UNIVERSITÄT LEIPZIG
Fakultät für Mathematik und Informatik
Institut für Informatik

Kompression von Hyperspektraldaten und Übertragung in einem Kanal mit reduzierten Ressourcen

Diplomarbeit

Leipzig, Juni 2002

vorgelegt von
Rink, Karsten
geboren am: 02. Juni 1976
Studiengang: Informatik

Abstract

Die Auswertung von Hyperspektraldaten spielt heute in Wirtschaft und Forschung eine immer grössere Rolle. Die Anwendungsgebiete reichen von der Einschätzung von Sturm- und Brandschäden über die Bestimmung von Landnutzungsklassen bis hin zur Entdeckung von Mineralien und Bodenschätzen.

Aufgrund ihres Datenvolumens stellt die Übertragung von hyperspektralen Datensätzen jedoch ein nicht zu unterschätzendes Problem dar. Deshalb wird eine Kompression der Daten vorgenommen, um die zu übertragende Datenmenge zu reduzieren. Da bei der Anwendung verlustfreier Kompressionsverfahren im allgemeinen keine befriedigenden Kompressionsraten erzielt werden können, muss auf verlustbehaftete Kompressionsmethoden zurückgegriffen werden. Die Anwendung solcher Methoden ist in der Fernerkundung jedoch noch immer umstritten, da keine standardisierten Verfahren zur Bewertung des Informationsverlustes existieren.

Ziel dieser Arbeit ist es nun, ein geeignetes Verfahren vorzustellen, das die speziellen Eigenschaften von Hyperspektraldaten nutzt und es ermöglicht, bei der Kompression derartiger Daten hohe Kompressionsraten zu erzielen, ohne nennenswerte Qualitätsverluste in Kauf nehmen zu müssen. Somit wird eine Übertragung der Daten im Internet in kürzerer Zeit ohne Einbußen in Bezug auf die Anwendbarkeit der Daten möglich.

Der verwendete Algorithmus ermöglicht dem Anwender, die Qualität der zu übertragenden Daten durch eine Vielzahl an Parametern selbst zu bestimmen. Neben einer möglichen verlustfreien Übertragung lassen sich auch bei hohen Kompressionsraten qualitativ hochwertige Ergebnisse erzielen. Für die Anwendung in typischen Browserapplikationen werden bei einer Kompression von 60:1 noch akzeptable Ergebnisse erreicht.

Inhaltsverzeichnis

Abbildungsverzeichnis

Kapitel 1

Einleitung

In der Fernerkundung wird heutzutage mit Hilfe von Meßstationen, Satelliten und anderen Hilfsmitteln eine Vielzahl von Informationen über die Erde gesammelt. Mit ihrer Hilfe können Aussagen über eine Vielzahl von Eigenschaften und Phänomenen im Zielgebiet gemacht werden.

Dazu zählen beispielsweise die Klassifizierung der Landnutzung, Aussagen über die Vegitation und deren Zustand, Beobachtungen bezüglich Wasserverschmutzung oder auch die Analyse von Phänomenen wie beispielsweise *El Niño*.

Auf Grund ihrer breiten Anwendungspalette werden Fernerkundungsdaten bzw. die daraus gewonnenen Erkenntnisse heute in vielen Bereichen in Wissenschaft und Technologie genutzt.

Die grossen Fortschritte in computergestützter Bildverarbeitung und die Möglichkeit heutiger Computer grosse Datenmengen zu speichern und zu verarbeiten, führten dazu, dass geographische Informationssysteme (GIS) immer beliebter wurden. Mit ihrer Hilfe werden Informationen gesammelt, gespeichert, bearbeitet, verwaltet und analysiert. Eine Nutzung derartiger GIS ist sowohl in geowissenschaftlichen Fachbereichen (z.B. Klimatologie, Ökologie, Geologie) als auch in vielen anwendungsbezogenen Einsatzbereichen (z.B. Vermessung, Transport- und Planungswesen) möglich.

Einen ähnlichen Ansatz verfolgte auch *ATLAS2000*, ein Gemeinschaftsprojekt des Instituts für Physische Geographie der Universität Freiburg/Brsg. und dem Institut für Informatik der Universität Leipzig. Das Ziel war die Entwicklung des Prototypen eines web-basierten digitalen Atlas. Dafür wurden digitale Karten und Klimameßreihen in beispielhaften Modellen unterschiedlicher Komplexität für den Einsatz in Web-Atlanten aufbereitet.

Das Hauptaugenmerk der vorliegenden Arbeit, die noch im Rahmen von *ATLAS2000* anzusiedeln ist, liegt nun auf der Kompression und Übertragung von Hyperspektraldaten.

Aufgrund des enormen Volumens derartiger Datensätze ist eine Übertragung der Daten oft sehr zeitaufwendig und kostenintensiv. Aber in Hinblick

auf die Vielzahl der Anwendungsgebiete von Hyperspektraldaten ist ein solcher Datentransfer oft notwendig, da Gewinnung und Auswertung der Daten in der Regel nicht am gleichen Ort stattfinden.

Deshalb sollen die Daten in komprimierter Form von einem zentralen Server an den Benutzer übertragen werden. Die dabei verwendeteten Kompressionsverfahren sind teilweise verlustbehaftet, um ein hohes Maß an Datenreduktion zu gewährleisten. Die Kompressionsrate sowie die damit einhergehende Qualität der Daten soll aber vom Benutzer steuerbar sein, so dass die Daten stets in der vom Anwender gewünschten Qualität geliefert werden können.

Neben der Implementation eines geeigneten Verfahrens zur Kompression der Daten ist es Ziel dieser Arbeit, für ausgewählte Benutzergruppen[1] geeignete Parameter zu bestimmen, um einerseits eine gute Qualität der Daten nach der Übertragung zu gewährleisten und andererseits die Daten derart zu komprimieren, dass für eben diese Übertragung nur möglichst wenig Zeit benötigt wird.

Bevor nun das verwendeten Verfahren erläutert wird, soll zunächst eine Betrachtung der geographischen Grundlagen erfolgen. Dabei soll erklärt werden, wie Hyperspektraldaten aufgebaut sind, welche Anwendungsgebiete es dafür gibt und auf welche Weise die Daten typischerweise verarbeitet werden.

Es folgt eine Betrachtung heute verwendeter verlustfreier und verlustgehafteter Kompressionsverfahren und eine Beurteilung der Eignung dieser Verfahren in Hinblick auf die speziellen Eigenschaften von Hyperspektraldaten bzw. deren spätere Verwendung.

Weiterhin sollen bereits existierende Verfahren zur Kompression von Multi- und Hyperspektraldaten kurz vorgestellt werden, gefolgt von der Beschreibung des im praktischen Teil der Arbeit implementierten Algorithmus. Neben einer Beschreibung aller verwendeten Teilverfahren wird auch auf die Wahl der Programmiersprache sowie die Grundlagen der Client/Server-Architektur eingegangen.

Schliesslich folgt eine Betrachtung der für die einzelnen Verfahren relevanten Parameter und den Einfluss des Anwenders darauf. Die Auswirkungen dieser Parameter bei der Anwendung des Kompressionsalgorithmus auf die vorhandenen Testdatensätze sowie geeignete Einstellungen für bestimmte Benutzergruppen sollen abschliessend ebenfalls dargestellt werden.

[1] Als Benutzergruppen wurden hier sogenannte "interessierte Laien"und Experten gewählt. Die erste Gruppe ist dabei lediglich an einer Visualisierung der Daten interessiert (also etwa Studenten oder geowissenschaftlich interessierte Anwender), während die zweite Gruppe tatsächlich an einer Weiterverarbeitung der übertragenen Daten interessiert ist, um Eigenschaften des Zielgebietes zu bestimmen.

Kapitel 2

Geographische Grundlagen

2.1 Fernerkundung

Fernerkundung (*Remote Sensing*) beschäftigt sich, einfach ausgedrückt, mit dem Sammeln von Informationen über die Erdoberfläche, ohne jedoch unmittelbar damit in Kontrakt zu stehen. Stattdessen wird die von der Erde reflektierte oder emittierte Energie gemessen und die so gewonnen Informationen werden gespeichert, um die Daten später weiterverarbeiten und analysieren zu können [14].

Das "Remote Sensing Tutorial" der NASA gibt dazu eine weitaus umfangreichere Definition. Hier wird der Begriff Fernerkundung wie folgt definiert:

Definition 1 *The acquisition and measurement of data/information on some property(ies) of a phenomenon, object, or material by a recording device not in physical, intimate contact with the feature(s) under surveillance; techniques involve amassing knowledge pertinent to environments by measuring force fields, electromagnetic radiation, or acoustic energy employing cameras, radiometers and scanners, lasers, radio frequency receivers, radar systems, sonar, thermal devices, seismographs, magnetometers, gravimeters, scintillometers, and other instruments.*

Stark vereinfacht betrachtet, betreibt also jeder Mensch Fernerkundung, wenn er beispielsweise auf den Bildschirm seines Computers schaut. Die von dort ausgesandte Strahlung wird mit den Augen wahrgenommen und das Gesehene wird im Gehirn weiterverarbeitet. Allerdings kann das menschliche Augen nur einen vergleichsweise kleinen Anteil des elektromagnetischen Spektrums wahrnehmen und unser Gehirn "generiert" nun aus diesen Information Echtfarbenbilder. Wie sich zeigen wird, ist dieser Arbeitsablauf der Verarbeitung von Hyperspektraldaten nicht unähnlich. Natürlich wird in der Fernerkundung ein ungleich grösserer Teil des elektromagnetischen

Kanal	Wellenlänge in μm	Informationsgehalt, Anwendung
1	0.45 - 0.52 (blue)	Küstenwasserkartierung, Trennung von Boden/Vegitation, Trennung von Laubwald/Nadelwald, Reflektion durch Chlorophyll
2	0.52 - 0.60 (green)	Reflektion durch Chlorophyll
3	0.63 - 0.69 (red)	Trennung von Vegitation / Nichtvegitation, Vegitationsunterschiede, Chlorophyll-Absorption
4	0.76 - 0.90 (near IR)	Identifikation von Pflanzen- und Vegitationsarten, Wasserkörper, Bodenfeuchtigkeit
5	1.55 - 1.75 (short wave IR)	Wassergehalt von Vegitation, Trennung von Schnee/Wolken
6	10.4 - 12.5 (thermal IR)	Thermale Kartierung, Wärmestress bei Pflanzen
7	2.08 - 2.35 (short wave IR)	Hydrothermale Kartierung, Unterscheidung von Mineralien- und Gesteinsarten

Tabelle 2.1: *Die Spektralkanäle des Landsat-TM Sensors und die darin enthaltenen Informationen [14]. Die Abkürzung IR bezeichnet den infroraten Spektralbereich.*

Spektrums in einer viel feineren Auflösung gescannt und auch bei der Weiterverarbeitung der Daten gibt es eine Vielzahl von Möglichkeiten, mit der sich eine überraschende Menge an Informationen gewinnen lässt.

Derartige Verfahren zur Gewinnung und Weiterverarbeitung von hyperspektralen Daten sollen nun im folgenden näher erläutert werden.

2.2 Hyperspektraldaten

Seit Apollo 9 im Jahr 1968 mit einer Anordnung von vier Kameras die ersten Multispektralbilder der Erde vom All aus machte, hat die Entwicklung von Sensoren zur Gewinnung derartiger Daten grosse Fortschritte gemacht.

Bereits im Juli 1972 wurde mit ERTS-1 der erste Landsat-Satellit ins All geschickt. Landsat1 war bereits mit einem Multispektral Scanner ausgestattet und war damit der erste Satellit, der eigens zur Gewinnung von Multispektraldaten gestartet worden war. Andere Satelliten folgten in den darauffolgenden Jahren und die Sensoren zur Datengewinnung wurden immer weiter verbessert und die gewonnen Daten für immer mehr Bereiche in Wirtschaft und Wissenschaft interessant.

Heute ist die Beschaffung und Auswertung von Multi- und Hyperspekt-

raldaten ein wichtiges Teilgebiet der Fernerkundung. Multikanal-Datensätze[1] werden heute bereits in vielen Bereichen der Wirtschaft und Forschung genutzt.

Um die weiteren Ausführungen besser verstehen zu können, soll nun erklärt werden, was Hyperspektraldaten eigentlich sind: Mit Hilfe spezieller Sensoren wird eine Vielzahl Bilder eines bestimmten Gebietes auf der Erdoberfläche in mehreren Spektralbereichen simultan aufgenommen. Dies geschieht überlicherweise von Flugzeugen, Space Shuttles oder von Satelliten aus. Die bekanntesten Quellen für Multispektraldaten sind dabei wahrscheinlich die Landsat Satelliten der NATO, die seit 1972 die Erde umkreisen und Aufnahmen von verschiedensten Teilen der Oberfläche unseres Planeten machen (Eine Fülle von Beispielen hierzu findet sich unter http://landsat.gsfc.nasa.gov/).

Bei diesen Aufnahmen sind neben dem sichtbaren Spektrum des Lichts (ca. 0.4 μm - 0.7 μm) vor allem auch der ultraviolette sowie der reflektierte Infrarotbereich (0.7 μm - 3.0 μm) und der thermale Infrarotberich (3.0 μm - 100 μm) des elektromagnetischen Spektrums interessant.

Multispektrale Bilder sind also dreidimensionale Datensets, bei denen die ersten beiden Dimensionen eine räumliche Ausdehnung darstellen (nämlich den aufgenommenen Teil der Erdoberfläche), während die dritte Dimension reflektierte Strahlung eines Zielobjektes in verschiedenen Spektralbereichen beschreibt.

Als einfaches Beispiel kann man sich ein RGB Bild vorstellen, dass in drei Ebenen aufgespalten werden kann. Diese drei Ebenen sind der rote, grüne und blaue Farbkanal, die übereinandergelegt wieder das Originalbild ergeben. Das gleiche Prinzip lässt sich auch auf multispektrale Fernerkundungsdaten (z.B. LandSat TM Datensätze) anwenden. Auch hier gibt es einen roten, grünen und blauen Farbkanal, jedoch existieren neben diesen noch weitere Kanäle, die Bilder von anderen Spektralbereichen aufnehmen. Die einzelnen Kanäle dieser Datensätze werden überlicherweise als Bänder bezeichnet.

So bestehen beispielsweise Landsat Multispektral Scans[2] (MSS) aus vier Bändern, Landsat Thematic Mapper[3] (TM) Daten aus sieben Bändern und

[1] Mit dem Begriff Multikanal-Daten sind im Rahmen dieser Arbeit sowohl Multispektraldaten als auch Hyperspektraldaten gemeint.

[2] Der Multispectral Scanner war der Hauptsensor von Landsat 1, 2 und 3. Beim MSS werden sechs Scan-Linien (*scan lines*) durch einen oszilierenden Spiegel simultan durchgewechselt. Jede Scan-Linie deckt dabei einen Bereich von 79 m der Erdoberfläche ab, d.h. alle sechs Linien korrespondieren zu 474 m der Oberfläche. Die Breite der Aufnahmen beträgt 185 km. Daten können beim MSS jedoch nur in einer Scanrichtung gewonnen werden. Für die vier Bänder bei Landsat 1 wurden 24 Signal-Dedektoren benötigt. Diese Dedektoren werden mit der vom oszilierenden Spiegel reflektierten Strahlung beleuchtet und produzieren ein kontinuierlich variierendes elektrisches Signal, das mit der Energie, die entlang der 79 m breiten zugehoerigen Linie empfangen wird, korrespondiert. [16]

[3] Der Thematic Mapper ist ein gegenüber dem MSS weiterentwickelter Sensor mit ver-

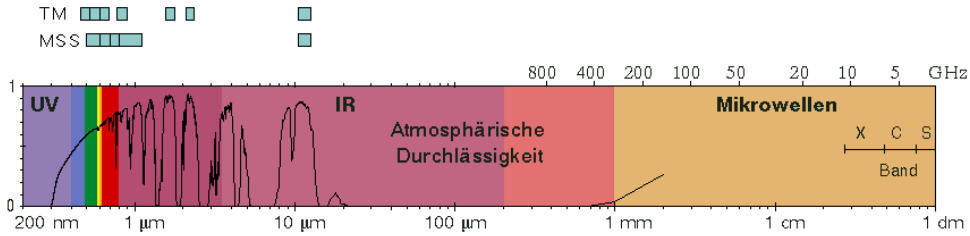


Abbildung 2.1: *Das elektromagnetische Spektrum des Lichts. Mit dem menschlichen Auge ist lediglich der Bereich von 0.4 µm - 0.7 µm (blau bis rot) sichtbar. Über dem Diagram sind die vom Landsat Multispectral Scanner (MSS) bzw. Landsat Thematic Mapper (TM) aufgenommenen Bereiche markiert. [20]*

aircraft scanner Daten aus ca. zwölf Bändern. Bei allen diesen Daten spricht man von Multispektraldaten.

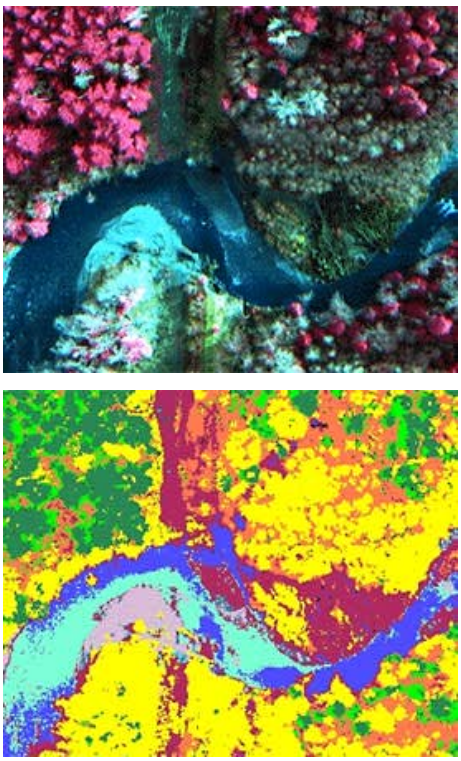
Die Datensätze, die dieser Arbeit zugrunde liegen, bestehen hingegen aus 79 (DAIS[4]) bzw. sogar 128 Bändern (HyMap[5]). Der im Dezember 2000 von der NASA gestartete EO-1 - Satellit verfügt sogar über mehr als 200 Kanäle [20]. Hier spricht man nun nicht mehr von Multispektral- sondern von Hyperspektraldaten.

Hyperspektraldaten sind für die meisten Anwendungen aufgrund ihrer spektralen Auflösung den Multispektraldaten überlegen. Die Aufnahmen sind wesentlich detailierter und man erhält feine spektrale Strukturen über einem grossen Wellenbereich. Der Spektralbereich ist bei Multi- und Hyperspektralaufnahmen dabei fast identisch: Landsat TM Datensätze bestehen aus Bändern mit Wellenlängen von 0.42 μm - 2.35 μm (hinzu kommt das Thermalband mit einer Wellenlänge von 10.4 - 12.5 μm, das hier einmal vernachlässigt werden soll, da es als einziges Band auch eine völlig andere Auflösung hat), die Wellenlängen von Hyperspektral-Datensätzen reichen von 0.38 - 2.5 μm (bei dem in dieser Arbeit verwendeten HyMap-Datensatz $0.42\mu m$ - 2.48 μm). Während Hyperspektraldaten nahezu kontinuierliche Signale liefern (der durchschnittliche Abstand zwischen zwei Bändern eines Hyperspektral-Datensatzes beträgt im Durchschnitt ca. $10nm = 0.01\mu m$), werden bei multispektralen Aufnahmen relativ grosse Intervalle von Wellenlängen durch einen einzigen Wert repräsentiert. Aufgrund der erheblich

besserten spektralen, räumlichen und radiometrischen Eigenschaften. Hier arbeiten 16 Scan-Linien simultan über einen Bereich von 480 m. Ausserdem können Daten hier in beiden Scanrichtungen gewonnen werden. Der Thematic Mapper verfügt über eine geringere *mirror scan rate* als der MSS, d.h. die Verweilzeit auf einem bestimmten Punkt ist länger, wodurch eine höhere räumliche Auflösung sowie eine verbesserte dynamische Reichweite ermöglicht wird. [16]

[4]Digital Airborne Imaging Spectrometer

[5]Hyperspectral Mapper



Dunkelgrün:	Nadelbäume
Grün:	*lower branches*
Violett:	Kies
Gelb:	Laubbäume
Orange:	Trockener Boden
Rot:	Feuchter Boden
Hellblau:	Wasser
Dunkelblau:	Tiefes Wasser

Abbildung 2.2: *Beispiel für die Nutzung von Hyperspektraldaten. Diese Aufnahme von 1995 zeigt einen Nadelwald auf* Vancouver Island. *Bild a zeigt eine Falschfarbendarstellung, bei der die Douglas-Tannen bereits deutlich durch ihre violette Farbe auffallen. Bild b stellt eine Stammzählung dar, mit deren Hilfe man sich ein recht genaues Bild von der tatsächlichen Anzahl der Tannen machen kann. Auf Bild c sieht man schliesslich Klassifizierung der Bodenbedeckung. Die genaue Bedeutung der Farben ist neben dem Bild aufgeführt. [14]*

besseren spektralen Auflösung sind die Möglichkeiten zur Klassifizierung bzw. Identifizierung bei Hyperspektraldaten gegenüber multispektralen Daten stark verbessert.

Die Ziele der Auswertung solcher Daten sind dabei natürlich grundsätzlich ähnlich: Aufgrund der Aufnahmen können Aussagen über bestimmte Eigenschaften des Zielgebietes gemacht werden, wie sie mit anderen Mitteln, beispielsweise mit Satellitenfotos, nicht möglich wären. Die Anwendungsmöglichkeiten sind mannigfaltig und reichen von der Bestimmung der Pflanzenarten über die Entdeckung von Bodenschätzen bis hin zur Einschätzung von Sturmschäden. Viele dieser Informationen kann man jedoch nur bei der Betrachtung bestimmter ausgewählter Bänder bzw. durch die Kombination bestimmter Bänder erhalten. Und hier ist natürlich die feine spektrale Auflösung von Hyperspektraldaten ein unschätzbarer Vorteil.

Bei der Bestimmung von Bodenschätzen[6] haben bespielsweise einige Materialien von Interesse Absorptionseigenschaften, die nur in einem Bereich von 20 - 40 nm Breite erkennbar sind [20]. Dieser Bereich ist so schmal, dass diese Eigenschaften bei einem grösseren Intervall zwischen den Spektralkanälen nicht mehr erkennbar wären. Besonders interessant ist auch die Kombination verschiedener Bänder. Offensichtlich kann man Bänder, die Aufnahmen des roten, grünen und blauen Wellenbereichs des Lichts darstellen (z.B. die Landsat TM Bänder 3, 2, 1), zu Echtfarbenbildern des aufgenommenen Gebietes kombinieren. Daneben sind aber insbesondere sogenannte Falschfarbenbilder, bei denen etwa Infrarotbilder für Farbkanäle genutzt werden, sehr aufschlussreich, da man bei diesen Bildern bestimmte Klassifikationen vornehmen kann, die auf Echtfarbenbildern oder einzelnen Bändern nicht sichtbar werden.

2.3 Weiterverarbeitung

Bevor geeignete Kompressionsmöglichkeiten für Hyperspektraldaten diskutiert werden, soll noch kurz auf die Auswertung von Multikanal-Daten eingegangen werden. Ferner soll kurz erläutert werden, auf welche Art und Weise die Daten später weiterverarbeitet werden und welche Eigenschaften bei der Auswertung speziell genutzt werden.

Wenn Fernerkundungsdaten durch Computer interpretiert werden, spricht man im allgemeinen von einer “quantitativen Analyse“. Systeme, die solche quantitativen Analysetechniken anwenden, reichen von Software Paketen bis hin zu vollständig interaktiven auf Bildverarbeitung spezialisierten Computern [16].

Bei der Weiterverarbeitug werden die gemessenen Daten in aufeinander aufbauende Verarbeitungsstufen unterteilt. Diese Einteilung ist unabhängig vom eingesetzten Sensor und sind international einheitlich. Im einzelnen beeinhalten die Level folgende Daten [7]:

- *Level 0*: Die Rohdaten, die vom Satelliten zur Empfangsstation übertragen werden. Diese Rohdaten enthalten neben den wissenschaftlichen Daten auch Informationen über den Zustand der Instrumente bzw. des Satelliten.

- *Level 1a*: Level 0 Daten werden in einem geeigneten Format abgespeichert und von den Zusatzinformationen getrennt.[7]

[6] Bespielsweise Sulfate, eisenhaltige Mineralien und Oxide

[7] Beim DAIS-Sensor werden die Daten wie folgt aufgeteilt [9]:
1. Die Spektraldaten werden als Binärdaten in einem BSQ-File gespeichert.
2. Es wird ein ASCII-Header im HDR-Format erstellt, der Informationen über den Datensatz und die einzelnen Bänder enthält.
3. Die zusätzlichen Daten werden im XDR-Format (*External Data Representation*) abgespeichert.

- *Level 1b*: Hier werden die Messdaten des Sensors in physikalische Werte umgerechnet, die sowohl spektral als auch radiometrisch kalibriert sind. Diese sogenannten "kalibrierten Strahldichten" werden nun georeferenziert, d.h. einer geographischen Position auf der Erde zugeordnet.

- *Level 2*: Level 2 Daten sind abgeleitete geophysikalische Variablen (wie z.B. Temperatur oder Spurengaskonzentration) mit der gleichen Auflösung und Ortszugehörigkeit wie die Level 1b Quelldaten.

- *Level 3*: Level 3 Daten sind kartierte Level 2 Daten, die auf ein einheitliches räumliches und zeitliches Gitter projiziert wurden. Derartige Daten können beispielsweise wöchentlich oder monatlich gemittelte globale Karten von Spurengaskonzentrationen sein.

- *Level 4*: Durch aufwendige Analysen niedrigerer Datenlevel werden Level 4 Daten gewonnen. Sie enthalten Aussagen, die nicht direkt vom Sensor gemessen werden können. Solche Daten sind üblicherweise Ergebnisse von Modellen und entstehen nicht in der Routine-Verarbeitung sondern als wissenschaftliche Ergebnisse.

Nachdem also die Bilder in der Vorverarbeitung geometrischen und radiometrischen Korrekturen unterzogen wurden, wie zum Beispiel das Herausfiltern atmosphärischer Störungen, werden die Daten in den höheren Levels zur Dedektion bestimmter Eigenschaften bzw. zur Extraktion von Informationen verändert . Als Beispiele sollen an dieser Stelle zwei Verfahren, *contrast stretching* und *spatial filtering*, kurz näher erläutert werden.

Contrast stretching ist eine beliebte Methode um den Kontrast in Bildern zu verstärken und Details besser sichtbar zu machen. Auf diese Weise können zum Beispiel einige Unterschiede zwischen Vegitation und Bodentypen sichtbar gemacht werden. Es gibt dafür mehrere Möglichkeiten, von denen hier aber nur eine näher erläutert werden soll. Beim *linear contrast stretch* werden der minimale und der maximale Helligkeitswert aus dem Histogramm des Bildes ermittelt. In Abbildung 2.3 werden im Original a) beispielsweise nur die Helligkeitswerte 84 bis 153 tatsächlich benutzt. Bei einem *linear contrast stretch* werden diese 70 Werte nun gleichmässig auf den gesamten Bereich von 0 bis 255 verteilt. Das Ergebnis ist Bild b) [16].

Weitere Möglichkeiten sind eine logarithmische oder exponentielle Anpassung des Kontrastes, um nur bestimmte dunkle bzw. helle Regionen des Bildes zu verstärken. Ferner besteht die Möglichekeit, nur einen Ausschnitt des Histogramms zu strecen. Hat man beispielsweise eine Aufnahme der Erdoberfläche, durch die ein Fluss verläuft, so kann man die Helligkeitswerte, die dem Wasser zugewiesen sind, auf den gesamten Grauwertbereich von 0 bis 255 strecken. Durch diese Modifikation werden die Details des Flusses

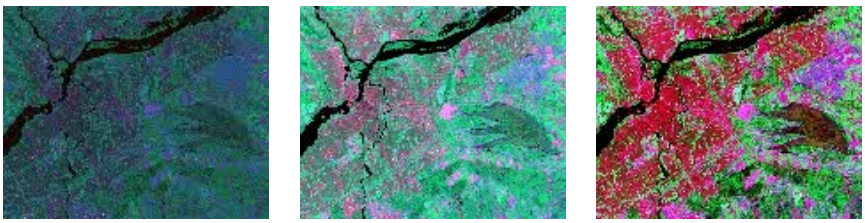


Abbildung 2.3: *Anwendung von* contrast stretching *und* spatial filtering. *Bild a zeigt das Orginal. Nach einem* linear contrast stretch *entsteht Bild b mit verbessertem Kontrast. Wir darauf ein Filter zur Sichtbarmachung bestimmter Effekte angewendet, erhält man als Ergebnis Bild c [14]*

in der Aufnahme deutlich besser sichtbar. Alle anderen Farbwerte wurden jedoch auf 0 bzw. 255 gesetzt und sind somit verloren.

Beim *spatial filtering* soll ebenfalls der visuelle Eindruck des Bildes verbessert werden. Hierzu wurden Filter entwickelt, die die Pixel eines Bildes unter Berücksichtigung ihrer Nachbarpixel entsprechend bestimmter mathematischer Berechnungen verändern. Dabei können die Nachbarpixel unterschiedlich gewichtet sein, so dass durch verschiedene Filter bestimmte Eigenschaften verstärken oder unterdrücken können. Das können vergleichsweise einfache Filter sein, beispielsweise kantenverstärkende Filter oder helligkeitsverändernde Filter, es können aber auch kompliziertere Verfahren angewendet werden, wie z.B. eine Fourier Analyse. Bild c) in Abbildung 2.3 wurde einem *spacial filtering* unterzogen [20].

Eine Vielzahl anderer Möglichkeiten zur Gewinnung von Informationen beruhen auf der Kombination mehrerer Bänder, die durch verschiedenste Verfahren zu einem Bild kombiniert werden. Dazu werden im beispielsweise Bänder bestimmten Farben zugeordnet (Falschfarben-Bilder, etc.) oder verschiedene Bänder mittels arithmetischer Operationen wie Addition oder Subtraktion miteinander verknüpft.

Beispiel 1 *Gesunde Vegitation wird im Nahinfrarot-Bereich (NIR) des Lichtes stark reflektiert, wohingegen das sichtbare rote Licht stark absorbiert wird. Andere Oberflächen, wie etwa Erde oder Wasser, zeigen in diesen Spektralbereichen des Lichts annähernd gleiche Reflektionseigenschaften. Nimmt man nun Band 7 des Landsat Multispektral Scanners (NIR, 0.8 µm - 1.1 µm) und teilt es durch Band 5 (Rot, 0.6 µm - 0.7 µm), so erhält man Werte grösser als 1.0 für Vegitation und Werte kleiner als 1.0 für Erde und Wasser. Die Möglichkeiten zur Unterscheidung von Vegitation und Wasser sind also auf dem Ergebnisbild stark verbessert. Ferner kann man sogar kranke oder gestresste Vegitation besser erkennen, da diese schlechtere Reflektionseigenschaften im Nahinfrarot-Bereich hat und damit die entsprechenden Werte nach der Division kleiner sind als für gesunde Vegitation [14].*

Eine weitere Möglichkeit zur Weiterverarbeitung ist schliesslich die Kombination verschiedener Sensordaten. Dazu werden etwa Hyperspektraldaten und Radardaten miteinander kombiniert. Als Ergebnis erhält man wieder ein neues Bild, bei dem zum einen die spektrale Information (z.B. Landnutzung), als auch die Oberflächenstrukturen gut sichtbar sind.

Nachdem nun Gewinnung und Verwendungsmöglichkeiten von Hyperspektraldaten aufgezeigt wurden, stellt sich die Frage wie sich hyperspektrale Datensätze effizient übertragen lassen. Sowohl aufgrund der Grösse des dargestellten Gebietes der Erdoberfläche bei einer räumlichen Auflösung von nur wenigen Metern pro Pixel als auch aufgrund der grossen Anzahl an spektralen Kanälen erreichen hyperspektrale Datensätze Grössenordnungen von mehreren hundert Megabyte. Um nun eine möglichst schnnelle Übertragung in einem Netzwerk gewährleisten zu können, lässt sich eine Kompression der Daten nicht vermeiden. Die dabei auftretenden Probleme sowie einige Grundlagen zur Datenkompression sollen im nächsten Kapitel diskutiert werden.

Kapitel 3

Kompressionsgrundlagen

3.1 Motivation

Durch ihre breiten Anwendungsmöglichkeiten können Hyperspektraldaten in sehr vielen unterschiedlichen Bereichen in der Wissenschaft und Wirtschaft genutzt werden. Die hyperspektrale Fernerkundung wird in der Geologie bereits heute kommerziell genutzt. Explorationsfirmen wie Texaco und Anglo American setzen zunehmend zur Erdölerkundung bzw. bei der Suche nach neuen Diamant- und Goldvorkommen auf Fernerkundungsdaten [5]. Insbesondere in den Vereinigten Staaten haben mehrere grosse Unternehmen bereits eigene Satelliten im Orbit, von denen viele mit multi- oder hyperspektralen Sensoren ausgestattet sind [20].

Allein für das Jahr 2000 war der Start von vier Satelliten zur Gewinnung von Hyperspektraldaten geplant. Daran zeigt sich die rasante Entwicklung neuer Sensoren zur Deckung des Bedarfs an derartigen Daten. Bis vor wenigen Jahren konnten hyperspektrale Daten nur von Flugzeugen aus aufgenommen werden[1]. Mit den nun im Erdorbit vorhandenen Satelliten ist die Gewinnung von Hyperspektraldaten wesentlich kompfortabler und einfacher geworden.

Durch die zunehmende Verwendung hyperspektraler Daten in Wissenschaft und Wirtschaft steigt natürlich auch der Aufwand zur Übertragung dieser Daten. Aufgrund der Grösse der Datensätze stellt das ein nicht zu unterschätzendes Problem dar.

Durch die in den letzten Jahren vorgenommenen Verbesserung der spektralen und räumlichen Auflösung von hyperspektralen Sensoren ist dieses Datenvolumen sogar noch gestiegen. So bestehen beispielsweise die Aufnahmen des Hyperion Sensors, Teil des im Dezember 2000 von der NASA gestarteten EO-1 Satelliten, aus 220 Bändern. Das aufgenomme Gebiet der Erdoberfläche ist 7,5 x 100 km bei einer Auflösung von 30 m/Pixel [20]. Der

[1] Die in dieser Arbeit verwendeten Testdatensätze sind beispielsweise solche von Flugzeugen aus aufgenommene Daten

Platzbedarf bei der Speicherung solcher Datensaätze ist folglich enorm.

3.2 Vorüberlegungen

Multikanal-Datensätze bestehen aus immer wieder dem gleichen Bild der Erdoberfläche, das lediglich unter verschiedenen Wellenlängen des elektromagnetischen Spektrums aufgenommen wurde. Verschiedene Bänder von Hyperspektraldatensätzen sind daher oft hochgradig korreliert und enthalten ähnliche Informationen. So sehen zum Beispiel die Landsat MSS Bänder 4 (grün) und 5 (rot) typischerweise sehr ähnlich aus, da Reflektionen von gleichen Oberflächen bei diesen Bändern fast identisch sind [14].

Es stellt sich nun also die Frage, wie man diese Eigenschaft von hyperspektralen Daten möglichst effizient zur Kompression nutzen kann. Offensichtlich ist es nicht sinnvoll zu versuchen, herkömmliche ein- oder zweidimensionale Kompressionsmethoden auf die dritte Dimension zu erweitern.

Ferner sollte man bedenken, dass die Datensätze für gewöhnlich nicht dafür bestimmt sind, von Menschen betrachtet zu werden, sondern als Input für eine Computeranalyse dienen sollen. Man kann also nicht die gleichen Masstäbe anlegen, die für die Kompression “gewöhnlicher“ Bilder gelten, da hier nicht die Minimierung des Speicherplatzes im Mittelpunkt steht, sondern die Relevanz eines Datensatzes, um ein bestimmtes Problem zu erklären. Es ist also möglich, dass bei einem visuell eventuell nicht wahrnehmbarer Informationsverlust, wie er von den meisten aktuellen verlustbehafteten Kompressionsverfahren angestrebt wird, wertvolle Informationen für eine Computeranalyse “wegkomprimiert“ werden.

Stattdessen sollten besser Bildtransformationsverfahren benutzt werden, die die statistischen Charakteristikien von Multiband Daten ausnutzen und die die Redundanz und Korrelation zwischen den Bändern reduzieren. Eine solche Transformation ist beispielsweise die Hauptkomponentenanalyse, zu der später noch eine ausführliche Erklärung folgt.

3.2.1 Datenredundanzen

Damit eine Kompression der Daten sinnvoll anwendbar ist, müssen Redundanzen zwischen den Daten vorhanden sein. In der Bildverarbeitung wird dabei im wesentlich zwischen drei Formen der Redundanz unterschieden: Coding Redundanz, Interpixel Redundanz, sowie Psychovisuelle Redundanz. Im folgenden werden diese drei Formen kurz erläutert [25]:

Coding Redundanz

Hier werden Redundanzen betrachtet, die bei der Abspeicherung der Daten im Code entstehen. Das beruht einerseits auf der Tatsache, das die Pixel eines Bildes nur eine bestimmte Anzahl von Farbwerten (Pixelintensitäten)

haben, andererseits entstehen solche Redundanzen auch durch verschiedene Formate beim Abspeichern der Daten. Je nach Format werden verschieden lange Codes für Farb- bzw. Bytewerte verwendet. Speichert man beispielsweise ein 8-bit Bild (8 Bits/Pixel = 256 Farben) in einem Format ab, das 16-bit Farbtiefe zulässt (16 Bits/Pixel = 65k Farben), so werden zum Abspeichern der gleichen Information jetzt doppelt soviele Bits/Pixel benötigt. Es ist folglich sinnvoll, stets das in Abhängigkeit von der Anzahl der Farben/Graustufen kleinste notwendige Dateiformat zu wählen.

So ist es zum Beispiel für Landsat-TM Aufnahmen (256 Grauwerte) sinnvoll, das Byte Binary Format (8 Bits/Pixel) zu verwenden.

Interpixel Redundanz

Interpixel Redundanz beschreibt die Korrelation benachbarter Pixel in einem Bild. Offensichtlich korrelieren Pixel in einem homogenen Bild ohne nennenswerte Kanten oder Texturen wesentlich stärker miteinander, als in einem sehr kontrastreichen Bild. Je ausgeprägter diese Korrelation ist, um so leichter können Pixelwerte anhand ihrer benachbarten Pixel vorausgesagt bzw. rekonstruiert werden. Diese Form der Redundanz wird insbesondere von Predictive Kodierern wie der Differential Pulse Code Modulation (DPCM) oder von Transformationskodierern, in denen die diskreten Kosinustransformation (DCT) oder die diskreten Wavelettransformation (DWT) implementiert ist, genutzt. Diese Form der Redundanz ist in Hinsicht auf Hyperspektralbilder besonders interessant, da hierzu aufgrund der geringen spektralen Abstände zwischen den Bändern eines Datensatzes auch die spektrale Redundanz zwischen benachbarten Bändern von Multi-Band-Daten zählt.

Psychovisuelle Redundanz

Psychovisuelle Redundanz bezieht sich auf die Tatsache, dass digitale Bilder oft in einer Auflösung vorliegen, die das menschliche Auge nicht auflösen kann. So ist es zum Beispiel nicht möglich, Grauwertsprünge im Nachkommastellenbereicht von reellwertigen Grauwertbildern zu unterscheiden. Diese Tatsache wird von allen verlustbehafteten Kompressionsverfahren genutzt, indem sie die hochfrequenten Strukturen in Bildern etwa durch Quantisierung unterdrücken.

Man sieht sofort, dass die Ausnutzung psychovisueller Redundanzen bei geowissenschaftlichen Bildern nicht sinnvoll ist, da einerseits zum Erkennen bestimmter Phänomene pixelgenaue Informationen notwendig sein können und da andererseits die Daten in der Regel nicht von Menschen ausgewertet werden und damit auch nicht den Beschränkungen des menschlichen Auges angepasst werden sollten.

3.3 Kompressionsgrundlagen

Im folgenden sollen nun kurz Grundlagen der Datenkompression erläutert werden, damit klar wird, welche prinzipiellen Möglichkeiten man hat und wo sich die Kompression von Hyperspektraldaten einordnen läßt.

3.3.1 Verlustfreie Kompression

Grundsätzlich hat man bei der Kompression von Daten zwei Möglichkeiten: Die Daten können verlustfrei (*lossless*) oder verlustbehaftet (*lossy*) komprimiert werden.

Obwohl die bei verlustfreier Kompression erzielten Kompressionsraten eher gering sind, ist die Anwendung insbesondere dann sehr sinnvoll, wenn später eine Computeranalyse der Daten vorgenommen werden soll, bei der natürlich alle Informationen möglichst orginalgetreu erhalten sein sollen.

Inbesondere bei medizinischen Daten ist es sowohl moralisch als auch rechtlich fragwürdig, ob ein Datenverlust tolerierbar ist.

Verlustfreie Kompression versucht, wie der Name schon sagt, das Datenvolumen zu verringern, jedoch ohne dabei irgendwelche Teile der Daten zu verlieren. Grundsätzlich versucht man daher, häufig vorkommende Symbole möglichst kurz zu kodieren, d.h. die *Coding Redundanz* der Daten zu verringern.

Die Möglichkeiten und Beschränkungen, die daraus resultieren, sollen nun kurz näher erläutert werden.

Im Jahr 1948 definierte Claude Shannon in *"A Mathematical Theory Of Communication"* den Informationsgehalt für ein Ereignis X als $log_a \frac{1}{P(X)}$. $P(X)$ ist dabei die Wahrscheinlichkeit für das Ereignis X und a die Basis des Logarithmus. D.h. für $a = 2$ ist die Einheit der Information Bits. Die *Entropie* einer Folge von Ereignissen $X^n = X_0, X_1, ..., X_{n-1}$ ist erklärt durch

$$H(S) = \lim_{n\to\infty} \sum P(X^n) log_2 \frac{1}{P(X^n)}$$

Shannon hat gezeigt, dass $H(S)$ Bits/Symbol die minimale durchschnittliche Rate ist, mit der der Output einer Quelle S encoded werden kann [19]. Sind die Output-Symbole X_i unabhängig voneinander, so reduziert sich die Entropie auf

$$H(S) = \sum P(X_i) log_2 \frac{1}{P(X_i)}$$

Daher besteht verlustfreie Kompression in der Regel aus zwei Schritten: Dekorrelation und Kodierung. Die Dekorellation der Daten ist dabei als *entropy reduction* zu verstehen, durch die Korellation zwischen den Daten entfernt wird. Die dekorellierten Daten können nun schliesslich noch

mit Hilfe eines Entropiekodierer komprimiert werden, wobei die Kompressionsrate bei den dekorrelierten Daten im allgemeinen grösser ist als bei den Originaldaten. Die Datenmenge nach der Kodierung ist also kleiner [17].

Bekannte Entropiekodierer sind beispielsweise Huffman Coder, Aritmetische Coder oder auch die Familie der Lempel-Ziv Coder (*gzip*, *compress*, etc.). Mittels Huffman Code kann beispielsweise eine minimale durchschnittliche Codelänge R von $H(S) < R < H(S) + 1$ erreicht werden, mit einem arithmetischen Coder sogar $H(S) < R < H(S) + 2/n$, wobei n die Länge der zu kodierenden Sequenz ist.

Derartige verlustfreie Kompressionsverfahren werden seit den 60er Jahren eingesetzt, allerdings lassen sich damit nur geringe Kompressionsraten von 1.1:1 bis maximal 3.5:1 erzielen [25].

Das Standardverfahren bei der Behandlung von Multispektraldaten ist die *Differential Pulse Code Modulation* (DPCM). Die Idee dieses Verfahrens ist es, bei einer Reihe aufeinanderfolgender Pixel $x_0, ..., x_{n-1}$ den folgenden Wert x_n mit Hilfe einer Schätzfunktion f durch einen Wert $\tilde{x}_n$ zu approximieren.

$$\tilde{x}_n = f(x_0, ..., x_n)$$

Anschliessend wird lediglich die Differenz

$$d_n = x_n - \tilde{x}_n$$

von $\tilde{x}_n$ zum Originalwert kodiert.

Anwendung findet die DPCM bei Multikanal-Daten insbesondere in spektraler Richtung, wo die Werte stark miteinander korrelieren.

3.3.2 Verlustbehaftete Kompression

Offenbar können bei verlustfreien Kompressionsverfahren keine befriedigenden Kompressionsraten für grosse Datenmengen erzielt. Daher greift man, wenn möglich, auf verlustbehaftete Verfahren zurück.

Mit verlustbehafteter Kompression wird bewusst ein Datenverlust in Kauf genommen, wodurch die Kompressionsrate der Daten jedoch erheblich erhöht werden kann. Dabei werden Algorithmen verwendet, die bei einer vorgegebenen Kompressionsrate die Bildqualität möglichst gut erhalten. Bei einer Analyse durch Menschen ist Datenverlust oft tolerierbar, da bei sinnvoller Auswahl des Verfahrens und Anpassung der Kompressionsrate dieser Verlust durch das menschliche Auge nicht wahrnehmbar ist.

In bestimmten Fällen kann sich der Informationsverlust sogar vorteilhaft auswirken, da beispielsweise bei der Kompression von SAR-Daten die verlorenen Informationen oft Rauschen sind [17].

Die Anwendung verlustbehafteter Kompressionsmethoden ist in der Fernerkundung jedoch noch immer umstritten, da keine standardisierten Verfahren zur Bewertung des Informationsverlustes existieren. Durch die Vielzahl der Anwendungsmöglichkeiten (Computeranalyse vs. Sichtung durch

Menschen) ist diese Bewertung anwendungsabhängig und es muss daher individuell entschieden werden, ab wann ein Informationsverlust vorliegt bzw. ob dieser tolerierbar ist.

Grundsätzlich gibt es eine Vielzahl von verlustbehafteten Kompressionsmethoden. Zu den bekanntesten Verfahren zählen sicherlich Vektorquantisierung, Fraktale Kompression und Transformationskodierer. Bei einer näheren Betrachtung der Verfahren stellt sich jedoch schnell heraus, dass nicht alle davon für die Kompression von Hyperspektraldaten geeignet sind.

Die einfachste Methode bei verlustbehafteter Kompression von Hyperspektraldaten ist es, nicht alle Bänder zu übertragen. Diese Methode ist auch als *spectral editing* bekannt. Dabei wird durch einen Algorithmus festgestellt, welche Bänder nur eine geringe Bedeutung haben. Diese werden dann nicht übertragen. Da aber nicht grundsätzlich vorher feststeht, welche Daten benötigt werden und da es auch wissenschaftlich fragwürdig ist, grosse Datenmengen einfach wegzuwerfen, ist diese Methode offenbar nicht sinnvoll.

Stattdessen wurden andere Methoden entwickelt, um die Daten zu komprimieren und dabei so viel Informationen wie möglich zu erhalten.

Vektorquantisierung

Vektorquantisierung (VQ) ist ein Kompressionsverfahren, das heute bereits in der Fernerkundung Anwendung findet. Verschiedene Varianten der Vektorquantisierung werden zur on-board Kodierung von SAR Aufnahmen genutzt. Es gibt ausserdem Verfahren zur Kodierung von Landsat TM- und Hyperspektraldatensätzen. Die grundlegende Idee der Vektorquantisierung soll deshalb hier kurz erklärt werden. Zunächst wird ein Orginalbild in N Blöcke $B_0, ..., B_{N-1}$ aufgeteilt. Die Blöcke haben im einfachsten Fall alle die gleiche Grösse. Ferner existiert ein Codebuch, das eine Menge repräsentativer Bildblöcke $\{X_0, \ldots, X_{M-1}\}$ enthält. Nun wird zu jedem Bildblock B_j der Codebucheintrag X_i gesucht, der ihm am "ähnlichsten" ist. Das heisst man sucht den Eintrag X_i, der bezüglich eines festgelegten Ähnlichkeitsmaßes den geringsten Abstand zu Block B_j hat.

Die Problem bei diesem Kompressionsverfahren und bei allen gängigen Varianten ist das Design eines geeigneten Codebuchs. In den meisten heute benutzten Verfahren wird das Codebuch aus dem zu kodierenden Bild selbst erstellt. Bei einer Übertragung der komprimierten Daten muss das Codebuch dann natürlich stets mitgesendet werden.

Fraktale Kompression

Fraktale Kompressionsverfahren sind der Vektorquantisierung nicht unähnlich. Das Bild wird auch hier wieder in Blöcke aufgeteilt. Diese Blöcke werden als *Range*-Blöcke bezeichnet. Allerdings werden diese *Range*- Blöcke

jetzt nicht anhand eines Codebuch-Eintrags aproximiert, sondern anhand eines transformierten *Domain*-Blocks. *Domain*-Blöcke sind Teile des gleichen Bildes, jedoch haben sie die doppelte Grösse wie der *Range*-Block, den sie approximieren. Aus der Menge der affinen Transformationen, die jeweils einen *Domain*-Block auf einen *Range*-Block abbilden, setzt sich schliesslich der fraktale Code zusammen.

Obwohl sich durch fraktale Kompressionsmethoden sehr gute Kompressionsraten erzielen lassen, ist die Verwendung derartiger Verfahren bei der Kompression von geowissenschaftlichen Daten eher ungeeignet. Da hier Struktukturen einer Bildregion mit Hilfe ähnlicher Strukturen im Bild repräsentiert werden, entstehen bei der Kompression zwar zwar kein unscharfen Stellen im Bild, allerdings ist der räumliche Verlauf der Strukturen möglicherweise verändert [25].

Transformationskodierer

Die wohl derzeit am weitesten verbreiteten Verfahren zur Kompression von Bilddaten sind Transformatonskodierer, wie die diskrete Kosinustransformation, die beim JPEG Verfahren genutzt wird, die Karhunen-Loeve Transformation oder die verschiedenen Varianten der Wavelettransformation.

Alle Transformationskodierer arbeiten dabei nach dem gleichen Schema. Die Daten werden zunächst durch eine lineare Transformation dekorreliert, d.h. man überführt jede Bildzeile $x = (x_0, \ldots, x_{N-1})$ mit Hilfe einer Transformationsmatrix A in eine dekorrelierte Bildzeile $y = (y_1, \ldots, y_{N-1})$.

$$y = Ax \;\; mit \;\; y_j = \sum_{i=0}^{N-1} a_{i,j} x_i$$

Sollen mehrdimensionale Daten transformiert werden (etwa bei einem zweidimensionalen Bild, siehe Abbildung 3.1), so wird die Transformation für jede Dimension seperat durchgeführt. Bei einem Bild wird die Transformation also zuerst über alle Zeilen durchgeführt und anschliessend über alle Spalten.

Die Transformation sollte ferner so gewählt werden, dass die Daten nicht nur dekorreliert werden, sondern auch so, dass die Energie auf wenige Koeffizienten konzentriert wird. Das bedeutet, dass der Betrag der meisten Koeffizienten sehr klein wird, was sich bei der anschliessend folgenden Quantisierung vorteilhaft auswirkt.

Diese Quantisierung ist oft das Herz des Kompressionsprozesses. Dabei wird eine Wertemenge (in diesem Fall die Farbwerte eines Bildes) auf eine kleinere Menge von Werten abgebildet. Ist die Schrittweite der Intervalle konstant, spricht man von einem *uniform quantizer*, ist sie es nicht, spricht man von einem *non-uniform quantizer*.

Abschliessend folgt noch eine Entropiekodierung der quantisierten Daten, etwa durch Huffman Coding oder arithmetische Kodierung.

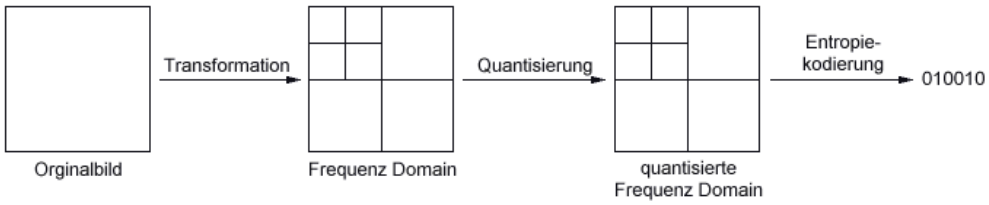


Abbildung 3.1: *Schema der Transformationskodierung*

3.3.3 Bewertung

Offensichtlich wäre *lossless compression* zwar wünschenswert, allerdings würden damit nicht die notwendigen Kompressionsraten erreicht werden, um die Daten in einem Netzwerk wie dem Internet in annehmbarer Zeit zu verschicken. Hyperspektraldatensätze sind oft mehrere hundert Megabyte gross und selbst bei optimalen Kompressionsraten von 4:1 ist die Verwendung derartiger Verfahren nicht sinnvoll.

Um die gewünschten Kompressionsraten erreichen zu können, ist also die Verwendung verlustbehafteter Kompressionsverfahren notwendig. Auch bei einer anwenderabhängigen Implementierung, wie sie in dieser Arbeit umgesetzt werden soll, spricht nichts gegen diese Art der Kompression, da natürlich auch bei diesen Verfahren eine verlustfreie Übertragung der Daten möglich ist. Wichtige Kriterien für die Auswahl eines bestimmten Verfahrens sind also folglich:

- Wie gut ist die Qualität des dekodierten Bildes nach der Kompression mit einer vorgegebenen Kompressionsrate?
- Wie hoch ist der Rechenaufwand für Kodierung bzw. Dekodierung der Daten?

Für diese Arbeit ist es ausserdem interessant, welche Zeit für die Kompression bzw. Dekompression benötigt wird und wieviele Daten pro Zeiteinheit übertragen werden können, da der Anwender genau diese Zeit warten muss, um ein Ergebnis seiner Anfrage zu sehen.

Eine beliebte Möglichkeit, die Qualität des rekonstruierten Bildes zu messen, ist die Berechnung des mittleren quadratischen Fehlers (*mean squared error*) zwischen dem Orginalbild $S = (x_{i,j})_{\substack{i=1,\dots,M \\ j=1,\dots,N}}$ und dem rekonstruierten Bild $D = (\tilde{x}_{i,j})_{\substack{i=1,\dots,M \\ j=1,\dots,N}}$.

$$MSE(S,D) = \frac{1}{M \cdot N} \sum_{i=1}^{M} \sum_{j=1}^{N} (x_{i,j} - \tilde{x}_{i,j})^2$$

Aus dem mittleren quadratischen Fehler lässt sich die *Peak Signal To Noise Ratio* (PSNR) ableiten, die das Verhältnis zwischen dem maximalen Grauwert im Orginalbild und der mittleren Grauwertabweichung in Dezibel angibt.

$$PSNR(S, D) = 20log_{10}\frac{max(S)}{\sqrt{MSE(S, D)}}$$

MSE und PSNR sind jedoch nicht für alle Anwendungen gute Maße, da hier nur der durchschnittliche Fehler innerhalb der Daten gemessen wird.

Nachdem nun einige grundlegende Verfahren zur Datenkompression vorgestellt wurden und auch auf die besonderen Eigenschaften von hyperspektralen Daten eingegangen wurde, folgt nun im folgenden Kapitel die Vorstellung eines Verfahrens zur verlustbehafteten Kompression von Hyperspektraldaten.

Da fraktale Kompressionsmethoden bereits ausgeschlossen werden konnten, wird im nächsten Abschnitt gezeigt, welche Ergebnisse sich mit Vektorquantisierung bzw. Transformationskodierern erzielen lassen und welche Kombinationen verschiedener Verfahren geeignet sind, um möglichst optimale Ergebnisse zu erzielen.

Kapitel 4

Implementation

4.1 Kompressionsmöglichkeiten

Nachdem die grundlegenden Möglichkeiten zur Kompression von Daten vorgestellt wurden, soll nun diskutiert werden, welche dieser Verfahren sich für die Kompression von Hyperspektraldaten eignen.

Aus den im letzten Kapitel beschriebenen Gründen sollen hier nur verlustbehaftete Methoden der Kompression von Multi- bzw. Hyperspektraldaten behandelt werden. In der Literatur finden sich dazu zwei grundlegende Ideen, von denen die eine auf Vektorquantisierung mit Vorhersagestrategien und die andere auf einer Kombination von Karhunen-Loeve Transformation und einer Frequenztransformation beruht. Im folgenden soll für jedes der beiden Verfahren ein Algorithmus vorgestellt werden.

Verfahren 1: Predictive vector quantisation

Vektorquantisierung (VQ) ist ein bewährtes Kompressionsverfahren, mit der man theoretisch optimale *coding performance* erreichen kann. Allerdings hat dieser Ansatz auch Nachteile, wie beispielsweise die hohe Komplexität beim Encoden. Bei der Kompression vom Hyperspektraldaten wird nun Vektorquantisierung in Verbindung mit Verhersagestrategien benutzt, wobei die Vektorquantisierung zur Dekorrelation der räumlichen Dimensionen benutzt wird und die Verhersage in der spektralen Dimension.

Dazu wird zunächst jedes Band in $P \times P$ nichtüberlappende Blöcke eingeteilt. Jeder dieser Blöcke wird dann getrennt VQ-kodiert. Ferner werden die Spektralbänder in *feature bands* und *nonfeature bands* unterteilt. Diese Unterteilung wird anhand der Gesamtzahl der Bänder und der Korrelation zwischen den Bändern getroffen. Jedes *feature band* wird anhand eines eigenen Codebuchs VQ encoded. Die *nonfeature bands* werden anhand der enkodierten *feature bands* vorhergesagt. Dabei wird die Vorhersage von den tatsächlichen Daten subtrahiert und es wird für jedes *nonfeature band* ein Fehlerblock angelegt. Von den *nonfeature bands* wird nur dieser Fehlerblock

gespeichert. Falls die Energie des Fehlerblocks eine gewisse vorgegebene Schranke überschreitet, wird dieser nochmals mittels eines eigenen Kodebuchs VQ-kodiert [24].

Verfahren 2: Transform-based compression

Auch hier wird zunächst jedes Band in $P \times P$ nichtüberlappende Blöcke eingeteilt und jeder Block mittels einer Frequenztransformation wie beispielsweise der diskreten Kosinustransformation kodiert. Anschliessend wird für jeden Block eine Karhunen-Loeve Transformation in spektraler Richtung durchgeführt, wodurch verschiedene Regionen des Bildes verschieden transformiert werden. Auf diese Weise kann sich das Verfahren an die verschiedenen Oberflächen in dem aufgenommenen Datensatz anpassen. Anschliessend werden, wie bereits oben beschrieben, die dekorellierten Daten quantisiert und entropiekodiert. In diesem Fall wurde zu diesem Zweck das JPEG-Verfahren verwendet [24].

Für diese Arbeit wurde der zweite Ansatz gewählt, d.h. die Hyperspektraldaten sollen mittels Frequenz-Transformationen in den räumlichen Dimensionen und mittels KL-Transformation in der spektralen Dimension dekorelliert werden.

Für eine Implementierung dieses Verfahrens gibt es nun eine Vielzahl von Möglichkeiten. So stellt sich bespielsweise die Frage mit welchem Verfahren die Coding Redundanz vermindert werden soll, welche Transformationen auf die Daten angewendet werden sollen, in welcher Reihenfolge diese bevorzugt durchzuführen sind bzw. ob die Reihenfolge der Transformationen überhaupt Auswirkungen auf die Kompressionsrate hat.

In [24] werden Kompressionsverfahren für Hyperspektraldaten in folgende grundlegende Klassen unterteilt, die unter bestimmten Voraussetzungen beide optimal sein können.

Spectral-spatial transform:

Bei dieser Methode wird als erstes eine Karhunen-Loeve Transformation auf die spektrale Dimension angewendet. Anschliessend wird jede dekorrelierte Komponente einzelt mittels eines Transformationskodierers komprimiert. Der Kompression der einzelnen Komponenten kann dabei entweder eine DCT oder eine Wavelet-Transformation zugrunde liegen. Diese Methode ist dann besonders geeignet, wenn alle Bänder die gleichen Auflösungen und die richtige Reihenfolge haben.

Spatial-spectral transform:

Hier wird zunächst eine Transformation (z.B. DCT) auf die räumlichen Dimensionen angewendet. Danach wird jede der spektralen Komponenten eines

Orts-Frequenz-Bandes mittels einer eigenen KLT dekorreliert. D.h. für jeden Koeffizienten der DCT bzw. für jedes Band einer Wavelet-Transformation wird eine andere KL-Transformation benutzt. Diese Methode ist dann sinnvoll, wenn die unterschiedlichen spektralen Komponenten unterschiedliche Orts-Frequenz-Inhalte haben. Das ist beispielsweise der Fall, wenn die Infrarot-Bänder eine geringere räumliche Auflösung haben als die Bänder des sichtbaren Spektrums des gleichen Datensatzes. Für diesen Fall kann man durch die verschiedenen KL-Transformationen bessere Kompressionsraten erzielen.

Die Eigenschaften der hier verwendeten Testdatensätze des DAIS- und HyMap-Sensors sprechen dafür, das erste Verfahren anzuwenden, d.h. es wird zunächst die spektrale Dimension und anschliessend die räumlichen Dimensionen dekorreliert.

Zur spektralen Dekorrelation wird dabei wie vorgeschlagen die Karhunen-Loeve Transformation verwendet, in den räumlichen Dimensionen wird die Dekorrelation mittels einer Wavelet-Transformation durchgeführt. Anschliessend werden die Daten mittels einen arithmetischen Koders komprimiert. Wahlweise kann nach der Dekorrelation noch eine Quantisierung durchgeführt werden, durch die die Kompressionsrate stark verbessert werden kann.

In [6] wird ein Algorithmus für die Kompression von Multispektraldaten vorgeschlagen, der dem in dieser Arbeit verwendeten Vorgehensweise sehr ähnlich ist. Die Testdaten waren bei [6] 512×512 Pixel grosse Landsat TM Aufnahmen, die wie folgt bearbeitet wurden:

1. Subtrahieren des Mittelwertes von jedem der sieben Orginalbänder,
2. Generierung der sieben Hauptkomponenten durch Anwendung der KLT auf die um den Mittelwert verkleinerten Bänder,
3. Räumliche Dekorrelation der sieben Hauptkomponenten durch diskrete Wavelet Transformation,
4. Quantisierung der dekorrelierten Samples und Lauflängenkodierung der Ketten von Nullen und Nicht-Nullen,
5. Huffman Kodierung für jede a) Kette von Nullen, b) Kette von Nicht-Nullen und c) Werte dieser Nicht-Null Ketten,
6. Der resultierende Bitstrom enthält nun eine verkleinerte Repräsentation der sieben Orginalbänder. Zur späteren Dekodierung und Restauration der Bilder werden die Mittelwerte und Eigenwerte ebenfalls übertragen,

7. Die Huffman- und lauflängenkodierten Bitströme werden dekodiert und anschliessend invers wavelettransformiert. Als Ergebnis erhält man die nun verlustbehafteten sieben Hauptkomponenten,

8. Die Orginalbilder werden nun durch inverse KLT wiedergewonnen,

9. Die Mittelwerte werden wieder hinzuaddiert

Im wesentlichen wird das gleiche Verfahren auch in dieser Arbeit verwendet, jedoch wurden einige Änderungen vorgenommen, auf die im weiteren Verlauf dieses Kapitels noch näher eingegangen wird. So wird beispielsweise ein arithmetischer Kodierer anstelle eines Huffman-Kodierers verwendet, die Lauflängenkodierung wurde anders implementiert, es wurde ein spezieller Quantisierungsalgorithmus verwendet, usw.

Ferner haben erste Versuche bereits gezeigt, dass durch kleinere Blockgrössen die Qualität nach der Kompression und das Zeitverhalten verbessert werden.

Im folgenden sollen nun die verwendeten Teilverfahren näher erklärt werden, wobei auch auf die Änderungen gegenüber dem Verfahren in [6] eingegangen wird.

4.2 Mathematische Grundlagen

4.2.1 Wavelet Transformation

Grundlagen zur Wavelet Transformation

Die grundsätzliche Idee einer Transformation ist, eine neue Repräsentation für einen gegebenen Vektor $\mathbf{x}$ zu finden. Die Transformation verändert die Daten nun derart, dass der transformierte Vektor $\bar{\mathbf{x}}$ Eigenschaften hat, die $\mathbf{x}$ nicht hatte. Ist die Transformation invertierbar, kann man $\bar{\mathbf{x}}$ ohne Informationsverlust wieder in $\mathbf{x}$ zurücktransformieren [23].

Da hier die Kompression von Daten im Mittelpunkt steht, wäre eine interessante Eigenschaft des transformierten Vektors beispielsweise, dass die Energie[1] von $\mathbf{x}$ in $\bar{\mathbf{x}}$ auf wenige Koeffizienten konzentriert wird, d.h. viele Koeffizienten werden betragsmässig sehr klein. Tatsächlich ist es so, dass man noch immer eine fast perfekte Rekonstruktion von $\mathbf{x}$ erhält, wenn man diese kleinen Koeffizienten auf 0 setzt. Das Ergebnis ist in diesem Fall also eine verlustbehaftete Kompression.

Nach eben diesem Prinzip funktioniert auch die diskreten Wavelet Transformation (DWT). Man kann die Transformation als eine Matrix betrachten, mit deren Hilfe ein Vektor der Länge 2^i in einen anderen Vektor der gleichen Länge transformiert wird. Die Transformation ist invertierbar und

[1] Die Energie $\mathbf{E}$ eines Vektors x ist definiert als $\mathbf{E} = \sum_{i=0}^{N-1} x_i^2 = x^T x$

orthogonal, d.h. die Matrix für die inverse Transformation ist gerade die Transponierte der Transformationsmatrix[2].

Bei reinen Frequenztransformationen, wie beispielsweise der DCT oder Fourier Transformation, findet eine reine Frequenzzerlegung von Bildern statt. Daher sind bei diesen Transformationen nur globale Aussagen über das Frequenzverhalten möglich. Bei der Wavelet Transformation hingegen besteht die Basis der Wavelet-Domain aus einer Menge von Funktionen, die in ihrem Orts- und Frequenzverhalten stark lokalisiert sind. Statt einer reinen Frequenzzerlegung entsteht eine Orts-Frequenz-Zerlegung [21].

Ähnlich wie bei der Fourier Transformation kann man die (orthogonale) Wavelet Transformation als eine Drehung im Funktionsraum betrachten. Die Basisfunktionen im Ortsraum waren gerade die Einheitsvektoren, in der neuen Domäne sind es hingegen kompliziertere Funktionen. Für die Fourier Transformation sind die Basisfunktionen Sinus- und Kosinusfunktionen, bei der DWT sind es sogenannte Wavelets bzw. Waveletfunktionen [25].

Eine solche Waveletbasis ist eine Menge von linear unabhängigen Funktionen $\omega_{jk}(t)$ in fortlaufender Zeit, die sich nun dadurch auszeichnen, dass aus einem Mutterwavelet $\omega(t)$ durch Verschiebung (Translation) und Skalierung (Dilatation) hervorgehen. Dieses Wavelet ist eine kleine Welle (*wave*), die normalerweise bei $t = 0$ beginnt und bei $t = N$ endet [21].

Verschobene Wavelets ω_{0k} beginnen hingegen bei $t = k$ und enden bei $t = k+N$, skalierte Wavelets ω_{j0} beginnen bei $t = 0$ und enden bei $t = N/2^j$.

$$\omega_{0k} = \omega(t - k) \qquad \omega_{j0} = \omega(2^j t).$$

Für ein typisches Wavelet ω_{jk} gilt also:

$$\omega_{jk} = \omega(2^j t - k)$$

Eine solche Menge von Wavelets kann durch eine bestimmte Menge von Zahlen spezifiziert werden, die als Wavelet-Filter-Koeffizienten bezeichnet werden.

Daubechies Wavelet Filter Koeffizienten

In dieser Arbeit werden Wavelet-Filter-Koeffizienten verwendet, die zur Klasse der *Daubechies Wavelets* gehören. Der einfachste Filter dieser Klasse, *DAUB4* oder Daubechies-4-tap[3], besteht aus nur vier Koeffizienten $c_0, \ldots, c_3$.

Anhand dieses Filters soll die Wavelet Transformation, wie sie in dieser Arbeit verwendet wird, erklärt werden.

[2] Für orthogonale Matrizen gilt gerade $A^T = A^{-1}$

[3] Im allgemeinen versteht man unter *n-tap*-Filtern Filter mit n Koeffizienten

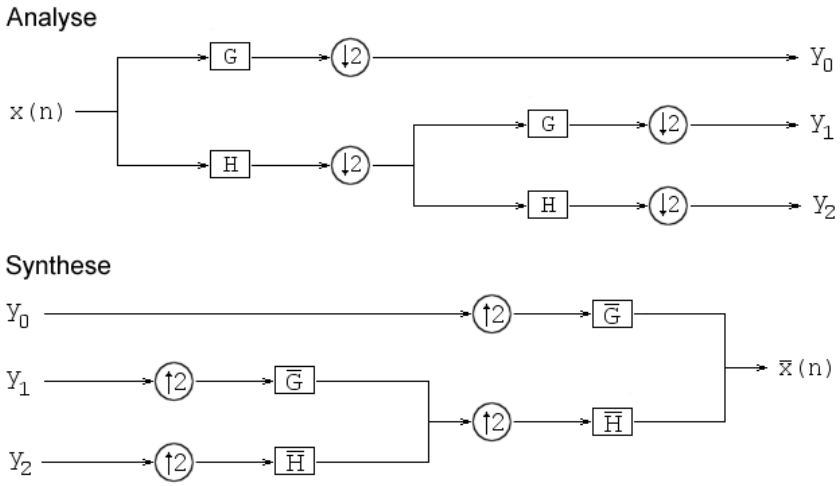


Abbildung 4.1: *Schema der eindimensionalen Wavelettransformation. H ist der low pass-Filter, G der high pass- Filter. Bei der Analyse wird nach jeder Anwendung eines Filters ein Downsampling durchgeführt. Bei mehrfacher Anwendung wird die Transformation nur auf den low pass der vorangegangenen Transformation angewendet. Bei der Synthese wird vor der Anwendung der inversen Filter $\bar{H}$ und $\bar{G}$ ein Upsampling durchgeführt.*

Zunächst sei folgende Matrix $\mathbf{A}$ gegeben:

$$\mathbf{A} = \begin{bmatrix} c_0 & c_1 & c_2 & c_3 & & & & & & & & \\ c_3 & -c_2 & c_1 & -c_0 & & & & & & & & \\ & & c_0 & c_1 & c_2 & c_3 & & & & & & \\ & & c_3 & -c_2 & c_1 & -c_0 & & & & & & \\ \vdots & & & & & & \ddots & & & & & \\ & & & & & & & c_0 & c_1 & c_2 & c_3 & \\ & & & & & & & c_3 & -c_2 & c_1 & -c_0 & \\ c_2 & c_3 & & & & & & & & c_0 & c_1 & \\ c_1 & -c_0 & & & & & & & & c_3 & -c_2 & \end{bmatrix}$$

Bei der Anwendung von von $\mathbf{A}$ auf einen Vektor kann man die Zeilen der Matrix als Filter betrachten. Im folgenden sollen der Filter c_0, c_1, c_2, c_3 als H und der Filter $c_3, -c_2, c_1, -c_0$ als G bezeichnet werden[4].

Der Filter H in den ungeraden Reihen verstärkt gerade die groben Strukturen (*low pass*), während der Filter G in den geraden Reihen die Details hervorhebt (*high pass*). G ist so gewählt, dass für einen glatten Datenvektor (d.h. bei kleiner Varianz der Koeffizienten) eine Antwort zurückgegeben wird, die möglichst nah an 0 liegt. Das wird erreicht, indem indem die Mo-

[4] G und H werden in der Signalverarbeitung als *quadrature mirror filter* bezeichnet

mente von G 0 werden. Da während der Anwendung ein *Downsampling*[5] durchgeführt wird, geben beide Filter jeweils nur die Hälfte der Werte der Eingabedaten zurück, allerdings lässt sich aus den $N/2$ *low pass* Werten und den $N/2$ *high pass* Werten der Orginalvektor der Länge N perfekt rekonstruieren (vergleiche Abbildung 4.1).

Dies geschieht mit Hilfe der Matrix **B**:

$$\mathbf{B} = \begin{bmatrix} c_0 & c_3 & & & \cdots & & & & & c_2 & c_1 \\ c_1 & -c_2 & & & & & & & & c_3 & -c_0 \\ c_2 & c_1 & c_0 & c_3 & & & & & & & \\ c_3 & -c_0 & c_1 & -c_2 & & & & & & & \\ & & & & \ddots & & & & & & \\ & & & & & c_2 & c_1 & c_0 & c_3 & & \\ & & & & & c_3 & -c_0 & c_1 & -c_2 & & \\ & & & & & & & c_2 & c_1 & c_0 & c_3 \\ & & & & & & & c_3 & -c_0 & c_1 & -c_2 \end{bmatrix}$$

B ist gerade die Inverse der Matrix **A**, genau dann wenn folgende Bedingungen gelten:

$$c_0^2 + c_1^2 + c_2^2 + c_3^2 = 1$$

$$c_2 c_0 + c_3 c_1 = 0$$

Soll zusätzlich noch gelten, dass G zwei verschwindende Momente besitzt[6], werden noch zwei weitere Bedingungen benötigt:

$$c_3 - c_2 + c_1 - c_0 = 0$$

$$0c_3 - 1c_2 + 2c_1 - 3c_0 = 0$$

Mit Hilfe dieser vier Gleichungen kann man jetzt die vier Unbekannten $c_0, \ldots, c_3$ bestimmen. Als eindeutige Lösung ergibt sich:

$$c_0 = (1 + \sqrt{3})/4\sqrt{2}$$

$$c_1 = (3 + \sqrt{3})/4\sqrt{2}$$

$$c_2 = (3 - \sqrt{3})/4\sqrt{2}$$

$$c_3 = (1 - \sqrt{3})/4\sqrt{2}$$

Mit Hilfe dieser vier Werte kann man nun also, setzt man sie in die Transformationsmatrix **A** ein, beliebige Daten einer Wavelet Transformation unterziehen [15].

[5] D.h. das jeder zweite Koeffizient der Antwort nicht betrachtet wird. So wird z.B. aus der Antwort $[2, 5, 4, 3]$ nach dem *Downsampling* $[2, 4]$. Bei der inversen Transformation wird analog dazu ein *Upsampling* durchgeführt, indem für die verlorengegangenen Koeffizienten Nullen eingesetzt werden. Im Beispiel wird aus $[2, 4]$ also $[2, 0, 4, 0]$.

[6] Für *DAUB4* ist die Länge der Antwort von G nach dem *Downsampling* gerade 2.

Diskrete Wavelet Transformation

Bei der diskrete Wavelet Transformation wird eine aus Wavelet-Koeffizienten bestehende Matrix hierarchiesch auf einen Datenvektor der Länge N angewendet (vgl. Abbildung 4.2), d.h. zunächst auf den gesamten Vektor der Länge N und dann jeweils auf den entstandenen *low pass*, d.h. auf den Vektor der Länge $N/2$, im nächsten Schritt auf den Vektor der Länge $N/4$, usw.

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \end{bmatrix} \overset{(1)}{\longrightarrow} \begin{bmatrix} l_1 \\ h_1 \\ l_2 \\ h_2 \\ l_3 \\ h_3 \\ l_4 \\ h_4 \end{bmatrix} \overset{(2)}{\longrightarrow} \begin{bmatrix} l_1 \\ l_2 \\ l_3 \\ l_4 \\ h_1 \\ h_2 \\ h_3 \\ h_4 \end{bmatrix} \overset{(1)}{\longrightarrow} \begin{bmatrix} L_1 \\ H_1 \\ L_2 \\ H_2 \\ h_1 \\ h_2 \\ h_3 \\ h_4 \end{bmatrix} \overset{(2)}{\longrightarrow} \begin{bmatrix} L_1 \\ L_2 \\ H_1 \\ H_2 \\ h_1 \\ h_2 \\ h_3 \\ h_4 \end{bmatrix}$$

Abbildung 4.2: *Darstellung des Pyramidenschemas bei einer Wavelet Transformation. (1) beschreibt die Anwendung der Transformationsmatrix auf den Vektor, (2) ist die Permutation des Vektors, derart dass low- und high pass getrennt sind. l_i sind dabei die low pass Werte nach der ersten Transformation, h_i sind die high pass Werte. Analog teilen sich l_i nach nochmaliger Transformation in L_i und H_i.*

Die Transformation kann durchgeführt werden, bis der *low pass* nur noch aus $n/2$ Werten besteht, wobei n die Anzahl der Filter-Koeffizienten ist.

Schliesslich stellt sich noch die Frage, wie die Wavelet Transformation bei höherdimensionalen Datensätzen anzuwenden ist. Wie in Abbildung 4.3 dargestellt ist, wird die eindimensionale Wavelet Transformation einfach sukzessive auf alle Dimensionen des Datensatzes angewendet. Im Fall eines zweidimensionalen Datensatzes heisst das also, dass die Transformation zunächst über alle Zeilen durchgeführt wird. Im nächsten Schritt wird die Transformation dann auf alle Spalten des gefilterten Bildes angewendet. Das Bild ist nun in vier Frequenzanteile zerlegt, die entsprechend der darauf angewendeten Filter mit LL (zweimalige Anwendung des *low pass* Filters), LH (Anwendung des *high pass* Filters bei der Transformation über die Zeilen und des *low pass* Filters bei der Transformation über die Spalten), HL (*low pass* über die Zeilen und *high pass* über die Spalten) und HH bezeichnet werden.

Soll das Bild nochmals transformiert werden, wird die Transformation nur auf das LL-Band angewendet. Die sich ergebenden Frequenzanteile werden analog zum letzten Schritt mit LLLL, LHLL, HLLL und HHLL bezeichnet.

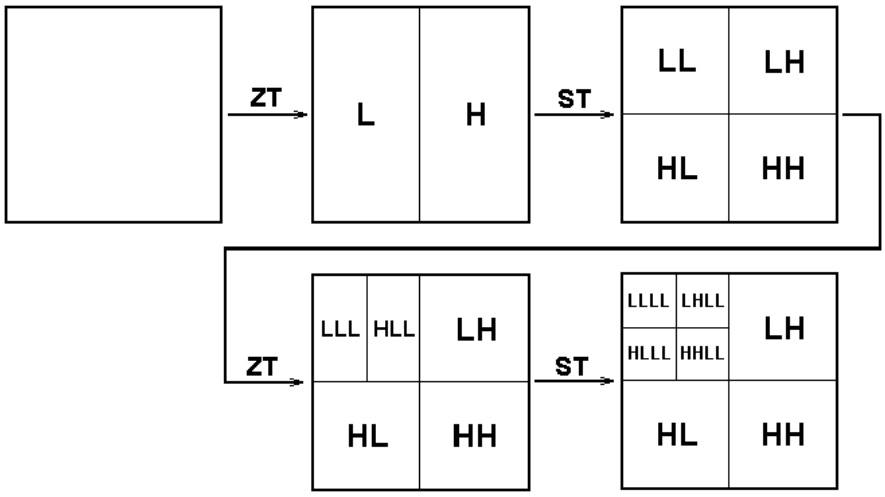


Abbildung 4.3: *Schema der zweidimensionalen Wavelettransformation. Die Bezeichnung ZT steht für Zeilentransformation, ST beschreibt eine Spaltentransformation. HL, LH und HH stellen die hochfrequenten Bildanteile dar, LL die niederfrequenten. Bei einer nochmaligen Transformation wird LL abermals unterteilt. Man bezeichnet nun LHLL, HLLL und HHLL als mittlere Frequenzen und LLLL als niederfrequenten Anteil.*

Bei einer Visualisierung kann man in jedem Frequenzanteil (mehr oder weniger gut) die Struktur des Orginalbildes erkennen, d.h. jeder Koeffizient repräsentiert einen Frequenzanteil eines Raumausschnittes. Anhand dessen wird noch einmal veranschaulicht, dass durch die Wavelet Transformation, wie zu Beginn dieses Abschnitts erklärt, eine Orts-Frequenz-Zerlegung der Daten durchgeführt wird.

Quantisierung

Bereits im Kapitel 3.3.2 wurde kurz erläutert, wie Daten überlicherweise quantisiert werden. Aufgrund der besonderen Eigenschaften der Daten nach der Wavelet Transformation bietet sich hier jedoch ein leicht abgewandeltes Verfahren an, wie es von Jacob Ström in [22] vorgestellt wird.

Dort wird die Verwendung eines *dead zone quantizers* vorgeschlagen, um die Tatsache, dass eine grosse Menge der Werte nach der Wavelet Transformation betragsmässig sehr klein werden, besser ausnutzen zu können.

Bei dem in Kapitel 3.3.2 beschriebenen *uniform quantizer* wird die Menge aller Daten in eine kleinere Datenmenge abgebildet, indem Werte in vorher festgelegten, gleich grossen Intervallen auf einen einzigen Wert abgebildet werden. Da die Daten im Anschluss an die Quantisierung noch *runlength* encodet werden sollen (siehe dazu auch Abschnitt 4.2.4), ist es im Interesse der Kompression, dass die Daten möglichst lange Ketten von Nullen enthalten.

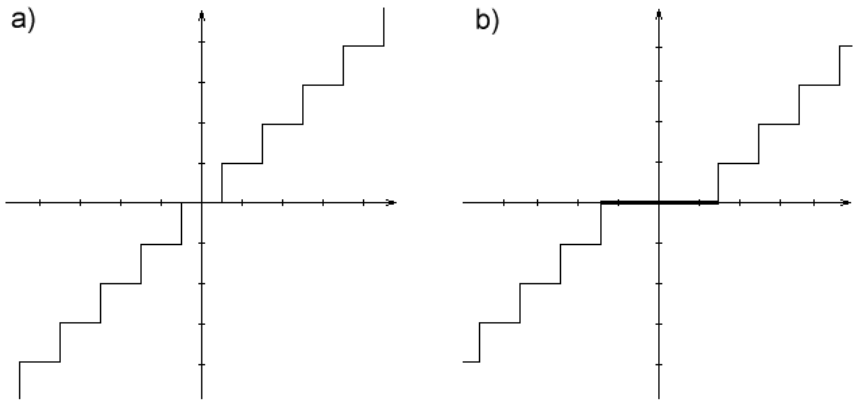


Abbildung 4.4: *Abbildung a) zeigt einen* uniform quantizer*, Abbildung b) einen* dead zone quantizer *mit einer* dead zone *von drei Schritten.*

Jedesmal wenn eine solche Kette unterbrochen wird, wird die Kompressionsrate negativ beeinflusst. Andererseits sind gerade die betragsmässig sehr kleinen Werte weniger wichtig für eine gute Rekonstruktion der Orginaldaten bei der Rücktransformation als die grossen Werte.

Die Idee des *dead zone quantizers* ist nun, das Intervall der Werte, die auf 0 abgebildet werden, zu vergrössern. Auf diese Weise entstehen nach der Quantisierung längere Ketten von Nullen als mit einem *uniform quantizer* ohne dass die Qualität der zu rekonstruierenden Daten darunter leidet [26]. Im Gegenteil: nach den in [22] durchgeführten Testreihen war die PSNR für eine gegebene Bitrate im Durchschnitt 0.5 dB höher als bei einem *uniform quantizer*, wenn die Grösse der *dead zone* richtig gewählt wurde.

In dieser Arbeit wird zur Quantisierung der wavelettransformierten Daten ein solcher *dead zone quantizer* verwendet. Die verwendete *dead zone* hat die in [22] empfohlene Länge von 1.9 Schritten[7], bei der die PSNR annähernd maximal wird[8].

Abschliessend sollen noch einige Anwendungsmöglichkeiten für die Wavelet Transformation genannt werden. Neben der Datenkompression lässt sich die Wavelet Transformation insbesondere auch zur Bildanalyse einsetzen. Hier wird sie zur Kantenerkennung, zur Texturanalyse sowie zur Bildsegmentierung benutzt. In der medizinischen Bildverarbeitung gibt es ferner Verfahren, in denen die Wavelet Transformation beispielsweise zur EKG Analyse eingesetzt wird.

[7]Ein Schritt entspricht der Länge eines Intervalls bei einem skalaren Quantisierer.

[8]Die optimale Länge der *dead zone* variiert natürlich abhängig von den zu quantisierenden Daten. Allerdings ist dieses Intervall nur klein und die Wahl von 1.9 ist in jedem Fall annähernd optimal [2].

Bei dieser Arbeit wird die Wavelet-Transformation räumlichen Dekorrelation der Hyperspektraldaten genutzt. Die Dekorrelation in der spektralen Dimension wird mit Hilfe der Karhunen-Loeve Transformation durchgeführt, die nun als nächstes erklärt werden soll.

4.2.2 Karhunen-Loeve Transformation

Ebenso wie bei Wavelet Transformation wird auch mit Hilfe der Karhunen-Loeve Transformation[9] eine Transformationskodierung vorgenommen, d.h. die Daten werden durch die Transformation dekorreliert, anschliessend können sie quantisiert und entropiekodiert werden, wodurch hohe Kompressionsraten erzielt werden können.

Das besondere an der KLT ist nun, dass die Transformationsmatrix nicht fest vorgegeben ist. Stattdessen wird sie anhand der statistischen Eigenschaften der zu transformierenden Daten berechnet. Das hat den Vorteil, dass diese Transformation optimal bezüglich der Energiekonzentration ist.

Diese Abhängigkeit von den zu transformierenden Daten ist aber auch ein Nachteil der KLT, da die Berechnung der Transformationsmatrix aufgrunddessen vergleichsweise rechenaufwendig ist. Ferner muss die Transformationsmatrix stets mit den transformierten Daten übertragen werden, um diese später wieder zurücktransformieren zu können.

Oft werden daher Frequenztransformationen wie die diskrete Kosinustransformation benutzt, um die KL-Transformation zu approximieren. In räumlichen Dimensionen (also beispielsweise bei Bildern) gelingt das auch sehr gut. Für die bei Hyperspektraldaten vorhandene dritte (spektrale) Dimension gilt das jedoch nicht, da die Daten hier nicht als stationär modelliert werden können. So hängt beispielsweise die Korrelation zweier benachbarter Spektralbänder in hohem Maße davon ab, welche Bänder gerade betrachtet werden. Benachbarte Bänder im Infrarot Bereich können beispielsweise wesentlich stärker miteinander korrelieren, als benachbarte Bänder im sichtbaren Bereich des Lichts. D.h. die Korrelation ist abhängig von der absoluten Position in der spektralen Dimension [24].

Soll also ein L-kanaliges Bild der Grösse $M \times N$ in spektraler Richtung transformiert werden, muss dazu jeder Vektor $\mathbf{x_{ij}} = (x_{ij}^0, \ldots, x_{ij}^{L-1})$ transformiert werden. Die dazu benötigte Transformationsmatrix wird nun über einen statistischen Ansatz bestimmt. Man betrachtet dazu alle Vektoren als Realisation einer Zufallsvariablen

$$\mathbf{x} = (\mathbf{x}^0, \ldots, \mathbf{x}^{L-1})^T$$

[9] Die KLT ist auch als Hauptkomponententransformation oder Hotelling Transformation bekannt

Die Variable $\mathbf{x}$ hat den Erwartungswert

$$\bar{\mathbf{x}} = E(\mathbf{x}) = \frac{1}{M \cdot N} \left(\sum_{\substack{0 \le i < M-1 \\ 0 \le j < N-1}} x_{ij}^0, \ \ldots \ , \sum_{\substack{0 \le i < M-1 \\ 0 \le j < N-1}} x_{ij}^{L-1} \right)^T$$

Damit lässt sich nun die Korrelationsmatrix C_x wie folgt berechnen:

$$\mathbf{C_x} = E((\mathbf{x} - \bar{\mathbf{x}})(\mathbf{x} - \bar{\mathbf{x}})^T)$$

Das Ziel der Transformation ist nun $\mathbf{x}$ nach $\mathbf{y}$ zu transformieren, derart dass die Korrelationsmatrix $\mathbf{C_y}$ von $\mathbf{y}$ nur Einträge auf der Hauptdiagonalen hat. Das heisst, die Komponenten von $\mathbf{y}$ sind unkorreliert.

$\mathbf{C_x}$ ist eine reellwertige symmetrische Matrix, es lassen sich also n Eigenwerte $\lambda_0, \ldots, \lambda_{L-1}$ von $\mathbf{C_x}$ mit zugehörigen Eigenvektoren $\mathbf{e_0}, \ldots \mathbf{e_{L-1}}$ bestimmen, so dass

- $\mathbf{e_i} \perp \mathbf{e_j}, \quad i \neq j$
- $\| \mathbf{e_i} \|_2 = 1$
- $\lambda_0 \ge \lambda_1 \ge \ldots \ge \lambda_{L-1}$

Die Eigenvektoren $\mathbf{e_i}$ bilden nun gerade die Zeilen der Transformationsmatrix für die zu transformierenden Daten. Damit ist die Hauptkomponententransformation definiert als:

$$\mathbf{y} = \mathbf{A}(\mathbf{x} - \bar{\mathbf{x}}), \quad mit \ \ \mathbf{A} = \begin{pmatrix} \mathbf{e_0}^T \\ \vdots \\ \mathbf{e_{L-1}}^T \end{pmatrix}$$

Für die Korrelationsmatrix von $\mathbf{y}$ gilt nun gerade

$$\mathbf{C_y} = \mathbf{A C_x A^T} = \begin{bmatrix} \lambda_0 & & & 0 \\ & \lambda_1 & & \\ & & . & \\ & & & . \\ 0 & & & \lambda_{L-1} \end{bmatrix},$$

d.h. die Einträge auf der Hauptdiagonalen von $\mathbf{C_y}$ sind gerade die Eigenwerte von $\mathbf{C_x}$. Da die Eigenwerte der Grösse nach geordnet waren, tragen die ersten Komponenten in $\mathbf{y}$ die meisten Informationen von $\mathbf{x}$.

Man kann daher eine Datenreduktion mit der Kompressionsrate $L : K$ durchführen, indem man lediglich die ersten K Komponenten von $\mathbf{y}$ betrachtet.

Bei der Rücktransformation von $\hat{y} = (y_0, \ldots, y_K, 0, \ldots, 0)$ gilt damit für den mittleren quadratischen Fehler:

$$mse = \sum_{k=K}^{N-1} \lambda_k$$

Aus der Transformationsgleichung lässt sich schliesslich die Gleichung für die Rücktransformation ableiten:

$$\mathbf{x} = \mathbf{A}^T\mathbf{y} + \bar{\mathbf{x}}$$

Die Karhunen-Loeve Transformation ist in der Fernerkundung ein gängiges Verfahren zur Variablenkompression [25]. Bei der Kompression von hyperspektralen Daten bleibt ihre Anwendung aufgrund des Rechenaufwandes jedoch auf die spektrale Dimension beschränkt. Für die räumlichen Dimensionen wird stattdessen, wie bereits beschrieben, eine Frequenztransformation verwendet.

4.2.3 Arithmetische Kodierung

Um die in Abschnitt 3.2.1 beschriebene Coding Redundanz zu vermindern, sollen die bereits dekorrelierten Daten nun noch mittels eines Entropie-Kodierers komprimiert werden.

Entropie-Kodierer gehören zur Gruppe der verlustfreien Kompressionsverfahren, d.h. im Gegensatz zu den bisher beschriebenen Verfahren erzielt man hier keine Reduktion der Daten, indem man etwa durch Quantisierung Teile der Daten opfert.

Die Idee bei einem Entropie-Kodierer ist, eine Kette von Symbolen (also beispielsweise ASCII-Zeichen oder Bytes) in eine andere Zeichenfolge umzuwandeln. Diese neue Zeichenkette sollte im allgemeinen kürzer sein als das Orginal (die Daten werden also komprimiert) und sie muss wieder in die Orginalzeichenkette überführbar sein.

Im allgemeinen heisst in diesem Fall, dass nicht jede Zeichenkette komprimierbar sein muss. Eine Kompression ist nur dann möglich, wenn einige Symbole häufiger vorkommen als andere. Diese können dann mit weniger Bits encodet werden als weniger häufig vorkommende Symbole, so dass die Gesamtlänge der Daten sich verkürzt [15].

Zu den bekanntesten Verfahren gehört dabei die in dieser Arbeit verwendete arithmetische Kodierung, aber auch der Huffman-Code und der Lempel-Ziv-Code.

Bevor nun eine Erklärung des verwendeten Verfahrens folgt, sollte geklärt werden, was Entropie eigentlich bedeutet. Bei einem gegebenen Eingabealphabet mit N Symbolen und einer Zeichenkette, in der dieses Alphabet verwendet wird, lässt sich für jedes Symbol des Eingabealphabets eine Wahrscheinlichkeit $p_i,\ i = 1, \ldots, N$ berechnen, so dass $\sum_{i=1}^{N} p_i = 1$.

Die Entropie H ist nun definiert als eine untere Schranke für die durchschnittliche Länge einer Zeichenkette, die aus diesen Symbolen besteht und es gilt

$$H = -\sum_{i=1}^{N} p_i \, log_2 \, p_i.$$

Der ungünstigste Fall, der auftreten kann, ist, dass alle Symbole die gleiche Wahrscheinlichkeit $p_i = 1/N$ haben. Die Entropie ist dann $H = log_2 \, N$, d.h. es ist keine Kompression möglich. Für alle anderen Wahrscheinlichkeitsverteilungen wird die Entropie kleiner sein und es kann eine Kompression der Daten durchgeführt werden.

Bei der arithmetischen Kodierung wird ein Input beliebiger Länge als eine reelle Zahl R kodiert, mit $0 \leq R < 1$. Je länger die Eingabe ist, um so grössere Genauigkeit ist für R erforderlich.

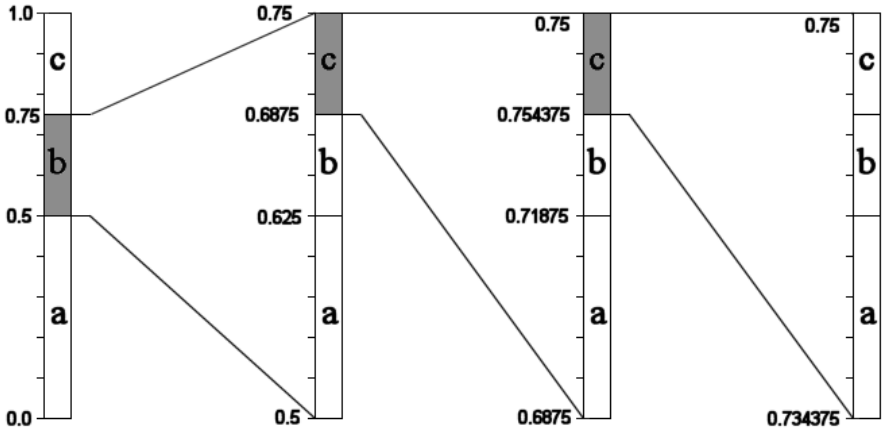


Abbildung 4.5: *Arithmetische Kodierung der Zeichenkette "bcc..." bei einem gegebenen Alphabet (a,b,c)*

Die Vorgehensweise soll nun an einem Beispiel erklärt werden. Gegeben ist ein Eingabealphabet (a, b, c) mit den Wahrscheinlichkeiten $p(a) = \frac{1}{2}$, $p(b) = \frac{1}{4}$ und $p(c) = \frac{1}{4}$. Möchte man nun die Nachricht "bcc" kodieren, so wird das Intervall [0,1) entsprechend den Wahrscheinlichkeiten der Alphabetsymbole in drei Segmente unterteilt, wobei die Länge eines Segments i der Wahrscheinlichkeit p_i korrespondiert. Das erste Symbol "b" verkleinert das Intervall für R auf $0.5 \leq R < 0.75$ (vgl. Abbildung 4.5).

Dieses Intervall wird nun wieder in drei Segmente unterteilt, deren Länge wieder proportional zu den p_i ist. Nach dem Enkoden des zweiten Symbols "c" wird Intervall erneut eingeschränkt. Für R gilt jetzt $0.6875 \leq R < 0.75$. Analog wird jetzt mit dem letzten Zeichen der Nachricht verfahren. Nach dem enkoden von "c" gilt $0.734375 \leq R < 0.75$. Man kann sich jetzt eine beliebige Zahl aus diesem Intervall wählen. Diese wird dann in Binärdarstellung übertragen und stellt den arithmetischen Code für die Nachricht

“bcc“ dar. Genaugenommen repräsentiert dieses Intervall jedoch nicht nur die Zeichenkette “bcc“, sondern “bcc...“, d.h. nach den ersten drei Zeichen könnten noch weitere folgen. Daher empfielt sich das Anhängen eines *end of message* Symbols (EOM), das dem Alphabet hinzugefügt wird, aber nur eine sehr geringe Wahrscheinlichkeit hat [15].

Bei genauerer Betrachtung des Verfahrens sieht man schnell, dass Symbole mit hoher Wahrscheinlichkeit das Intervall nicht stark verkleinern, Symbole mit geringer Wahrscheinlichkeit hingegen schon. Ein grosses Intervall kann aber durch wenige Dezimalstellen dargestellt werden, während man für ein kleines Intervall viele Dezimalstellen benötigt. Damit ist die Forderung vom Beginn dieses Abschnitts erfüllt, dass häufig vorkommende Symbole mit sehr wenigen Bits kodiert werden können, allerdings selten vorkommende Symbole dafür mehr Bits benötigen.

4.2.4 Lauflängen-Kodierung (*run-length coding*)

Ebenso wie Arithmetsche Kodierung ist auch Lauflängen-Kodierung eine Möglichkeit, die Coding Redundanz gegebener, hochkorellierter Daten zu verringern. Das Verfahren ist jedoch ungleich einfacher, als Arithmetische Kodierung bzw. Huffman-Kodierung.

Statt jedes Bit des Datenstroms einzeln zu übertragen, werden die Daten in eine Reihe von Integerwerten transformiert, von denen jeder Wert angibt, wieviele aufeinanderfolgende Bits den gleichen Wert haben.

Dieses Verfahren ist besonders geeignet für schwarz/weiß Bilder, bei denen aufeinanderfolgende Zahlen im Code stets einen Farbwechsel implizieren. Da aber nur zwei Farben existieren, muss lediglich bekannt sein, mit welchem Farbwert der Code beginnt, um die Daten vollständig dekodieren zu können.

In dieser Arbeit wird *run-length coding* in etwas abgewandelter Form benutzt: Nach der Quantisierung der wavelet-transformierten Daten, haben sehr viele Bits den Wert 0. Kodiert werden nun also nur *runs* von Nullen und *runs* von “Nicht-Nullen“. Die so komprimierten Daten werden anschliessend arithmetisch kodiert, wodurch man eine bessere Kompressionsrate gegenüber der Verwendung eines einzelnen Verfahrens erzielt [6].

4.3 Die Programmierumgebung

Die Implementierung der beschriebenen Verfahren erfolgt in Java. Der Grund dafür ist, dass diese Programmiersprache viele Vorteile anderer Sprachen übernommen hat und mit der Möglichkeit plattformunabhängiger Programmierung vereinigt.

Die Java-Architektur gliedert sich zunächst in zwei Bestandteile: das Java Application Programming Interface (API) und die Java Virtual Ma-

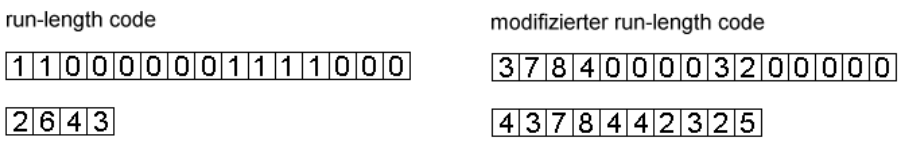


Abbildung 4.6: *Beispiel für die Verwendung von* run-length code*: Auf der linken Seite sind einige Bits eines schwarz/weiss Bildes dargestellt, darunter dieselben Daten, nachdem sie* run-length *kodiert wurden. Die erste Zahl steht fuer die 2 Einsen, die nächste für die 6 Nullen, usw. Auf der rechten Seite ist eine modifizierte Version dargestellt, bei der lediglich zwischen Null und Nicht-Null unterschieden wird. Hier steht bei den kodierten Daten die erste Zahl für einen* run *von 4 Nicht-Nullen, gefolgt von eben diesen 4 Zahlen. Die nun folgende 4 steht für einen* run *von 4 Nullen, usw.*

chine (JVM). Java verfügt dabei über alle Konzepte, die man von anderen objektorientierten Programmiersprachen wie C++ kennt (z.B. Abstraktion, Vererbung, Kapselung, usw.), wodurch ein gut gegliederter, modularer Aufbau des Programms möglich wird.

Mit Hilfe eines Java-Compilers wird dieser Quellcode in einen Bytecode übersetzt. Dieser Bytecode wird dann bei der Ausführung des Programms von der Virtual Machine interpretiert. Der Vorteil gegenüber anderen Programmiersprachen besteht nun offensichtlich darin, dass der vom Compiler generierte Bytecode noch immer plattformunabhängig[10] ist und erst zur Laufzeit in Maschienencode umgesetzt wird. Die einzige Voraussetzung zur Ausführung eines Java-Programms ist folglich das Vorhandensein einer Virtual Machine, die der Java-Definition der Firma Sun entspricht, wie sie heute in allen gängigen Internetbrowsern vorhanden ist oder in Form eines Plug-Ins integriert werden kann.

Nun ist das Konzept von interpretierbaren Sprachen nicht neu. Im Gegensatz zu anderen Sprachen wird jedoch bei Java nicht erst zur Laufzeit eine Syntaxüberprüfung des Quellcodes vorgenommen. Diese Syntaxüberprüfung ist der grosse Nachteil anderer Interpreter, da sie sehr zeitaufwendig ist und sich damit in der Regel sehr negativ auf die Performance auswirkt. In Java findet diese Überprüfung aber bereits während der Übersetzung in den Bytecode statt, hat also keinen Einfluss auf die Laufzeit des Programms.

Ein weiterer Vorteil von Java besteht darin, dass sich der Programmierer keine Gedanken über die Umsetzung von betriebssystemspezifischen Diensten machen. So kann beispielsweise mit Hilfe der “java.awt“-Bibliotheken die Oberfläche eines Programms gestaltet werden. Zur Laufzeit wird dann mit Hilfe der Virtual Machine diese Oberfläche an das verwendete Betriebssystem angepasst dargestellt. Der Programmierer kann sich also auf die Funktionalität der Anwendung konzentrieren, ohne sich Gedanken um

[10]Natürlich gilt das nur, wenn keine proprietären Bibliotheken genutzt werden.

eine spezifische Umsetzung machen zu müssen.

Die Zielsetzung für diese Arbeit war gerade, dass das Programm viele Nutzergruppen ansprechen soll. D.h. es soll sowohl im wissenschaftlich Bereich nutzbar sein, wenn Daten möglichst exakt übertragen werden sollen, anderseits soll es auch für private Nutzer interessant sein, die sich im Rahmen eines Projekt wie *ATLAS2000* über Hyperspektraldaten informieren wollen.

Während im privaten Bereich ein Betriebssystem wie *Microsoft Windows* üblich ist, werden im wissenschaftlichen Bereich oft unixbasierte Betriebssysteme genutzt.

Dieses Problem wird durch eine Implementierung in Java vollständig umgangen, da die Benutzeroberfläche - als Java-Applet implementiert - über jeden beliebigen Broser mit Java Plug-In aufgerufen werden kann, völlig unabhängig vom verwendeten Betriebssystem.

4.4 Client/Server Architektur

Bei der Implementierung des zu dieser Arbeit entwickelten Programms zur Kompression von Hyperspektraldaten werden die Daten in einer zentralen Datenbank gehalten und Nutzer können über ein Netzwerk darauf zugreifen. Das ist die klassische Struktur einer Client/Server Architektur. Derartige Architekturen teilen sich stets in mehrere Schichten, die jeweils konkrete Aufgaben haben.

Im einfachsten Fall existieren zwei Schichten. Ein Beispiel dafür wäre, das von vielen Rechnern aus mittels eines Programmes oder eines Web-Browsers auf eine Datenbank zugegriffen wird. Hierbei bietet die Datenbank eine Schicht und der Browser bzw. das Programm die zweite.

Üblicher ist jedoch das dreischichtige Modell (*3tier*). Auch hier bildet die Datenbank eine Schicht (den sogenannten *Data Storage Layer*). In ihr werden alle Daten gespeichert und verwaltet. Für alle Operationen jeder Transaktion bieten moderne Datenbanken eine Reihe von Eigenschaften[11], die den sicheren Ablauf von Transaktionen gewährleisten und die die Persistenz der Datenbank sichern.

Die nächste Schicht ist der *Application Logic Layer*, der die Programmlogik auf der Serverseite repräsentiert. Nur von hier aus kann auf die Datenbank zugegriffen werden, d.h. Clients (wie z.B. Browser) haben keine direkte Verbindung zur Datenbank mehr, sondern alle Zugriffe werden über diese Schicht realisiert. Hier überlicherweise auch der Grossteil der Funktionalität eines Programms eingebettet. Die Implementierung des Servers geschieht mit Hilfe von Programmiersprachen wie *C++*, *Java* oder *Delphi*. Wird durch

[11] Für jede Transaktion eines Datenbanksystems gelten die sogenannenten ACID-Eigenschaften: **A**tomicity, **C**onsistency, **I**solated Execution und **D**urability

Browser auf diesen Layer zugegriffen, so gehört auch ein Webserver zu dieser Schicht.

Die dritte Schicht schliesslich wird als *Presentation Logic Layer* bezeichnet. Wie der Name schon sagt, findet hier die "Präsentation" des Programmes statt. Zu dieser Schicht gehört üblicherweise die Benutzeroberfläche. Von hier können Anwender mit dem Server kommunizieren, d.h. die Funktionalität des *Application Logic Layer* nutzen bzw. mit dessen Hilfe auf die Datenbank zugreifen. Der *Presentation Logic Layer* stellt für gewöhnlich die Client-Seite eines Programms dar.

Man unterscheidet dabei je nach Implementation zwischen *Thin-* und *Rich-Client*. Der *Thin-Client* enthält keine eigene Logik. Er implementiert lediglich das GUI (*Graphical User Interface*) des Programms, d.h. Nutzereingaben werden zum Server geschickt und die daraufhin empfangenen Daten werden wiederum lediglich visualisiert. Das ist beispielsweise der Fall, wenn mit Hilfe von Browsern auf Server zugegriffen wird. Der Browser wird ausschliesslich zur Repräsentation der Benutzeroberfläche und zur Darstellung der Ergebnisdaten genutzt, hier ist keinerlei Funktionalität enthalten.

Im Gegensatz dazu verfügt der *Rich-Client* durchaus über eigene Logik. Das kann einerseits eine Vorverarbeitung der *Requests* sein, die an den Server geschickt werden sollen, andererseits können auch mit den vom Server empfangenen Daten noch weitere Berechnungen vorgenommen werden [13].

Der Vorteil dieser Architektur liegt in der logischen Partitionierung der Schichten, die nur mit Hilfe von *Interfaces* miteinander kommunizieren. Jede der Schichten stellt einen eigenständigen Teil des Programms dar und jede Schicht ist austauschbar. Heute ist es üblich, dass die *Application Logic* nochmals in mehrere Schichten unterteilt wird, die ebenfalls streng voneinander getrennt werden, wodurch sogenannte n-schichtige Architekturen entstehen.

Für diese Arbeit wird ein *3tier* Modell benutzt. Der *Presentation Logic Layer* ist in Form eines *Rich Client* implementiert, da hier die Dekompression der Daten durchgeführt wird.

4.5 Übertragung der Daten im Netzwerk

Da die komprimierten Daten zwischen verschiedenen Rechnern in einem Netzwerk (beispielsweise dem Internet) übertragen werden, sollen an dieser Stelle die Eigenschaften des verwendeten Protokolls erklärt werden.

Da Java als Programmiersprache für das Internet entwickelt wurde, werden im Sprachumfang bereits Klassen zur Verfügung gestellt, mit denen Java-Programme über Sockets mit anderen Java-Programmen bzw. mit anderen Servern oder Clients, die auf Basis von TCP oder UDP arbeiten, kommunizieren können.

Diese Sockets setzen auf TCP/IP auf, d.h. für die Übertragung gelten alle für TCP/IP geltenden Eigenschaften. Das sind zum einen das Routing zum Zielrechner und eventuell notwendige Änderungen im Routing oder auch Statusabfragen, wie sie durch das *Internet Protokoll* (IP) implementiert werden. Ferner werden aber auch die Vorzüge des *Transmission Control Protokol* (TCP) genutzt, d.h. es findet eine Fehlererkennung und -korrektur auf dem gesamten Übertragungsweg statt und es wird gewährleistet, dass alle Pakete den Zielrechner erreichen und das diese Pakete sich in der richtigen Reihenfolge befinden.

Andererseits bietet TCP keinerlei Zusicherungen in Bezug auf die Übertragungszeit[12], so dass TCP bei Übertragungsproblemen schnell zum Flaschenhals werden kann.

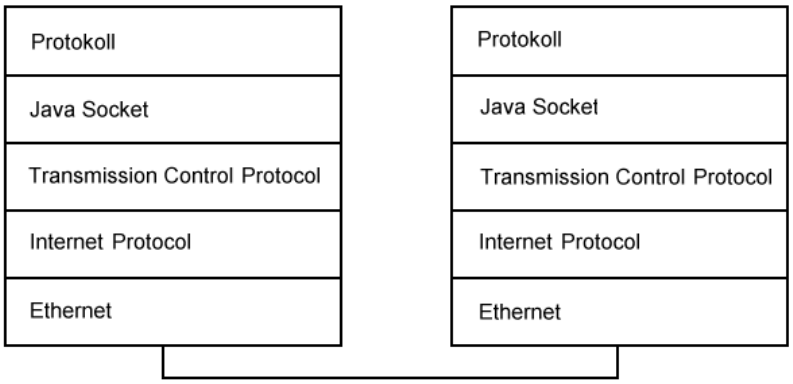


Abbildung 4.7: *Schema der Datenübertragung im Netzwerk. Auf dem Ausgangsrechner werden die Daten in OutputStreams umgewandelt und dann mittels des im Programm implementierten Protokolls übertragen. Dieses Protokol nutzt die in Java implementierten Sockets, die widerum auf TCP/IP aufsetzen. Auf dem Zielrechner durchlaufen die Daten dann alle Schritte nochmals in umgekehrter Reihenfolge.*

Der Datentransfer im Netzwerk ist also in Java sehr komportabel implementiert. Die zu übertragenden Daten werden auf dem Ausgangsrechner vor der Übertragung in *OutputStreams* umgewandelt. Reihenfolge und Länge der Daten können dabei festgelegt werden. Der Zielrechner emfängt die Daten als *InputStreams*, aus denen die ursprünglichen Daten leicht wiedergewonnen werden können. Auf diese Weise lassen sich leicht eigene Protokolle in Java implementieren, die auf die Sockets zugreifen.

[12]Das andere Extrem ist UDP, das zwar eine bessere Performance als TCP bietet, allerdings nicht korrekte Reihenfolge bzw. das Eintreffen aller Pakete beim Zielrechner gewährleistet.

4.6 Die Benutzeroberfläche

Vor der Kompression der Daten hat der Benutzer die Möglichkeit, durch die Einstellung einiger Parameter Einfluss auf die Auswahl der Daten und auf die Kompression zu nehmen. Dies geschieht mit Hilfe der Eingabefelder der Benutzeroberfläche. In erster Linie sind das natürlich die Abmessungen des zu übertragenden Datenblockes, für den jeweils Anfangs- und Endpunkt in jeder Dimension[13] anzugeben ist.

Ferner ist es möglich, die Grösse der Blöcke vorzugeben, die seperat komprimiert werden sollen, d.h. der ausgewählte Datenblock wird während Kompression und Übertragung nochmals in kleinere Teile zerlegt. Es wird später noch gezeigt, dass die Wahl der Grösse dieser Teilblöcke einen nicht zu unterschätzenden Einfluss auf das Kompressionsverhalten des Algorithmus hat.

Ebenso beeinflusst die Auswahl des Waveletfilters Qualität und Kompressionsverhalten. Zur Zeit besteht die Möglichkeit, zwischen dem *DAUB4*- und *DAUB12*-Filter (siehe dazu auch Abschnitt 4.2.1) zu wählen, die spätere Einbindung weiterer Filter ist jedoch möglich.

Die wichtigsten Einstellungen in Bezug auf die Kompression und die Qualität der Daten sind schliesslich die Quantisierungsoptionen. Die Quantisierung kann sowohl bei der Karhunen-Loeve Transformation als auch bei der Wavelet Transformation durchgeführt werden. Die Durchführung ist dabei, wie im Abschnitt 4.2 beschrieben, d.h. bei der KLT werden die Komponenten mit der geringsten Energie auf 0 gesetzt, bei der DWT wird eine *deadzone*-Quantisierung durchgeführt.

Ein Überblick über die einstellbaren Parameter wird noch einmal in Tabelle 4.1 gegeben.

Nach der Visulisierung von Daten ist es ausserdem möglich, diese zu drehen, so dass auch die spektrale Dimension betrachtet werden kann. Ausserdem können die Daten auch abgespeichert werden, um sie später beispielsweise in GIS weiterverarbeiten zu können.

Eine Abbildung der Benutzeroberfläche ist im Anhang auf Seite 67 dargestellt.

4.7 Der Algorithmus

4.7.1 Kompression und Dekompression

Die auf den vorhergehenden Seiten beschriebenen verlustbehafteten und verlustfreien mathematischen Verfahren zur Datentransformation und -kompression werden nun kombiniert, um einen guten Kompressionsalgorithmus

[13] Die Dimensionen werden mit *layers*, *rows* und *columns* bezeichnet. *Rows*(Zeilen) und *columns*(Spalten) bilden dabei die beiden räumlichen Dimensionen, während *layers* die spektrale Ausdehnung angibt.

Parameter	Funktion
Blocksize	Die Blockgrösse, in die der ausgewählte Datenblock zur Kompression und Übertragung nochmals unterteilt wird
WaveletKern	Der verwendete Waveletfilter.
KL-Quantize	Die bei der KLT durchgeführte Quantisierung. $k = 0$: keine Quantisierung, $0 < k < L$: die k kleinsten Komponenten jedes Vektors werden auf 0 gesetzt, L ist die Gesamtzahl der Komponenten (siehe dazu auch Abschnitt 4.2.2).
WT-Quantize	Die bei der DWT durchgeführte Quantisierung. $n = 0$: keine Quantisierung, $n > 0$: Quantisierung mit n Quantisierungsstufen, $Deadzone = 1.9$ ist fest eingestellt (siehe Abschnitt 4.2.1).

Tabelle 4.1: *Überblick über die Kompressionsparameter, die mittels Benutzeroberfläche einstellbar sind.*

für Hyperspektraldaten zu konstruieren. Aufgrund der Vorbetrachtungen in Abschnitt 4.1 wird bereits recht klar vorgegeben, welche Kombinationen der vorgestellten Verfahren unter bestimmten Bedingungen sinnvoll sind.

Wurden die gewünschten Voreinstellungen im GUI getroffen, wird die Anforderung an den Server übertragen. Dort werden die gewünschten Daten aus der Datenbank gelesen und in Teilblöcke aufgespalten.

Diese Unterteilung der Daten ist wichtig für das Verständnis der weiteren Ausführungen. Die ausgewählten Daten werden nicht als Ganzes komprimiert, sondern (wie bereits erwähnt) in Teilblöcke unterteilt. Diese Teilblöcke haben die in der Benutzeroberfläche angegebene Kantenlänge und werden seperat komprimiert und übertragen. Zum einen wird dadurch gewährleistet, dass das Datenvolumen bei den einzelnen Transformationen nicht zu gross und die Berechnungen nicht zu aufwendig werden[14], zum anderen verändert sich die Qualität quantisierter Daten bei verschiedenen Blockgrössen.

Die Daten jedes Teilblocks werden als erstes KL-transformiert und gegebenenfalls quantisiert. Anschliessend folgt eine Wavelet Transformation mit dem eingestellten Waveletfilter. Die Transformation wird sooft durchgeführt, wie es mit Hilfe des ausgewählten Filters möglich ist[15]. Auch hier kann anschliessend gegebenenfalls eine Quantisierung durchgeführt werden.

Abschliessend folgt eine *run-length* Kodierung und eine Arithmetische Kodierung der Daten, durch die die transformierten und quantisierten Daten nochmals komprimiert werden. Wie bereits im Abschnitt 4.2 beschrieben, arbeiten diese beiden Kompressionsverfahren verlustfrei, allerdings sind

[14]Insbesondere bei einer Blockgrösse von 256×256 Pixel und ausreichender spektraler Tiefe können dennoch während der KLT Speicherprobleme auftreten. Zu Testzwecken ist diese Einstellung dennoch implementiert.

[15]Die Transformationslevel sind durch die Länge des verwendeten Filters beschränkt. Siehe dazu auch Abschnitt 4.2.1

die damit erreichten Kompressionsraten vergleichsweise gering, während die durch Quantisierung der Daten erzielten Kompressionsraten sehr hoch sein können.

Die komprimierten Daten werden nun zusammen mit den zur Dekompression benötigten Werten[16] an den Benutzer übertragen. Das dafür definierte Protokoll ist ebenfalls Teil des Programms und setzt auf die in Java definierten Sockets auf (siehe dazu auch Abschnitt 4.5).

Die Dekompression der Daten erfolgt nun auf der Client-Seite, d.h. die für die Dekompression benötigte Zeit hängt maßgeblich von der Rechenleistung des Client-Rechners ab. Bei der Dekompression werden im wesentlichen alle auf der Serverseite durchgeführten Transformationen und Kodierungen in umgekehrter Reihenfolge invers durchgeführt. Konkret werden die Daten also zunächst arithmetisch dekodiert. Anschliessend werden die *runlength* kodierten Daten expandiert und die so gewonnenen Daten dann invers wavelet- und kl-transformiert. Wurde bei der Abarbeitung des Kompressionsalgorithmus an einer Stelle eine Quantisierung durchgeführt, sind die Daten nun verlustbehaftet.

Eine Visualisierung des beschriebenen Kompressions- und Dekompressionsalgorithmus wird auf Seite 49 dargestellt. Eine nähere Betrachtung der Parameter, die Einfluss auf Kompression und Übertragung der Daten haben, folgt in Kapitel 5.

4.7.2 Umsetzung des Algorithmus

Der beschriebene Algorithmus wurde bei der Umsetzung in Java modular aufgebaut. Jede Transformation bzw. jedes Kompressionsverfahren wurden dafür in jeweils eigenen Klassen implementiert, die in der Regel wie eine *Black Box* aufgebaut sind, d.h. es existiert eine Methode zur Kompression / Transformation und eine Methode zur Dekompression / inversen Transformation. Diesen Methoden müssen im Allgemeinen nur die zu bearbeitenden Daten übergeben werden, als Ausgabe erhält man dann die transformierten bzw. komprimierten Daten.

Die einzige Ausnahme bildet dabei die KL Transformation, die zusätzlich zu den transformierten Daten auch noch Mittelwerte und Transformationsmatrizen zurückgibt.

Bedingt durch diesen Aufbau, ist es sehr einfach, einzelne Teile des Algorithmus zu ersetzen bzw. Teile hinzuzufügen (etwa zusätzliche Wavelet-Filter). Sollte für eines der Teilverfahren also später ein besserer Algorithmus zur Verfügung stehen, ist es lediglich notwendig eine Interfacemethode für Kompression bzw. Dekompression zu implementieren, ohne Seiteneffekte auf andere Teile des Programms befürchten zu müssen.

[16] Beispielsweise die Transformationsmatrizen für die KL-Transformation, der verwendete Waveletfilter, die Grösse der Teilblöcke usw.

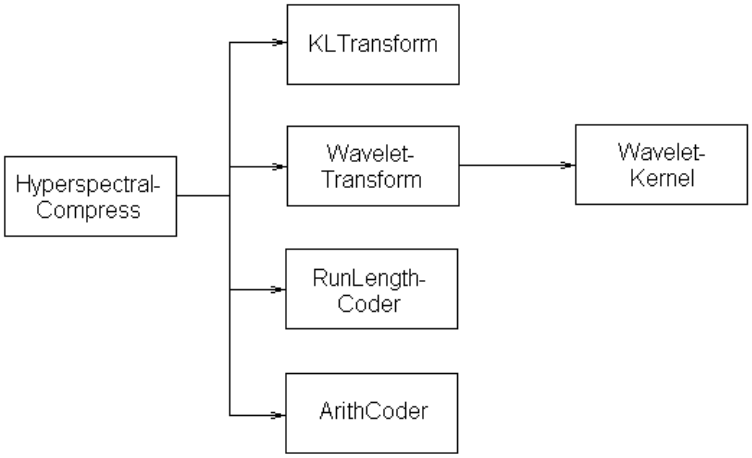


Abbildung 4.8: *Beispiel für den modularen Aufbau der einzelnen Programmteile anhand der für den Kompressionsalgorithmus benötigten Klassen.*

In Abbildung 4.8 wird der allgemeine Aufbau anhand des Kompressionsalgorithmus einmal visualisiert.

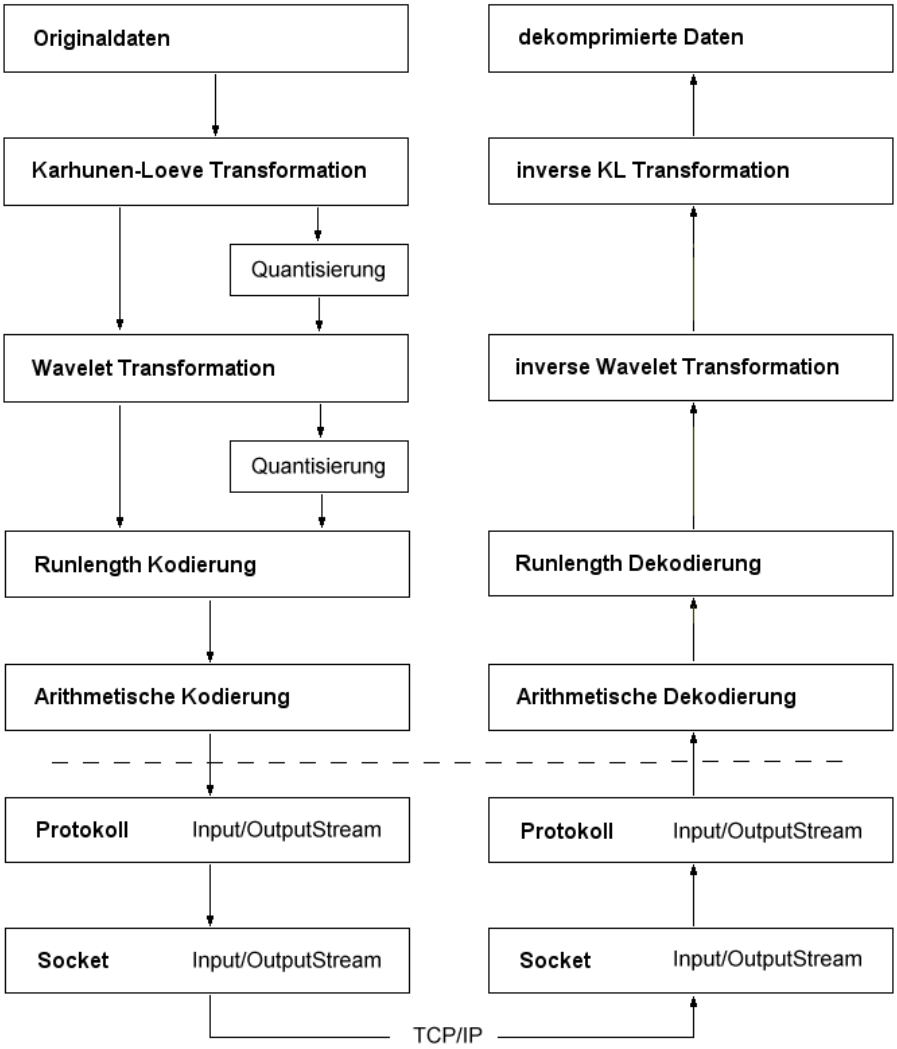


Abbildung 4.9: *Schema des verwendeten Kompressions- und Dekompressionsalgorithmus.*

Kapitel 5

Auswertung

5.1 Die Testdatensätze

Für diese Arbeit standen zwei hyperspektrale Datensätze zu Testzwecken zur Verfügung. Im folgenden soll kurz erklärt werden, mit welchen Sensoren diese Daten aufgenommen wurden und welche Eigenschaften sie haben.

5.1.1 DAIS

Das *Digital Airborne Imaging Spectrometer* (DAIS7915) ist ein in Europa entwickelter Sensor, der seit 1995 zur Gewinnung von Fernerkundungsdaten von Flugzeugen aus eingesetzt wird. Die gewonnenen Daten dienen unter anderem zur Karthographierung von Land und Gewässern, Erforschung von Bodenschätzen und um Daten für geographische Informationssysteme (GIS) zu sammeln. Mit Hilfe eines Beugungsgitters lassen sich damit hyperspektrale Daten im Bereich von 450 - 2500 nm auf 72 Kanäle verteilt aufnehmen. Die aufgenomme Strahlung wird dabei in drei Intervalle unterteilt. Im erste Intervall sind 32 Bänder auf den Bereich 400 - 1200 nm verteilt, die Bandbreite beträgt hier 15 - 30 nm. Das zweite Intervall deckt den Bereich von 1500 - 1800 nm mit acht Bändern und einer Bandbreite von 45 nm ab und das dritte Intervall, abermals mit 32 Bändern, den Bereich von 2000 - 2500 nm mit einer Bandbreite von 20 nm. Die Lücken zwischen den Intervallen stimmen mit dem Bereich der atmosphärischen Absorption überein. Ferner gehören zum DAIS weitere Sensoren, die die gleiche Optik nutzen. Das sind ein einzelner Breitband-Kanal der den Bereich von 3000 - 5000 nm abdeckt sowie sechs Kanäle, die im Bereich von 8000 - 14000 nm arbeiten. Diese Bänder beinhalten wichtige Informationen über thermale Emissionen von Bodenobjekten. Bei niedriger Flughöhe können Aufnahmen mit 1 m Auflösung in Streifen von einigen Kilometern Breite gemacht werden [20] [9].

Nähere Informationen zum DAIS-Sensor finden sich auf der Homepage

des Deutschen Forschungszentrums für Luft- und Raumfahrt[1]. Der Aufbau eines DAIS-Sensors wird im Anhang auf Seite 72 dargestellt.

5.1.2 HyMap

Der *Hyperspectral Mapper* (HyMap) wurde ursprünglich in Australien entwickelt. Als Platform dient auch hier ein Flugzeug, beispielsweise ein leichtes zweimotiges Modell wie eine Cessna 404. Der Einsatz dieser Sensoren zu kommerziellen Zwecken ist nicht ungewöhnlich. Der HyMap Sensor deckt den Spektralbereich von 0.45 - 2.5 μm nahezu zusammenhängend ab und hat eine hohe Leistungsfähigkeit in Bezug auf Bildqualität und *signal to noise ratio*. Ferner verfügt der Sensor über zwei thermale Bänder im Bereich von 3 - 5 μm und 8 - 10 μm.

Der Begriff HyMap bezeichnet jedoch nicht ein spezifisches Instrument, sondern eher das Design und Konfigurations-Konzept des Sensors. Durch einen modularen Aufbau ist es möglich, den spektralen und räumlichen Charakteristiken des Sensors genau auf die Bedürfnisse der Betreiber anzupassen. So kann beispielsweise die Anzahl der Bänder zwischen 100 und 200 variiert werden, die räumliche Auflösung bewegt sich im Bereich von 2 - 10 m und die spektrale Bandbreite kann zwischen 10 und 20 nm variieren. Der Sichtbereich bewegt sich dabei zwischen 60 und 70 Grad [4].

Auch bei diesem Sensor wird der Spektralbereich wieder in mehrere Intervalle unterteilt. Beim HyMap-Sensor sind das die folgen vier Bereiche: Das sichtbare Spektrum (VIS) von 0.45 - 0.89 μm, der *near infrared* Bereich (NIR) von 0.89 - 1.35 μm und der *short wave infrared* Bereich (SWIR), der in SWIR1 (1.40 - 1.80 μm) und SWIR2 (1.95 - 2.48 μm) aufgeteilt wird [11].

Die hier zu Testzwecken verwendendete Aufnahme zeigt Oberpfaffenhofen. Sie besteht aus 128 Spektralbändern (je 32 in jedem der vier Intervalle) bei einem FOV von 60 Grad (das entspricht 512 Pixel) und 1024 Scan Linien.

5.2 Die Vorverarbeitung der Daten

In Abschnitt 2.3 wurde bereits beschrieben, in welcher Weise die Daten in den einzelnen Levels der Datenverarbeitung verändert werden.

Die dieser Arbeit zugrundeliegenden Testdatensätze, auf die in Kapitel 5.1 bereits eingegangen wurde, liegen nach dieser Einteilung als *Level 1a*-Daten vor. Die ursprünglichen Daten waren im BSQ- bzw. HDR-Format abgespeichert, d.h. die eigentlichen Messdaten lagen in einer Binärdatei vor. In der zugehörigen Header-Datei waren Zusatzinformationen über den Datensatz und die einzelnen Bänder abgespeichert, beispielsweise die räumlichen und spektralen Dimensionen der Daten, die Wellenlängen der einzelnen Bänder u.v.m.

[1] http://www.op.dlr.de/dais/

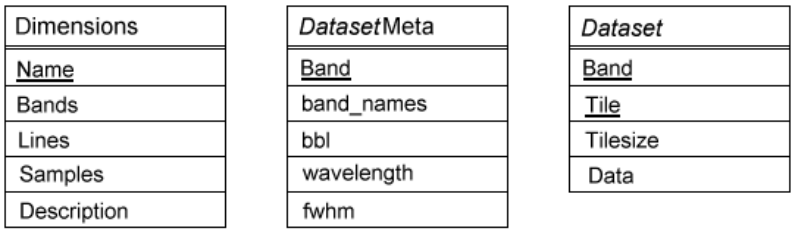


Abbildung 5.1: *Darstellung der Tabellenstruktur der verwendeten Datenbank. Die Tabelle* Dimensions *enthält Dimensionen und Beschreibung aller Datensätze,* Dataset*Meta (für Hymap also* HymapMeta*) enthält alle zusätzlichen Informationen zu den Bändern, d.h. Bandname, Wellenlänge, FWHM (*Full Width Half Maximum*) und bbl (*Barrel*). In der Tabelle* Dataset *sind schliesslich die Daten der einzelnen Bänder gekachelt gespeichert. Durch die Zusatzinformationen Tile und Tilesize ist es ausserdem möglich, dass auch verschieden grosse Layer oder Layer mit unterschiedlicher Kachelgrösse innerhalb eines Datensatzes in der Datenbank abgelegt werden können.*

Diese Daten wurden in eine Datenbank übertragen, so dass der Zugriff auf die Daten nun schneller und einfacher ist als zuvor. Ein grosser Nachteil der Binärdatei war die Zugriffsdauer, die aufgrund der Grösse der Dateien[2] um so länger war, je grösser die Wellenlänge (d.h. je weiter hinten die Daten im Datensatz standen) war.

Als Datenbank wurde *InterBase*[3] verwendet, die Tabellenstruktur wurde möglichst einfach gehalten. In der Tabelle *Dimensions* werden zu jedem Datensatz Name und Dimensionen abgespeichert. Für jeden in dieser Tabelle enthaltenen Datensatz werden zwei weitere Tabellen angelegt, von denen die eine die Informationen zu den einzelnen Bändern enthält (z.B. die Wellenlänge) und die andere die eigentlichen Binärdaten. Diese Binärdaten sind in Kacheln von 64×64 Pixeln als *Binary Large Objects* (BLOB) abgespeichert, die ursprünglichen Daten wurden also nicht verändert, sondern lediglich umgeordnet. Der genaue Aufbau des Datenbankschemas wird in Abbildung 5.1 nochmals vollständig dargestellt.

Durch die Übertragung in die Datenbank ist das Datenvolumen nochmals angewachsen, jedoch wird dieser Nachteil durch den nun stark beschleunigten und wahlfreien Zugriff auf die Daten aufgewogen[4]. Hinzu kommt,

[2]Die Datei DAIS.bsq hat beispielsweise ein Grösse von 176 MB

[3]Nähere Informationen zu *Borland Interbase* finden sich unter http://www.interbase.com

[4]Vergleich der Zugriffsgeschwindigkeit anhand des HyMap-Datensatzes:
Für das Lesen aus der Binärdatei wurden folgende Zeiten gemessen: Das Lesen von Band 0 (also das erste Band in der Binärdatei) dauert 4 Sekunden, bei Band 60 sind es bereits 10 Sekunden. Zum Lesen von Band 127 (das letzte Band) werden schliesslich sogar 20 benötigt.
Beim Lesen aus der Datenbank ist die Zugriffsgeschwindigkeit für alle Bänder gleich. Zum

dass bei der Übertragung an einen Benutzer auch nur die komprimierten Binärdaten übertragen werden, d.h. das Datenvolumen macht sich nur bei der Abspeichung auf dem Server negativ bemerkbar.

5.3 Messparameter

Bei einem Vergleich der komprimierten Daten nach der Übertragung spielen drei Faktoren eine Rolle, die hier einmal beschrieben werden sollen:

- **Kompressionsrate:** Die Kompressionsrate beschreibt das Verhältnis zwischen der Grösse der Originaldaten und der Grösse der Daten nach der Kompression. Offensichtlich ist es wünschenswert, eine möglichst grosse Kompressionsrate zu erzielen. Wie bereits in Kapitel 3.3 beschrieben, muss dafür jedoch in der Regel ein Datenverlust in Kauf genommen werden. Ob dieser Verlust akzeptabel ist, muss vom jeweiligen Anwender entschieden werden.

- **Qualität:** Dieser Parameter gibt an, wie gut die Qualität der dekomprimierten, verlustbehafteten Daten ist. Mögliche objektive Maße sind beispielsweise der MSE oder die PSNR (siehe dazu auch Abschnitt 3.3.3), mit deren Hilfe sich die Qualität der Daten vergleichbar wird. Subjektiv ist hier aber auch die Qualität des Bildes bei einer Visualisierung der Daten wichtig.

- **Zeit:** Die Zeit, die vom Anfordern der Daten bis zur Visualisierung beim Benutzer benötigt wird. Darin enthalten ist also nicht nur die Zeit, die der eigentliche Datentransfer benötigt, sondern auch der Zugriff auf die Datenbank und die zur Kompression und Dekompression benötigte Zeit. Insbesondere sind dabei natürlich die Kompressions- und Dekompressionszeit von Interesse, da zu erwarten ist, dass die Transferrate nicht direkt beeinflussbar ist. In der weiteren Betrachtung werden daher auch die einelnen Zeiten getrennt gemessen und betrachtet.

Aufgrund der verschiedenen Verfahren, die bei der Kompression der Daten verwendet werden, ist es ausserdem interessant, welche Variationen und Kombinationen dieser Verfahren dazu führen, dass die Qualität der Daten bei einer gegebenen Kompressionsrate möglichst optimal ist.

Ferner sollte bestimmt werden, welche Kompressionsparameter bei verschiedenen Benutzergruppen sinnvoll sind. Ein interessierter Laie (etwa ein Student) ist vermutlich nur einem optisch guten Ergebnis interessiert, während bei einer geowissenschaftlichen Anwendung bestimmte Merkmale für

Lesen eines einzelnen Bandes werden hier 4 Sekunden benötigt.

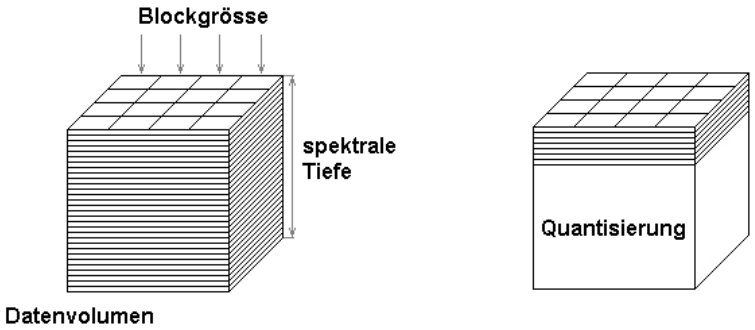


Abbildung 5.2: *Darstellung der Parameter mit Einfluss auf die Karhunen-Loeve Transformation.*

den jeweiligen Verwendungszweck interessant sind. Hier ist eine Unterteilung in verschiedene Nutzerprofile notwendig, um jedem Anwender möglichst schnell ein für ihn gutes Ergebnis zu liefern und die Manipulationsmöglichkeiten zu bieten, die er erwartet.

Im weiteren soll nun auf die Parameter der einzelnen Teilverfahren eingegangen werden, die mehr oder weniger offensichtlich die oben aufgeführten Faktoren verändern können.

Der Parameter, der wohl am offensichtlichsten Einfluss auf die Übertragungenszeit hat, ist die Grösse der angeforderten Daten. Die Bearbeitung eines grösseren Datenvolumens benötigt bei allen verwendeten Teilverfahren mehr Zeit, als die Bearbeitung von nur wenigen Daten.

Ein weiterer Faktor mit Einfluss auf alle Teile des Verfahrens ist die verwendetete Blockgrösse. Hier gilt es festzustellen, inwiefern durch diesen Parameter die einzelnen Verfahren beeinflusst werden und ob sich eine optimale Blockgrösse für das Kompressions- bzw. Zeitverhalten finden lässt.

An dieser Stelle soll nun auf die Variationsmöglichkeiten der einzelnen Teilverfahren eingegangen werden und wie sich Veränderungen auf das Kompressions- und Zeitverhalten des jeweiligen Verfahrens auswirken.

Karhunen-Loewe Transformation

Bei der KL Transformation gibt es eine Reihe von Parametern, die durch den Benutzer einstellbar sind. Neben dem bereits erwähnten Datenvolumen und der Blockgrösse ist hier vor allem die spektrale Tiefe der ausgewählten Daten ein wichtiger Faktor.

Es lässt sich leicht überlegen, dass die Blockgrösse einen Einfluss auf die Berechnung der Mittelwerte haben wird, der aber aufgrund der einfachen Operationen bei der Berechnung voraussichtlich auch bei grossen Blockgrössen nicht signifikant sein wird. Hinzu kommt jedoch, dass bei grösserer

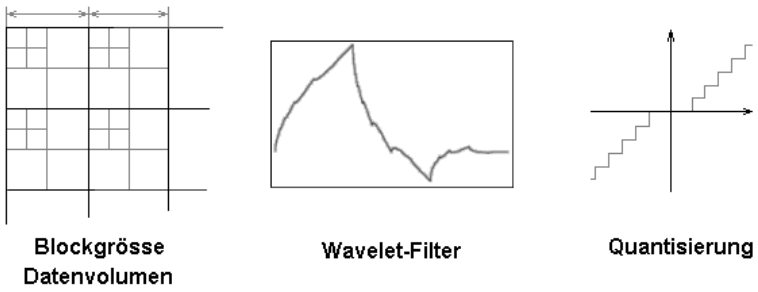


Abbildung 5.3: *Darstellung der Parameter mit Einfluss auf die Wavelet Transformation.*

Blockgrösse mehr Vektoren mit der gleichen Matrix transformiert werden, an dieser Stelle der Berechnungsaufwand also sinkt. Und schliesslich sollte man bedenken, dass sich bei grösseren Blöcken der Abstand zwischen Mittelwert und Vektoren vergrössern wird und die durch die Transformation erzielte Energiekonzentration nicht so optimal ist wie bei kleinen Blöcken.

Einen grösseren Einfluss auf die Berechnung hat die spektrale Tiefe der ausgewählten Daten. Bei Beachtung der Tatsache, dass bei einer spektralen Tiefe von N die Eigenvektoren einer $N \times N$ Matrix zur Transformation berechnet werden müssen, so kann schlussfolgern, dass für grosse N die Berechnungszeit exponentiell zunimmt.

Und schliesslich hat auch die Quantisierung der Daten einen erheblichen Einfluss auf Kompressionsrate und auf die Kompressionszeit der später folgenden Verfahren, da die Daten hier unter Umständen stark vereinfacht werden. Da sich die KLT optimal in Bezug auf die Energiekonzentration der transformierten Daten verhält, ist es offensichtlich, dass sich Daten mit grösserer spektraler Tiefe ohne signifanten Qualitätsverlust stärker quantisieren lassen. Ebenso wie bei allen anderen bisher erwähnten Faktoren werden Messungen belegen, in welchem Maße sich die jeweiligen Variablen auf die Qualität der transformierten Daten sowie auf Kompressionrate und -zeit auswirken.

Wavelet Transformation

Da die Wavelet Transformation nur im für die räumlichen Dimensionen der Daten angewand wird, folgt sofort, dass der Kompressionsaufwand linear mit wachsendem Datenvolumen steigen wird. Interessant hingegen ist der Einfluss der Blockgrösse auf das Verfahren. Auch hier folgt aus grösseren Blöcken eine aufwendigere Berechnung, allerdings ist die für die DWT benötigte Zeit im Vergleich zu den anderen Verfahren eher gering. Der tatsächliche Einfluss auf die Kompressionszeit wird also ebensfalls gering sein. Jedoch kann bei grösseren Blöcken die Wavelet Transformation öfter

durchgeführt werden und aufgrund dessen ist eine bessere Dekorrelation der Daten möglich. Diese Dekorrelation hat, wie bereits bei der KLT, Einfluss auf die Qualität der Daten nach einer Quantisierung.

Und schliesslich hat auch der verwendete Waveletfilter einen Einfluss auf das Verhalten der Transformation, den es zu messen gilt. So werden beispielsweise bei kürzeren Filtern die Daten besser dekorreliert, da der Filter öfter angewendet werden kann.

Lauflängen Kodierung

Während bei den beiden vorangegangenen Verfahren aufgrund der Parametervielfalt eine Reihe von Überlegungen angestellt werden konnten, gibt es beim *run-length coding* keine Parameter, auf die direkt Einfluss genommen werden kann. Natürlich gilt auch hier, dass mit zunehmendem Datenvolumen auch die Berechnungszeit steigt, aber ähnlich wie bei der DWT ist diese selbst bei grossen Datenmengen nicht signifikant.

Jedoch haben die gegebenenfalls nach den beiden Transformationen durchgeführten Quantisierungen einen ganz entscheidenden Einfluss auf den Erfolg des Verfahrens. Je stärker die Daten quantisiert wurden, um so erfolgreicher wird eine nun folgende Lauflängen-Kodierung sein, während bei nichtquantisierten Daten das Datenvolumen nach Anwendung dieses Verfahrens sogar grösser sein kann als zuvor.

Arithmetische Kodierung

Der letzte Schritt ist schliesslich die arithmetische Kodierung. Auch hier gibt es keine direkten Parameter, auf die der Anwender Einfluss nehmen kann, jedoch gilt auch hier, dass der Erfolg des Verfahrens von den vorangegangenen Schritten abhängig ist.

Das Datenvolumen hat hier einen spürbaren Einfluss auf die Kompressionszeit, da diese mit zunehmendem Datenvolumen linear steigt, Eine Quantisierung der Daten und die damit einhergehende Verkleinerung des Datenvolumens während der Lauflängenkodierung wirkt sich also sehr vorteilhaft auf die Laufzeit der arithmetischen Kodierung aus.

Da auch dieses Verfahren, ebenso wie die Lauflängen-Kodierung, verlustfrei arbeitet, ist auch hier eine Qualitätsbetrachtung nicht notwendig.

5.4 Ergebnisse

Bevor nun auf den Einfluss der obengenannten Parameter auf die Faktoren Kompressionsrate, Qualität und Zeit eingegangen wird, soll kurz betrachtet werden, wie sich das Verfahren bei verlustfreier Kompression verhält:

Die Kompressionsrate ist in diesem Fall sehr gering. Mit zunehmender spektraler Tiefe verbessert sich die Rate aufgrund der besseren Dekorrela-

tion ein wenig, aber selbst unter dieser Bedingung beträgt die Kompressionsrate nur ca. 1.05 : 1. Das ist teilweise der verwendeten Lauflängenkodierung geschuldet, die den Code bei fehlender Quantisierung teilweise sogar verlängert.

Ebenso wie die Kompressionsrate durch zunehmende spektrale Tiefe verbessert wird, kann sie auch durch grössere Teilblöcke verbessert werden. Allerdings ist auch dieser Effekt vergleichsweise gering und durch den gesteigerten Rechenaufwand wird auch für die Kompression der Daten geringfügig mehr Zeit benötigt.

Der Zeitaufwand für Kompression, Transfer und Dekompression steigt bei zunehmendem Datenvolumen nahezu linear (Abb. 6.7), d.h. man kann anhand eines gemessenen Wertes sehr genau abschätzen, wieviel Zeit für ein bestimmtes Datenvolumen benötigt wird.

Diese Aussage verliert natürlich ihre Gültigkeit, wenn die Daten zur besseren Kompression quantisiert werden. Welchen Einfluss diese Quantisierung und all die anderen in Abschnitt 5.3 genannten Parameter nun genau auf das Zeit- und Kompressionsverhalten sowie auf die Qualität der Daten haben, soll wieder anhand der einzelnen Teilverfahren erklärt werden.

Karhunen-Loeve Transformation

Der interessanteste Aspekt der KLT in Bezug auf das Gesamtverfahren ist offenbar die Qualität der Daten nach einer Quantisierung.

Die Quantisierung selbst benötigt nur einen Bruchteil der Kompressionszeit[5], aber aufgrund der daraus folgenden Reduzierung des Datenvolumens ist die für die folgenden Verfahren benötigte Zeit deutlich geringer.

Die Qualität ist trotz des Datenverlustes vergleichsweise gut und nimmt auch bei stärkerer Quantisierung nur langsam ab. Abbildung 6.12 zeigt Kompressionsrate und Qualität der dekomprimierten Daten bei Quantisierung nach der KLT.

Aufgrund der Verfahrensweise bei einer Quantisierung nach der KL-Transformation lassen sich die Daten natürlich um so besser komprimieren, je grösser ihre spektrale Tiefe ist (Abb. 6.13). Diese spektrale Tiefe ist aber auch, wie bereits vermutet wurde, ein wichtiger Punkt, wenn man die für die Kompression benötigte Zeit betrachtet (Abb. 6.11).

Hier spielt auch die verwendete Blockgrösse eine wichtige Rolle. Während die Qualität der Daten und die Kompressionsrate unabhängig von der Blockgrösse immer annähernd gleich sind, steigt die Rechenzeit bei grossen Blöckgrössen spürbar an (Abb. 6.9).

Da mit zunehmender spektraler Tiefe der Speicherbedarf für die KLT exponentiell wächst, können bei grossen Blockgrössen (insbesondere bei 256 ×

[5]Unabhängig von der Datenmenge wird für die Quantisierung der Daten weniger als 5 Prozent der Gesamtzeit benötigt.

256 Blöcken) bereits bei der Auswahl von nur wenigen Layern bereits Speicherprobleme auftreten.

Zusammenfassend lässt sich sagen, dass die Zeitersparnis bei den folgenden Verfahren die bei einer grossen spektralen Tiefe verlorene Zeit mehr als aufwiegt, wenn im Anschluss an die KLT die Daten quantisiert werden. Ferner ist eine kleine Blockgrösse bei der Kompression von Vorteil.

Die für die inverse KLT benötigte Zeit ist aufgrund der vorberechneten Matrizen nur sehr gering und unabhängig von der verwendeten Blockgrösse. Auch die für die Quantisierung benötigte Zeit ist nicht signifikant[6].

Wavelet Transformation

Die Wavelet Transformation ist unabhängig von der gewählten spektralen Tiefe, da hier alle *Layer* nacheinander abgearbeitet werden. Hier spielt aber, wie bereits vermutet, die Blockgrösse eine wichtige Rolle. Die damit in Zusammenhang stehenden Effekte werden jedoch nur sichtbar, wenn eine Quantisierung im Anschluss an die Wavelet Transformation durchgeführt wird. Die Kompression der Daten erzielt bei grösseren Blöcken bessere Ergebnisse als bei kleinen Blöcken. Dieser Effekt fällt besonders bei vielen Quantisierungsstufen auf.

So erzielt man beispielsweise bei 128 Stufen mit 16×16 grossen Blöcken nahezu doppelt so hohe Kompressionsraten wie bei 8×8 Blöcken. Mit zunehmender Blockgrösse nimmt dieser Effekt jedoch stark ab und ist bei grösseren Blöcken nicht mehr signifikant (Abb. 6.14). Auch bei einer stärkeren Quantisierung (d.h. mit weniger Stufen) wird dieser Effekt nicht so deutlich sichtbar.

Im Gegensatz dazu verschlechtert sich aber die Qualität der dekomprimierten Daten mit zunehmender Blockgrösse langsam. Auch dieser Effekt fällt besonders bei den kleinen Blockgrössen auf, jedoch wird hier die Ausprägung um so stärker, je weniger Quantisierungsstufen gewählt werden.

Auf die zur Transformation und Quantisierung benötigten Zeit hat die Blockgrösse hingegen keinen Einfluss.

Interessant ist auch der Einfluss des verwendeten Wavelet-Filters. Die bei Verwendung des *DAUB4*-Filters beobachteten Kompressionsraten sind geringfügig höher als die bei Verwendung des *DAUB12*-Filters erreichten. Je stärker die Daten jedoch quantisiert werden, um so mehr nähern sich die Kompressionsraten beider Filter aneinander an. Dennoch ist die Qualität der Daten bei Verwendung von *DAUB4* in jedem Fall deutlich besser. Dieser Effekt ist sogar derart signifikant, dass man bei der Verwendung grösserer Blöcke mit *DAUB4*-Filter bessere Ergebnisse erzielt als bei kleinen Blöcken in Zusammenhang mit *DAUB12* (Abb. 6.16).

Bei den gemessenen Zeiten wurden für Synthese und Analyse annähernd

[6] Zur Quantisierung von 15 MB Daten wurden im Durchschnitt 0.4 Sekunden benötigt

gleiche Zeiten gemessen, wobei die für die DWT benötigte Zeit vergleichsweise gering ist[7].

Lauflängen-Kodierung

Zu diesem Teilverfahren gibt es keine überraschenden Beobachtungen. Die zur Kompression benötigte Zeit ist sehr gering[8], zur Dekompression wird sogar noch weniger Zeit benötigt.

Natürlich wird hier der Hauptteil der Datenreduktion durchgeführt, die erzielten Ergebnisse sind jedoch allein von den zuvor durchgeführten Quantisierungen abhängig. Wie bereits im Abschnitt 5.3 bemerkt wurde, kann das Datenvolumen ohne Quantisierung bei diesem Verfahren sogar zunehmen.

Arithmetische Kodierung

Für die abschliessende arithmetische Kodierung wurde eine Java Implementierung des Source Codes aus dem CACM Artikels *"Arithmetic coding for data compression"* von Witten, Neal and Cleary aus dem Jahr 1987 verwendet [18].

Dieser arithmetische Coder fällt leider vor allem durch die dafür benötigte Zeit auf. Während die zur Kompression benötigte Zeit zwar sehr lang aber noch akzeptabel ist, wird für die Dekompression der Daten mehr Zeit benötigt, als für eine Internetanwendung tolerierbar ist. So dauert die Kompression von zwei Megabyte Daten ca. 10 Sekunden, für die Dekompression werden hingegen 80 Sekunden benötigt. Werden die Daten quantisiert, so ist die für dieses Verfahren benötigte Zeit aufgrund der Datenreduktion während der Lauflängenkodierung natürlich ungleich kürzer. D.h. die Dekompressionszeit verhält sich umgekehrt proportional zum Informationsverlust.

Um die Applikation dennoch für Internetanwendungen interessant zu machen, wurde deshalb die Benutzung eines sogenannten *Fast Mode* für die Kompression in die Benutzeroberfläche integriert. In dieser Variante wird die arithmetische Kompression und Dekompression übersprungen und das gesamte Verfahren daher um ein Vielfaches beschleunigt[9]. Natürlich ist die auf diese Weise erzielte Kompressionsrate geringer als bei der Durchführung des gesamten Verfahrens.

[7] Da die benötigte Zeit weder von spektraler Tiefe noch von Blockgrösse abhängt, kann man für die Transformation von 15 MB Daten ca. 3 Sekunden annehmen, für eine folgende Quantisierung wird nochmal ca. eine Sekunde benötigt.

[8] Die Kompression von 5 MB Daten dauert ca. 0.5 Sekunden.

[9] Die Dekompression von 5 MB Daten wird damit statt in 3 Minuten in ca. 3 Sekunden vorgenommen.

5.5 Benutzerprofile

Neben der Implementierung des Algorithmus und den obigen Aussagen in Bezug auf Zeit, Qualität und Kompression der Daten bei verschiedenen Einstellungen war ein Ziel dieser Arbeit, Vorschläge für geeignete Einstellungen bei typsischen Benutzergruppen zu machen. Diese Einstellungen sollen nun abschliessend diskutiert werden.

Als typsche Benutzergruppen wurden dabei "interessierte Laien" und Experten ausgewählt. Als Laien sollen in diesem Zusammenhang Benutzer angesehen werden, die lediglich daran interessiert sind, die Daten anzuschauen, um beispielsweise die Visualisierung des Zielgebietes in verschiedenen Bereichen des elektromagnetischen Spektrums zu vergleichen.

Experten hingegen sind auch an einer Auswertung der Daten interessiert. Sie werden in der Regel die übertragenen Daten abspeichern und mit Hilfe geeigneter Programme verändern, so dass Eigenschaften des Zielgebietes festgestellt werden können, wie sie in Kapitel 2 beschrieben werden.

Die auf den vorangegangenen Seiten beschriebenen Auswirkungen der einzelnen Parameter lassen bereits darauf schliessen, dass für beide Gruppen sehr unterschiedliche Einstellungen geeignet sind.

"Interessierte Laien"

Ist der Benutzer lediglich an einem visuell ansprechendem Ergebnis interessiert, etwa für eine Darstellung in einem Webbrowser, lässt sich die Frage der optimalen Quantisierung relativ leicht beantworten.

Für die KLT kann in diesem Fall eine sehr starke Quantisierung vorgenommen werden. Selbst bei einer grossen spektralen Tiefe, wie etwa 50 Layer, ist die Qualität der Daten visuell noch immer akzeptabel, wenn man nur die ersten Komponenten jedes Vektors überträgt (siehe dazu auch Abbildung 6.13 im Anhang auf Seite 77).

Grundsätzlich lässt sich sagen, das die 5 Prozent der KL-transformierten Daten mit der stärksten Energiekonzentration genügen, um ein visuell hervorragendes Ergebnis für die übertragenen Daten zu liefern. Die mit Hilfe der PSNR gemessene Qualität beträgt in diesem Fall etwa 120 dB.

Für die wavelettransformierten Daten kann keine derart pauschale Aussage getroffen werden. Mit abnehmender Zahl an Quantisierungsstufen verbessert sich die Kompressionsrate sehr stark, allerdings sinkt in gleichem Maße auch die Qualität der Daten. Einen guten Kompromiss in dieser Hinsicht bilden beispielsweise 64 Stufen, aber auch mit 32 Stufen lassen sich noch gute Ergebnisse erzielen.

Die Wahl des Waveletfilters ist bei den beiden momentan implementierten *Daubechies*-Filtern einfach, da der kürzere *DAUB4*-Filter nach einer Quantisierung sowohl bessere Kompressionsraten als auch qualitativ bessere Ergebnisse liefert. Die Blockgrösse, die ja bei der KLT nahezu keine

Bedeutung hatte, sollte für die DWT klein gewählt werden. Hier kann eine Blockgrösse von 16×16 empfohlen werden. Bei kleineren Blöcken verbessert sich die Qualität der Daten um einige Dezibel, allerdings auf Kosten der Datenkompression.

Bei Anwendung der obigen Einstellungen auf die Testdatensätze wurde typischerweise ein Qualität von etwa 95 db gemessen, eine Visualisierung der Daten liefert sehr gute Ergebnisse[10]. Die Kompressionsraten sind dabei, wie schon zuvor bemerkt, in hohem Maße von der spektralen Tiefe der ausgewählten Daten abhängig. Bei der Auswahl von 20 Layern werden Raten von etwa 25:1 erzielt, bei 40 Layern bereits 45:1.

Die gegebenen Einstellungen sind natürlich nur Empfehlungen und bei stärkerer Quantisierung werden deutlich höhere Raten erzielt, wobei die Bildqualität der visualisierten Daten noch immer akzeptabel ist.

Experten

Für diese Benutzergruppe ist es nun sehr schwierig, allgemeingültige Aussagen zu treffen. Wie bereits in Kapitel 3 dargestellt wurde, existiert bisher kein Verfahren, das es ermöglicht, den Informationsverlust in verlustbehaftet komprimierten Daten zu bewerten.

Es wird daher nur eine schwache Kompression der Daten mit Kompressionsraten von 5:1 bis maximal 10:1 empfohlen[11], da bei hier visuell noch kein Datenverlust wahrnehmbar ist. Wie jedoch in Abschnitt 3.2.1 bereits erklärt wurde, kann anhand des visuellen Eindrucks keine tatsächliche Aussage über den Informationsverlust getroffen werden.

Für ernsthaft an einer Auswertung der Daten interessierte Benutzer kann folglich nur empfohlen werden, einen kleinen Ausschnitt der gewünschten Daten verlustfrei zu übertragen (etwa ein einzelner Block in der ausgewählten Blockgrösse) und mit dem verlustbehafteten Ergebnis auf eine Verwendbarkeit zu überprüfen. Sollte das Ergebnis nicht wie erwartet ausfallen, muss eine geringere Quantisierung der Daten vorgenommen werden. Dieser Schritt muss so oft wiederholt werden, bis die komprimierten Daten ein Ergebnis liefern, das das mit den Orginaldaten erzielte Ergebnis zufriedenstellend approximiert.

[10] Bei einer Visualisierung der Daten kann keine Artefaktbildung beobachtet werden, im Direktvergleich mit dem Orginal zeigen sich lediglich einige Unterschiede in sehr heterogenen Bildregionen wie etwa dem Stadtgebiet von Oberpfaffenhofen.

[11] Auf die Einstellungen im Programm bezogen heisst das: Bei den KL-transformierten Daten werden die 10 Prozent der Daten mit der stärksten Energiekonzentration betrachtet und nach der DWT wird auf 128 oder mehr Stufen quantisiert. Als Blockgrösse wird auch hier 8 oder 16 empfohlen.

Kapitel 6

Zusammenfassung und Ausblick

6.1 Zusammenfassung

Der ständig steigende Bedarf an Fernerkundungsdaten für verschiedenste geowissenschaftliche und andere fachspezifische Anwendungsbereiche und der damit verbundene Aufwand bei der Übertragung der Daten waren Ausgangspunkt der vorliegenden Arbeit.

Es wurden Beispiele für die Anwendung von Hyperspektraldaten gegeben, sowohl im geowissenschaftlichen Bereich (z.B. bei der Klassifizierung der Landnutzung eines Gebietes) als auch in der Wirtschaft (z.B. bei der Suche nach Erdölvorkommen). Ebenso wurde beispielhaft demostriert, auf welche Weise die Daten typischerweise modifiziert werden, um Eigenschaften des betrachteten Gebietes aufzeigen zu können.

Nach einer Betrachtung heute verwendeter verlustfreier und verlustbehafteter Kompressionsverfahren wurde schnell klar, dass aufgrund der Grösse solcher hyperspektralen Datensätze verlustbehaftete Techniken eingesetzt werden sollten, um eine sinnvolle Kompression der Daten durchführen zu können.

Aus den in der Literatur vorgestellten beiden grundsätzlichen Ideen zur Kompression von spektralen Daten wurde die auf Transformationskodierer mit anschliessender Quantisierung beruhende Methode ausgewählt und in Verbindung mit anderen Teilverfahren implementiert. Hierzu wurden die räumlichen Dimensionen und die spektrale Dimension getrennt betrachtet und erstere einer Wavelet Transformation unterzogen, während auf die zweite die Karhunen-Loeve Transformation angewendet wurde, da die damit erzielten Ergebnisse deutlich besser sind als bei anderen Transformationen. Bei beiden Transformationen ist eine anschliessende Quantisierung der Daten möglich, durch die ein Grossteil der letztendlich erzielten Datenkompression realisiert wird. Den Transformationen folgte die Anwendung zweier

Entropiekodierer, durch die das Datenvolumen nochmals reduziert wurde.

Dieser Algorithmus wurde in einem Java Programm in Form einer Client/Server Architektur implementiert. Anwendern ist es damit möglich, Hyperspektraldaten von einem Server anzufordern und vor der Übertragung auf einen Bruchteil ihrer tatsächlichen Grösse zu komprimieren. Der zur Dekompression notwendige Algorithmus, der die zu den genannten Verfahren gehörigen inversen Transformations- bzw. Expansionsverfahren realisiert, ist in der Client-Anwendung des Programmes vorhanden. Ferner ermöglicht dieser Client eine Visualisierung der Daten sowie die Berechnung einiger statistischer Werte wie z.B. PSNR und Kompressionsrate.

Die für die Übertragung benötigte Zeit ist aufgrund des verwendeten Algorithmus entscheidend kürzer als die Zeit, die für die Übertragung der Orginaldaten in einem Netzwerk benötigt wird.

So würde für die Übertragung eines $10 \times 512 \times 512$ grossen Ausschnitts der Daten für einen Nutzer, der über ISDN-Anschluss verfügt, mindestens 10.5 Minuten benötigt. Wird eine für nichtwissenschaftliche Auswertung der Daten realistische Kompressionsrate von 20:1 angenommen, so dauert die Übertragung der Daten weniger als eine halbe Minute.

Der bei der Kompression entstehende Datenverlust ist vom Anwender mit Hilfe der im Programm einstellbaren Parameter steuerbar, so dass jeder Anwender die Daten in der Qualität anfordern kann, in der sie benötigt werden. Es ist also auch möglich, die Daten verlustfrei zu übertragen, wobei beachtet werden sollte, dass das Programm für diese Art der Kompression nicht konzipiert ist und die erreichten Kompressionsraten nur sehr gering sind.

Schliesslich wurden noch Empfehlungen für die Anwendung des Programmes gegeben, indem für verschiedene Benutzergruppen typische Einstellungen genannt werden, mit dem zusätzlichen Hinweis, dass der Anwender selbst eine Bewertung der verlorenen Information durchführen muss.

Für die Weiterverarbeitung ist es möglich, die übertragenen Daten in Form einer Binärdatei abzuspeichern, die dann beispielsweise in geographischen Informationssystemen wie *ERDAS IMAGINE* importiert werden kann.

6.2 Ausblick

Das Programm im augenblicklichen Zustand hat zwei Schwachpunkte. Das ist zum einen die lange Rechenzeit bei der arithmetischen Dekompression und zum zweiten der Speicherbedarf bei der Karhunen-Loeve Transformation, der es verhindert, dass sehr grosse Datenmengen transformiert werden können.

Aufgrund der Architektur des Programmes, die in Abschnitt 4.7.2 kurz vorgestellt wurde, ist es jedoch möglich, ohne grosse Kenntnis des Program-

mes einzelne Teile auszutauschen, so dass eine “bessere“ Implementation dieser beiden Programmteile durchaus ohne grossen programmiertechnischen Aufwand möglich ist.

Ein weiterer interessanter Punkt wäre die Implementation weiterer Wavelet-Filter (z.B. nach dem *Lifting Schema*) und die damit erzielten Ergebnisse. Auch könnten mit Hilfe des bestehenden Programmes Betrachtungen zur Art des Informationsverlustes durchgeführt werden, der bei Quantisierung der KL- bzw. wavelettransformierten Daten entsteht. So wurde beispielsweise in [25] festgestellt, dass sich nach der Wavelet Transformation von Landsat-TM-Aufnahmen durch die Unterdrückung der hochfrequenten Strukturen einzelne Stadtareale in den Aufnahmen leichter separieren lassen. Ferner wurde eine Glättung leicht verrauschter Bildbereiche erreicht. Eine Betrachtung derartiger Effekte und die damit verbundenen Empfehlungen in Bezug auf die Parameterwahl bei der Übertragung der Daten wären Punkte, die eine weitere Betrachtung des Programmes interessant machen würden.

Und schliesslich wäre die Anwendung dieses oder eines ähnlichen Algorithmus auf andere Arten von Daten, z.B. Radar-Daten, ebenfalls eine Herausforderung, die bei der Präsentation dieser Arbeit am Institut für Physische Geographie der Universität Freiburg bereits angesprochen wurde.

Literaturverzeichnis

[1] G.P. Abousleman. Adaptive Wavelet Coding of Hyperspectral Imagery. In *Proceedings of the SPIE - The Innternational Society for Optical Engineering*, volume 2762, pages 545–556, 1996. ISSN 0277-786X.

[2] W. K. Carey and S. S. Hemami. Smoothness-Constrained Wavelet Image Compression. In *Proc. IEEE Int. Conf. on Image Processing*, Lausanne, Switzerland, September 1996.

[3] R. N. Clark. U. S. Geological Survey, 1995. U.S. Department of the Interior, http://www.usgs.gov/.

[4] T. Cocks, R. Jenssen, A. Stewart, I. Wilsion, and T. Shields. The HyMap Airborne Hyperspectral Sensor: The System, Calibration and Performance. In *1st EARSEL Workshop on Imaging Spectroscopy*, Zurich, October 1998.

[5] S. Dech. Mission zum Planeten Erde. *Deutsche Forschungsanstalt für Luft- und Raumfahrt e.V.*, 2000.

[6] B. R. Epstein, R. Hingorani, J. M. Shapiro, and M. Czigler. Multispectral KLT-Wavelet Data Compression for Landsat Thematic Mapper Images. In *Data Compression Conference*, pages 200–208, 1992.

[7] W. Schneider F. Pätzold, F. Porsch. Fernerkundung - Hintergründe und Informationen, 1996.

[8] N. K. Giang and D. Saupe. Rapid High Quality Compression of Volume Data for Visualization, 2001.

[9] M. Habermeyer. DLR's DAIS 7915 Home Page. In *Airborne Imaging Spectroscopy at DLR*. Deutsche Forschungsanstalt für Luft- und Raumfahrt e.V., October 2001. http://www.op.dlr.de/dais/.

[10] I. Ihm and S. Park. Wavelet-Based 3D Compression Scheme for Very Large Volume Data. In *Graphics Interface*, pages 107–116, 1998.

[11] Analytical Imaging and Geophysics LLC. Data Acquisition and Processing, 2000. http://www.aigllc.com/.

[12] Y. Kim and W. A. Pearlman. Lossless Volumetric Medical Image Compression. In *Proc. of SPIE, Applications of Digital Image Processing*, volume 3808, 1999.

[13] M. Melle. *Atlanten und geowissenschaftliche Anwendungen im Internet: Methoden und Frameworks für die Integration digitaler Atlanten im Web.* Dissertation, Universität Leipzig, 2002.

[14] Canada Centre of Remote Sensing. Fundamentals of Remote Sensing, 1998. http://www.ccrs.nrcan.gc.ca/ccrs/.

[15] W.H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes in C.* Cambridge University Press, second edition, 1992.

[16] J. A. Richards and X. Jia. *Remote Sension Digital Image Analysis: An Introduction.* Springer Verlag, third edition, 1999.

[17] K. Sayood. Data Compression in Remote Sensing Applications. *IEEE GRS-S Newsletter*, pages 7–15, 1993.

[18] D. Scott and Tim Tyler. BIAC - A Bijective Arithmetic Compresser, 2001. The Mandala Centre.

[19] C. Shannon. A Mathematical Theory Of Information. *The Bell System Technical Journal*, 27:379–423, 623–656, 1948.

[20] N. M. Short. *The Remote Sensing Tutorial.* NASA GSFC, GST, USAF Academy, September 1999. http://rst.gsfc.nasa.gov/.

[21] G. Strang and T. Nguyen. *Wavelets and Filter Banks.* Wellesley-Cambridge Press, 1996.

[22] Jacob Ström. Dead Zone Quantization in Wavelet Image Compression, 1996.

[23] W. Sweldens and P. Schröder. Building Your Own Wavelets At Home. In *Siggraph Conference on Computer Graphics and Interactive Techniques*, 1996.

[24] D. Tretter, N. Memon, and C. Bouman. Multispectral Image Coding. In *The Image and Video Processing Handbook.* Academic Press, 2000.

[25] B. Triebfürst. *Entwicklung waveletbasierter Kompressionsverfahren für Fernerkundungsdaten und deren Einsatz in einem geographischen Informationssystem.* Dissertation, Albert-Ludwigs-Universität Freiburg im Breisgau, 1999. http://www.freidok.uni-freiburg.de/volltexte/21/.

[26] M. Vetterli and J. Kovacevic. *Wavelets and Subband Coding.* Prentice Hall, 1995.

Anhang

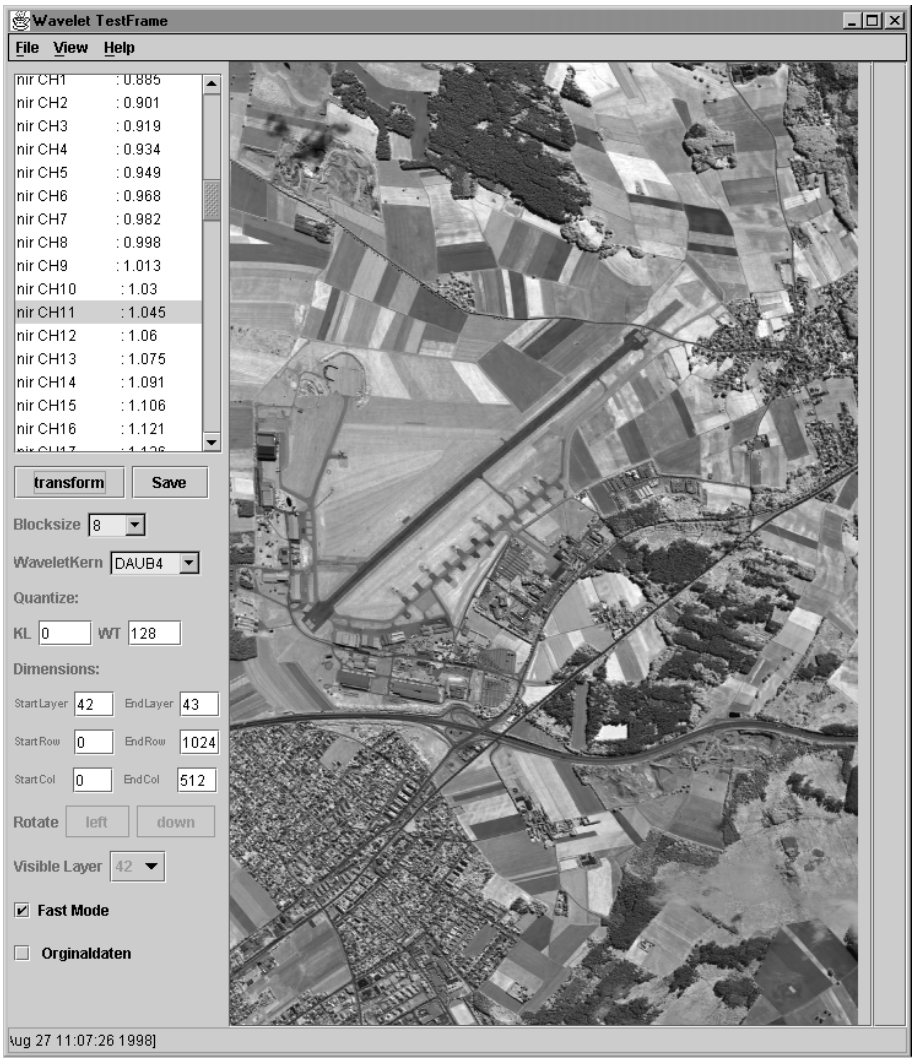


Abbildung 6.1: *Darstellung der Benutzeroberfläche. Auf der linken Seite erkennt man die in Abschnitt 4.6 beschriebenen Elemente, mit denen der Benutzer Einfluss auf Datenauswahl und Kompression nehmen kann.*

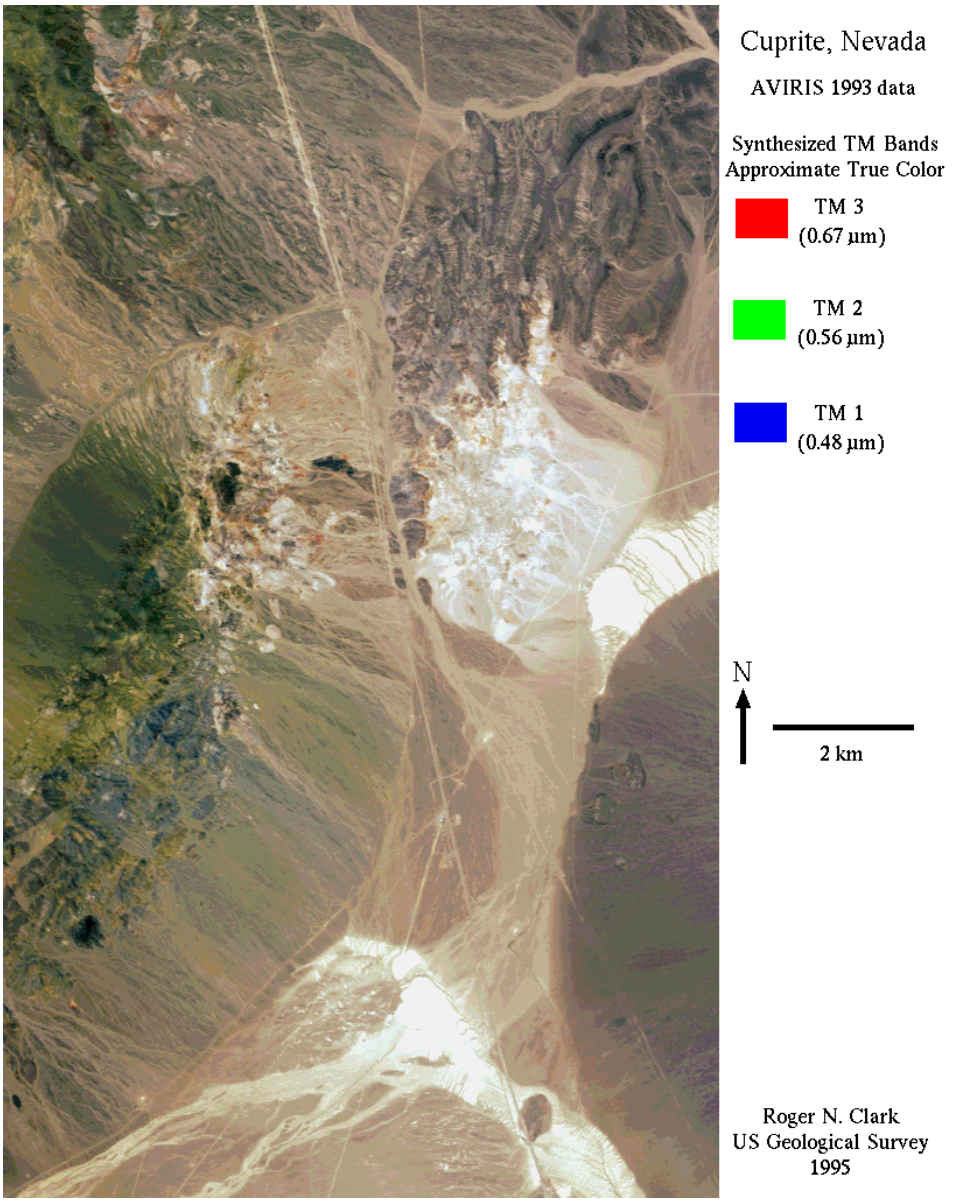


Abbildung 6.2: *Die hier dargestellte Abbildung von 1993 ist ein Pseudo-Echtfarbenbild von Cuprite, Nevada, das aus einem AVIRIS-Datensatz gewonnen wurde. Das Zielgebiet ist 10.5 km breit und 17 km lang. Norden ist oben. [3]*

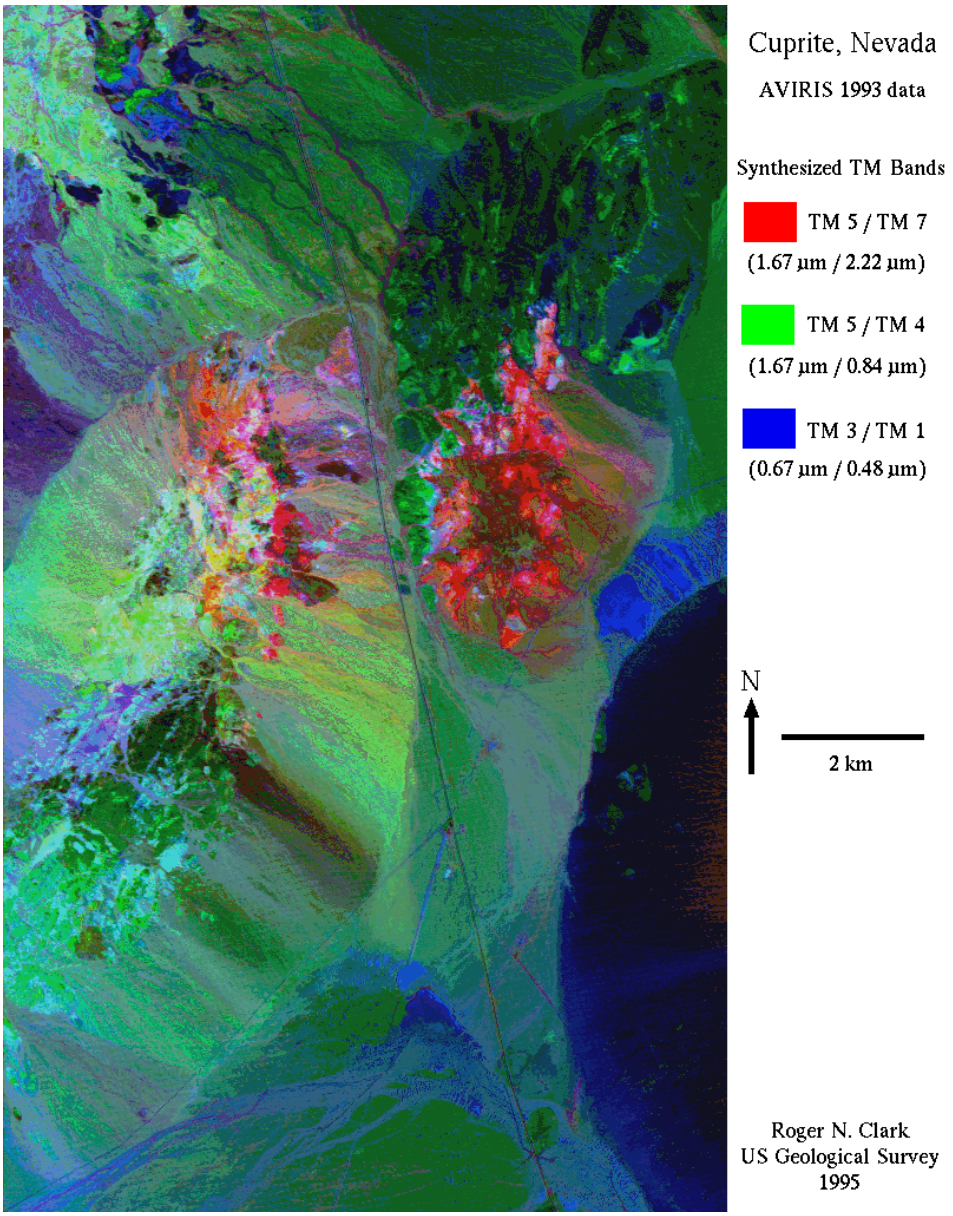


Abbildung 6.3: *Diese Abbildung ist ein Falschfarbenbild von Cuprite, Nevada, das durch die Anwendung arithmetischer Operationen auf die TM-Bänder eines AVIRIS-Datensatzes von 1993 gewonnen wurde. Das Zielgebiet ist 10.5 km breit und 17 km lang. Norden ist oben. [3]*

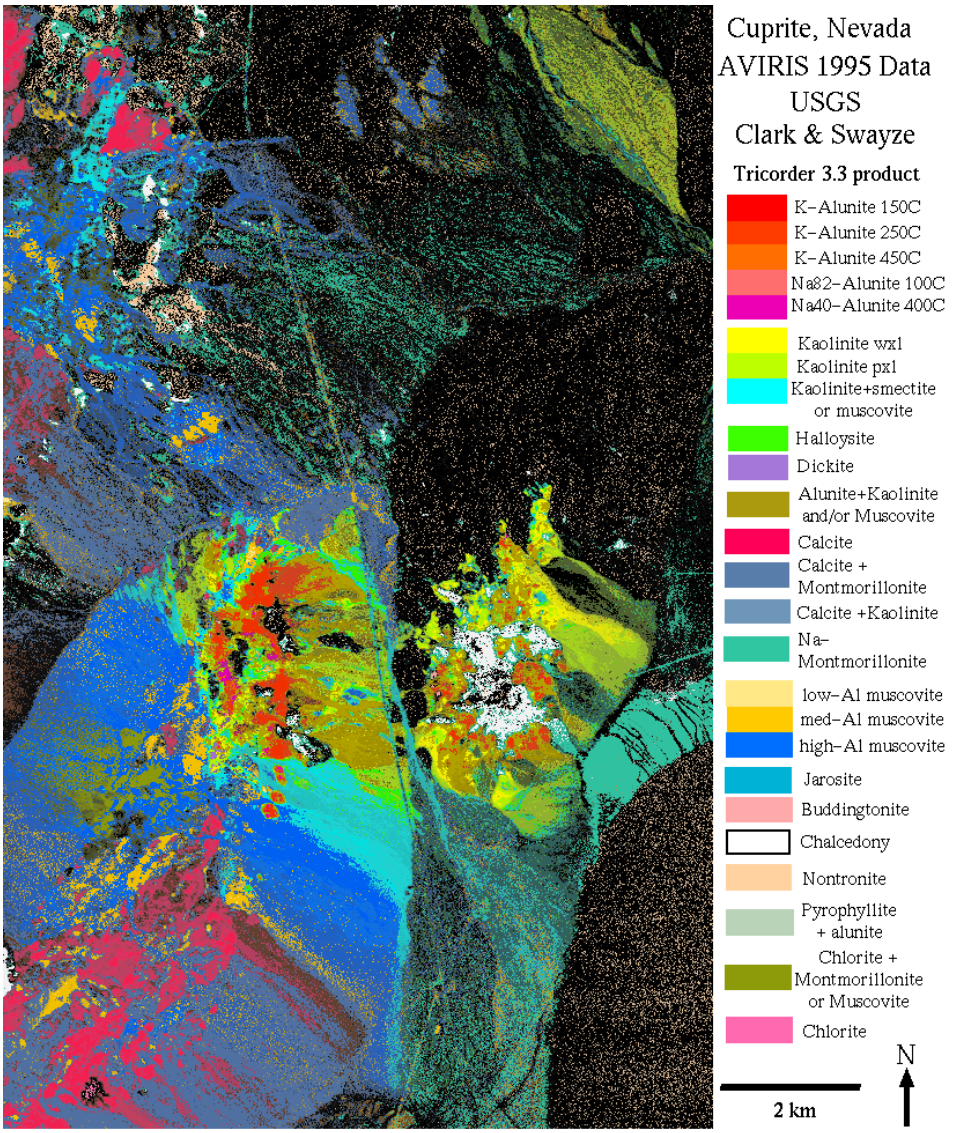


Abbildung 6.4: *In dieser Abbildung werden Mineralien im Zielgebiet dargestellt. Aufgrund ihrer seismischen Absorptionseigenschaften, die typischerweise im Intervall von 2 - 2.5 µm sichtbar werden, können spezielle Mineralien unterschieden werden. Diese Absorptionseigenschaften können sogar so genau differenziert werden, dass beispielsweise sichtbar wird, ob die Mineralien einen hohen, mittleren oder geringen Anteil an Aluminium enthalten. [3]*

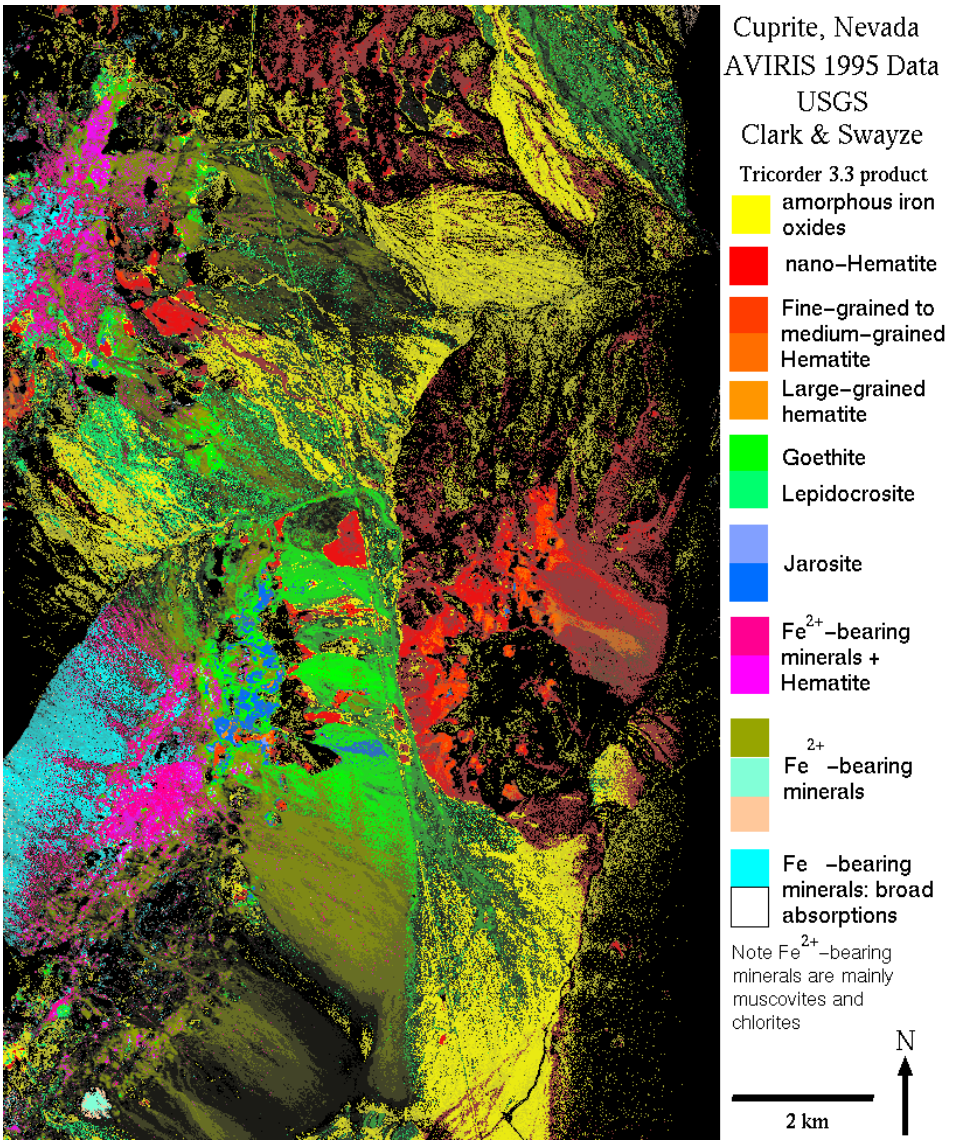


Abbildung 6.5: *Auch in dieser Abbildung werden Mineralien im Zielgebiet dargestellt, dieses Mal identifiziert anhand ihrer elektrischen Leitungseigenschaften, die typischerweise im Bereich von 0.4 - 1.2 µm liegen. Zu erkennen sind hier eisenhaltige Mineralien bzw. Eisenoxyde. Da die elektrischen Eigenschaften oft über einem breiten Bereich erkennbar sind, ist es teilweise schwierig, einzelne Mineralien voneinander zu unterscheiden. [3]*

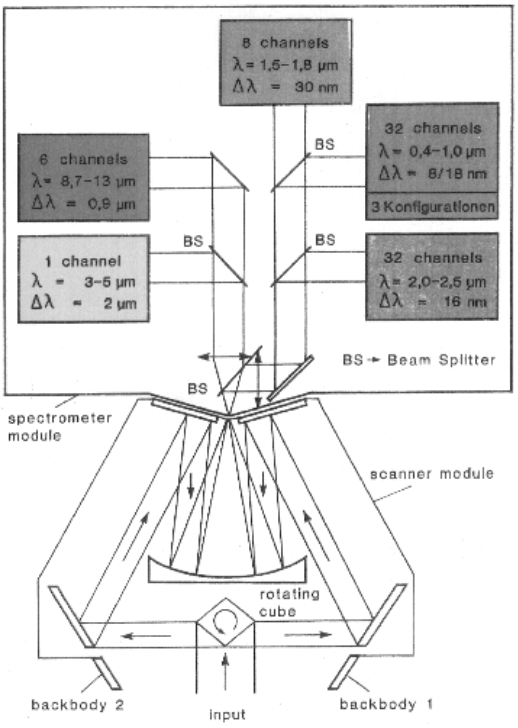


Abbildung 6.6: *Der Aufbau des DAIS7915 Sensors. Ein kubischer Polygon-Spiegel scannt die Oberfläche unter dem Sensor durch ein geöffnetes Fenster im Flugzeugboden. Die während des Scans im FOV (*field of view*) des Scanners empfangene Energie wird mit dem Spiegel reflektiert und mit Hilfe des Parabolspiegels auf die Öffnung des Spektrometers fokussiert. Die empfangene Strahlung wird nun mit Hilfe von Splittern (BS =* beam splitter*) aufgeteilt und je nach Wellenlänge in die entsprechende Untereinheit des Spektrometers geleitet. Dort wird sie in die einzelnen Bänder aufgeteilt.*

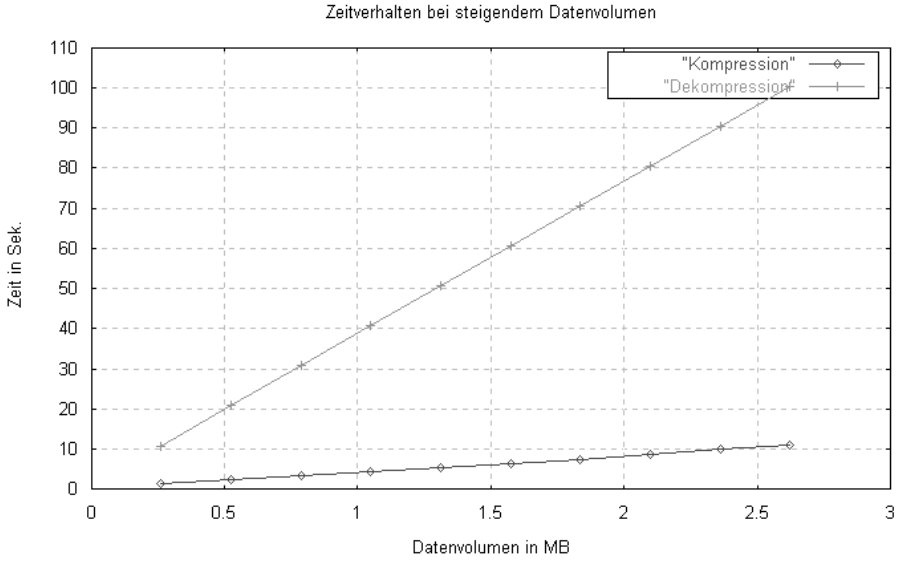


Abbildung 6.7: *Zeitverhalten des Gesamtalgorithmus bei wachsendem Datenvolumen.*

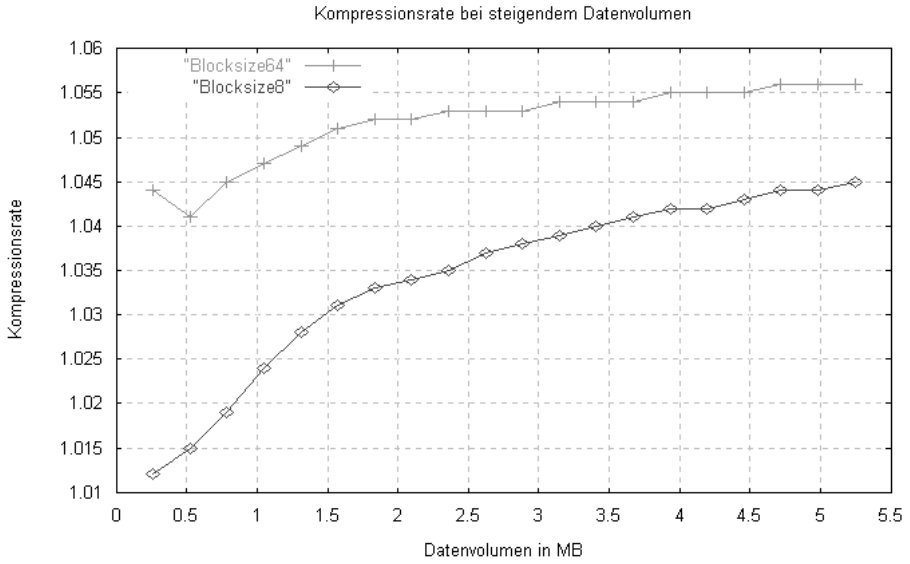


Abbildung 6.8: *Kompressionsraten bei verlustfreier Kompression der Daten bei verschiedenen Blockgrössen. Die Kompressionsraten bei grossen Blockgrössen sind höher als bei kleinen Blöcken. In jedem Fall aber sind die erzielten Raten nur sehr gering, da das gesamte Verfahren nicht für diese Art der Kompression ausgelegt ist.*

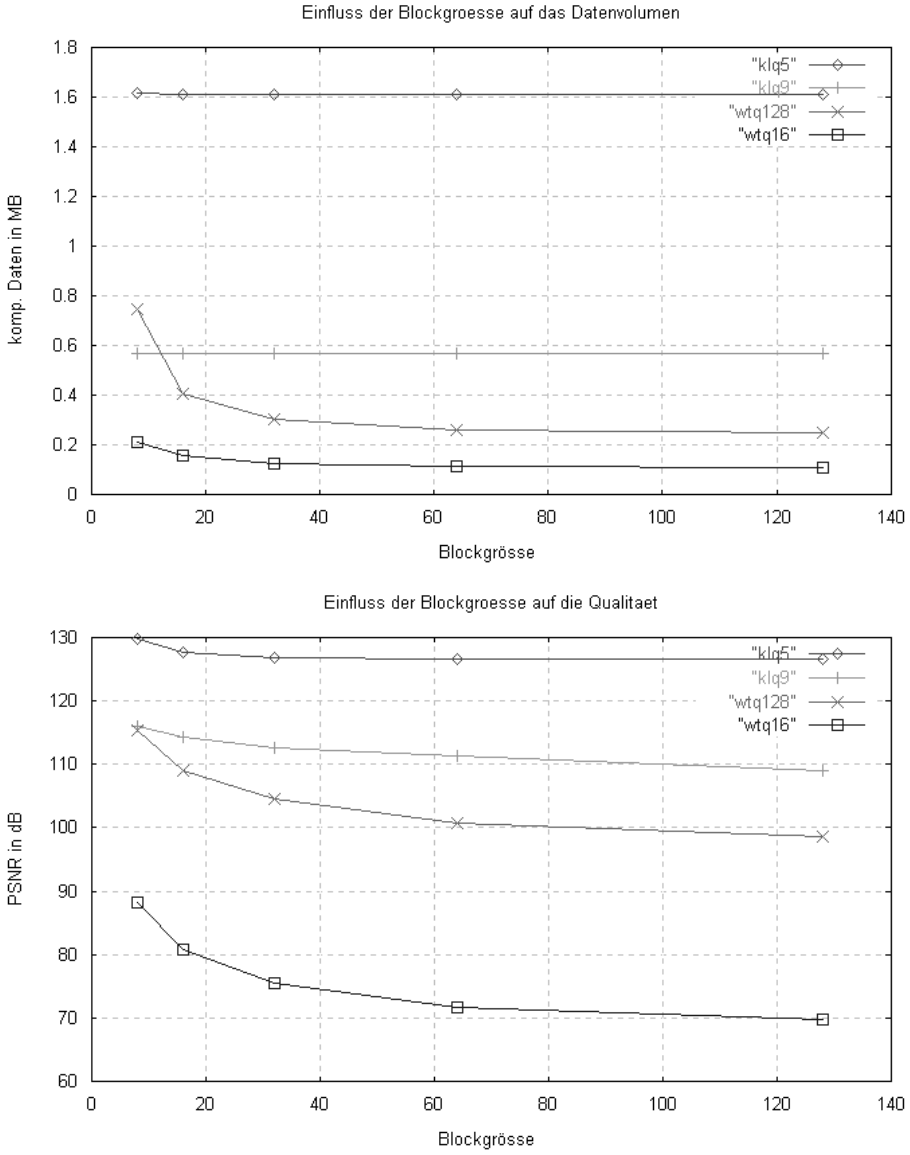


Abbildung 6.9: *Einfluss der Blockgrösse auf Kompression (oben) und Qualität (unten) der Daten. Der Testdatensatz war* $10 \times 256 \times 256$ *Pixel gross, die verwendete Blockgrösse war* 8×8 *Pixel. KLQ bezeichnet die Quantisierung nach der KLT mit 5 bzw. 9 Bändern, WTQ ist analag die Quantisierung nach der DWT mit 128 und 16 Quantisierungsstufen. Man sieht, dass die Blockgrösse nahezu keinen Einfluss auf die KLT hat, wohl aber auf die DWT. Hier ist der Effekt um so grösser, je mehr Quantisiserungsstufen benutzt werden.*

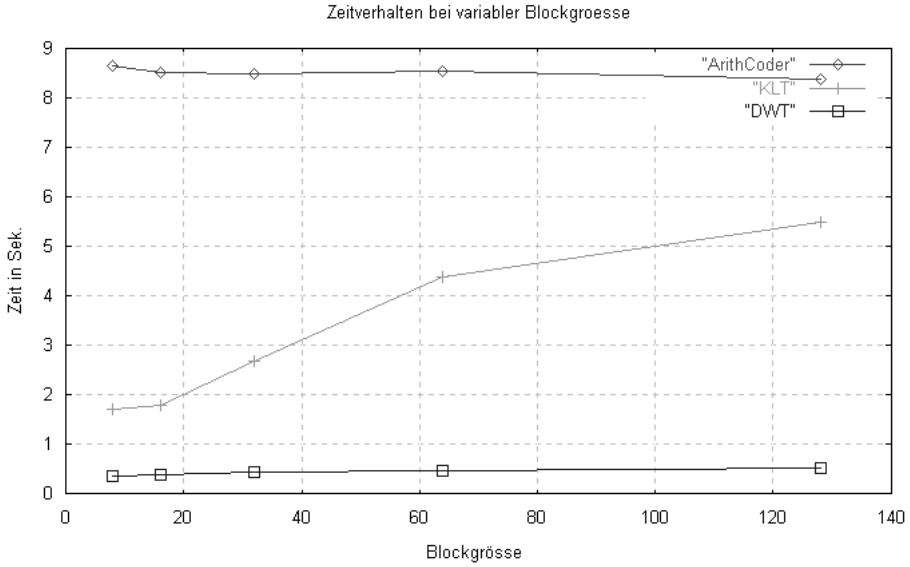


Abbildung 6.10: *Einfluss der Blockgrösse auf die Kompressionszeit der Daten. Der Testdatensatz war* $10 \times 256 \times 256$ *Pixel gross, die verwendete Blockgrösse war* 8×8 *Pixel. Es ist zu erkennen, dass eine steigende Blockgrösse einen geringen negativen Einfluss auf die DWT und einen geringen positiven Einfluss auf den arithmethischen Koder hat. Lediglich bei der KLT ist dieser Effekt tatsächlich signifikant, da sich hier die Kompressionszeit mit steigender Blockgrösse deutlich verlängert.*

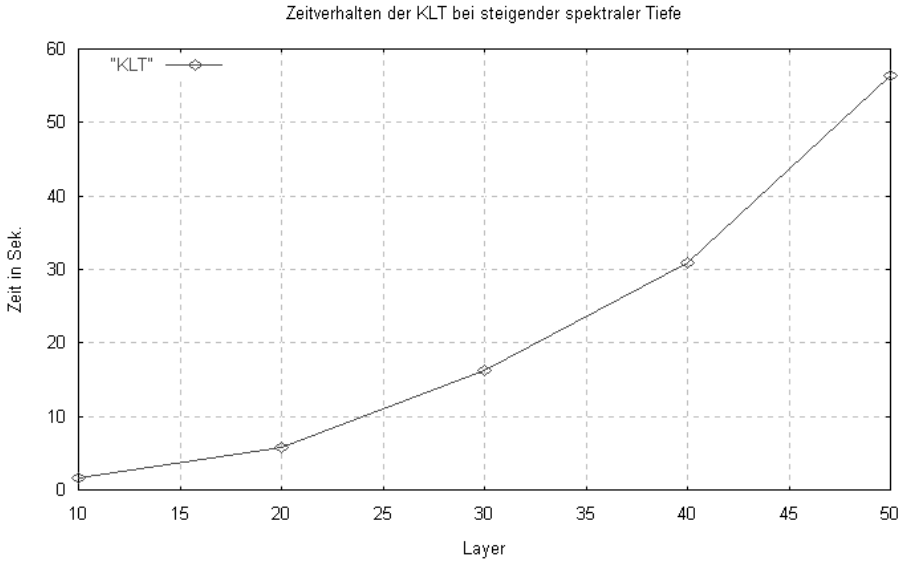


Abbildung 6.11: *Zeitverhalten der KLT bei zunehmender spektraler Tiefe. Der Testdatensatz war* 256×256 *Pixel gross, die verwendete Blockgrösse war* 8×8 *Pixel.*

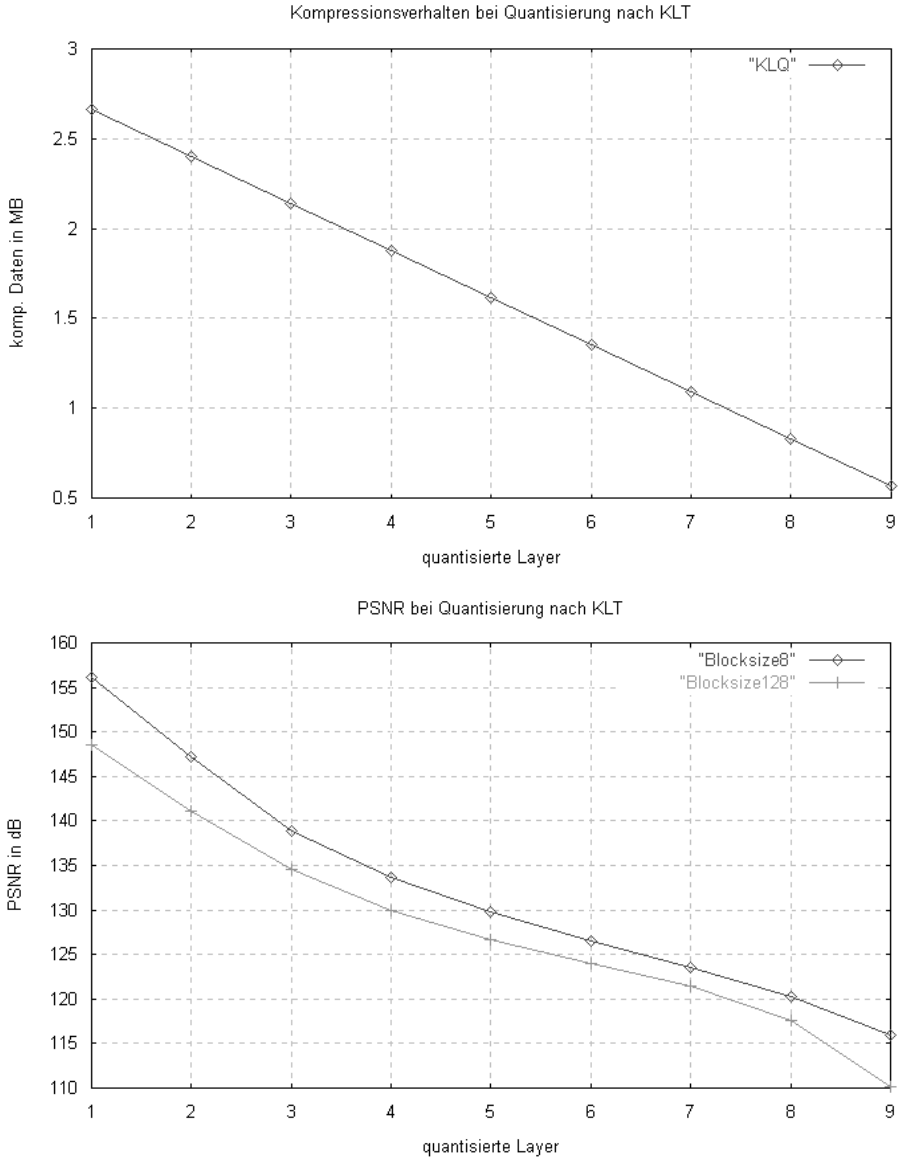


Abbildung 6.12: *Einfluss der Quantisierung auf Kompression (oben) und Qualität (unten) der KL-transformierten Daten. Der Testdatensatz war* $10 \times 256 \times 256$ *Pixel gross. Mit steigender Quantisierung der Daten nimmt das Datenvolumen aufgrund des verwendeten Verfahrens linear ab. Die Qualität der Daten sinkt nur langsam und auch die verwendete Blockgrösse hat nahezu keinen Einfluss.*

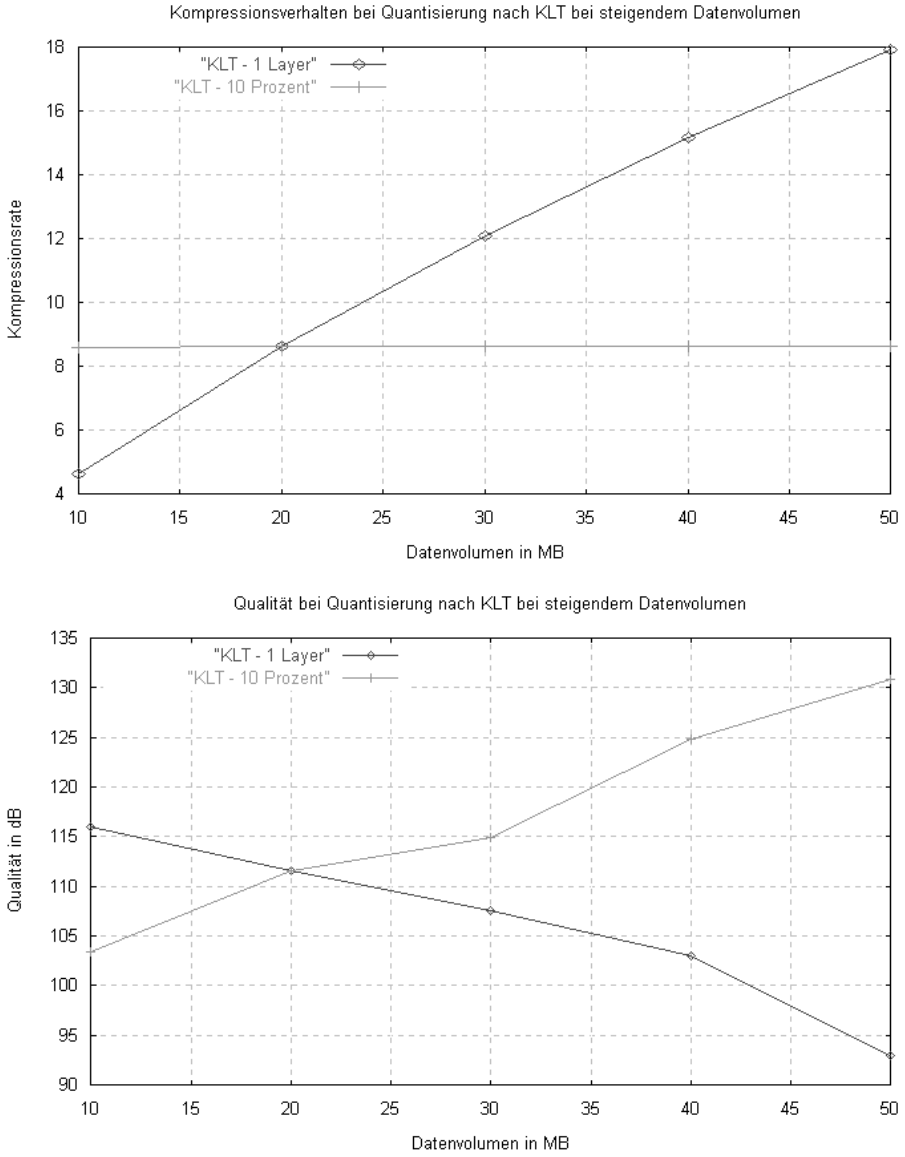


Abbildung 6.13: *Kompression (oben) und Qualität (unten) von KL-transformierten Daten bei steigendem Datenvolumen. Der Testdatensatz war* 256×256 *Pixel gross, die verwendete Blockgrösse war* 8×8 *Pixel. Der KLT-Graph zeigt die Ergebnisse, wenn jeweils nur die Komponente mit der grössten Energie erhalten wird, bei KLT10 werden die 10 Prozent der transformierten Komponenten mit der grössten Energie erhalten. Man sieht, dass die Qualität der Daten auch bei sehr hohem Datenverlust noch immer akzeptabel ist. Der KLT10 Graph verläuft bei der Kompressionsrate wie zu erwarten waagerecht und zeigt beispielhaft die Ergebnisse für die in Abschnitt 5.5 gegebenen Empfehlungen für die Quantisierung.*

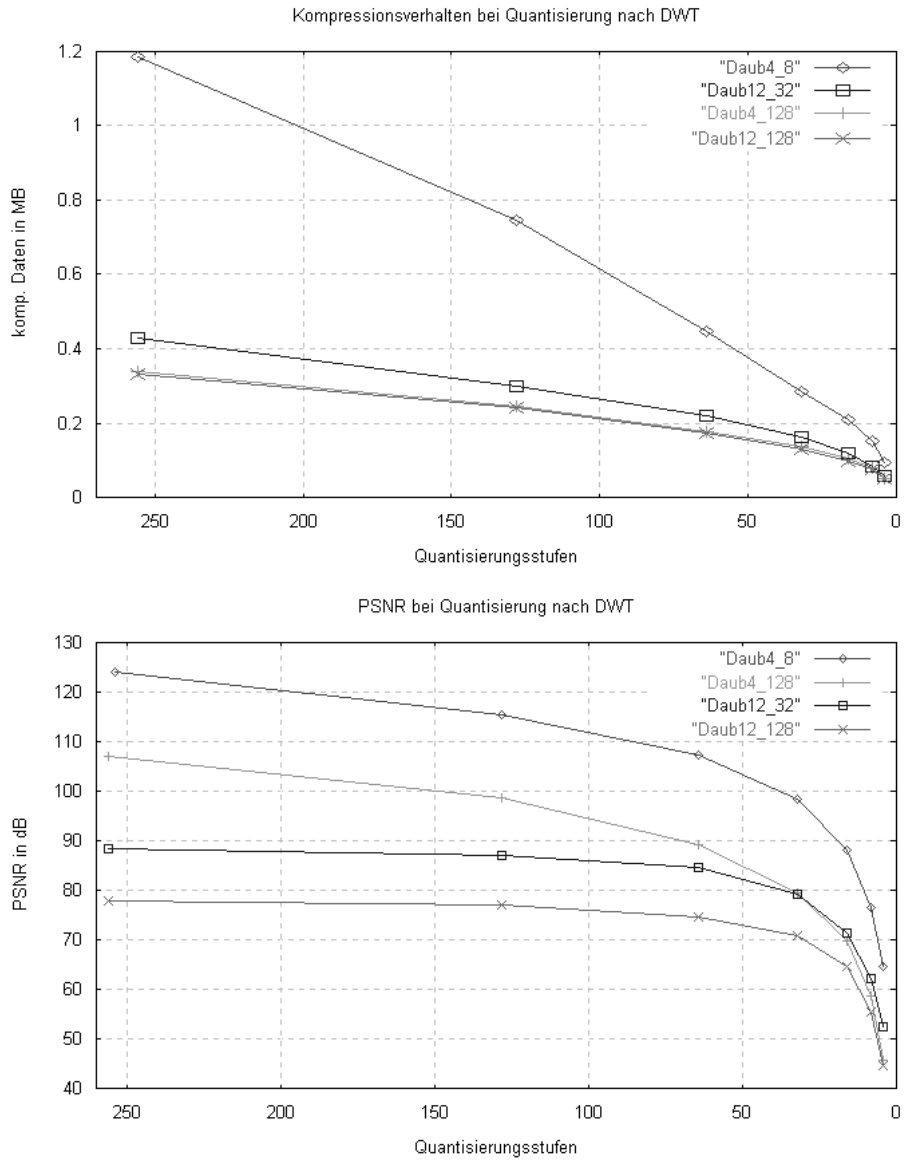


Abbildung 6.14: *Einfluss der Quantisierung auf Kompression (oben) und Qualität (unten) der wavelet-transformierten Daten. Der Testdatensatz war* $10 \times 256 \times 256$ *Pixel gross. Bei Blockgrösse* 8×8 *steigt die Kompressionsrate bei stärkerer Quantisierung am schnellsten, mit steigender Blockgrösse wird dieser Effekt immer geringer. Die Kompressionsrate ist bei gleicher Blockgrösse mit* DAUB4 *geringfügig besser als mit* DAUB12*-Filter, die Qualität der Daten ist mit* DAUB4*-Filter sogar deutlich besser. Ferner wird mit grösseren Blöcken eine bessere Kompression der Daten möglich, allerdings bei wesentlich schlechterer Qualität.*

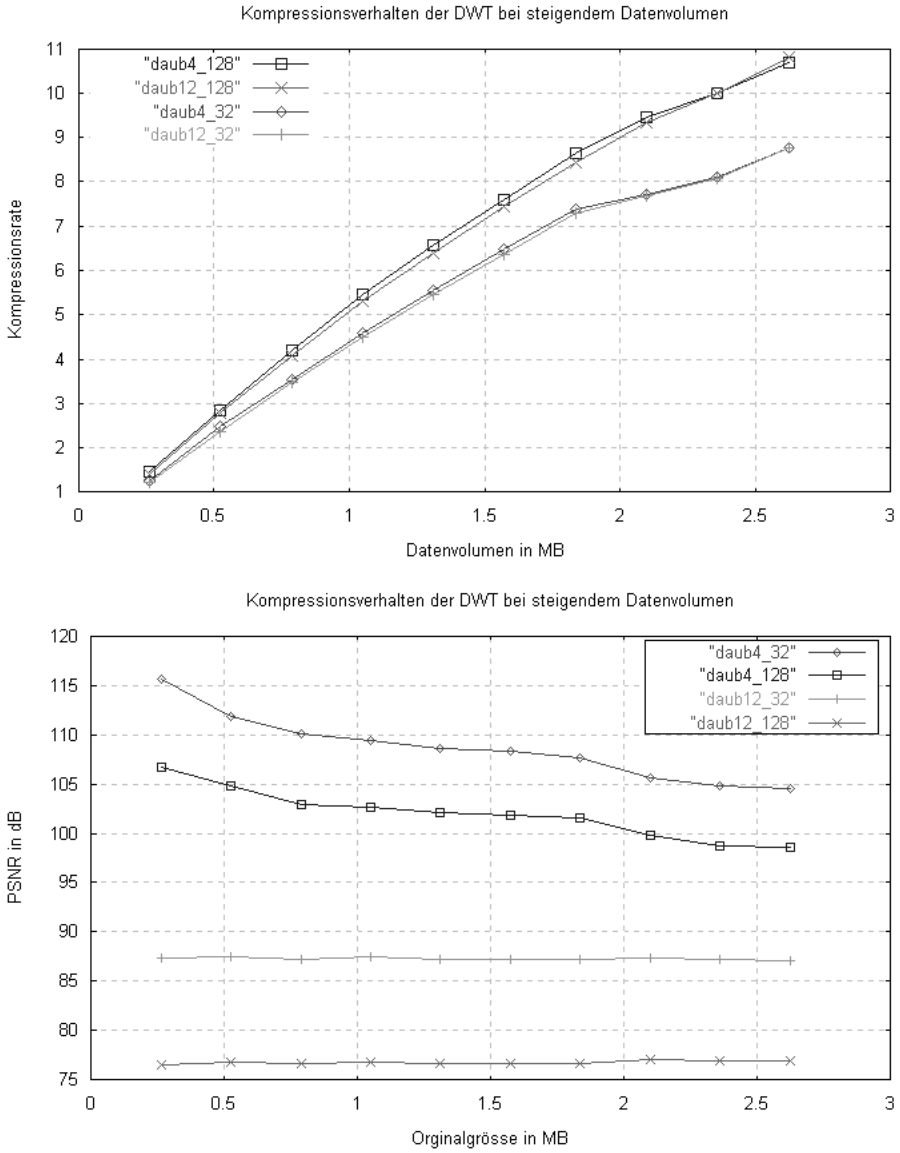


Abbildung 6.15: *Einfluss der Quantisierung auf Kompression (oben) und Qualität (unten) der wavelet-transformierten Daten bei steigendem Datenvolumen. Die verwendete Blockgrösse war* 8×8 *Pixel. Bei gleicher Blockgrösse ist die Kompressionsverhalten bei Verwendung des* DAUB4*-Filters stehts etwas höher als bei* DAUB12*. Die Qualität der Daten hingegen ist bei* DAUB12 *entscheidend schlechter.*

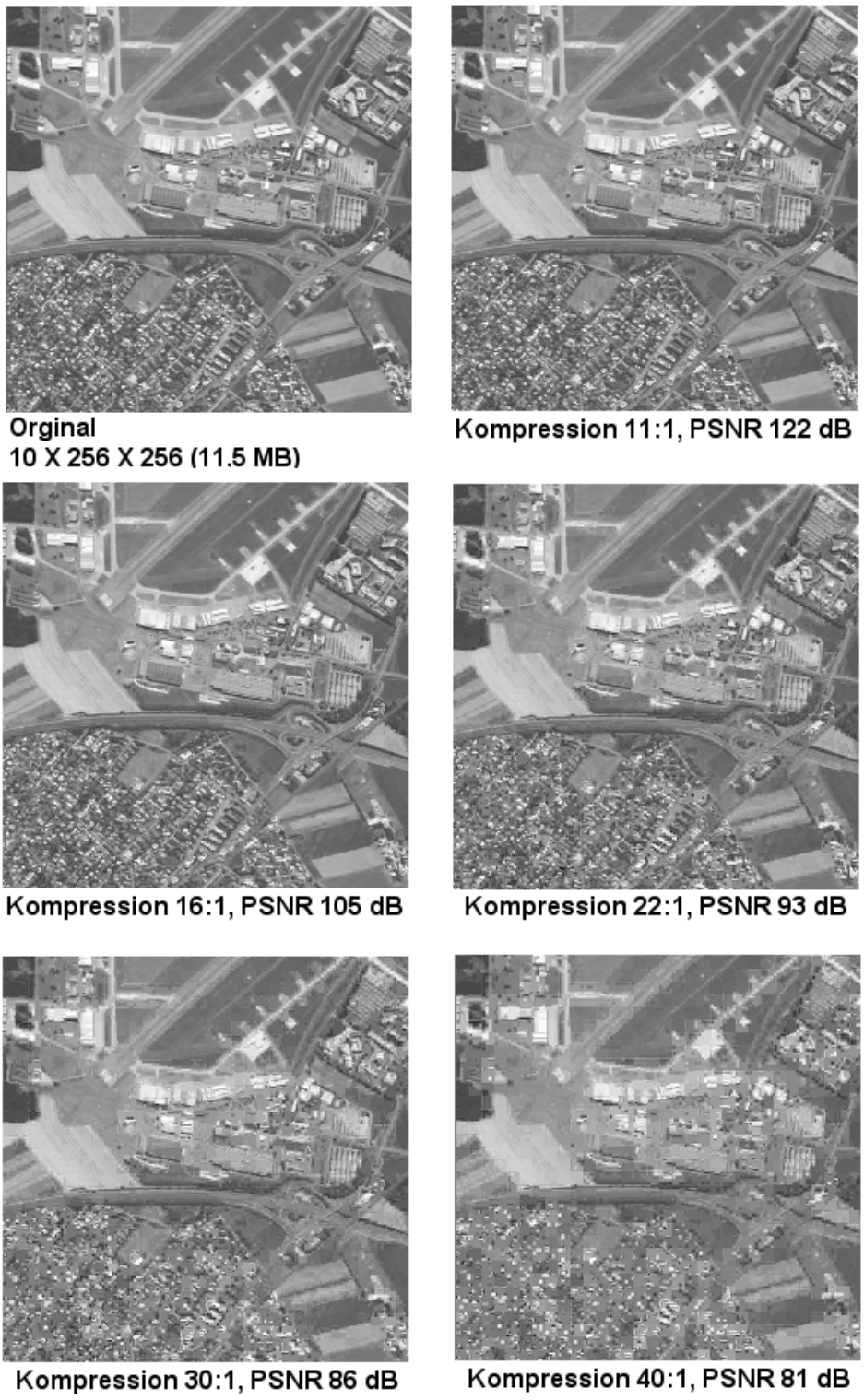


Abbildung 6.16: *Beispiel für die Bildqualität bei der Visualisierung von verlustbehafteten Daten.*

Erklärung

Ich versichere, dass ich die vorliegende Arbeit selbständig und nur unter Verwendung der angegebenen Quellen und Hilfsmittel angefertigt habe.

Leipzig, 27. Juni 2002

Karsten Rink