie zugeordnet werden kann. 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].

Da die Punktmenge für Low-End Grafikkarten zu gross ist, führt der JHelioviewer ein weiteres Subsampling durch, welches im Abschnitt 3.2.1 beschrieben ist.

#### 3.2 Lösungsansatz: Adaptives Subsampling

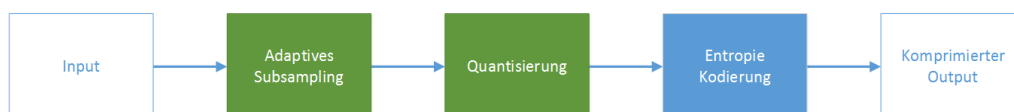


Abbildung 6: Aufbau des Lösungsansatzes: Adaptives Subsampling.

Dieser Lösungsansatz verwendet die selbe Pipeline wie die Ist-Kompression 3.1. Der Unterschied ist, dass ein anderes Subsampling Verfahren gewählt wurde und eine andere Entropie-Kodierung. Die Abbildung 6 zeigt den neuen Ablauf.

##### 3.2.1 Adaptives Subsampling

Ziel des adaptiven Subsamplings ist es, die Daten durch eine Folge von Strecken zu approximieren. An Stellen, welche die Feldlinie gekrümmt ist, braucht es mehr Strecken. An Stellen, welche die Feldlinie linear verläuft, können Punkte gespart werden. Das Diagramm der Abbildung 7 stellt das Subsampling im zweidimensionalen

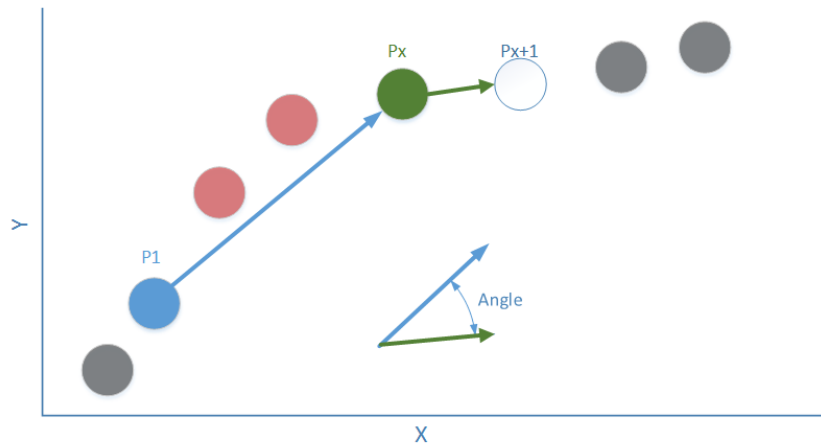


Abbildung 7: Darstellung des Adaptiven Subsapmlings im 2D Raum. Rot sind die Punkte, welche geprüft und gelöscht wurden. Grün ist der Punkt dargestellt, welcher geprüft wird.

Raum dar. Zu sehen sind die Punkte der Feldlinie. 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  $\alpha$ , 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 Lösungsanz: Diskrete Kosinus Transformation

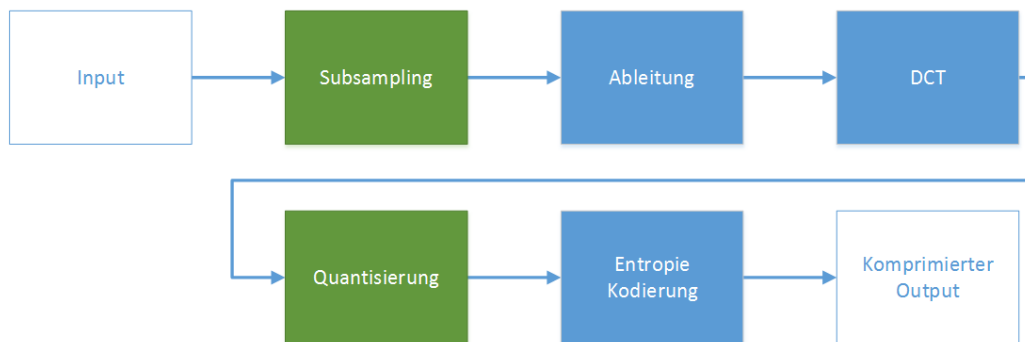


Abbildung 8: Aufbau des Lösungsansatzes: Adaptives Subsampling.

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 die Ableitung Kosinus Transformiert 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 erleutert.

Es gibt Möglichkeiten, die Ringing Artefakte zu Dämpfen oder gar zu beheben: Die simpelste Variante ist es, das Signal zum Beispiel mit einem Gauss Filter [10] zu glätten. Die Glättung kann das Signal verfälschen und ist deshalb nicht die optimale Lösung. In der Bild- und Audioverarbeitung wird aktiv nach Filter geforscht, welche die Ringing Artefakte in der Dekompression vermindern [11] [12].

In dieser Arbeit wurde für die drei Typen von Feldlinien unterschiedliche Quantisierungsfaktoren umgesetzt.

#### 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 schneller.

Falls die Laufzeit der Dekompression weiter verbessert werden soll, kann die Fast-Cosine-Transformation umgesetzt werden. Diese hat eine Komplexität von  $O(n \log(n))$ . 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. Vermutlich braucht es für die Approximation der Blöcke insgesamt mehr Kosinus-Funktionen, als für die Approximation der gesamten Feldlinie.

#### 3.3.2 Ableitung

Die Feldlinie wird abgeleitet und alle folgenden Transformationen werden auf den Steigungen der Feldlinie ausgeführt. Damit die Transformation umkehrbar ist, muss der Startpunkt zusätzlich abgespeichert werden. Für das menschliche Auge sind die Artefakte, die in der abgeleiteten Feldlinie entstehen, weniger störend. Die Feldlinie bleibt tendenziell glatt. Artefakte äussern sich meist in veränderten Amplituden, welche erst erkennbar sind, wenn die Originalfeldlinie zum Vergleich bereit steht.

### 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 Vorwärts und die DCT-III (3.2) als Rückwärtstransformation verwendet [1].

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

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

Wobei  $N$  die Länge des diskreten Signals ist.  $x_n$  bezeichnet einen Wert im diskreten Signal und  $X_k$  ist der Anteil der Frequenz  $k$ . Ein Inputsignal der Länge  $N$  resultiert in  $N$  Kosinus-Funktionen.

Die DCT transformiert ein periodisches, unendliches Signal. Um ein endliches Signal zu transformieren, wird das Signal konzeptionell wiederholt. Welche Transformationen gewählt werden, wirkt sich aus, in welcher Art das Signal wiederholt wird. In diesem Fall wird das Signal jeweils in umgekehrter Reihenfolge wiederholt. Wie die Ränder aussehen, wirkt sich auf die Dekompression aus und kann je nach dem ungewünschte Artefakte mit sich bringen. Im Diagramm der Abbildung 17 im Abschnitt 6.2.1 ist ein Beispiel zu sehen für mögliche Artefakte bei ungeeigneten Rändern.

### 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. Es wird deshalb für diesen Typ von Feldlinien eine andere Quantisierung durchgeführt.

### 3.3.5 Entropie Kodierung

Daten werden geordnet, genaue Spezifikation ist im Anhang zu finden. 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 quantisierten Koeffizienten einer Feldlinien-Kanals können in zwei Blöcke unterteilt werden: Der erste Block enthält alle nicht-Null Koeffizienten, sie approximieren den Kanal. Der zweite Block besteht aus hochfrequenten Schwingungen, welche auf Null quantisiert wurden. Die Längenkodierung schneidet den Null block 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 Nullen Anfügen, bis die ursprüngliche Länge des Kanals erreicht wurde. Die Anzahl Punkte jeder Feldlinie ist ebenfalls gespeichert.

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.

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. Durch die Quantisierung kann dieser Fall nur bei Kanälen mit weniger als 50 Punkten auftreten.

### Adaptive Genauigkeits-Kodierung

Die quantisierten Koeffizienten sind meistens klein und liegen zwischen  $-50$  und  $+50$ . Acht Bit Genauigkeit würden meistens ausreichen, nur wenige Koeffizienten brauchen mehr Genauigkeit. 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.

## 4 Qualitätsmessung der dekomprimierten Daten

Bei verlustbehafteten Kompressionen muss die Qualität des dekomprimierten Bildes sichergestellt werden. Im optimalen Fall ähneln die dekomprimierten Daten ihren Originalen, sodass für das menschliche Auge keine Artefakte sichtbar sind.

Im Verlauf der Arbeit wurden zwei Metriken verwendet: Die Standardabweichung und eine angepasste PSNR-HVS-M. Für die ersten Tests wurde nur die Standardabweichung gemessen. Es stellte sich heraus, dass die Standardabweichung nicht ausreicht: Oszillationen um die Originallinie fallen nicht ins Gewicht. Der absolute Fehler bleibt klein, für das menschliche Auge jedoch sind solche Artefakte störend. Ein Beispiel für die Artefakte ist in Abbildung 23 im Abschnitt 6.2.6 zu finden. Wie die Standardabweichung berechnet wird, ist im Abschnitt 4.2 beschrieben. Für weitere Tests wurde zusätzlich die PSNR-HVS-M Metrik berechnet. Die Metrik 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 anfangen 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. Das Mass bewertet seltene, grosse Abstände stärker.

Die originale und dekomprimierte Feldlinie können unterschiedliche Abtastraten aufweisen. Die Standardabweichung muss deshalb unabhängig von der Abtastrate berechnet werden.

#### 4.2.1 Allgemeiner Fall

Um die Abweichung unabhängig von der Abtastrate zu berechnen, wird zwischen den dekomprimierten Punkte eine Linie gezogen und den Abstand von dieser berechnet. Der Vorgang ist dargestellt im Diagramm der Abbildung 9. Für jeden Punkt  $P1'$  aus den dekomprimierten Punkten  $D$ , nehme  $P1'$  und den folgenden Punkt  $P2'$ . Ziehe eine Strecke  $s$  durch  $P1'$  und  $P2'$ . Suche von  $p1'$  den Originalpunkt  $P1$  aus allen Originalpunkten

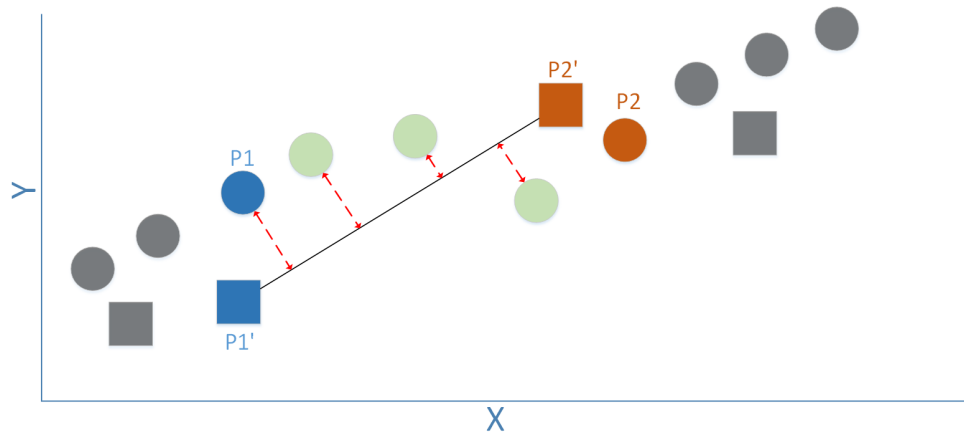


Abbildung 9: Darstellung der Fehlerberechnung. Die Punkte sind die Originaldaten, die Quadrate sind die Punkte nach der Kompression.

$O$  und rechne den Abstand aus zur Strecke  $s$ . Führe das für alle folgenden Originalpunkte durch, bis  $P2$  erreicht wurde. Der Abstand  $s$  zu  $P2$  wird nicht mehr berechnet.

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  $P$  berechnet.

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

$$distance = \frac{|\vec{BA} \times \vec{BP}|}{|\vec{BP}|} \quad (4.2)$$

$A$  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.

#### 4.2.2 Randbehandlung

Es ist möglich, dass die originalen Endpunkte der Feldlinien durch ein Subsampling verworfen wurden. Es ist möglich, dass am Anfang und am Ende Originalpunkte existieren, für die nie eine Distanz berechnet wurde. Das Diagramm der Abbildung 10 zeigt das Problem. Deshalb müssen die Abstände der Eckpunkte von der Komprimierten- zur Original-Linie berechnet werden.

#### 4.2.3 Berechnung der Standardabweichung

$$\sigma(X) = \sqrt{\text{variance}(X)} \quad (4.3)$$

$$\text{variance}(X) = \sum (x_i - E(x_i))^2$$

Die Standardabweichung  $\sigma$  einer Beobachtungsreihe  $X$  ( $x_1, x_2, x_3, \dots, x_n - 1$ ) 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

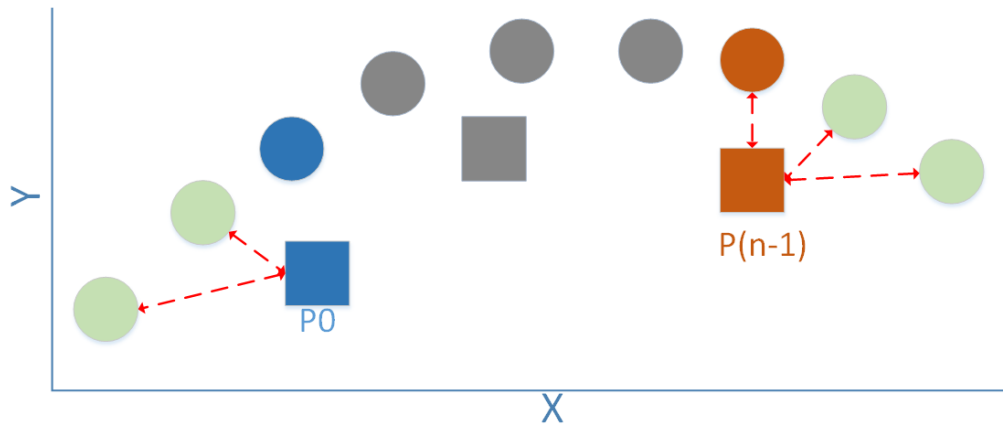


Abbildung 10: Darstellung der Fehlerberechnung. Die Punkte sind die Originaldaten, die Quadrate sind die Punkte nach der Kompression.

der dekomprimierten Linie, während der Erwartungswert der Originalpunkt ist. Die Distanz wird mit dem besprochenen 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 kann für die Messung des Fehlers zwischen dekomprimierten Bild und dem Original eingesetzt werden. Das Problem der Metrik ist aber, dass es nicht immer mit der menschlichen Wahrnehmung korreliert. 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.

$$PSNR = 20 * \log_{10}(MAX_I) - 10 * \log_{10}(MSE)$$

$$MSE = \frac{1}{n} \sum_{i=0}^{N-1} [E(i) - D(i)]^2 \quad (4.4)$$

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  $D()$ . Der Unterschied zwischen der PSNR und der PSNR-HVS-M liegt in der Berechnung des durchschnittlichen quadratischen Fehlers. Das Diagramm der Abbildung 11 zeigt den Ablauf der neuen Berechnung.

PSNR-HVS-M berechnet die Differenz zwischen dem Originalbild  $E$  und dem verrauschten Bild  $D$  und führt die Daten mittels einer DCT in den Frequenzraum. Der nächste Schritt Contrast Masking 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:



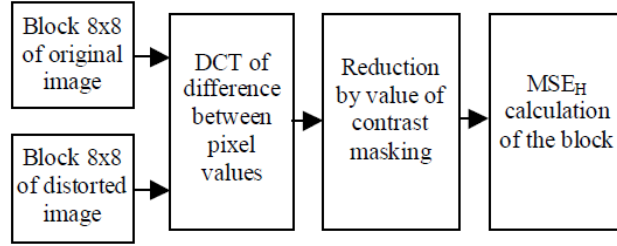


Abbildung 11: Flussdiagramm der PSNR-HVS-M Berechnung [13].

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

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

Wobei  $X$  die Kosinus-Koeffizienten eines Bildblocks sind und  $C$  die jeweiligen Gewichtungen zur Frequenz. 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.

$$E_w(X_e - X_d) < \max[E_m(X_e), E_m(X_d)] \quad (4.7)$$

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

$$X_{\Delta ij} = \begin{cases} X_{eij} - X_{dij}, & i = 0, j = 0 \\ 0, & |X_{eij} - X_{dij}| \leq E_{norm} / C_{ij} \\ X_{eij} - X_{dij} - E_{norm} / C_{ij}, & X_{eij} - X_{dij} > E_{norm} / C_{ij} \\ X_{eij} - X_{dij} + E_{norm} / C_{ij}, & \text{otherwise} \end{cases} \quad (4.8)$$

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 11 angedeutet, zuerst die Differenz der Pixelwerte berechnet und diese DC transformiert. Beide Wege führen zum selben Resultat. 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 Standard JPEG 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. Das menschliche Auge kann leichte Verschiebungen der Feldlinien kaum unterscheiden. Des weiteren musste für die Feldlinie eine eigene Quantisierungsmatrix gefunden werden, aus denen die Gewichtungen

berechnet werden. Die Quantisierungsmatrix wurde empirisch ermittelt, sodass die dekomprimierten Feldlinien keine sichtbaren Artefakte aufweisen.

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	<b>95.6</b> dB	94.5 dB
Kaum sichtbare Artefakte	95.8 dB	<b>94.4</b> dB	93.3 dB
Sichtbare Artefakte	90.0 dB	<b>88.3</b> 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 exakt ein Punkt abgespeichert wird, würde die PSNR-HVS-M trotzdem eine hohe Ähnlichkeit zwischen Original und dekomprimierten Feldlinien ergeben. Dies ist Vertretbar, da die Standardabweichung eine hohe Distanz ergeben würde. 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

Um die Feldlinien darzustellen müssen diese heruntergeladen und dekomprimiert werden. Die zwei Operationen sind die Hauptverantwortlichen für die Wartezeit. Um die Wartezeit zu verkürzen, werden Feldlinien bereits im Voraus asynchron heruntergeladen. Somit sind die Daten bereits im Arbeitsspeicher, bevor die Visualisierung sie benötigt.

Die Wartezeit kann mit zusätzlichen Massnahmen weiter verkürzt werden. Folgende Massnahmen wurden implementiert:

1. Asynchrone Dekompression.
2. Vorladen der Dekomprimierten Feldlinien.
3. Mehrstufiges Caching.

Während der Benutzer sich eine Simulation der Feldlinie sieht, soll asynchron bereits die nächsten Simulationen heruntergeladen und dekomprimiert werden. So ist der Wechsel von Simulation zu Simulation so kurz wie möglich. Damit die Dekompression den Wechsel nicht verlangsamt, werden mehr als nur die aktuelle Simulation dekomprimiert. Die komprimierten Daten sollen ebenfalls im Vorfeld heruntergeladen werden. So sind die Daten bereits im Arbeitsspeicher, sobald die Dekompression gestartet wird.

Durch das Caching der unkomprimierten und der komprimierten Feldlinien soll der Wechsel beschleunigt werden, wenn der Benutzer nicht die nächste Simulation, sondern die vorhergehende nochmals anzeigen möchte. Je nach grössse des Arbeitsspeichers könnten die unkomprimierten Daten komplett zwischengespeichert werden, sodass nur noch die Dekompression durchgeführt werden muss.

### 5.1 Software Architektur

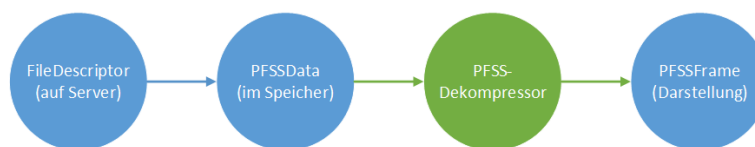


Abbildung 12: Zustandsdiagramm der Feldliniendaten

Die Daten der Feldlinien durchlaufen im JHelioviewer vier Zustände, welche durch vier Klassen abgebildet wurde. Die Klassen sowie die Zustandswechsel sind im Diagramm der Abbildung 12 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. Für das Herunterladen ist ebenfalls die "PFSSData" Klasse zuständig. "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 Mehrstufiges Vorladen und Caching

Um den Flaschenhals herunterladen und Dekomprimieren zu umgehen, wird ein mehrstufiges Read-Ahead und Caching eingeführt. Es soll ein Vorladen und Caching für die unkomprimierten "PFSSFrame" Feldlinien und eines für die "PFSSData" Objekte implementiert werden. Die Implementation ist im Diagramm der

Abbildung 13 dargestellt. Der FileDescriptorManager sucht die Feldlinien auf dem Server und stellt die File-

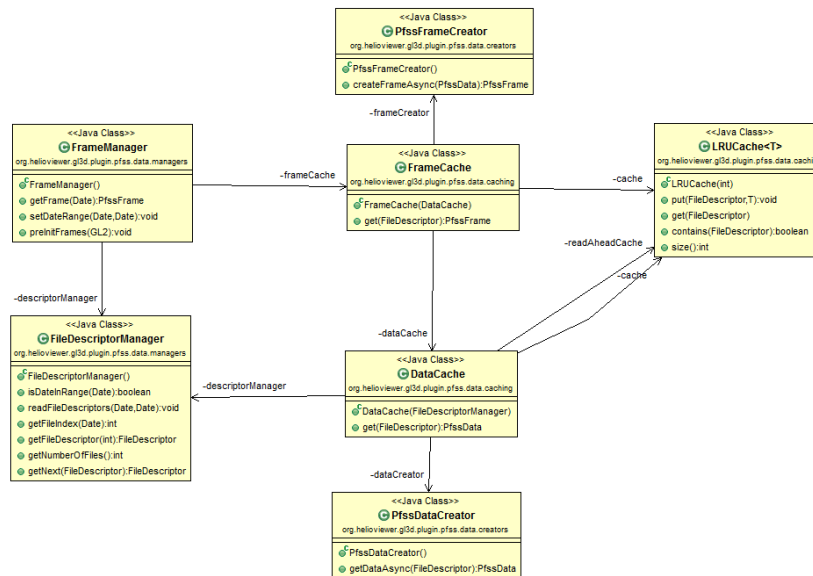


Abbildung 13: Diagramm der Implementation vom Vorladen und Caching

Descriptors zur Verfügung. Der Manager weiss ebenfalls, in welcher Reihenfolge die FileDescriptors zu liegen kommen.

Die Klasse "FrameManager" ist die Facade der Vorladens- und Caching- Implementation. Sie abstrahiert das Zusammenspiel der verschiedenen Caches und den verschiedenen Zustände der Feldliniendaten, indem eine vereinfachte Schnittstelle angeboten wird. Für das Vorladen von PFSSFrame Objekte wurde wegen dem Prinzip von Low-Coupling ebenfalls im FrameManager umgesetzt. Die "PFSSFrame" Objekte müssen jeweils Speicher auf der Grafikkarte allozieren und abräumen. Die FrameManager Klasse muss genau wissen, wann ein PFSSFrame Objekt neu geladen oder nicht mehr gebraucht wird. Die Klasse FrameCache nur noch zuständig, PFSSFrame Objekte zwischenspeichern, welche keine Ressourcen der Grafikkarte alloziert haben. Es ist auch eine Architektur denkbar, in der der FrameCache das Vorladen und das Abräumen des Grafikkartenspeichers übernimmt. Der Speicherplatz der Grafikkarte würde aber die Grösse des FrameCaches beschränken.

Das Vorladen und Caching der PFSSData Objekten wird von der Klasse DataCache übernommen. Die PFSSData Objekte allozieren nur Arbeitsspeicher, die vom Garbage-Collector aufgeräumt werden können. Das Vorladen ist deshalb simpler und wird direkt im DataCache implementiert durch eine weitere Instanz des LRU Caches. Der eigentliche Cache beinhaltet alle PFSSData Objekte, welche gebraucht wurden und der readAheadCache alle, welche noch gebraucht werden können.

In dieser Arbeit wurde ein Least-Recently-Used (LRU) Cache Algorithmus verwendet. Wenn der Cache voll ist, löscht der Algorithmus das längste nicht verwendete Objekt. Da der JHelioviewer im allgemeinen Fall sequenziell nach den Datenobjekten abfragt, kann der LRU Cache mit einer First-in-First-out Queue implementiert werden. Das Objekt, welches am längsten nicht mehr gebraucht 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 ein 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 wäre.

### 5.1.2 Asynchrone Aufrufe mittels Executor Services

Im Diagramm der Abbildung 13 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

Es gibt  $x$  Lösungsansätze mit  $x$  Varianten. Hier sind die Kompressionsraten der jeweils besten Varianten der Ansätze.

Bis auf den Lösungsansatz unter 6.1 Unterschiedliche Qualitätsstufen Im Verlauf wurde eine neue Metrik entwickelt, deshalb gibt es manchmal zwei Plots. Einmal mit der Standardabweichung, und einmal mit PSNR-HVS-M. Standardabweichung ist der beste Ort unten Links, bei PSNR-HVS-M Plots Oben Links. Bis aus den Lösungsansatz 6.1 sind alle Ansätze mit unterschiedlichen Qualitätsstufen getestet.

### 6.1 Lösungsansatz: Adaptive 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-

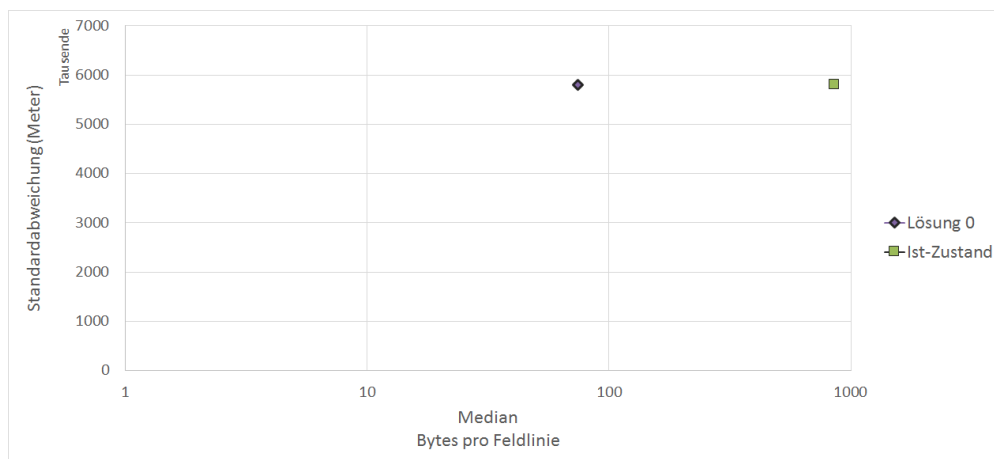


Abbildung 14: Vergleich des Lösungsansatzes: Adaptive Subsampling zur Ist-Kompression.

gramm 14 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 Komplexität dieses Ansatzes bleibt  $O(n)$  ( $n$  ist die Anzahl Punkte) und bleibt somit in der selben Komplexitätsklasse wie die Ist-Kompression. Dieser Lösungsansatz ist aber bei der Dekompression schneller, da  $n$  etwa vier mal kleiner ist. und ist somit Da bei der Dekompression  $n$  etwa vier Mal weniger Punkte bearbeiten muss, ist die Dekompression sogar schneller als die Ist-Lösung.

Die Abbildung 15 zeigt die Artefakte, die bei der Komprimierung der Lösung 0 entstehen. Es ist anzumerken, dass der Ist-Zustand die selben Artefakte aufweist.

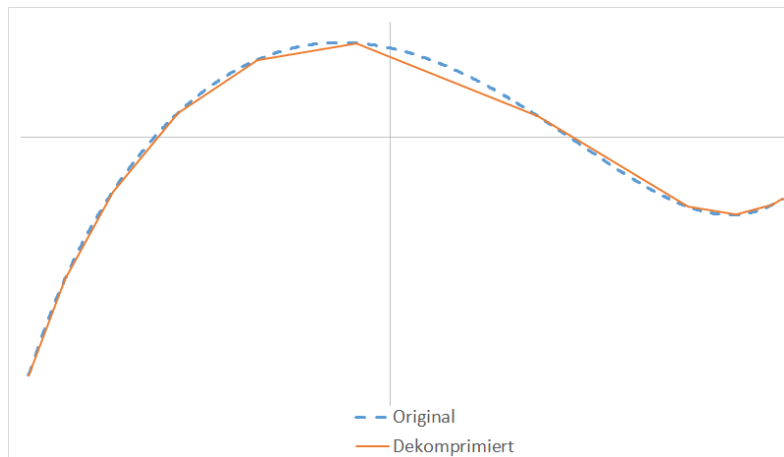


Abbildung 15: Artefakte der Lösung 0

## 6.2 Lösungsansatz: 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 erhöht. 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

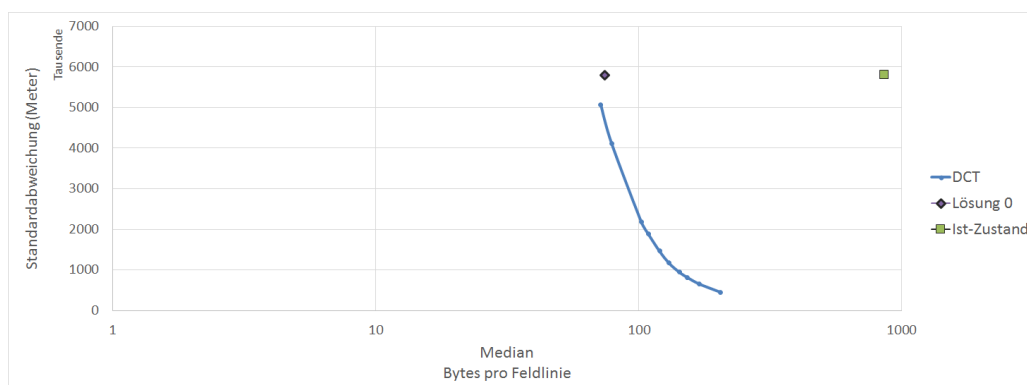


Abbildung 16: Vergleich der DCT Kompression mit der Lösung0

Abbildung 16 zeigt den Vergleich der DCT Kompression mit dem Lösungsansatz des Adaptiven Subsampling (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 17 entnommen werden. Un den meisten Fällen

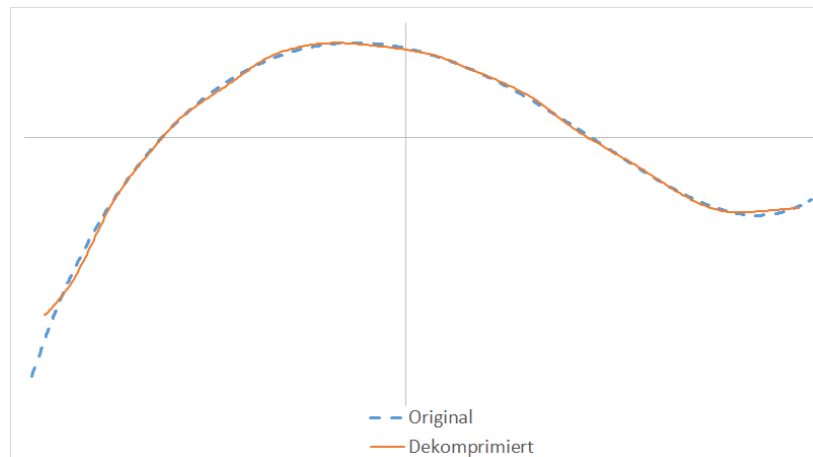


Abbildung 17: Artefakte der DCT Dekompression anhand Beispieldaten

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 implementierte 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 17.

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 Darstellung 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 17 gelöst werden. Der Nachteil ist, dass Ungenauigkeiten sich durch die Kurve durchziehen und summieren. Am Ende kann die Approximation ungenauer sein, wie am Anfang der Feldlinie.

Das Diagramm der Abbildung 18 zeigt, dass die abgeleiteten Feldlinien besser approximiert werden können

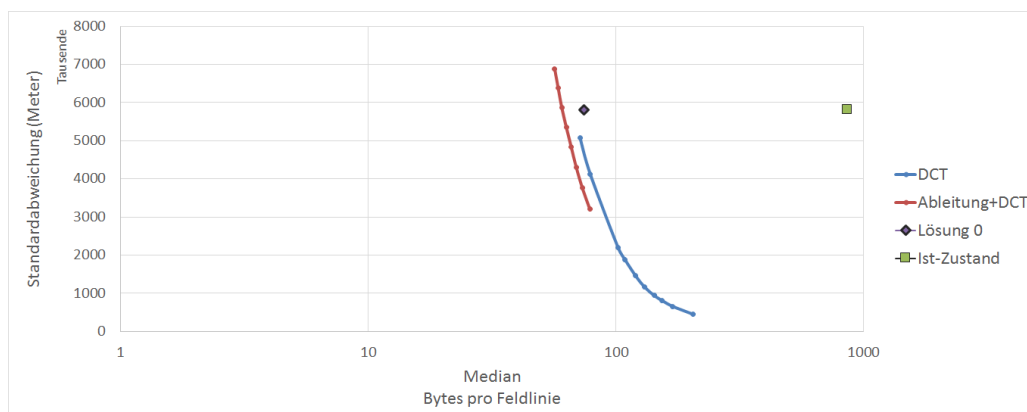


Abbildung 18: Vergleich der DCT Kompression der Ableitung mit der DCT Kompression

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 Abbildung 19 zu finden.



Es ist deutlich zu sehen, dass die Kurve durch die Quantisierung gedämpft wird. das Maximum der Kur-

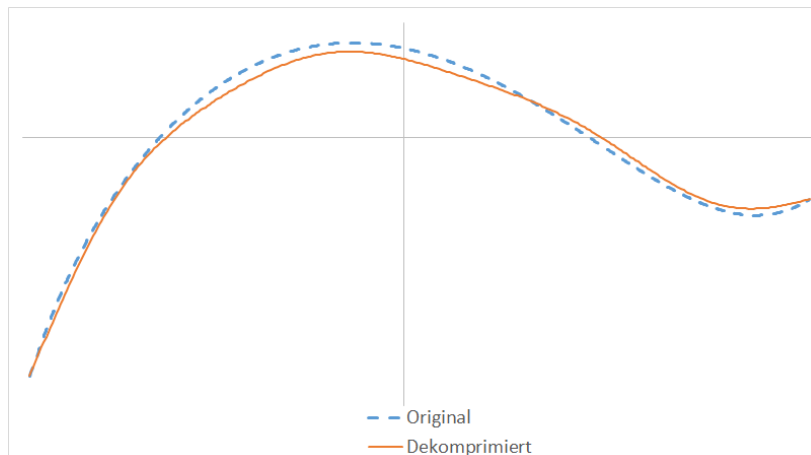


Abbildung 19: Artefakte der DCT Kompression der Ableitung

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 wären 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

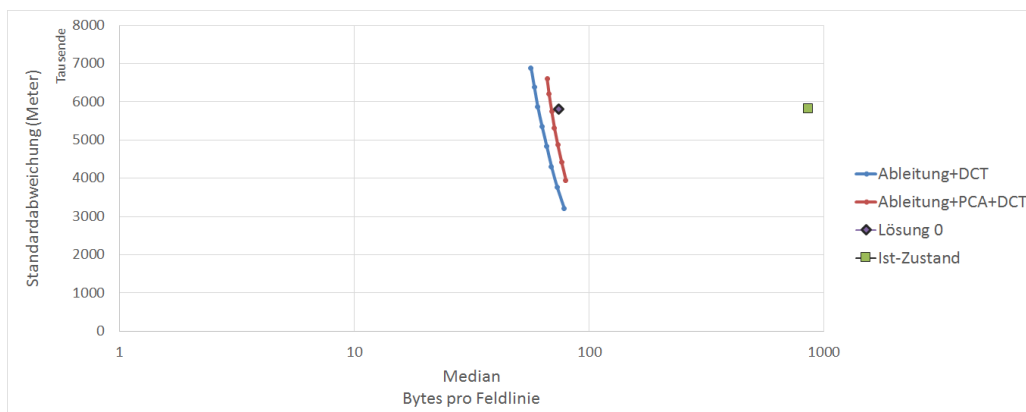


Abbildung 20: Vergleich der PCA DCT Kompression der Ableitung mit der DCT Kompression der Ableitung

verbessern zu können. Der Grund ist, dass die Variante ohne PCA zwischen 5 und 20 Kosinus-Funktionen braucht, um eine Feldlinie zu approximieren. Durch die PCA braucht es weniger zwar weniger Parameter,

jedoch verbrauchen die zusätzlichen Koordinatenachsen und Verschiebungen mehr Speicher, als die Transformation gewinnt.

#### 6.2.4 Variante: Ableitung+DCT+Byte Kodierung

Eine Feldlinie kann mit 5 bis 20 Koeffizienten approximiert werden. Alle anderen DCT-Koeffizienten sind 0. Mit einer Byte Kodierung soll nun zusätzlich Speicherplatz gespart werden. Die Kodierung ist im Abschnitt 3.3.5 beschrieben. Das Diagramm der Abbildung 21 zeigt eine deutliche Verbesserung mit der Kodierung. Bei einer

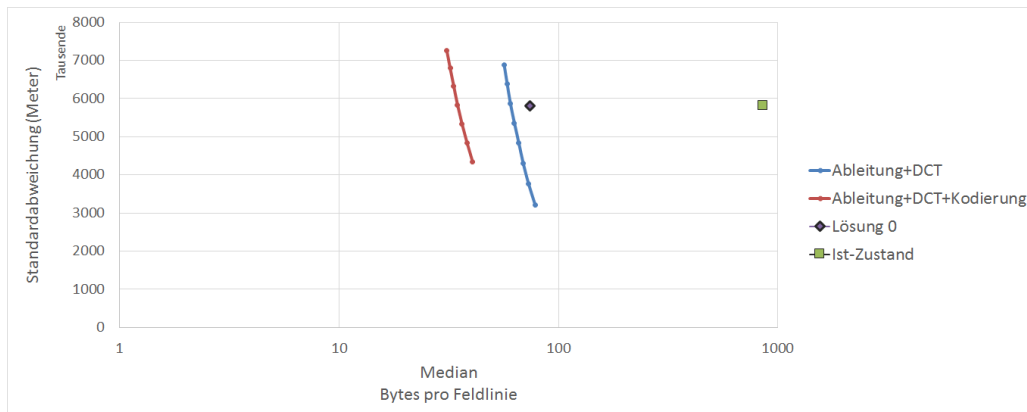


Abbildung 21: Vergleich der Kompression mit und ohne Byte-Kodierung

Vergleichbaren Genauigkeit weist diese Variante eine Kompressionsrate von 24.5 auf.

#### 6.2.5 Variante: Randbehandlung+DCT+Byte Kodierung

Wenn die Artefakte 19 und 15 vergleicht, fällt auf, dass die Variante 6.2.1 die Feldlinie genauer approximiert, falls die Ränder der Feldlinie besser dargestellt werden könnten. 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 22 zeigt den Vergleich

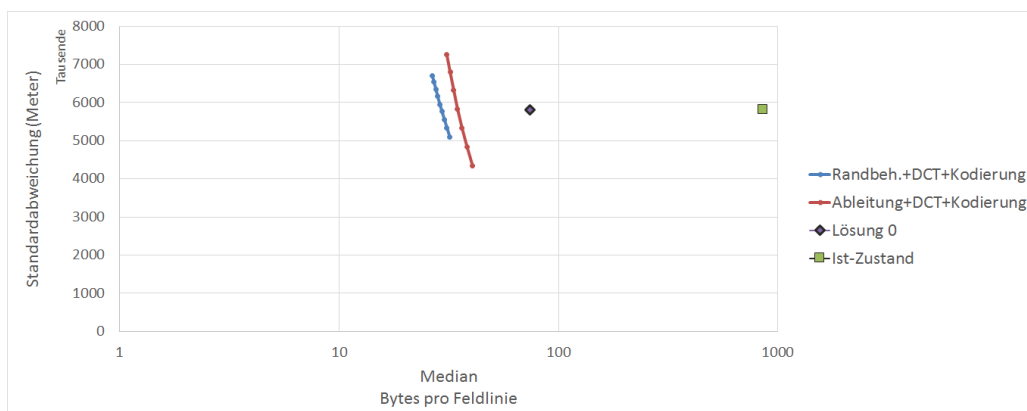


Abbildung 22: Vergleich der Kompression mit und ohne Byte-Kodierung

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

Obwohl die Variante ?? eine vergleichbare Genauigkeit aufweist, wie Ist-Kompression, sind auf der JHelioviewer Visualisierung deutliche Artefakte zu sehen. Die Abbildung 23 vergleicht die originalen mit dekomprimierten Feldlinien. Die Dekompression lässt die Feldlinien um das originale Signal schwingen. In diesem

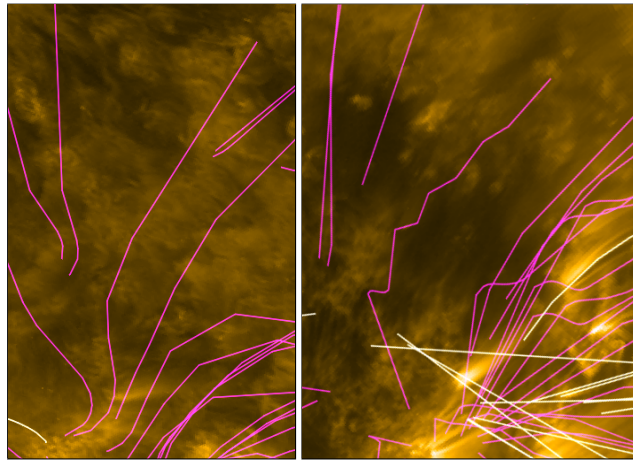


Abbildung 23: Artefakte der Kompression, links sind die originalen Feldlinien, rechts die Dekomprimierten.

Fall scheint die Standardabweichung als Fehlermass zu versagen: Da die Schwingungen nahe am originalen Signal liegen, bleiben die Abstände klein. Jedoch sind die Artefakte für das menschliche Auge inakzeptabel. Interessant ist, dass die Variante der Ableitung (Abschnitt 6.2.4) ähnliche Artefakte aufweist. Im Diagramm der Abbildung 19, welches die Artefakte anhand einer Beispiellinie zeigt, sind keine Schwingungen zu entdecken. In der JHelioviewer Visualisierung jedoch, sind auch bei dieser Variante deutliche Schwingungen zu erkennen. Die Abbildung 24 zeigt die Artefakte. Es ist anzumerken, dass die Artefakte weniger ausgeprägt sind, aber dennoch störend für das menschliche Auge.

Es wird vermutet, dass es sich um Ringing Artifacts [9] handelt. Es sind typische Artefakte einer DCT Kompression. Es können abrupte Steigungen im Inputsignal nicht gut approximiert werden. Die Approximation dieser abrupten Steigungen führt zu Oszillationen im Outputsignal. Dieser Effekt wird als Ringing Artefakte bezeichnet und sind typische 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. Diese verhalten sich oft monoton, mit teils abrupten Richtungswechsel in der Nähe der Sonnenoberfläche. Die Abbildung 25 zeigt ein Beispiel solcher Feldlinien. Den abrupten Anstieg gefolgt von einer monotonen Steigung kann die DCT nur durch viele hochfrequente Anteile darstellen. Wenn diese durch die Quantisierung an Genauigkeit verlieren, können oszillierende Artefakte entstehen.

## 6.2.7 Behandlung der Ringing Artefakte

Um eine optimale Kompression der Feldlinien zu erreichen, müssen die Ringing Artefakte behandelt werden. Im Abschnitt 3.3 wurde erwähnt, dass Reduktion von Ringing Artefakten ein aktives Forschungsfeld ist. Es existiert keine allgemeingültige Lösung um Ringing Artefakt zu dämpfen oder gar zu vermeiden. Abschnitt 6.2.6

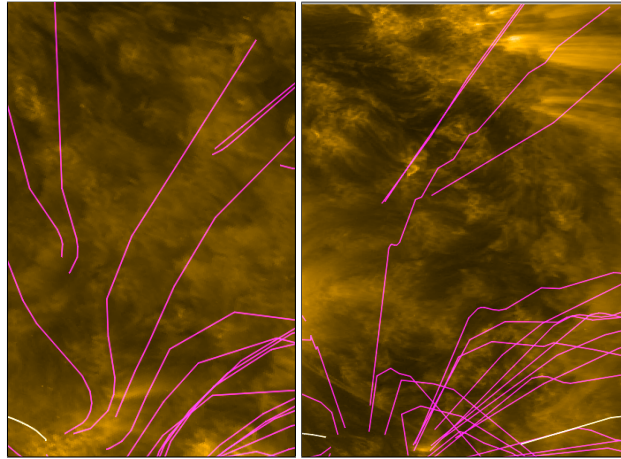


Abbildung 24: Artefakte der Kompression, links sind die originalen Feldlinien, rechts die Dekomprimierten der Variante 6.2.4.

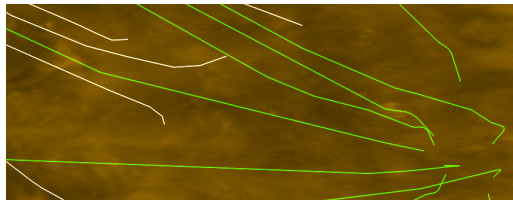


Abbildung 25: Abrupte Steigungen bei Feldlinien, welche von der Sonne ins Weltall führen.

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 6.2.5 jeweils mit und ohne PCA Transformation. Die Transformation wird hier nochmals geprüft. Die Transformation könnte die Ringing Artefakte jeweils auf einen Kanal beschränken.

Das Diagramm der Abbildung 26 zeigt, wie gut die Varianten die Feldlinien approximieren können, welche

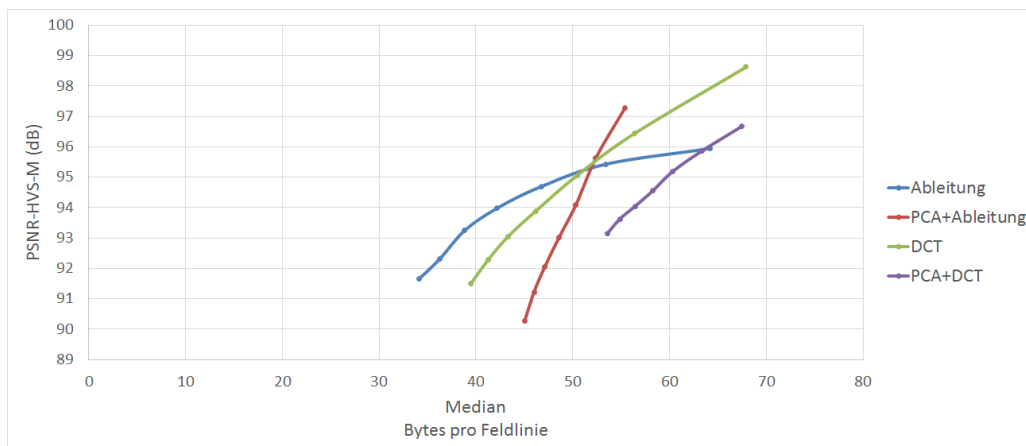


Abbildung 26: Approximation der Feldlinien "Sonnenoberfläche zu Sonnenoberfläche". Je höher die PSNR-HVS-M, desto besser ist die Approximation.

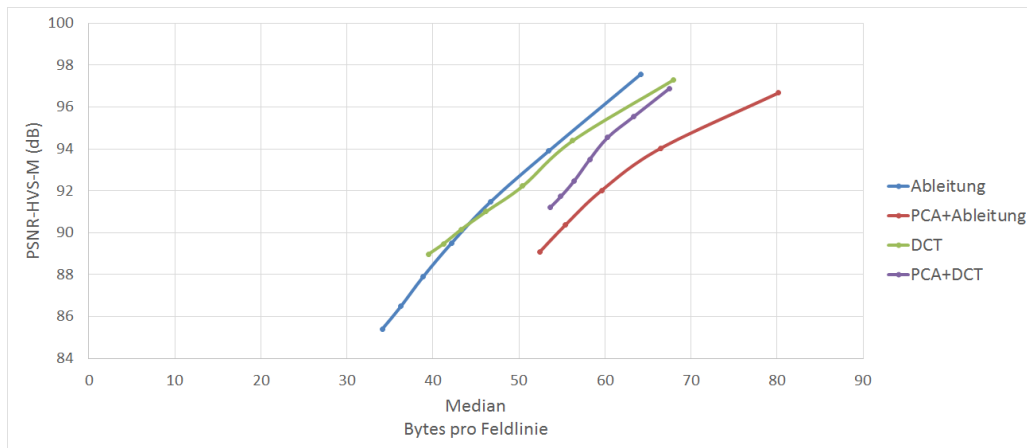


Abbildung 27: Approximation der der Feldlinien "Sonnenoberfläche ins Weltall" oder "Weltall zur Sonnenoberfläche". Je höher die PSNR-HVS-M, desto besser ist die Approximation.

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. 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 27 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 26. 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. spezielle Quantisierung. Das Diagramm der Abbildung 28 zeigt die Standardabweichung der abschliessenden DCT Variante. Zu einer besseren Kompression kann sie deutlich genauer Punkte ablegen und erreicht eine Kompressionsrate von 13.9. Die Artefakte sind mit einer PSNR-HVS-M von 94.1 ebenfalls in Grenzen gehalten. Jedoch sind wegen dem Zoom-Feature des JHelioviewers immer noch Oszillationen zu erkennen. Das Linke Bild der Abbildung 29 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 weniger 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

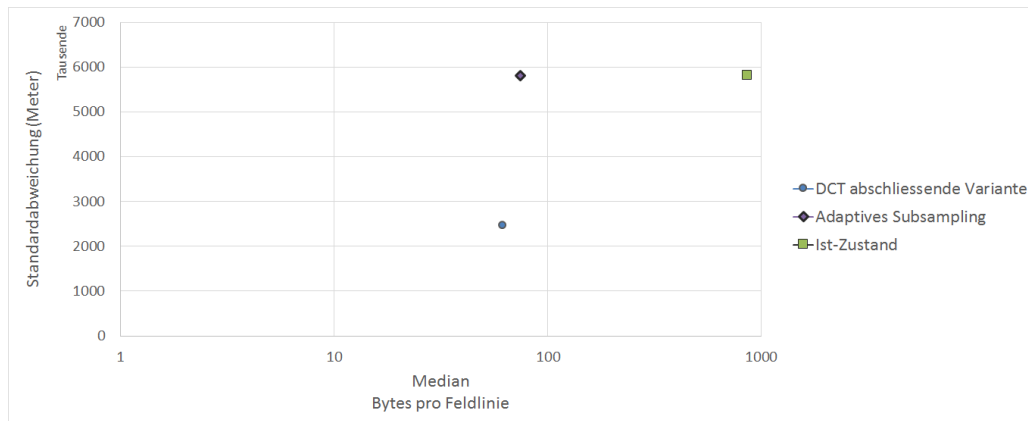


Abbildung 28: Standardabweichung der abschliessenden DCT Variante.

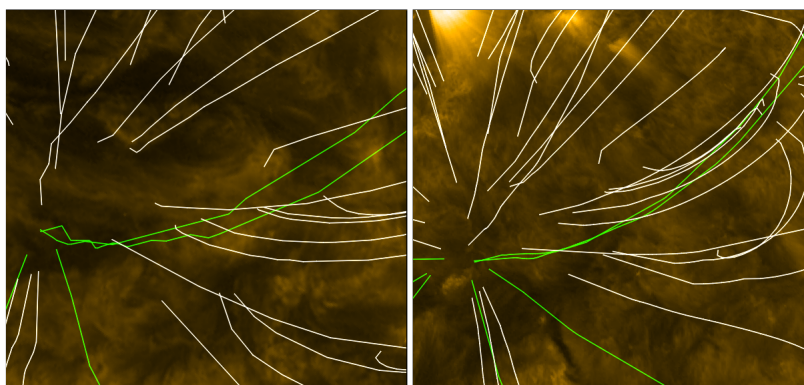


Abbildung 29: Artefakte der abschliessenden Variante. Links ohne Glättung, rechts mit Glättung

möglichst wenigen Artefakten zu approximieren. Eine Kompression mit der Diskreten Kosinus Transformation 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 29 verdeutlicht. Dadurch können die Artefakte auch bei höheren Zoomstufen verschleiert werden. Jedoch bedeutet das auch, dass die Feldlinie weiter verfälscht wird.

## 7 Diskussion

### 7.1 Diskussion Lösungsansatz Adaptive Subsampling

Der Lösungsansatz des adaptiven Subsamplings ist ein simples aber effektives Verfahren. Besonders die Dekompression hält sich simpel, was sich auch auf die Laufzeit auswirkt. Bei einem 17 Prozessor braucht die Dekompression durchschnittlich 19 Millisekunden. Ein Thread kann dadurch etwa 50 Simulationen pro Sekunde dekomprimieren.

Der Nachteil dieses Ansatzes liegt daran, dass nur wenige Punkte übertragen werden. Wenn der Benutzer hohe Zoomstufen einstellt, muss der Client selbst eine Interpolation durchführen, welche bitrate braucht es für ein movie. Den Nachteil kann man mit zb. durch clientseitige Interpolation lösen, was aber die Komplexität wieder erhöht. Bei einer durchschnittlichen Grösse von 90 Kibytes braucht es eine Leitung von etwa 14 Megabit für 20 Simulationen pro Sekunden.

Beste Lösung wenn der Client wenige Arbeitsspeicher und Rechenressourcen aufweist.

### 7.2 Diskussion Lösungsansatz DCT

Die DCT Kompression kann im Vergleich mit dem adaptiven Subsampling eine höhere Kompressionsrate erreichen, obwohl eine höhere Anzahl an Punkte übertragen werden. 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 Kosinusfunktion verbraucht. Bei der Implementation in dieser Arbeit werden die Funktionen über SoftReferences gecached. Der Cache passt sich somit an den Arbeitsspeicher an. Wie schnell die Dekompression ist, hängt im Wesentlichen vom freien Arbeitsspeicher ab. Im besten Fall ergibt das eine Laufzeit von 65 Millisekunden und im schlechtesten Fall 350. Das führt zu einem Durchsatz zwischen 3 und 15 Simulationen, pro Sekunde pro Thread.

Momentaner zustand 12 MegaBit

Die DCT Kompression kann Ringing Artefakte (siehe Abschnitt 6.2.6) einführen, welche vom menschlichen Auge als störend empfunden werden. Je stärker die Feldlinien komprimiert werden, desto deutlicher werden die Artefakte.

JHelioviewer bietet an, in die Feldlinien hineinzuzoomen. Der Benutzer kann auch kleine Artefakte zu Gesicht bekommen, während gewisse Artefakte bei kleinen zoomeinstellungen verschwinden würden. Die Frage ist also, zu welcher Zoomstufe man eine artefaktfreie Ansicht erhalten möchte. Die jetzige Implementation ist noch konservativ und enthält nur kleine Artefakte, welche erst bei grösseren Zoomstufen sichtbar werden.

Da mehr Punkte übertragen werden als der JHelioviewer überhaupt darstellen kann, wird für die Visualisierung eine Glättung vorgenommen. Die momentane Implementation enthält nur kleine Artefakte, welche erst bei höherem Zoom sichtbar werden. Zusammen mit der Glättung werden die Ringing Artefakte fast komplett verschleiert. Wäre man aber bereit mehr zu opfern, so könnte eine deutlich bessere Kompression erreicht werden.

Verbessertes Postprocessing, welches die Ringing Artefakte vermindert.

Welche Bitrate braucht es für movie. Prozessorleistung/Memoryverbrauch. Zoomstufen 3 sekunden 65 – 350 ms

Allgemein Zielerreichung



## 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 Burtcher. 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	Typische Teilschritte einer 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	Darstellung der Fehlerberechnung. Die Punkte sind die Originaldaten, die Quadrate sind die Punkte nach der Kompression. . . . .	12
10	Darstellung der Fehlerberechnung. Die Punkte sind die Originaldaten, die Quadrate sind die Punkte nach der Kompression. . . . .	13
11	Flussdiagramm der PSNR-HVS-M Berechnung [13]. . . . .	14
12	Zustandsdiagramm der Feldliniendaten . . . . .	16
13	Diagramm der Implementation vom Vorladen und Caching . . . . .	17
14	Vergleich des Lösungsansatzes: Adaptives Subsampling zur Ist-Kompression. . . . .	19
15	Artefakte der Lösung 0 . . . . .	20
16	Vergleich der DCT Kompression mit der Lösung0 . . . . .	20
17	Artefakte der DCT Dekompression anhand Beispieldaten . . . . .	21
18	Vergleich der DCT Kompression der Ableitung mit der DCT Kompression . . . . .	21
19	Artefakte der DCT Kompression der Ableitung . . . . .	22
20	Vergleich der PCA DCT Kompression der Ableitung mit der DCT Kompression der Ableitung . . . . .	22
21	Vergleich der Kompression mit und ohne Byte-Kodierung . . . . .	23
22	Vergleich der Kompression mit und ohne Byte-Kodierung . . . . .	23
23	Artefakte der Kompression, links sind die originalen Feldlinien, rechts die Dekomprimierten. . . . .	24
24	Artefakte der Kompression, links sind die originalen Feldlinien, rechts die Dekomprimierten der Variante 6.2.4. . . . .	25
25	Abrupte Steigungen bei Feldlinien, welche von der Sonne ins Weltall führen. . . . .	25
26	Approximation der Feldlinien "Sonnenoberfläche zu Sonnenoberfläche". Je höher die PSNR-HVS-M, desto besser ist die Approximation. . . . .	25
27	Approximation der der Feldlinien "Sonnenoberfläche ins Weltall" oder "Weltall zur Sonnenoberfläche". Je höher die PSNR-HVS-M, desto besser ist die Approximation. . . . .	26
28	Standardabweichung der abschliessenden DCT Variante. . . . .	27
29	Artefakte der abschliessenden Variante. Links ohne Glättung, rechts mit Glättung . . . . .	27

## 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. . . . .	15

## 9 Anhang

subsectionInstallationsanleitung

## 10 Ehrlichkeitserklärung