MPEG-4 Szenenbeschreibung in Telepräsenz-Szenarien

Jan Leupold

TECHNISCHE UNIVERSITÄT MÜNCHEN

Lehrstuhl für Realzeit-Computersysteme

MPEG-4 Szenenbeschreibung in Telepräsenz-Szenarien

Jan Leupold

Vollständiger Abdruck der von der Fakultät für Elektrotechnik und Informationstechnik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktor-Ingenieurs (Dr.-Ing.)

genehmigten Dissertation.

Vorsitzender:	Univ.-Prof. Dr. sc.techn. (ETH) A. Herkersdorf
Prüfer der Dissertation:	1. Univ.-Prof. Dr.-Ing. G. Färber
	2. Univ.-Prof. Dr.-Ing. K. Diepold

Die Dissertation wurde am 12.03.2008 bei der Technischen Universität München eingereicht und durch die Fakultät für Elektrotechnik und Informationstechnik am 04.06.2008 angenommen.

Danksagung

Diese Dissertation entstand als Ergebnis meiner Tätigkeit als wissenschaftlicher Mitarbeiter am Lehrstuhl für Realzeit–Computersysteme der Technischen Universität München. Teile der Arbeit wurden von der *Deutschen Forschungsgemeinschaft* (DFG) als Teil des Sonderforschungsbereichs „Wirklichkeitsnahe Telepräsenz und Teleaktion“ gefördert.

An erster Stelle möchte ich mich hier bei Professor Färber bedanken. Nicht nur als Doktorvater für die Erstellung dieser Dissertation, sondern auch in allen anderen Aspekten des Lehrstuhllebens hat er stets ein ausgewogenes Verhältnis von Freiräumen und notwendiger Pflicht gefunden. Bei Professor Diepold will ich insbesondere deswegen bedanken, dass er mir die Idee mit MPEG-4 „in den Kopf gesetzt hat“. Natürlich darf hier auch nicht unerwähnt bleiben, dass ich ohne die Informationen, die mir Prof. Diepold im Umfeld zu MPEG-4 zukommen hat lassen, nur schwerlich weitergekommen wäre.

Bei allen meinen Kollegen am RCS will ich mich für die gute Zusammenarbeit, das angenehme Arbeitsklima und das ebenso angenehme „Nicht-Arbeits“-Klima bedanken. Ebenso will ich diesen Dank an die zahlreichen Kollegen richten, mit denen ich im Verlauf meiner bisherigen Tätigkeit in Kooperationsprojekten zusammengearbeitet habe.

Was das Telepräsenz-Team am RCS angeht – die manchmal auch nur „Teletubbies“ genannt werden – bedanke ich mich bei Georg Passig und Tim Burkert für die angenehme Teamarbeit in der dritten Phase des SFB 453. Für die vierte Phase geht mein besonderer Dank an Stephan Behrendt: einen besseren Freund, Bürokollegen und Teletubbie kann man sich nicht wünschen.

Mein Studium und die anschließende Dissertation wäre nicht ohne die stetige Unterstützung durch meine Eltern möglich gewesen. Mir war immer bewusst, dass ich auf Euch zählen kann.

La última y persona más importante a la que quiero dar las gracias, es mi mujer Cecilia. Te agradezco muchísimo el apoyo que me has dado durante todos estos años.

München, im März 2008

Inhaltsverzeichnis

Verzeichnis der verwendeten Symbole

AU	Access Unit, Seite 16
B-rep	Boundary Representation, Seite 5
BIFS	Binary Format for Scenes
CSG	Constructive Solid Geometry
DMIF	Delivery Multimedia Integration Framework
IEC	International Engineering Consortium
IFS	Indexed Face Set, Seite 20
ISO	International Organization for Standardization
HVA	Haptisch-Visuell-Auditorischer Arbeitsraum, Seite 3
LASeR	Lightweight Application Scene Representation
LOD	Level of Detail
OCR	Object Clock Reference, Seite 37
ODS	Object Descriptor Stream, Seite 32
PMF	Predictive MFField, Seite 30
PROTO	Benutzerspezifischer Knotentyp, Seite 18
QP	Quantization Parameter, Seite 21
ROUTE	Signalpfad für Benutzerinteraktion, Seite 22
SL	SyncLayer: zeitliche Synchronisierung von Elementary Streams
SL-Paket	Datenpakte mit SyncLayer-Informationsheader
SVG	Scalable Vector Graphics
VRML	Virtual Reality Modeling Language

Zusammenfassung

Für Telepräsenz-Szenarien steht ein Ziel immer sehr deutlich im Vordergrund: der Bediener eines ferngesteuerten Roboters soll sich fühlen, als ob er an Stelle des Roboters die gestellte Aufgabe selber ausführen würde. Um dies technisch umzusetzen, müssen viele Daten von der Umgebung des Roboters erfasst und dem Bediener mit hoher Qualität dargestellt werden. Zu diesen Daten gehören auch Informationen über die Geometrie der Roboterumgebung. Sobald ein dreidimensionales Umgebungsmodell verfügbar ist, kann dem Benutzer durch leistungsfähige Computergrafik eine Ansicht der entfernten Umgebung aus beliebiger Perspektive dargestellt werden. Die vorliegende Arbeit befasst sich mit der Problemstellung, die Daten eines dreidimensionalen Umgebungsmodells vom Standort des Roboters zum Standort des Benutzers mit möglichst geringem Ressourcenbedarf und trotzdem ausreichender Qualität zu übertragen.

Als Teil des MPEG-4 Standards wird mit BIFS (Binary Format for Scenes) unter anderem ein Protokoll definiert, um dreidimensionale Geometriedaten zu übertragen. Das BIFS Protokoll enthält bereits einige Kompressionswerkzeuge, um das benötige Datenvolumen für Geometriedaten zu reduzieren. In dieser Arbeit wird vorgestellt, wie diese Werkzeuge für Telepräsenz-Szenarien besonders effizient eingesetzt werden können.

In komplexen Telepräsenz-Szenarien agieren mehrere Roboter in ein- und derselben Umgebung, um eine gestellte Aufgabe kooperativ zu lösen. Jeder Roboter ist dabei genau einem Benutzer zugeordnet. Die Herausforderung hier ist, alle in der Roboterumgebung erfassten Eindrücke der Umgebung in einer, für alle Benutzer gemeinsam genutzten Repräsentation zusammenzuführen. Es wird ein Konzept vorgestellt, wie das ebenfalls in MPEG-4 beschriebene Object Descriptor Framework zu diesem Zweck eingesetzt werden kann. Weiterhin werden Methoden vorgestellt, um die benötigte Bandbreite für Datenübertragungen zwischen dem Roboterstandort und den Benutzerstandorten auf das notwendige Minimum zu reduzieren. Der Vorteil bei der Verwendung des Object Descriptor Frameworks ist, dass keine besonderen Maßnahmen für die Benutzer erforderlich sind, um auf die gemeinsame Repräsentation *aller* Umgebungsdaten zugreifen zu können.

Für alle vorgestellten Methoden und Konzepte wurde eine Implementierung entwickelt, die nicht nur die erfolgreiche Umsetzung demonstriert, sondern vor allem für die Messung wichtiger Kenngrößen herangezogen wurde. Zu diesen Kenngrößen gehören Kompressionsrate, benötigte Rechenzeit für die Kodierung, sowie die Zeitdifferenz zwischen Erfassung der Daten und deren Darstellung am Benutzerstandort.

1 Einleitung

In dieser Arbeit wird beschrieben, wie der MPEG-4 Standard in Telepräsenz-Szenarien eingesetzt werden kann. MPEG-4 ist ein umfangreicher ISO/IEC Standard zur Übertragung von Multimedia-Inhalten. Telepräsenz vereint Forschungsarbeiten auf vielen Gebieten. Insbesondere Techniken zur wirklichkeitsnahen Darstellung von multimodalen Daten, aber auch zur Gewinnung dieser Daten spielen eine wichtige Rolle. Die Schnittmengen der Themenkomplexe MPEG-4 und Telepräsenz werden im Folgenden motiviert.

Für eine Definition von Telepräsenz folgt hier ein Ausschnitt aus dem Internetauftritt des Sonderforschungsbereichs (SFB) 453 „Wirklichkeitsnahe Telepräsenz und Teleaktion“, der durch die Deutsche Forschungsgemeinschaft (DFG) gefördert wird:

> „Telepräsenz wird erreicht, wenn es einem menschlichen Operator durch technische Mittel ermöglicht wird, mit seinem subjektiven Empfinden in einer anderen, entfernten oder nicht zugänglichen Remote - Umgebung präsent zu sein. Teleaktion bedeutet, daß dieser menschliche Operator nicht nur passiv präsent ist, sondern daß er an dem entfernten Ort auch aktiv eingreifen kann. Wirklichkeitsnah sind diese Eindrücke dann, wenn der menschliche Operator nicht mehr leicht unterscheiden kann, ob seine sensorischen Eindrücke und die Rückmeldungen von seinem Handeln in direkter Wechselwirkung mit der Wirklichkeit oder über technische Mittel entstehen.“[68]

Für die technische Umsetzung dieses Ziels in einem Telepräsenz-Szenario bedeutet dies, dass die für die gestellte Aufgabe relevanten Sinneseindrücke, so wie sie der Operator in der entfernten Umgebung auf natürliche Weise empfinden würde, in der lokalen Umgebung des Operators durch technische Mittel reproduziert werden müssen. Insbesondere die Sinnesmodalitäten der Haptik, Akustik und Vision werden häufig für die Durchführung einer Teleaktion benötigt. Die vorliegende Arbeit befasst sich mit der visuellen Modalität.

Die klassische Implementierung der visuellen Modalität besteht darin, dass in der entfernten Umgebung eine Videokamera als Sensor montiert wird (siehe Abbildung 1.1). Diese Kamera nimmt während der Telemanipulation Bilder des zu manipulierenden Werkstücks, des Teleoperators und der restlichen entfernten Umgebung auf. Diese Bilder werden als Videodatenstrom zum Operator übertragen und dort dargestellt. Dadurch kann der Operator den Verlauf der Telemanipulation visuell verfolgen, zumindest aus der Position und Blickrichtung, an der die Videokamera montiert ist. Für eine alternative Perspektive, die z. B. eine präzisere Manipulation ermöglicht, kann eine zweite Kamera montiert werden.

Im Rahmen des Teilprojekts M3 im SFB 453 (sowie dessen Vorgänger C2), wurde ein anderer Ansatz für die visuelle Modalität verfolgt. Hierzu wird von der entfernten Um-

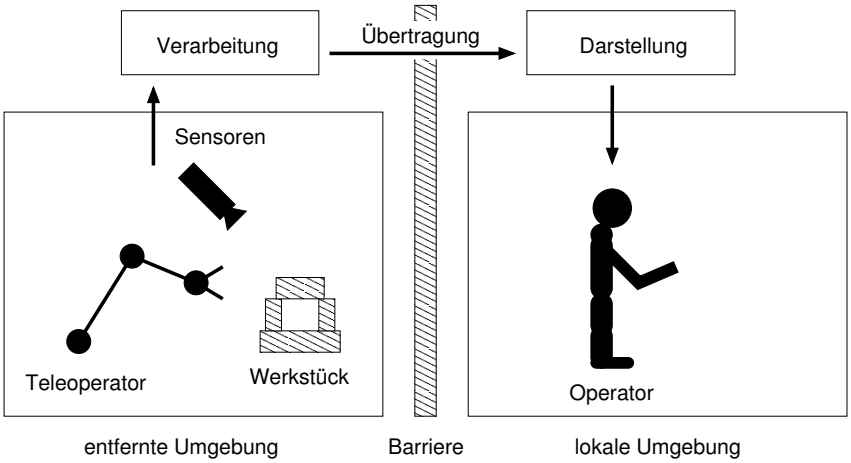


Bild 1.1: Vereinfachte Darstellung eines typischen Telepräsenz-Szenarios

gebung ein dreidimensionales Modell der Oberflächen aller Objekte rekonstruiert. Dieses 3D-Modell wird dem Operator wie eine virtuelle Welt aus beliebiger Perspektive dargestellt. Damit die Darstellung der virtuellen Welt mit annähernd photorealistischer Qualität erfolgen kann, werden entweder für die einzelnen Oberflächen Texturen extrahiert (Teilprojekt C2, [4]) oder Kamerabilder projektiv auf die Oberflächen gezeichnet (Teilprojekt M3, [1]). Insbesondere die beliebige Wahl der Betrachterperspektive ermöglicht es dem Operator, die Szene von einer ihm angenehmen Position zu betrachten, anstatt auf die vorgegebene Position der Kamera beschränkt zu sein. Dies kommt der geforderten Transparenz der technischen Mittel in der Definition von wirklichkeitsnaher Telepräsenz nahe.

Diese Arbeit entstand im Rahmen des Teilprojekts M3 aus der Notwendigkeit, die Daten der rekonstruierten 3D-Modelle effizient von der entfernten Umgebung zum Operator zu übertragen. Während in C2 noch eine Übertragung durch eine Client-Server-Lösung realisiert werden konnte, musste aufgrund neuerer, detailreicher und deutlich beschleunigter Rekonstruktionsalgorithmen ein alternativer Weg beschritten werden. Um aufwendige und unnötige Neuentwicklungen von Übertragungsprotokollen zu vermeiden wurde die Verwendung von BIFS vorgeschlagen.

BIFS (BInary Format for Scenes) ist Teil des ISO/IEC Standards 14496, besser bekannt als MPEG-4. Zum einen werden dort Datenstrukturen für einen hierarchischen Szenengraphen beschrieben. Die rekonstruierten 3D-Modelle werden in diesen Datenstrukturen abgespeichert. Zum anderen wird dort auch ein Streamingprotokoll definiert, mit dem ein Szenengraph, und Änderungen in diesem Szenengraph, binär kodiert übertragen werden kann.

Der gewählte Ansatz, die entfernte Umgebung dem Operator als virtuelle Welt zu präsentieren, ist nicht ganz unproblematisch. Alle Manipulationen werden in der realen Welt der entfernten Umgebung ausgeführt. Da die Entscheidungen über Manipulationen vom Operator jedoch aufgrund der Wahrnehmung in der virtuellen Welt getroffen werden,

muss viel Wert auf eine korrekte Rekonstruktion der virtuellen Welt gelegt werden. Zum einen betrifft dies die Präzision der 3D-Rekonstruktion. Auf diese wird hier nicht weiter eingegangen, da dies ein eigenständiges Forschungsgebiet ist. Zum anderen müssen alle Zeitverzögerungen zwischen dem Erfassen einer Veränderung in der realen Welt und der Darstellung in der virtuellen Welt so klein wie möglich gehalten werden. Aus diesem Grund nehmen die benötigten Übertragungs- und Rechenzeiten in dieser Arbeit stets eine zentrale Rolle ein.

Allgemein gilt weiterhin für Telepräsenz-Szenarien, dass alle Daten, die zwischen Operator und Teleoperator ausgetauscht werden, über die sog. Barriere übertragen werden (siehe Abbildung 1.1). In einem Weltraum-Szenario manifestiert sich diese Barriere durch die große Entfernung zwischen Operator (Erde) und Teleoperator (Weltraum), bei minimalinvasiver Chirurgie durch die Haut des Patienten. Eine wesentliche Eigenschaft der Barriere ist ihre Bandbreite, welche als kostbare Ressource in vielen Szenarios begrenzt ist. Die Übertragung von detailreichen Geometriedaten zusammen mit weiteren Daten, z. B. für Haptik und Audio, kann eine Kompression der Geometriedaten erforderlich machen. Die für BIFS definierten Kompressionsverfahren werden in dieser Arbeit erläutert und es werden Vorgehensweisen vorgestellt, um ihren Einsatz zu optimieren.

Für komplexe Telemanipulationen entstehen Szenarien, in denen ein einzelner Teleoperator der gestellten Aufgabe alleine nicht mehr gewachsen ist. Auch die Anzahl der Operatoren kann, je nach Anforderung, variieren. Für eine kooperative Telemanipulation ist es zweckmäßig, wenn allen Operatoren Ansichten der entfernten Umgebung aus einer gemeinsam benutzten virtuellen Welt dargestellt werden. Die Informationen, die jeder Teleoperator über die entfernte Umgebung sammelt, sollen ebenfalls in dieser einen virtuellen Welt hinterlegt werden. Diese Überlegungen führen zu dem Schluss, dass eine Funktionalität im Sinne einer verteilten virtuellen Welt benötigt werden. Im oben erwähnten SFB 453 wird dazu der folgende Begriff verwendet:

> „[...] HVA = Haptisch-Visuell-Auditorischen Arbeitsraum. Dies ist als ein gemeinsamer Arbeitsraum zu verstehen, in dem eine verteilte telepräsente Kooperation mit den genannten Modalitäten unterstützt wird (Telekooperation). Mehrere geografisch verteilte Nutzer (Operatoren) müssen in einer gemeinsamen Umgebung eine komplexe Aufgabe lösen, werden dabei in der Wahrnehmung auditorischer, visueller und haptischer Eindrücke unterstützt und verfügen über jeweils einen Teleoperator, der die Manipulationen im gemeinsamen Arbeitsraum ausführen kann.“[68]

Mittels des MPEG-4 Object Descriptor Framework und der Möglichkeit zur Übertragung von 3D-Modellveränderungen zur entfernten Umgebung werden in dieser Arbeit Vorgehensweisen vorgestellt, um diese Funktionalität zu erreichen.

Im nächsten Kapitel folgt zuerst ein Überblick über verwandte Arbeiten. Kapitel 3 erläutert die für das weitere Verständnis notwendigen Konzepte des MPEG-4 Standards. Das Thema der komprimierten Geometrieübertragung wird in Kapitel 4 behandelt, während auf die oben geforderte Funktionalität als HVA in Kapitel 5 eingegangen wird. Die Arbeit wird durch eine Zusammenfassung mit Ausblick in Kapitel 6 abgeschlossen.

2 Stand der Technik und verwandte Arbeiten

In der Einleitung wurden bereits einige Themenbereiche angesprochen, die in Zusammenhang mit dieser Arbeit stehen. In den folgenden Absätzen werden nun Forschungsarbeiten, Standards, bekannte Implementierungen und grundlegende Arbeiten zu diesen Bereichen behandelt und in Relation zu dieser Arbeit diskutiert.

Geometrische Beschreibungsformen

Die wohl bekannteste Beschreibungsform für dreidimensionale Objekte ist die sog. *Boundary Representation*, oder kurz B-rep. Hierbei werden die Oberflächen solider Körper durch viele kleine, ebene Flächen angenähert [63, 34]. Diese Art der Modellierung ist besonders zur Visualisierung am Rechner geeignet, da moderne Computergrafiksysteme spezialisierte Hardware anbieten, um diese Beschreibungsform effizient darzustellen. Insbesondere werden Annäherungen der Oberflächen durch Dreiecke als Flächen bevorzugt. Da für diese Arbeit die Visualisierung der Modelldaten an erster Stelle steht, werden hier ausschließlich B-reps behandelt. Für viele andere Anwendungsgebiete, z. B. für Volumenberechnungen, sind B-reps nur eine indirekte Ausgangsbasis.

Zu den bekanntesten Alternativen der B-reps zählen *Constructive Solid Geometry (CSG)* [63, 34] und *Spatial Occupancy Enumeration* [63]. Bei CSG werden bekannte Grundprimitive (z. B. Kugel, Würfel, Konus) durch boolsche Operationen wie Vereinigung oder Schnittmenge kombiniert, um zur endgültigen Beschreibung eines Körpers zu gelangen. Wegen der exakten parametrischen Modellierung der Grundprimitive sind auch gekrümmte Oberflächen ohne Annäherung beschreibbar. Zur Visualisierung wird jedoch oft aus der CSG-Beschreibung eine B-rep generiert, die effizienter dargestellt werden kann. CSG wird gerne zum Entwurf oder zur Fertigung von mechanischen Bauteilen verwendet [10], wird jedoch in dieser Arbeit nicht weiter betrachtet.

Bei Spatial Occupancy Enumeration wird ein Raumbereich in eine Liste von kleinen Zellen (sog. *Voxel*, von „volume elements“) eingeteilt. Jeder Voxel kann entweder als frei oder belegt markiert sein. Gruppen von benachbarten, belegten Voxels repräsentieren die modellierten Objekte. Je nach Größe und Form der Voxel werden die Oberflächen des Objekts unterschiedlich gut angenähert. Da meistens eine feine Granularität erwünscht wird, müssen entsprechend viele Voxel verwaltet werden. Dazu werden gerne Quad- oder Octrees verwendet. In Kapitel 4.1.4 wird die Aufteilung der entfernten Umgebung des Teleoperators in ähnlicher Form aufgegriffen, eine Voxelbeschreibung im eigentlichen Sinn für die entfernte Umgebung wurde jedoch in dieser Arbeit nicht betrachtet.

Standards für B-reps (Datenstrukturen und Dateiformate)

In den vergangenen Jahren wurde eine Vielzahl von Lösungen entwickelt, um B-reps in Datenstrukturen abzulegen. Genau genommen werden in den meisten Standards anstatt der Datenstrukturen die Dateiformate beschrieben, also die Form in der die Modelldaten persistent auf Datenträgern gespeichert werden können. Jedoch liegt es nahe, die verwendeten Datenstrukturen in ähnlicher Weise wie ihre Dateiformate zu organisieren. Die folgende Auswahl beschreibt nur eine kleine Menge der bekannten und gebräuchlichen Standards.

Der offene Standard *VRML* (Virtual Reality Modeling Language) hat einen relativ hohen Bekanntheitsgrad erlangt. So bieten viele Werkzeuge aus den Bereichen Computer Aided Design (CAD), Modeling und Raytracing Import- und Exportfilter an. 1994 wurde die erste Version von VRML 1.0 veröffentlicht, die letzten Änderungen an dieser Version wurden 1996 gemacht. Die Datenstrukturen von Open Inventor, einem 3D-Toolkit von SGI, wurden als Basis zur Erstellung des Standards übernommen [92, 18]. Ein VRML-Modell besteht aus einem Szenengraph, in den verschiedene Knoten eingetragen werden können. Es gibt Knoten sowohl für Objekte, die visualisiert werden sollen, als auch um das Verhalten der Visualisierungssoftware zu beeinflussen. So kann z. B. der Einsatz von Backface-Culling durch sog. ShapeHints gesteuert werden. Die Knoten eines Szenengraphen können über Gruppierungsknoten in einer hierarchische Struktur angeordnet werden, z. B. um die relative Position zwischen Objekten auszudrücken. VRML wurde mit dem Ziel veröffentlicht, ein Analogon zu HTML (Hypertext Markup Language) für virtuelle Welten zu schaffen. Was mit HTML für text- und bildbasierte Inhalte im Internet sehr erfolgreich begonnen hatte, sollte mit VRML im 3D-Bereich weitergeführt werden. Um die Fertigstellung der ersten VRML Version nicht unnötig zu verzögern, wurde als einziges interaktives Element die Möglichkeit der Hyperlinks (im gleichen Sinne wie bei HTML) im Standard aufgenommen.

1996 wurde VRML 2.0 [93] veröffentlicht. Neben einer Vielzahl neuer Knoten für den Szenengraphen wurde hier spezifiziert, wie der Betrachter verstärkt interaktiv die virtuelle Welt beeinflussen kann, wenn notwendig sogar durch Skriptsprachen unterstützt. Verschiedene Sensorknoten im Szenengraphen können entweder anhand der Systemzeit oder über die Nähe des Betrachters zum Sensor Aktionen im Modell anstoßen. Bereits in VRML 1.0 wurden Texturen beschrieben, um Bilder einzubetten. VRML 2.0 nimmt nun auch Audioobjekte hinzu, die in der virtuellen Welt beliebig positioniert werden können.

Sehr bald danach wurde 1997 VRML97 [94] veröffentlicht. VRML97 erweitert VRML 2.0 durch die Definition einer generischen Programmierschnittstelle. Über diese Schnittstelle können externe Applikationen den Szenengraphen beeinflussen. Die Standards VRML 2.0 und VRML97 sind bis auf die erwähnte Programmierschnittstelle identisch, jedoch sollte beachtet werden, dass VRML97 ein zertifizierter ISO Standard ist (ISO/IEC 14772).

Der X3D Standard ([100], ISO/IEC 19775) führt die Entwicklung der VRML Standards weiter (letzte Änderungen: April 2007). Neben einer Vielzahl neuer Knoten für den Szenengraphen, wie z. B. menschenähnliche Avatars (H-Anim) oder zweidimensionale Geometrieelemente, wurden drei verschiedene Dateiformate spezifiziert. Das erste davon schlägt

eine Kompatibilitätsbrücke zum VRML-Dateiformat (ClassicVRML), ein anderes definiert eine Kodierung in XML (Extensible Markup Language). Das letzte beschreibt eine komprimierte, binäre Kodierung der Dateien. Ein komprimiertes Dateiformat wurde bereits für VRML97 diskutiert [82], jedoch konnte damals keine Einigung erreicht werden, weshalb es erst im X3D Standard untergebracht wurde.

Die bisher vorgestellten Standards wurden primär für den Zweck entwickelt, um 3D-Modelle über das Internet zu übertragen und in der Anwendung des Betrachters darzustellen. Nach der Übertragung ist keine Möglichkeit vorgesehen, diese Modelle durch neue Datenpakete zu verändern. Sowohl VRML97 als auch X3D bieten zwar Programmierschnittstellen an, über die nach dem Laden der Szene neue Daten hinzugefügt werden können, die tatsächliche Spezifikation einer Kommunikationsinfrastruktur, um solche Änderungen fortlaufend zu übertragen, wird erst in MPEG-4 BIFS (ISO/IEC 14496-11 [43]) aufgenommen. Bei einer Teleaktion müssen auf jeden Fall Änderungen im Modell von der entfernten Umgebung zum Operator übertragen werden. MPEG-4 BIFS führt die Konzepte der bereits erwähnten Standards weiter und wird ausführlicher in Kapitel 3 beschrieben.

Ein im Vergleich zu MPEG-4 noch sehr junger Standard ist das Universal 3D File Format [89] (vierte Ausgabe: 2007). Sowohl progressive Übertragung von Änderungen im Modell, als auch Datenkompression sind für diesen Standard vorgesehen. Da hier jedoch nur die Übertragung von Geometriedaten betrachtet wird, erscheint der MPEG-4 Standard durch seine starke Orientierung an der Übertragung multimedialer Inhalte interessanter für den Einsatz in Telepräsenz-Szenarien. Da Universal 3D bereits heute in *Portable Document Format (PDF)* Dokumente eingebunden werden kann, wird dieses Dateiformat in Zukunft jedoch wahrscheinlich mehr Aufmerksamkeit auf sich lenken, als das komplexe MPEG-4 BIFS Format bisher.

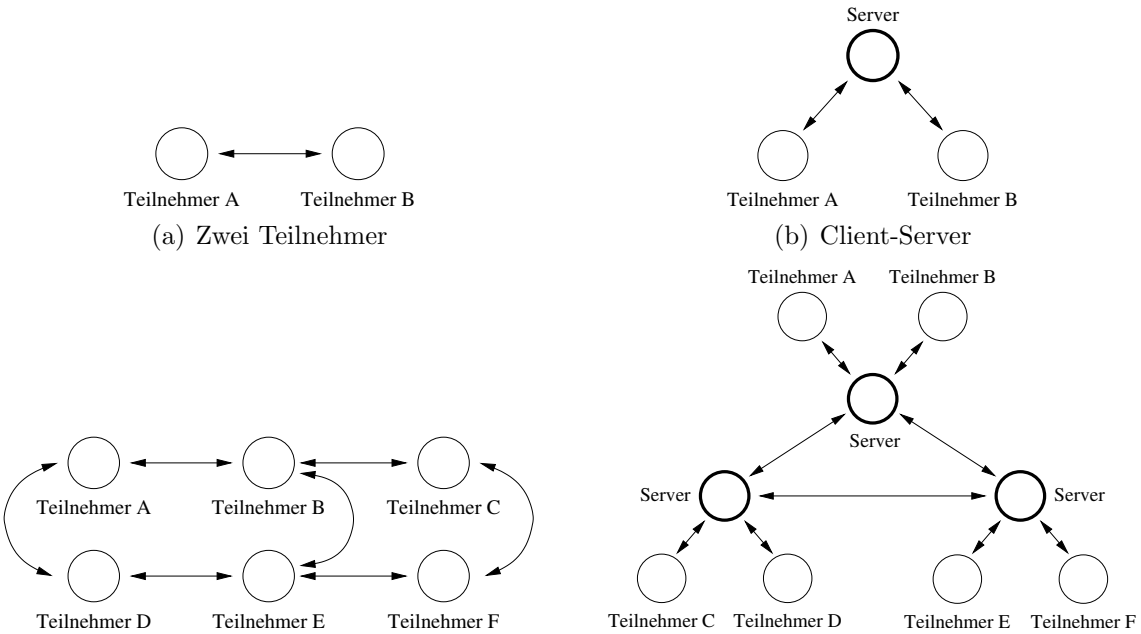
Im Gegensatz zu den bereits erwähnten Standards, mit denen sowohl zwei- als auch dreidimensionale Objekte beschrieben werden können, widmen sich andere Standards wie z. B. SVG (Scalable Vector Graphics) [80] und LASeR (Lightweight Application Scene Representation [45, 46]) ausschließlich zweidimensionalen Objekten. Insbesondere LASeR ist, als Konkurrent zu BIFS im gleichen Standard, auf Anwendungen im Bereich eingebetteter Systeme ausgerichtet, weshalb es auch ein platzsparendes, binäres Dateiformat anbietet. Da für Telepräsenz-Szenarien die entfernte Umgebung in den allermeisten Fällen als 3D-Modell rekonstruiert wird, werden sie hier nicht weiter betrachtet.

Andere Formate wie STL (Standard Tesselation Language) benutzen keinen Szenengraphen, sondern speichern nur eine Menge von soliden Objekten, deren 3D-Punkte in einem gemeinsamen Welt-Koordinatensystem angegeben werden. Dieses Vorgehen ist für die Anwendung 3D-Drucken, für welche STL ursprünglich entwickelt wurde, praktikabel. Für virtuelle Welten ist dies jedoch eher unpraktisch, da z. B. bei artikulierten Objekten (Roboterarme) gerne eine tiefe Hierarchie für die Modellierung verwendet wird. Einige Programme aus dem Modeling und CAD Bereich bieten Import- und Exportfilter für STL an.

Verteilte virtuelle Welten: Forschung

In der Einleitung wurde bereits motiviert, dass für eine kooperative Teleaktion mit mehreren Teleoperatoren und Operatoren eine verteilte virtuelle Welt, auf deren Inhalte alle Teilnehmern gleichermaßen zugreifen können, sinnvoll ist.

Nach Singhal [75] sind für die sog. „networked virtual environments“ prinzipiell vier verschiedene Realisierungen der Softwarearchitektur möglich. Die einfachste Lösung behandelt den Fall für genau zwei Teilnehmer, die sich Modelldaten und Veränderungen über eine Netzwerkverbindung direkt zuschicken (siehe Abbildung 2.1(a)). Im Fall der Telepräsenz entspricht dies einem Teleoperator und einem Operator. Sobald mehr Teilnehmer erforderlich sind, können die Möglichkeiten in Client-Server und Peer-to-Peer Systeme eingeteilt werden. Bei einer reinen Client-Server Lösung kommuniziert jeder Teilnehmer (Client) mit einem zentralen Server (siehe Abbildung 2.1(b)). Für die Kommunikation zwischen Client und Server können ähnliche Protokolle zum Datenaustausch wie bei der ersten Lösung verwendet werden. Bei Peer-to-Peer Lösungen kann theoretisch jeder Teilnehmer mit einem oder mehreren der anderen Teilnehmer kommunizieren (siehe Abbildung 2.1(c)). Für sehr viele Teilnehmer müssen hierzu Maßnahmen getroffen werden, um die Anzahl der Kommunikationsverbindungen auf ein sinnvolles Maß zu limitieren. Die letzte Lösungsmöglichkeit besteht in der Kombination aus Client-Server und Peer-to-Peer Technik: mehrere Server gleichen untereinander Modellinformationen über eine Peer-to-Peer Lösung ab und bieten diese Informationen jeweils ihren lokalen Clients an (siehe Abbildung 2.1(d)).



(a) Zwei Teilnehmer (b) Client-Server (c) Peer-to-Peer (d) Kombination aus (b) und (c)

Bild 2.1: Mögliche Realisierungsformen für verteilte virtuelle Welten

Allen Ansätzen stellt sich das grundsätzliches Problem des sog. „Consistency-Throughput Tradeoff“ [75]. Danach ist es unmöglich, dass Objekte in einer verteilten virtuellen Welt

mit hoher Rate ihren Zustand (z. B. ihre Position im Raum) verändern, und gleichzeitig zu garantieren, dass alle Teilnehmer identisches Wissen über diese Veränderung besitzen. Je nach Anforderung der Anwendung können entweder höhere Aktualisierungsraten des dynamischen Zustands erlaubt werden, oder es kann mehr Gewicht auf eine hohe Konsistenz der Zustandsinformationen zwischen den Teilnehmern gelegt werden.

Eines der ersten Projekte zu verteilten virtuelle Welten war das Projekt SIMNET (simulator networking) der DARPA [62], mit dessen Entwicklung 1983 begonnen wurde. Ziel war die Entwicklung eines Simulators für kleinere militärische Einheiten zu Trainingszwecken. Das System erlaubte bis zu 850 Teilnehmer [56]. Die simulierte Welt wurde in das unveränderliche Terrain und veränderbare Objekte (z.B. Panzer, oder auch zerstörbare Brücken) unterteilt. Zum Abgleich der Objektzustände wurde ein Protokoll verwendet, das sich sehr stark an der militärischen Anwendung orientiert. In jedem Datenpaket werden neben der aktuellen Positionsbeschreibung weitere Informationen über Geschwindigkeiten übermittelt. Mit Hilfe dieser Informationen sind die einzelnen Teilnehmer in der Lage, durch Extrapolation der Objektinformationen deren Positionen für einen kurzen Zeitraum zu schätzen. In Bezug auf den oben angesprochenen Consistency-Throughput Tradeoff werden hier durch die Extrapolation Inkonsistenzen zugelassen. Es ist jedoch den Teilnehmern erlaubt, ihre Zustandsdaten mit hoher Rate zu verschicken.

Das Nachfolgeprojekt zu SIMNET hieß *Distributed Interactive Simulation Protocol* (DIS), aus dem der gleichnamige IEEE Standard 1278 entstand [13]. Aufbauend auf SIMNET wurde durch den Standard eine generische, jedoch immer noch für militärische Zwecke geschaffene Schnittstelle veröffentlicht, die die Entwicklung von vielfältigen, teilweise autonom gesteuerten Teilnehmern ermöglichte. In abgewandelter Form konnten mit dem DIS Protokoll Szenarien mit bis zu 5 000 Teilnehmer realisiert werden [5]. Eine Weiterentwicklung des DIS Standards ist der HLA (high level architecture) Standard, der im Jahr 2000 als IEEE 1516 spezifiziert wurde und Basis für Forschungsprojekte in den verschiedensten Domänen ist (z. B. [53] oder [78]).

Auf der Seite der nichtmilitärischen Forschung sind die Projekte NPSNET [54, 6, 60], PARADISE [76, 35, 61], DIVE [23, 24], BrickNet [73, 74] und MR Toolkit Peer Package [70, 71] beispielhaft zu nennen. Eine Übersicht wurde in [55] veröffentlicht. Das NPSNET System entwickelte sich über mehrere Version von einer Simulation eines einzigen Flugkörpers hin zu einer virtuellen Welt mit vielen Aktionsmöglichkeiten für die Teilnehmer. Ebenso wie PARADISE orientiert es sich am DIS Protokoll, die virtuelle Welt ist ähnlich wie bei SIMNET aufgebaut. Bei PARADISE wurde allerdings die Aufteilung in statische und veränderbare Objekte aufgehoben, wodurch auch das Terrain verändert werden kann. Da das Datenvolumen für eine regelmäßige Erneuerung von Terraingeometrie zu umfangreich ist, kann hier eine variable Rate gewählt werden. DIVE realisierte in seinen ersten Versionen den Abgleich seiner Geometrieobjekte über ein netzwerkweites Shared Memory, wodurch die Anzahl der Teilnehmer eingeschränkt war. Neuere Implementierungen kommunizieren über zuverlässige Multicast-Protokolle. In [29] ist beschrieben, wie mit dem dedizierten Netzwerkserver DIVEBONE die Anzahl der Teilnehmer gesteigert werden konnte. Im BrickNet-System ist jeder Teilnehmer für einen Teil der virtuelle Welt zuständig. Werden Daten eines anderen Teilnehmers benötigt, wird über einen speziellen

Server ermittelt, wo die gesuchten Daten liegen. Das MR Toolkit Peer Package realisiert den Datenabgleich mittels Peer-to-Peer Kommunikation. Da hier keine Extrapolation der Objektzustände (vgl. SIMNET) verwendet wurde, mussten sehr hohe Aktualisierungsraten für jedes Objekt implementiert werden. Als Folge konnten aufgrund der hohen benötigten Netzwerkbandbreite nur sehr wenige Teilnehmer zugelassen werden.

Eine verteilte virtuell Welt neueren Datums entstand im Rahmen des blue-c Projekts [30, 31]. In [58] wird der grundlegende Aufbau beschrieben. Die Knoten in einem hierarchischen Szenengraphen, der auf OpenGL Performer [79] basiert, werden zwischen den Teilnehmern mittels UDP Multicast Nachrichten aktualisiert. Um zu garantieren, dass jeweils immer nur ein Teilnehmer einen Konten verändern kann, wurde ein Reservierungsschema entwickelt, um die Konsistenz der Modellinformationen zu gewährleisten. Nicht über UDP Multicast, sondern über Punkt-zu-Punkt Datenverbindungen baut das Myriad Projekt [65] sein Peer-to-Peer Netzwerk auf. Durch eine Filterung der Änderungsnachrichten kann dieses System besser für Datennetzwerke mit inhomogenen Bandbreiten skalieren. Je nach Abstufung dieser Filterung werden absichtlich Inkonsistenzen zugelassen, um Teilnehmer mit geringer Bandbreite nur die wichtigsten Änderungen gezielt zukommen zu lassen.

Verteilte virtuelle Welten: Kommerzielle Produkte

Vor allem im Bereich der spielerischen Erholung werden verteilte virtuelle Welten schon längere Zeit eingesetzt. Doom (id Software, [17]) war eines der ersten dieser sog. First-Person-Shooter (FPS) Spiele[1)]. Bei dieser Art von Spielen können sich mehrere Spieler in einer virtuellen Welt bewegen, und kooperativ ein bekanntes Missionsziel verfolgen. Die virtuellen Welten basierten anfangs noch nicht auf echten 3D-Modellen. Doom war bekannt dafür, dass es die Welt als zweidimensionalen Grundriss speichert. Als Folge dieser Einschränkung mussten alle Wände der virtuellen Welt senkrecht sein, und Elemente wie Brücken oder Tunnel mussten gesondert im Programm behandelt werden. Da für diese Spiele die virtuelle Welt bereits a-priori bekannt ist, wird sie zu Anfang einer Mission einmalig an alle Teilnehmer übertragen. Während der Mission werden nur noch bekannte Elemente, wie z. B. die Spieleravatars, bewegt. Als Kommunikationsarchitektur wird durchgehend das Client-Server Modell verwendet. Zu den neueren Varianten dieses Genre zählen Counter Strike (Valve), die Half Live Serie (Valve) und Far Cry (Crytek).

Derzeit ist World-Of-Warcraft (Blizzard, [99]) ein richtiger Publikumsmagnet. Als typischer Vertreter für ein sog. MMORPG (Massively Multi-User Online Role-Playing Game) betrachtet der Spieler das Geschehen nicht wie bei FPS Spielen aus der Ego-, sondern aus der Verfolgerperspektive. Auch für Spiele dieses Genre werden verteilte virtuelle Welten verwendet. Über die implementierte Technologie wird jedoch nur äußerst wenig Information publiziert. So lassen sich Angaben von mehreren tausenden gleichzeitig aktiven Spielern auf einem Server nur relativ schlecht einordnen. Weitere Vertreter dieses Genre sind Rappelz (gpotato), A Tale in The Desert 3 (eGenesis) sowie die Ultima Online

[1)] Sein Vorgänger Wolfenstein 3D wurde wegen der Verwendung von Kennzeichen verfassungswidriger Organisationen in Deutschland beschlagnahmt.

(Electronic Arts) und Everquest (Sony Online Entertainment) Serien.

Mehr Informationen sind für Second Life (Linden Lab, [66]) verfügbar, in dem eine virtuelle Welt von seinen Benutzern gestaltet wird. Ebenso wie bei FPS oder MMORPG Spielen wird das Client-Server Modell verwendet. Das auf UDP basierende Kommunikationsprotokoll wurde veröffentlicht [67], um die Entwicklung von Erweiterungen zu ermöglichen. Der Grundriss der Welt ist in rechteckige Regionen eingeteilt, deren Terrain über eine Höhenkarte (256 auf 256 Pixel, 13 Bit für jede Höhenangabe) modelliert wird. Auf das Terrain können die Bewohner ihre selbst gestalteten Objekte platzieren, für die sie jedoch auch bezahlen müssen. Ähnlich wie bei CSG werden Objekte aus Grundprimitiven zusammengesetzt. Eine Vielzahl weiterer virtuellen Welten, mit ähnlicher Zielsetzung wie Second Life, werden in [91] und [3] aufgeführt.

Ein weiteres Anwendungsgebiet, in dem auch 3D-Modelle verwendet werden, sind Geoinformationssysteme. Während der Fokus hier eigentlich auf der Verwaltung und Zusammenführung von 2D und 2,5D-Daten der Erdoberfläche in einer großen Datenbank liegt, werden für besondere Zwecke auch 3D-Datensätze von Gebäuden in die Datenbanken mit aufgenommen. Neben Datenbanken, deren Datensätze als normale Dateien verfügbar sind (z. B. GTOPO30 [32]), werden inzwischen auch Dienste angeboten, die benötigten Daten nach dem Client-Server Modell über spezielle Protokolle nach Bedarf abrufen. Bekannte Beispiele für solche Dienste sind Google Earth (Google, [28]), World Wind (NASA, [59]) oder Windows Live Maps (Microsoft, [96]). Einige dieser etablierten Datenbanken erlauben auch Änderungen und Ergänzungen durch die Benutzer.

Forschungsthemen im Bereich MPEG-4 BIFS

MPEG-4 BIFS wurde bereits weiter oben als Standard für eine Beschreibung durch B-reps erwähnt. Ein kurzer Überblick zu BIFS wurde in [72] veröffentlicht. Der gesamte MPEG-4 Standard umfasst jedoch ein wesentlich breiteres Spektrum an spezifizierter Funktionalität. In den Standard sind viele Ergebnisse aus der Forschung eingeflossen, und viele Forschungsarbeiten widmen sich weiterhin der Erweiterung und Verbesserung von MPEG-4. Zu den weitläufig bekanntesten Elementen von MPEG-4 zählen vor allem die Kompressionsverfahren für Video- und Audiodaten. Mehrere kommerzielle Produkte (z. B. DivX, [14]) und Open Source Projekte [101, 21] entwickeln seit geraumer Zeit Implementierungen für die Videokompression, die weitverbreitet Einsatz findet.

Für die Kodierung von Geometrie ist neben der C++ Referenzimplementierung (frei erhältlich von [47]), die Teil des Standards ist, mit GPAC [20, 51] ebenfalls ein Open Source Projekt bekannt. GPAC wird in C entwickelt und bietet zusätzlich zu MPEG-4 viele weitere 3D-Standards als Dateiformat (z.B. VRML, X3D) an. Ein weiteres Software Development Kit (SDK) wird in [22] beschrieben, das, aufbauend auf der Referenzimplementierung, insbesondere für die Arbeit mit animierten Avatars entwickelt wurde. Eine Übersicht zu Software im MPEG-4 Umfeld findet man in [52]. Neben Authoring, Encoding und Decoding werden hier auch einige Darstellungsapplikationen genannt. Leider werden einige der dort vorgestellten Anwendungen nicht mehr aktiv entwickelt. Insbesondere zum Thema Authoring wurden weitere Arbeiten veröffentlicht. Sowohl in [11, 12], als

auch in [86, 87], [2] und [98] werden Projekte beschriebenen, mit denen MPEG-4 Inhalte basierend auf dem BIFS Szenengraphen erstellt werden können. Die genannten Werkzeuge unterstützen das Authoring von 3D-Inhalten. Ebenfalls mit dem Thema Authoring beschäftigt sich [102], hier jedoch liegt der Fokus auf der Bereitstellung der MPEG-4 Datenströme über Netzwerkverbindungen. Die zuerst genannten Arbeiten konzentrieren sich auf die Ablage in MP4-Dateien.

Dass mit MPEG-4 kollaborative, virtuelle Welten implementiert werden können, wird von Hosseini und Georganas in [36] beschrieben. Hier werden auch Angaben zu erzielbaren Kompressionsraten von BIFS zu VRML Modellen, sowie Zeitangaben für die notwendigen Kodierungsalgorithmen angegeben. Sie setzen jedoch keines der BIFS Kompressionswerkzeuge ein. In [38] beschreiben sie weiterhin, welche Vorgehensweisen für die Erstellung von großen virtuellen Welten mit vielen bewegten Objekten günstig sind. Eine andere Verwendungsmöglichkeit von MPEG-4 beschreiben Hosseini und Georganas in [37], wo MPEG-4 als Basis genutzt wird, um Vorgänge in einer virtuellen Welt aufzuzeichnen. Die gesammelten Daten können zu einem späteren Zeitpunkt beliebig oft abgespielt und analysiert werden. Für das CAD-Umfeld wird in [15, 16] beschrieben, wie MPEG-4 für die kooperative Konstruktion von Bauteilen durch mehrere Teilnehmer verwendet werden kann.

Auch dem Gebiet der komprimierten Kodierung von MPEG-4 Animationen widmen sich einige Veröffentlichungen. In [48] gibt Jang einen Überblick über dieses Thema. Seine Arbeitsgruppe publizierte später in [50] und [49] ein Verfahren, um die für Animationen wichtigen Interpolatoren zu komprimieren. Bei diesem Verfahren treten bei gleicher Bitrate weniger Verluste durch die Kompression auf, als bei dem in MPEG-4 vorgestellten Predictive MFField Werkzeug[2)]. Für die komprimierte Kodierung der MPEG-4 Interpolatoren wurden seine Arbeiten in den Standard integriert [43]. Die in [8] veröffentlichte Arbeit von Concolato et al. hingegen untersucht, ähnlich wie diese Arbeit, die Performanz der bereits im MPEG-4 Standard definierten Kompressionswerkzeuge. Allerdings werden als Modelle nur animierte 2D Objekte, wie sie für Cartoons verwendet werden, untersucht, und es werden keine Angaben zu den benötigten Kodierungszeiten gemacht.

Bereits bei den Authoring Werkzeugen waren Arbeiten aus der Gruppe um Marius Preda vertreten. In den weiteren Veröffentlichungen wird in [7] eine Softwarearchitektur vorgestellt, mit der MPEG-4 Inhalte effizient dargestellt werden können. Um die Leistungsfähigkeit des MPEG-4 Szenenbeschreibung zu demonstrieren, wurde weiterhin in [88] und [84] publiziert, wie damit interaktive Spiele realisiert werden können. Einzig die Möglichkeit der persistenten Datenspeicherung (z. B. Spielstände) wurde als Erweiterung des Standards gefordert und in [85] beschrieben. Neuere Arbeiten integrieren MPEG-4 als Kommunikationsprotokoll in Netzwerkspiele [57], die sowohl auf herkömmlichen PCs, als auch auf Mobiltelefonen lauffähig sind. Insbesondere die Fähigkeiten für Level of Detail und skalierbare Datenübertragung sind hier die Stärken von MPEG-4.

In [77, 19] werden große Datenmengen mittels MPEG-4 zum Betrachter übertragen. Zur Darstellung hoch aufgelöster Panoramabilder wird ein Zylinder in einem BIFS Szenen-

[2)] In Kapitel 3.3.3 folgt eine Erläuterung des Predictive MFField Werkzeugs.

graph durch viele kleine Flächen modelliert. Auf diese Flächen werden Teile des Panoramas als Texturen gelegt. Da für die aktuelle Blickrichtung des Betrachters nicht das vollständige Panoramabild im Grafikspeicher vorhanden sein muss, benötigt die Darstellung nur wenig Ressourcen.

Eine Anwendung, für die über MPEG-4 eine komprimierte Geometrieübertragung realisiert wurde, wird in [9] beschrieben. Für Location Based Services (LBS) wird dem Betrachter auf seinem Mobiltelefon ein 3D-Modell der Umgebung parallel zu einem Videoclip dargestellt. Aufgrund der beschränkten Datenrate in dieser Anwendung müssen die 3D-Modelle stark komprimiert werden, wofür hier zwar nicht das in MPEG-4 beschriebene 3DMC Verfahren, jedoch der dazu ähnliche Delphi-Algorithmus verwendet wurde.

Kompression von B-reps

Geometriekompression ist ein großes Forschungsgebiet. Da in dieser Arbeit jedoch kein neuer, eigenständiger Algorithmus vorgestellt wird, folgt an dieser Stelle kein umfassender Überblick. Es folgt nur eine Auswahl von Arbeiten, die im Umfeld dieser Arbeit relevant sind. Insbesondere die Veröffentlichungen von Gabriel Taubin seien hier an erster Stelle genannt, da er – zusammen mit anderen – Autor von zwei Arbeiten ist, die die Basiselemente für das 3D Mesh Coding (3DMC) Verfahren bilden. 3DMC ist Teil des MPEG-4 Standards. Ausgehend von einem Dreiecksnetz zerschneidet der „Topological Surgery“ (TS) Algorithmus [83] das Netz entlang von Dreieckskanten in mehrere, zusammenhängende Bänder aus Dreiecken. Gespeichert werden Informationen zu den Schnittkanten, den (quantisierten) Eckpunkten aller Dreiecke und zur Rekonstruktion der restlichen Dreieckskanten. Besonders effizient wird der Algorithmus, wenn die Dreiecksnetze in wenige, aber dafür lange Dreiecksbänder zerschnitten werden kann. Mittels TS können beeindruckende Kompressionsraten von 50 : 1 und mehr erreicht werden. Allerdings benötigt das Verfahren zum Enkodieren wesentlich mehr Rechenzeit als die einfacheren Verfahren, die in dieser Arbeit behandelt werden. Aufbauend auf TS definiert das „Progressive Forest Split“ (PFS) Verfahren [81], wie aus einem zuerst grob modellierten Netz durch schrittweise Verfeinerungsoperationen mehr Dreiecke eingefügt werden können. Ähnlich wie bei TS wird das Netz stellenweise entlang von Dreieckskanten zerschnitten, und das entstehende Loch durch neue Dreiecke aufgefüllt. Die betroffenen Eckpunkte können als letzter Schritt noch verschoben werden. Abschließend sei hier noch erwähnt, dass für die 3DMC Geometriekompression in MPEG-4 auch Untersuchungen durchgeführt wurden, welchen Einfluss dieses Verfahren auf Wasserzeichen hat [25]. Das Ergebnis war, dass die gewählten Algorithmen für Wasserzeichen – natürlich nur bis zu einem gewissen Grad der 3DMC Quantisierung – recht resistent gegen die verlustbehaftete Kompression waren.

Ein alternatives Verfahren zur Kompression von Dreiecksnetzen wurde in [33] vorgestellt. Ausgehend von einem initialen Dreieck werden neue Dreiecke nacheinander hinzugefügt. Dazu stehen 7 verschiedene Operationen zur Verfügung. Die erreichten Kompressionsraten sind, bezogen auf die Kodierung der Verbindungsinformationen, vergleichbar mit denen von TS. Die veröffentlichten Kodierungszeiten sind jedoch wesentlich niedriger. Ein potenzieller Nachteil hier ist, dass die dekodierten Netze nicht in Form von Dreiecksbändern

vorliegen. Eben diese Bänder können effizient durch moderne Grafikhardware dargestellt werden.

Für weitere Arbeiten im Bereich Geometriekompression sei hier auf [26] und [27] verwiesen. Beide Internetseiten verweisen auf eine Vielzahl von Veröffentlichungen im Bereich Geometriekompression.

3 MPEG-4 Grundlagen

Der ISO/IEC Standard 14496 mit dem Titel „Information Technology - Coding of audio-visual Objects“ ist auch unter dem Begriff MPEG-4 bekannt. Er gliedert sich unter anderem in folgende Teile [42]:

- Part 1: Systems
- Part 2: Visual
- Part 3: Audio
- Part 6: Delivery Multimedia Integration Framework (DMIF)
- Part 11: Scene Description and Application Engine

Der Aufbau des Szenengraphen und die Datenstrukturen für die Beschreibung von Geometrie werden in Teil 11 spezifiziert. Dieser Teil wird für die komprimierte Übertragung des B-rep Modells der entfernten Umgebung benötigt. Wie diese Szenendaten mit Video- und Audiodaten (Teil 2 und 3) zu einer Multimedia-Präsentation kombiniert werden, beschreibt Teil 1 mit dem MPEG-4 System. Das MPEG-4 System wird ebenfalls hier eingesetzt. Allerdings liegt der Schwerpunkt nicht auf der Integration von Audio und Video, sondern in der Verwendung des sog. Object Descriptor Frameworks, um die Geometriedaten der entfernten Umgebung auf mehrere Datenströme aufzuteilen. Für die Übertragung der Datenströme benutzt ein MPEG-4 System die DMIF Schnittstelle (Teil 6). Die restlichen Teile des MPEG-4 Standards werden in dieser Arbeit nicht weiter betrachtet.

3.1 MPEG-4 Systemaufbau

In Teil 1 des Standards wird ein System beschrieben, um mehrere interaktive, audiovisuelle Inhalte zu Übertragen, die in einer Multimedia-Präsentation zusammengefasst werden. Ein typischer Anwendungsfall ist die Übertragung der Präsentation von einem Sendeterminal über Datennetzwerke zu einem Empfangsterminal. Das Sendeterminal stellt die Inhalte der Präsentation als einzelne, MPEG-4-konform kodierte Datenströme bereit, die vom Empfangsterminal empfangen, dekodiert und angezeigt werden.

Die audiovisuellen Inhalte sind sowohl zeitlich, als auch räumlich in eine Szene eingebettet. Somit ergeben sich die ersten drei möglichen Datentypen, die als einzelne Datenströme übertragen werden können:

- Eine geometrische Beschreibung der Szene, die präsentiert wird (2D oder 3D)

- Audio-Inhalte der Präsentation
- Video-Inhalte der Präsentation

Ein solcher Datenstrom wird *Elementary Stream* genannt und ist immer einem Medienobjekt vom gleichen Typ zugeordnet (Szenen-, Audio- oder Videoobjekt). Ein Medienobjekt kann mehr als nur einen Elementary Stream empfangen. So kann ein Audioobjekt aus mehreren Elementary Streams die gewünschte Sprache, oder ein Videoobjekt die gewünschte Bildqualität wählen. Eine Präsentation besteht typischerweise aus mehreren dieser Medienobjekte. Es wurden weitere Elementary Streams definiert, die zusätzliche Informationen für die Präsentation übermitteln. Als Beispiel sei hier auf den *Object Descriptor Stream* (ODS) verwiesen, der in Kapitel 3.4 beschrieben wird.

Abbildung 3.1 stellt den grundlegenden Aufbau eines Empfangsterminals dar. Die durchgezogenen Pfeile zwischen den Blöcken verdeutlichen dabei die notwendigen Schritte zur Verarbeitung der Elementary Streams. Die DMIF-Schnittstelle definiert eine generische, für viele Transportmechanismen verwendbare Schnittstelle, um Datenpakete zu empfangen (siehe Abschnitt 3.6). Diese Datenpakete werden als sog. *SyncLayer*-Pakete (SL) bezeichnet, da sie als erstes von dieser Ebene im System bearbeitet werden.

Der Zweck der SL-Pakete ist, dass sie neben den eigentlichen Daten für das Medienobjekt zusätzlich Informationen enthalten, die vor allem der zeitlichen Synchronisation dienen. Über regelmäßig empfangene Zeitstempel kann das Empfangsterminal den Verlauf der lokalen Uhr des Sendeterminals schätzen. Weiterhin enthalten SL-Pakete den gewünschten Darstellungszeitpunkt für ihren Inhalt, bezogen auf die rekonstruierte Uhr. Die SyncLayer ist dafür zuständig, die Nutzdaten zeitgerecht für die weitere Verarbeitung zu liefern.

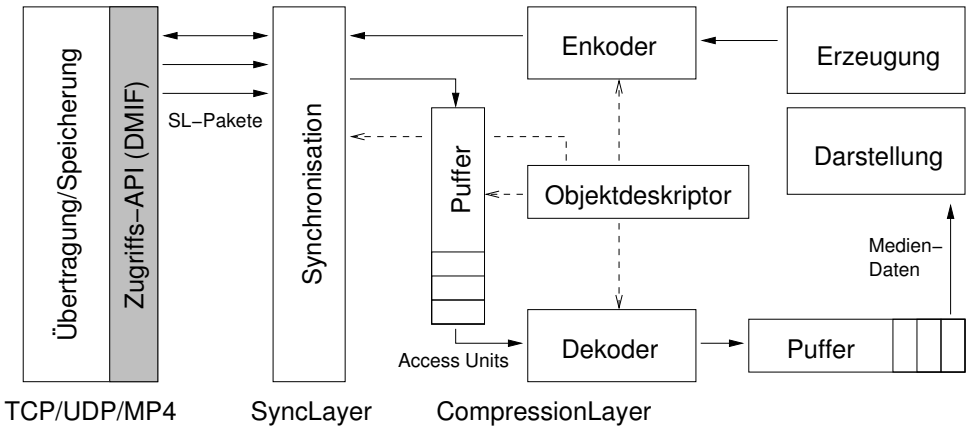


Bild 3.1: Empfangendes Terminal als Systemblockschaltbild

Die Nutzdaten der SL-Pakete sind beim Sendeterminal durch eine Zerteilung größerer Einheiten, der sog. *Access Unit*s (AU) entstanden. In der CompressionLayer muss diese Teilung zuerst in einem Eingangspuffer rückgängig gemacht werden. Die rekonstruierten Access Units werden anschließend von einem Dekoder verarbeitet. Bei Video Elementary Streams wird pro Access Unit Information für einen Videoframe übertragen, ähnlich

wie bei Audio Elementary Streams. Bei BIFS Elementary Streams beinhalten die Access Units *BIFS-Update* oder *BIFS-Anim* Kommandos (siehe Kapitel 3.2.6). Über BIFS-Update Kommandos wird ein Szenengraph aufgebaut und verändert. Über BIFS-Anim können kontinuierliche Veränderungen in diesem Szenengraphen kodiert werden.

Die aus den Access Units dekodierten Mediendaten (Videobilder, Audio Samples oder BIFS Kommandos) werden in einem Kompositionsspeicher abgelegt. Zum gegebenen Zeitpunkt stellt das Empfangsterminal diese Daten dann dem Betrachter dar.

Der Standard sieht auch die Möglichkeit vor, dass Daten zum Sendeterminal zurückgeschickt werden können. In Kapitel 5.2.1 wird diese Möglichkeit genauer betrachtet. Im Prinzip läuft der Verarbeitungsprozess hier umgekehrt. Sobald die zu verschickenden Daten generiert wurden, werden sie in eine Access Unit enkodiert. Nach einer für die Übertragung passende Zerteilung in kleinere Datenpakete werden diese mit SyncLayer-Informationen versehen und als SL-Pakete über die DMIF-Schnittstelle versendet. Der Standard legt auch Protokolle für diese sog. *Upstreams* fest. In dieser Arbeit wird das für Szenendaten definierte *ServerCommandRequest* Protokoll verwendet, um Änderungen im Szenengraphen an das Sendeterminal zu signalisieren.

Viele Parameter, die sowohl für den Empfang, als auch die Dekodierung der Elementary Streams notwendig sind, werden in *Objektdeskriptoren* verwaltet. Für jedes Medienobjekt existiert genau ein solcher Objektdeskriptor. In Kapitel 3.4 wird näher beschrieben, wie diese Objektdeskriptoren an die Empfangsterminals übertragen werden.

3.2 Binary Format for Scenes – BIFS

Unter dem Namen BIFS (Binary Format for Scenes) wird eine binäre Kodierung eines hierarchischen Szenengraphen als Folge von Bits (Bitstrom) beschrieben. Im Szenengraphen werden Geometriedaten einer Szene abgelegt, die zur Wiedergabe einer audiovisuellen MPEG-4 Präsentation notwendig sind.

Die binäre Kodierung umfasst einen Satz von Kommandos, mit denen der Szenengraph entweder vollständig oder nur Änderungen im Graphen als Bitstrom kodiert werden können. Diese Kommandos werden als BIFS-Update bezeichnet. Ein oder mehrere als Bitstrom kodierte BIFS-Updates bilden eine Access Unit für Szenendaten.

Die nächsten Abschnitte stellen die wichtigsten Eigenschaften vor, die ein Szenengraph für eine Kodierung in BIFS besitzen sollte. Danach folgt eine Beschreibung der BIFS-Update Kommandos.

3.2.1 Struktur des Szenengraphen

Der Szenengraph für BIFS orientiert sich stark am VRML97 Standard (ISO/IEC 14772 [94]). Grundlegende Datenstruktur ist ein Baum aus Knoten. Es sind mehr als hundert

verschiedene Knotentypen definiert, von denen hier jedoch nur wenige benötigt und beschrieben werden. Zum einen gibt es Knoten, in denen tatsächlich Geometriedaten gespeichert werden. Die überwiegende Mehrheit der Knoten jedoch beeinflusst die Darstellung und Kodierung der Geometrieknoten, oder legt Beziehungen zwischen Geometrieknoten fest.

Jedem Knoten kann eine eindeutige Nummer zugewiesen werden. Diese sog. Knoten-ID muss im gesamten Szenengraphen eindeutig sein. Benötigt werden Knoten-IDs, um einzelne Knoten im Szenengraphen zu referenzieren. Wenn ein Knoten nicht unbedingt referenziert werden muss, dann kann auf diese Zuweisung verzichtet werden. Der gültige Wertebereich der Knoten-IDs liegt im Intervall $[0 \dots 2^{nb} - 2]$. Dabei ist nb die Anzahl der Bits, mit der Knoten-IDs kodiert werden. Dieser Parameter wird vom Sendeterminal festgelegt (siehe Abschnitt 3.4.2). Der höchstmögliche kodierbare Wert $2^{nb} - 1$ ist für einen besonderen Zweck reserviert. Er wird verwendet, um explizit anzugeben, dass auf einen nicht existierenden Knoten verwiesen wird. Je größer nb gewählt wird, desto mehr Bits entfallen im Bitstrom auf die Kodierung der Knoten-IDs. Im Vergleich zur Kodierung der Geometrie spielt dieser Parameter jedoch meist eine sehr untergeordnete Rolle, und kann entsprechend großzügig gewählt werden.

Für jeden Knoten ist eine Liste von Feldern definiert. Im Standard wurden 14 verschiedene Feldtypen festgelegt, die jeweils unterschiedliche Daten speichern können. Jeder Typ ist in zwei Ausprägungen definiert: die Erste um genau ein Datenelement abzuspeichern. Deshalb wird der Typbezeichnung ein `SF` (wie *single value field*) vorangestellt. Bei der Zweiten Ausprägung können beliebig viele Datenelemente vom gleichen Datentyp als Liste abgelegt werden, und der Typbezeichnung wird ein `MF` (wie *multiple value field*) vorangestellt. Tabelle 3.1 zeigt alle definierten SF-Datentypen. Die angegebenen Eigenschaften der Datenelemente gelten ebenso für die korrespondierenden MF-Typen. Eine besondere Stellung nehmen die Feldtypen `SFNode` und `MFNode` ein. Sie beinhalten einen oder mehrere andere Knoten, wodurch die Möglichkeit zum Aufbau der Baumstruktur gegeben wird.

Der MPEG-4 Standard definiert für alle bis auf einen Knotentypen exakt, welche Felder verfügbar sind, und wie diese Felder kodiert werden sollen. Es wurde jedoch auch die Möglichkeit vorgesehen, dass anwendungsspezifische Knotentypen hinzugefügt werden können. Diese sog. PROTO Knoten werden an dieser Stelle jedoch nicht benötigt.

An der Wurzel des Szenengraphen darf im Kontext von MPEG-4 nur ein einziger Konten stehen. Bei VRML97 sind mehrere Wurzelknoten zulässig. Weiterhin darf der Wurzelknoten nur einer der folgenden Typen sein: `Group`, `OrderedGroup`, `Layer2D` oder `Layer3D`.

3.2.2 Mehrfachverwendung von Knoten

Um einen bestimmten Knoten mehrfach im Szenengraphen nutzen zu können, gibt es die Möglichkeit, einen besonderen Verweisknoten einzufügen. Der Verweisknoten referenziert den mehrfach verwendeten Knoten über dessen Knoten-ID. Deshalb kann ein Knoten auch nur dann mehrfach verwendet werden, wenn ihm eine gültige Knoten-ID zugewiesen

Feldtyp	Beschreibung und nominale Datenlänge
`SFBool`	Boolean-Wert: `wahr` oder `falsch`, 1 Bit
`SFFloat`	Fließkommazahl, 32 Bit
`SFTime`	Zeitangabe in Sekunden, 64 Bit Fließkommazahl
`SFInt32`	Ganzzahliger, vorzeichenbehafteter Wert, 32 Bit
`SFString`	Zeichenkette (UTF-8 kodiert, ISO/IEC 10646-1[90])
`SFVec2f`	2D Vektor, 2 Fließkommazahlen à 32 Bit
`SFVec3f`	3D Vektor, 3 Fließkommazahlen à 32 Bit
`SFVec4f`	4D Vektor, 4 Fließkommazahlen à 32 Bit
`SFColor`	RGB-Vektor, 3 Fließkommazahlen à 32 Bit, Wertebereich [0..1]
`SFRotation`	3D-Rotationsvektor und Winkel, 4 Fließkommazahlen à 32 Bit
`SFImage`	2D Pixelbild, verschiedene Farbtiefen möglich
`SFNode`	Enthält einen Knoten
`SFUrl`	Verweis auf eine andere Datenquelle; entweder als 10 Bit Identifizierungsnummer, oder wie `SFString`
`SFCommandBuffer`	enthält ein CommandFrame (siehe 3.2.6)
`SFScript`	Ein ECMA-Skript[a)] beliebiger Länge

Tabelle 3.1: Die verfügbaren SF-Felder

[a)] ISO/IEC 16262, besser bekannt als JavaScript

wurde.

Wird ein Verweisknoten im Szenengraph verwendet, beeinflusst er die Speicherverwaltung des mehrfach genutzten Knotens. Je nach angewendetem BIFS-Update Kommando wird zum Entfernen eines Knotens entweder der mehrfach genutzte Knoten mitsamt aller seiner Verweisknoten aus dem Szenengraph entfernt (`NodeDelete` und `NodeReplace`, siehe Tabelle 3.2), oder es wird nur genau die im Kommando beschriebene Instanz entfernt, während die restlichen Knoten ihre Gültigkeit behalten (`IdxValueDelete` und `IdxValueReplace`).

Wenn der gesamte Szenengraph in einen Bitstrom kodiert wird, werden ausgehend vom Wurzelknoten nach dem Schema der Tiefensuche die einzelnen Knoten und deren Felder kodiert. Für die kodierte Abfolge der Knoten im Bitstrom sollte unbedingt gelten, dass ein Verweisknoten erst nach der Kodierung des mehrfach verwendeten Knotens auftreten sollten, da der Inhalt des mehrfach verwendeten Knotens großen Einfluss auf den Kodierungsprozess haben kann. Siehe hierzu auch Anhang A.

3.2.3 In dieser Arbeit verwendete Knotentypen

Im Rahmen dieser Arbeit wird nur eine kleine Anzahl der definierten Knotentypen verwendet. Diese werden im Folgenden vorgestellt. Um eine übersichtliche Darstellung zu erreichen, werden nicht immer alle Felder der Knoten hier abgedruckt. Eine Auflistung sämtlicher Datenfelder befindet sich in Anhang C.1.

Group

`MFNode children` Das einzige für diesen Knoten definierte Feld `children` vom Typ `MFNode` ermöglicht eine Gruppierung von Kindknoten unterhalb dieses Knotens und damit den Aufbau der Baumstruktur. Weiterhin wird der `Group` Knoten an dieser Stelle erwähnt, weil er einer der wenigen Knoten ist, der im Wurzelpunkt des Szenengraphen auftreten darf.

Transform

`MFNode children`
`SFVec3f translation`
`SFRotation rotation`

Das `children` Feld hat hier die gleiche Bedeutung wie beim `Group` Knoten. Die Felder `translation` und `rotation` definieren eine Koordinatensystemtransformation, die auf alle Knoten in `children` angewendet wird. Der `Transform` Knoten wird dazu verwendet, um seine Kindknoten an eine bekannte Position in der virtuellen Szene zu verschieben.

Shape

`SFNode appearance`
`SFNode geometry`

Ein `Shape`-Knoten definiert ein virtuelles Objekt, das gezeichnet werden soll. Im Feld `appearance` werden Informationen über die Materialeigenschaften der Oberfläche des Objektes (Farbe, Reflexionsparameter, Texturen) abgelegt. Das `geometry` Feld verweist auf einen Knoten, der die Geometrie des Objektes beschreibt.

IndexedFaceSet

`SFNode coord`
`MFInt32 coordIndex`

Dieser Knoten (`IFS`) ist ein Geometrieknoten und beschreibt eine beliebige Menge von Polygonen. Er kann in das in das `geometry` Feld eines `Shape` Knoten eingetragen werden. Für die Polygone in diesem Knoten werden nur drei Annahmen gemacht:

1. Jedes Polygon hat mindestens 3 Eckpunkte.
2. Das Polygon ist plan.
3. Die Kanten des Polygons dürfen sich nicht schneiden.

Weder geschlossene, noch zusammenhängende Polygonnetze werden gefordert. Die 3D-Eckpunkte aller Polygone sind in einem `Coordinate` Knoten abgelegt, der im `coord` Feld zu finden ist. Die Kanten aller Polygone werden als Folge von Indizes im Feld `coordIndex` gespeichert. Die Indizes beziehen sich auf die Liste der 3D-Eckpunkte im `coord` Feld. Zwischen die Indexfolgen für jedes einzelne Polygon wird als Trennmarkierung der besondere Indexwert -1 gesetzt. Wegen der indizierten Referenzierung der 3D-Eckpunkten müssen diese nicht mehrfach gespeichert werden. Allerdings werden alle Indizes, also auch die Trennmarkierung, standardmäßig mit 32 Bit im Bitstrom kodiert. Das `CoordIndex` Verfahren (siehe Abschnitt 3.3.2) kann helfen, diese oftmals viel zu hohe Bitanzahl auf das notwendige Minimum zu senken.

Coordinate

`MFVec3f point`

`Coordinate` Knoten haben mit `point` nur ein einziges Feld, in dem eine Liste von 3D-Punkten abgelegt werden kann. Bei `IFS` Knoten entsprechen diese Vektoren den 3D-Eckpunkten der Polygone. Da 3D-Punkte immer in einem separaten `Coordinate` Knoten im Szenengraphen abgelegt sind, können sie über Verweisknoten von mehreren `IFS` Knoten wiederverwendet werden. Eine Vorgehensweise dazu wird in Kapitel 4.5 beschrieben.

QuantizationParameter

```
SFBool isLocal
SFBool useEfficientCoding
SFBool position3DQuant
SFVec3f position3DMin
SFVec3f position3DMax
SFInt32 position3DNbBits
...
```

Mit Hilfe dieses Knotens (`QP`) kann die Kodierung der nachfolgenden Knoten im Baum beeinflusst werden. Ist der Wert des Feldes `isLocal` wahr, wird nur der nächste Knoten (inklusive dessen Kindern) auf gleicher Baumebene beeinflusst. Ansonsten werden alle folgenden Knoten auf der gleichen Baumebene beeinflusst. Dies ist in Abbildung 3.2 dargestellt, die Knoten werden hier von links nach rechts nach dem Prinzip der Tiefensuche der Reihe nach kodiert. Die Parameter aller `QP` Knoten werden in einem Stapelspeicher verwaltet. Der Stapel wächst für jeden während der Kodierung neu hinzugekommenen `QP` Knoten und schrumpft, sobald der Algorithmus die Baumebene des `QP` Knotens zur Wurzel hin verlässt. Für die Kodierung gelten jeweils die Parameter des obersten `QP` Knoten im Stapel. Der `QP` Knoten hat sehr viele Felder, von denen einige hier angedeutet sind. In Kapitel 3.3.2 folgt eine ausführlichere Erläuterung.

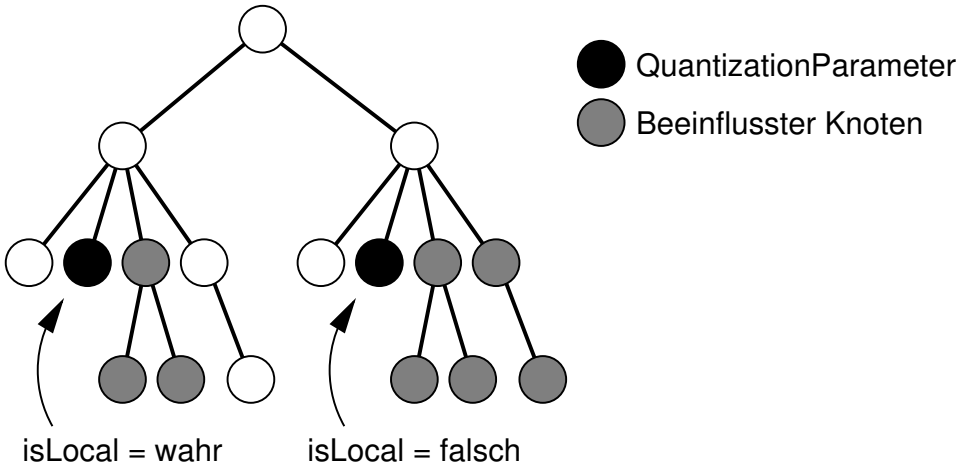


Bild 3.2: Wirkungsbereich eines `QuantizationParameter` Knotens

Inline

`MFUrl url` Ein `Inline` Knoten bettet einen weiteren Szenengraphen in den Szenengraphen des `Inline` Knotens ein. Im Feld `url` wird beschrieben, von woher die Daten des anderen Szenengraphen angefordert werden können. Für einen Betrachter der Szene ist die Existenz eines `Inline` Knoten nicht ersichtlich. Ein Visualisierungsalgorithmus wird im Fall eines `Inline` Knoten die Knotenhierarchie des referenzierten Szenengraphen abarbeiten, bevor er mit der Hierarchie des ursprünglichen Szenengraphen fortfährt. Die Gültigkeit von Knoten-IDs ist jedoch auf einen Szenengraphen beschränkt. Es ist deshalb nicht möglich, Knoten über verschiedene Szenengraphen hinweg mehrfach zu verwenden. `Inline` Knoten werden vor allem in Kapitel 5 verwendet.

3.2.4 Benutzerinteraktion

Um eine Interaktion des Betrachters mit den präsentierten Objekten zu ermöglichen, wurde eine ganze Reihe von Knoten definiert, die bei bestimmten Ereignissen Signale erzeugen. Solche Ereignisse können z. B. gedrückte Tasten auf der Tastatur, oder das Erreichen einer bestimmten Position der Maus am Bildschirm sein.

Ein erzeugtes Signals wird über sog. ROUTEs vom Quellknoten zu einem oder mehreren Zielknoten geführt. Die Zielknoten definieren die unterschiedlichsten Aktionen wenn ein Signal eintrifft. Am mächtigsten sind in diesem Zusammenhang PROTO Knoten, für die über JavaScript praktisch beliebig komplexe Abläufe angestoßen werden können. In ähnlicher Weise wurden mit Hilfe dieser Mechanismen interaktive Spiele realisiert [88, 84].

Auf eine präzisere Erläuterung dieser Möglichkeiten wird hier jedoch verzichtet, da sie für die weitere Arbeit nicht relevant sind. Eine mögliche Anwendung für Telepräsenz-Szenarien kann eine Menüstruktur sein, die direkt in die dreidimensionale Ansicht der entfernten Umgebung integriert ist. Mit Hilfe dieses Menüs könnte der Operator gezielt Aktionen anstoßen, z. B. um teilautonome Abläufe auszuführen.

3.2.5 Beispiel eines BIFS-Szenengraphen

Abbildung 3.3 zeigt beispielhaft, wie aus den oben erwähnten Knoten ein gültiger BIFS Szenengraph gebildet werden kann. Ganz oben im Baum steht ein `Group` Knoten, der als einzigen Knoten in seinem `children` Feld einen `Transform` Knoten hat. In diesem `Transform` Knoten wird die Position der folgenden virtuellen Objekte in ihrer Welt festgelegt. Die Liste im `children` Feld des `Transform` Knoten wird von einem `QuantizationParameter` Knoten angeführt. In dessen Felder gespeicherte Quantisierungsparameter gelten nur für den linken `Shape` Knoten, da das `isLocal` Feld auf `wahr` gesetzt ist. Der rechte `Shape` Knoten ist nicht davon betroffen. In beiden `Shape` Knoten werden keine Materialeigenschaften für die virtuellen Objekte angegeben, dafür aber wird Geometrie in Form von `IndexedFaceSet` Knoten definiert. Jeder `IFS` Knoten hat andere

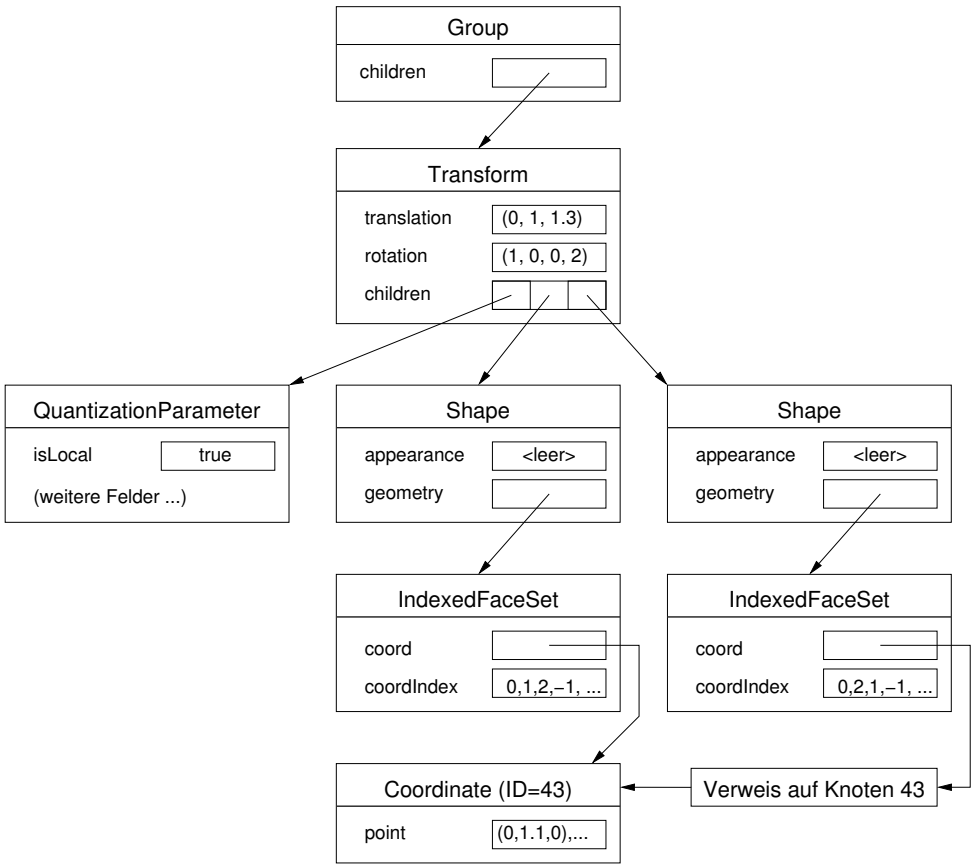


Bild 3.3: Beispielbaum eines BIFS-Szenengraphen

Werte in seinem `coordIndex` Feld, allerdings benützen beide `IFS` Knoten einen gemeinsamen `Coordinate` Knoten. Dieser `Coordinate` Knoten hat die Knoten-ID 43 bekommen. Der erste `IFS` Knoten enthält den eigentlichen `Coordinate` Knoten direkt, der zweite `IFS` Knoten benützt einen Verweisknoten auf den `Coordinate` Knoten.

3.2.6 Binäre Kodierung – BIFS-Update Kommandos

Zur Übertragung des gesamten Szenengraphen und nachfolgender Änderungen wurden mehrere *BIFS-Update* Kommandos definiert. Ein oder mehrere solcher Kommandos werden in ihrer binären Kodierung zu einem *CommandFrame* zusammengefasst. Ein CommandFrame entspricht exakt einer Access Unit, welche vom MPEG-4 System verarbeitet werden kann. CommandFrames sind jedoch nicht die einzigen AUs, die für Szenendaten übertragen werden können.

Unter dem Begriff BIFS-Anim definiert der Standard, wie einzelne Felder von referenzierbaren Knoten mit einer festgelegten Rate verändert werden können. BIFS-Anim AUs heißen *AnimationFrames* und werden parallel zu BIFS-Update AUs in separaten Elementary Streams verschickt. Insbesondere das in Abschnitt 3.3.3 beschriebene Predictive

MFField Verfahren wird hier verwendet, um die meist geringen Änderungen zwischen zwei BIFS-Anim AUs effizient zu kodieren. Eine mögliche Anwendung für Telepräsenz-Szenarien wäre die kontinuierliche Übertragung von Gelenkwinkel der Teleoperatoren. BIFS-Anim wurde in dieser Arbeit nicht weiterverfolgt, da im Vergleich zur rekonstruierten Teleoperatorumgebung die Anzahl der Bits zur Kodierung der Gelenkwinkel eines Teleoperators sehr gering ist. In Kapitel 5.2.5 folgt ein Beispiel mit genauen Angaben zu Kodierungsgrößen von Gelenkwinkel und Raumgeometrie.

In Tabelle 3.2 sind alle bisher definierten BIFS-Update Kommandos zusammen mit einer kurzen Erläuterung aufgeführt. Für jedes dieser Kommandos ist spezifiziert, welche

Name	Beschreibung
`NodeInsert`	Einfügen eines neuen Knotens
`IdxValueInsert`	Einfügen eines zusätzlichen Wertes in ein MF-Feld
`RouteInsert`	Einfügen einer neuen ROUTE in den Szenengraphen
`NodeDelete`	Löschen eines Knotens
`IdxValueDelete`	Löschen eines einzelnen Wertes in einem MF-Feld
`RouteDelete`	Löschen einer ROUTE im Szenengraphen
`NodeReplace`	Ersetzt einen bereits existierenden Knoten
`FieldReplace`	Ersetzt den Wert eines SF-Feldes
`IdxValueReplace`	Ersetzt einen Wert eines MF-Feldes
`RouteReplace`	Ersetzt eine bereits existierende ROUTE
`SceneReplace`	Ersetzt die gesamte Szene
`ProtoListInsert`	Neue PROTOs einfügen
`ProtoListDelete`	PROTOs entfernen
`ProtoDeleteAll`	Alle PROTOs entfernen
`MultipleFieldReplace`	Fasst mehrere `FieldReplace` für einen Knoten zusammen
`MultipleIndexedFieldReplace`	Fasst mehrere `IdxValueReplace` für ein MF-Feld zusammen
`GlobalQuantizationConfiguration`	Legt globale Quantisierungsparameter fest
`NodeDeletionEx`	Ähnlich wie `NodeDelete`, jedoch mit spezieller Wirkung für `OrderedGroup` Knoten

Tabelle 3.2: Die verfügbaren BIFS-Update Kommandos

Informationen binär kodiert werden. Zum Beispiel wird im Fall eines `SceneReplace` Kommandos der gesamte neue Szenengraph binär kodiert.

Als einziges dieser Kommandos kann `SceneReplace` vom Empfänger ohne weiteres Vorwissen ausgeführt werden. Es ist somit auch das einzige Kommando, ab dem ein neuer Empfänger an einer bereits laufenden Datenübertragung von BIFS-Update Kommandos teilnehmen kann. Vor der Ausführung des ersten `SceneReplace` Kommandos hat ein neuer

Empfänger für andere Access Units keine sinnvolle Verwendungsmöglichkeit. Eine ähnliche Situation entsteht direkt nach einem detektierten Übertragungs- oder Dekodierfehler. Erst nach dem nächsten `SceneReplace` ist wieder sicher gestellt, dass die Dekodierung von folgenden AUs gelingt.

Das `GlobalQuantizationConfiguration` Kommando ist erst später im Verlauf der Entwicklung des Standards hinzugekommen. Es setzt eine Beschränkung der normalen `QP` Knoten außer Kraft. Dadurch wird die Verwendung der Kompressionswerkzeuge einerseits vereinfacht. Andererseits jedoch kann ein neuer Empfänger sich nicht mehr darauf verlassen, dass er ein `SceneReplace` Kommando jederzeit dekodieren und ausführen kann. Aus diesem Grund wurde dieses Kommando hier nicht verwendet. Die oben genannte Beschränkung kann auch mit anderen Mitteln umgangen werden.

Einige Kommandos, z. B. `NodeDelete`, benötigen eine Referenz auf den manipulierten Knoten, wofür dessen Knoten-ID verwendet wird. Deshalb müssen alle Knoten, die im Nachhinein über BIFS-Update Kommandos verändert werden sollen, eine solche Knoten-ID zugewiesen bekommen werden. Weiterhin benötigen einige Kommandos, z. B. `IdxValueInsert`, eine sog. inID. Mit dieser Nummer wird das zu manipulierende Feld im Knoten referenziert. Die Vergabe dieser Nummern ist durch den Standard exakt festgelegt und nicht alle Felder jedes Knotens besitzen eine solche Nummer. Eine Auswirkung davon ist, dass z. B. das `coordIndex` Feld eines `IFS` Knotens nicht nachträglich verändert werden kann. In dieser Situation kann nur der ganze Knoten durch ein `NodeReplace` Kommando ersetzt werden.

3.3 BIFS Kompressionswerkzeuge

Im Standard sind einige Werkzeuge für die komprimierte Kodierung von Geometriedaten im Bitstrom definiert. Von diesen Werkzeugen wird eine sehr weite Spanne an Komplexität und Rechenzeitaufwand abgedeckt. Die folgenden Abschnitte beschreiben die in dieser Arbeit verwendeten Werkzeuge.

Zentrales Element für den Einsatz dieser Werkzeuge sind die `QuantizationParameter` Knoten. Sie enthalten nur Felder, die die Wirkung der Werkzeuge beeinflussen, jedoch keine eigentlichen Geometriedaten.

Es gibt ein paar Regeln für die Gültigkeit der `QP` Parameter:

1. Je nachdem, ob im `QP` Knoten das `isLocal` Feld gesetzt ist, oder nicht, erstreckt sich die Wirkung des `QP` Knotens nur über den folgenden Knoten, oder über alle folgenden Knoten auf der gleichen Ebene (siehe auch Abbildung 3.2).
2. Wenn ein Knoten kodiert wird und Regel 1 zutrifft, werden die Kompressionswerkzeuge jedoch nur dann angewendet, wenn der `QP` Knoten im gleichen BIFS-Update Kommando kodiert wurde. Wird z. B. nur ein `IFS` Knoten durch ein `NodeReplace` Kommando ersetzt, wird kein einziges Kompressionswerkzeug eingesetzt. Erst wenn ein `Group` Knoten über `NodeReplace` ersetzt wird, in den sowohl der `QP` Knoten, als

auch der `IFS` Knoten als Kinder eingetragen sind, werden die Kompressionswerkzeuge angewendet. Mit dem `GlobalQuantizationConfiguration` Kommando kann diese Regel umgangen werden.

3. Ist zur Kodierung eines Knoten mehr als ein `QP` Knoten nach den obigen Regeln aktiv, beschreibt der Standard eine Stapelspeicherung von `QP` Parametern. D. h. der oberste `QP` Knoten im Stapelspeicher definiert die gültigen Parameter.

3.3.1 EfficientFloat

Die generische Methode, um eine gebrochen rationale Zahl im Bitstrom abzulegen besteht darin, ihre binäre Repräsentation nach IEEE 754-1985 [39] ohne weitere Konvertierung in 32 Bit abzuspeichern. Diese Repräsentation umfasst 1 Vorzeichenbit, 8 Bit für die Kodierung eines vorzeichenbehafteten Exponenten und 23 Bit für die Mantisse. Sowohl Exponent als auch Mantisse werden ohne führende 1 abgelegt (siehe Abbildung 3.4).

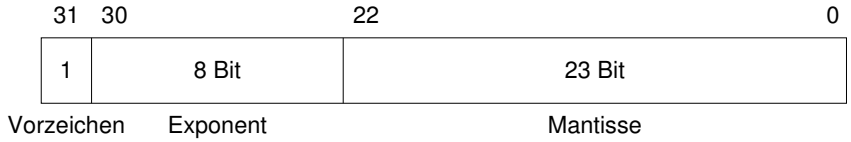


Bild 3.4: IEEE 754-1985 32 Bit Repräsentation

BIFS definiert eine alternative Kodierung für gebrochen rationale Zahlen. Diese trennt ebenso wie IEEE 754 nach Exponent, Mantisse und Vorzeichenbit, kodiert jedoch Exponent und Mantisse mit variabler Bitlänge. In Abbildung 3.5 ist die Abfolge der Bits veranschaulicht. Zuerst wird die Anzahl der Bits für Mantisse und deren Vorzeichen (b_m)

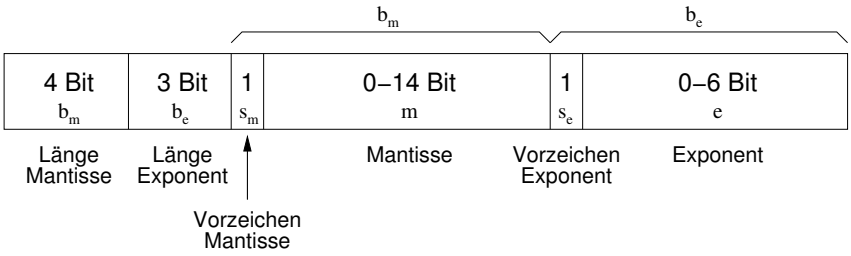


Bild 3.5: BIFS EfficientFloat Repräsentation

in 4 Bit kodiert. Ist b_m gleich 0, hat die Zahl den Wert 0.0 und keine weiteren Bits werden kodiert. Ansonsten wird in 3 Bit die Länge von Exponent und dessen Vorzeichen kodiert (b_e). Das folgende Bit ergibt das Vorzeichen der Mantisse (s_m), ein Bitwert von 0 korrespondiert mit einem positiven Vorzeichen. Nach der kodierten Mantisse (m) folgt eine ähnliche Bitfolge für den Exponent (s_e, e), aber nur wenn die oben gelesene Exponentlänge b_e größer als 0 ist. Die kodierte Zahl lässt sich dann mit folgender Formel ausrechnen [43]:

$$v = (1 - 2 \cdot s_m) \cdot (2^{b_m - 1} + m) \cdot 2^{(1-2 \cdot s_e) \cdot (2^{b_e - 1} + e)} \tag{3.1}$$

Die EfficientFloat Kodierung gilt für alle gebrochen rationalen Zahlen, die im Bitstrom kodiert werden. Betroffen sind also *alle* Felder vom Typ `SFFloat`, `SFVec2f`, `SFVec3f`, `SFVec4f`, `SFColor` und `SFRotation`, sowie deren MF-Äquivalente. Folglich können vor allem bei der Kodierung von 3D-Punkten (`SFVec3f`) Einsparungen in der Anzahl der geschriebenen Bits erzielt werden. Da allerdings auch Rotationswinkel, wie z. B. das `rotation` Feld im `Transform` Knoten, betroffen sind, kann bei allzu großzügigem Gebrauch dieses Werkzeugs auch die Lage von virtuellen Objekten in der Szene deutlich beeinflusst werden.

Das EfficientFloat Werkzeug wird aktiviert, wenn im `QP` Knoten das `useEfficientCoding` Feld auf wahr gesetzt wird. Wird zusätzlich auch Quantisierung (siehe nächster Abschnitt) aktiviert, erhält nach dem Standard die Quantisierung Vorrang.

Rechnerisch ergibt sich eine maximale Kompressionsrate von $\frac{32}{4} = 8$, wenn der Wert 0 kodiert wird. Da im ungünstigsten Fall 29 Bit zur Kodierung verwendet werden, ergibt sich eine minimale Kompressionsrate von $\frac{32}{29} \approx 1.1$. Dieser Fall tritt dann auf, wenn der Enkoder entscheidet, die Mantisse mit maximaler Bitanzahl zu kodieren und der Exponent der Zahl im Betrag einen sehr hohen Wert hat. Wenn die Koordinaten der 3D-Punkte in Meter rekonstruiert werden, sind allerdings eher Werte für die Exponenten zu erwarten, die z. B. mit 3 Bit kodiert werden können. In diesem Fall kann eine Kompressionsrate zwischen $\frac{32}{12} \approx 2.67$ und $\frac{32}{26} \approx 1.23$ erwartet werden. Die Entscheidung über die Anzahl der genutzten Bits für die Mantisse bestimmt schlussendlich über die erzielbare Kompressionsrate.

3.3.2 Quantisierung und das `CoordIndex` Verfahren

Mit dem Begriff Quantisierung wird im MPEG-4 Standard ein Kompressionswerkzeug bezeichnet, welches lineare Quantisierung im allgemein gebräuchlichen Sinn zur Kodierung verwendet. Wie bei linearer Quantisierung üblich, werden gebrochen rationalen Zahlen auf eine endliche Menge von ganzen Zahlen abgebildet. Dazu muss der gewünschte Wertebereich und die Anzahl der verfügbaren ganzen Zahlen bekannt sein. Die ganzen Zahlen können mit der minimal notwendigen Anzahl von Bits kodiert werden. Da diese Methodik auch für `SFInt32` und `MFInt32` Felder funktioniert, definiert MPEG-4 als Teil seiner Quantisierung auch zwei verlustlose Verfahren für diese Felder. Für die anderen Felder ist Quantisierung ein verlustbehaftetes Verfahren.

Zusätzlich ist es möglich, bei einigen wenigen Datentypen nur einen Teil der Daten bzw. eine alternative Repräsentation zu kodieren. So reicht es bei Normalenvektoren, wenn nur zwei Komponenten kodiert werden. Die dritte Komponente lässt sich mit einfachen Mitteln beim Empfänger rekonstruieren; eine Normalisierung der Normalenvektoren auf die Länge 1 wird hier vorausgesetzt. Für Rotationen wird für die Quantisierung eine Darstellung als Quaternion benutzt. Auch hier ist nur eine Kodierung von drei der vier Komponenten notwendig.

Im Standard wird für jedes Feld eines jeden Knotens definiert, mit welchem Verfahren es quantisiert wird. Die möglichen Verfahren sind in Tabelle 3.3 aufgeführt. Für die mei-

Name	Quantisierungsmethode
`none`	Werte in diesen Feldern werden nie quantisiert
`Position3D`	lineare Quantisierung aller drei Vektorkomponenten mit vorgegebener Bitanzahl innerhalb angegebener Wertgrenzen
`Position2D`	siehe `Position3D`, jedoch für zwei Dimensionen
`DrawingOrder`	siehe `Position3D`, jedoch für eine Dimension
`Color`	siehe `DrawingOrder`
`TextureCoordinate`	siehe `DrawingOrder`
`Angle`	siehe `DrawingOrder`
`Scale`	siehe `DrawingOrder`
`InterpolatorKeys`	siehe `DrawingOrder`
`Normals`	lineare Quantisierung mit vorgegebener Bitanzahl im Intervall [0..1] der ersten zwei Vektorkomponenten nach Normalisierung auf die Länge 1
`Rotations`	lineare Quantisierung mit vorgegebener Bitanzahl im Intervall [0..1] der ersten drei Quaternionenkomponenten
`ObjectSize3D`	siehe `Position3D`, identisches Intervall für alle Dimensionen
`ObjectSize2D`	siehe `Position2D`, identisches Intervall für alle Dimensionen
`LinearScalar`	Kodierung von ganzen Zahlen mit minimaler Bitanzahl
`CoordIndex`	Spezialfall von `LinearScalar`
`SFVec4f`	siehe `Position3D`, vierte Vektorkomponente wie `Scale`

Tabelle 3.3: In MPEG-4 definierte Quantisierungsverfahren für Felder

sten Quantisierungsverfahren kann entschieden werden, ob es während der Kodierung eines passenden Feldes angewendet werden soll, oder nicht. Ausnahme dazu sind die Verfahren `LinearScalar` und `CoordIndex`. Allein die Präsenz eines aktiven `QP` Knoten im Stapelspeicher des Kodierers reicht hier zur Aktivierung des entsprechenden Verfahrens aus. Bei den anderen Verfahren muss zur Aktivierung im entsprechenden `QP` Knoten ein Feld vom Typ `SFBool` auf `wahr` gesetzt werden. Im Folgenden wird näher auf einige der Quantisierungsverfahren eingegangen.

`Position3D`

Das `Position3D` Verfahren ist vor allem für die Kodierung von 3D-Punkten interessant, wie sie z. B. in einem `Coordinate` Knoten als Liste abgelegt sind. `Position3D` quantisiert jede der drei Komponenten der Punkte einzeln mit einer vorgegebenen Anzahl von Bits nb. Jeder Komponentenwert v wird dazu in eine ganze Zahl v_q zwischen $[0..2^{nb}-1]$ umgewandelt [41]:

$$v_q = \lfloor \frac{v - V_{min}}{V_{max} - V_{min}} (2^{nb} - 1) + 0.5 \rfloor \tag{3.2}$$

V_{max} und V_{min} sind dabei die pro Koordinatenachse festgelegten maximalen und minimalen Wertgrenzen. Der inverse Prozess rekonstruiert die Komponenten über folgende

Formel:

$$v = V_{min} + v_q \frac{V_{max} - V_{min}}{2^{nb} - 1}, nb \geq 1 \tag{3.3}$$

Sämtliche 3D-Punkte sollten zur Quantisierung innerhalb der vorgegebenen Wertgrenzen liegen, ansonsten werden entsprechend die Maximalwerte kodiert (0 bzw. $2^{nb} - 1$). Die Wertgrenzen V_{min}, V_{max} und die Anzahl der verwendeten Bits nb werden durch die Felder `position3DMin`, `position3DMax` und `position3DNbBits` im `QP` Knoten definiert.

`Normals/Rotations`

Die Verfahren für Normalenvektoren und Rotationen sind einander ähnlich. Für beide ist ein Vorverarbeitungsschritt bei der Enkodierung definiert. Normalenvektoren werden dabei auf die Länge 1 normalisiert, Rotationen werden in eine Quaternionendarstellung umgerechnet. Von den errechneten Werten werden die betragsmäßig größten Komponenten im Bitstrom mit der gleichen Vorschrift wie bei `Position3D` kodiert. Der jeweils übrig gebliebene letzte Wert kann beim Empfänger rekonstruiert werden. Da die minimalen und maximalen Wertgrenzen bekannt sind, bleibt als einziger wählbarer Parameter die Bitanzahl nb. Diese wird im `QP` Knoten im Feld `normalNbBits` für beide Verfahren zusammen festgelegt.

`LinearScalar/CoordIndex`

Die generische Methode, um `SFInt32` oder `MFInt32` Werte zu kodieren benötigt 32 Bit. Wird im entsprechenden Feld nicht der ganze 32 Bit Wertumfang genutzt, müssen nicht alle Bits kodiert werden. Zu diesem Zweck wurden in MPEG-4 das `LinearScalar` Verfahren, sowie sein Spezialfall `CoordIndex` definiert.

Für `LinearScalar` muss bekannt sein, welcher Wertebereich $[V_{min} \ldots V_{max}]$ für ein Feld zulässig ist. Das Verfahren subtrahiert V_{min} von jedem Feldwert und kodiert das Ergebnis mit $nb = \lceil log_2(V_{max} - V_{min} + 1) \rceil$ Bits. Im Standard sind für jedes `LinearScalar` Feld V_{min} und nb festgelegt.

Das `CoordIndex` Verfahren kommt nur bei den Feldern zum Einsatz, in denen Indizes in andere MF-Felder gespeichert werden. Das `coordIndex` Feld eines `IFS` Knotens ist ein typisches Beispiel. Als Trennmarkierung wird der Wert -1 verwendet. Da ansonsten nur nichtnegative Indizes vorkommen, ist der Wert V_{min} stets -1. Die obere Grenze V_{max} ergibt sich aus N_r, der Anzahl der Elemente im referenzierten MF-Feld.

$$V_{max} = N_r - 1$$

Die Anzahl der benötigten Bits pro Wert beträgt dann wie bei `LinearScalar`:

$$nb = \lceil log_2(V_{max} - V_{min} + 1) \rceil = \lceil log_2(N_r + 1) \rceil \tag{3.4}$$

Der Wert *nb* wird nicht im Bitstrom kodiert. Anstatt dessen merkt sich der Dekoder die Anzahl der Elemente N_r der zuletzt kodierten 3D- oder 2D-Punkteliste. Die Kodierung der Indexwerte erfolgt wie bei `LinearScalar`.

3.3.3 Predictive MFField

Als weiteres Werkzeug zur Komprimierung von MF-Feldern kann *Predictive MFField* (PMF) eingesetzt werden. Wie der Name bereits ausdrückt, wird hier eine einfache Form der Prädiktion eingesetzt. Mit wenigen Ausnahmen erfolgt die Prädiktion durch die Bildung der Differenz vom vorhergehenden Wert im MF-Feld zum aktuellen Wert. Für die Prädiktionsvorschrift von Normalenvektoren und Rotationen sei hier auf den Standard [43] verwiesen. Die Ergebnisse der Differenzen werden über einen arithmetischen Kodierer mit adaptivem Modell im Bitstrom abgelegt.

Adaptive arithmetische Kodierung

Die Arithmetische Kodierung steht in direkter Konkurrenz zur Huffman Kodierung [97]. Ziel ist die Übertragung einer Folge von Symbolen mit möglichst wenig Bits. Die möglichen Symbole sind sowohl dem Enkoder, als auch dem Dekoder bekannt. Beide Kodieralgorithmen verwenden Modelle der Symbolhäufigkeiten, um weniger wahrscheinliche Symbole mit mehr Bits als sehr wahrscheinliche Symbole zu kodieren. Eine untere Grenze für die notwendige Anzahl von Bits wurde von Shannon durch den Entropiegehalt der Symbolfolge bewiesen [69]. In [97] stellt Witten einige Vorteile der arithmetischen Kodierung gegenüber der Huffman Kodierung dar. Neben der effizienteren Kodierung in praktischen Anwendungen wird auch geäußert, dass die Arithmetische Kodierung besser mit adaptiven Modellen harmoniert. Anschließend an die Erläuterung des Algorithmus wird in dieser Arbeit auch C-Quellcode veröffentlicht, der mit leichten Abänderungen ebenfalls im MPEG-4 Standard abgedruckt wurde.

Das verwendete Modell hat entscheidenden Einfluss auf die Kodiereffizienz. Für die Huffman Kodierung werden üblicherweise statische Modelle verwendet. Die Häufigkeit jedes Symbols muss dazu a-priori bekannt sein. Das klassische Beispiel hierfür sind Modelle für Textdokumente einer bestimmten Sprache. Jeder Buchstabe entspricht hier einem Symbol, und die durchschnittliche Häufigkeit jedes Buchstaben wurde über eine Analyse vieler Dokumente in der gewählten Sprache ermittelt. Für Predictive MFField werden jedoch adaptive Modelle verwendet, die die Symbolhäufigkeiten während der laufenden Kodierung mit jedem kodierten Symbol anpassen.

Im Kontext von MPEG-4 werden ganze Zahlen als Symbole verwendet. Die Zahl 0 ist dabei das erste Symbol, die Zahl 1 das zweite usw. bis schließlich die größtmögliche Zahl dem letzten bekannten Symbol entspricht. Zu Beginn der Kodierung eines MF-Feldes wird eine Gleichverteilung der Häufigkeit aller bekannten Symbole angenommen. Sowohl Enkoder als auch Dekoder erhöhen nach Kodierung eines Symbols die entsprechende Häufigkeit im Modell. Aus Gründen der Laufzeiteffizienz werden im Modell kumulative Häufigkeiten

verwendet. Je mehr Symbole übertragen werden, desto besser nähert sich die im Modell abgelegte Häufigkeitsverteilung der tatsächlichen Entropie der Symbole an, wodurch effizienter kodiert werden kann. Für eine ausführlichere Beschreibung der adaptiven arithmetischen Kodierung wird hier auf [97] verwiesen.

Kodierung von Predictive MFField

Die Werte eines MF-Feldes können entweder als *Intra-* oder *Predictive*-Werte kodiert werden. In beiden Fällen wird der zu kodierende Wert zuerst quantisiert (siehe Abschnitt 3.3.2). Das Ergebnis der Quantisierung sind immer ganze, nichtnegative Zahlen. Für P-Werte wird anschließend die Prädiktion vom letzten quantisierten Wert zum aktuellen quantisierten Wert gebildet. Wie bereits oben erwähnt, bedeutet die Prädiktion in den meisten Fällen eine Differenzbildung. P-Werte werden, im Gegensatz zu I-Werten, immer durch einen adaptiven arithmetischen Kodieralgorithmus im Bitstrom abgelegt.

Zumindest der erste Wert eines MF-Feldes muss als I-Wert im Bitstrom abgelegt werden. Für die weitere Abfolge von I- und P-Werten sieht der Standard drei verschiedene Schemas vor:

1. IPPPPPP...
   Bei diesem Schema wird nur der erste Wert des Feldes als I-Wert kodiert. Alle folgenden Werte werden ausschließlich durch P-Werte kodiert.

2. IPPIPPIPP...
   Nach einem I-Wert wird eine festgelegte Anzahl n_i von P-Werten kodiert. Danach folgt wieder ein I-Wert und weitere n_i P-Werte, solange bis das gesamte Feld kodiert ist.

3. Wahlfrei
   Die letzte Möglichkeit sieht eine beliebige Abfolge von I- und P-Werten vor. Nach jedem kodierten Wert wird über ein Bit im Strom entschieden, ob ein I- oder ein P-Wert folgt.

Für alle Schemas gilt, dass vor jedem I-Wert neue Parameter für das adaptive Modell kodiert werden können. Dadurch kann in gewissem Grenzen eine Umkonfiguration im Verlauf der Kodierung eines MF-Feldes erfolgen. Dabei ist jedoch zu beachten, dass die bisher gesammelten Häufigkeiten des Modells nach einer Umkonfiguration wieder gleichverteilt sind, und damit die Kodierung der folgenden Werte an Effizienz verliert.

Predictive MFField kann nur in BIFS Elementary Streams ab Version 2 verwendet werden. Erst ab dieser Version ist für die Aktivierung des Verfahrens ein Flag vorgesehen. Ist dieses Flag gesetzt, wird mit einem Bit vor jedem MF-Feld entschieden, ob dieses Feld prädiktiv kodiert wird. Danach werden die notwendigen Parameter kodiert (siehe Tabelle 3.4), um das adaptive Modell zu initialisieren und eines der oben aufgeführten Schemas zu wählen.

Die Modelle werden für 2^{nb} mögliche Symbole initialisiert. Das erste Symbol wird über den Index 0, das letzte Symbol über den Index $2^{nb} - 1$ referenziert. Durch die Bildung

Parameter	Erläuterung
n	Die Anzahl der kodierten Werte[a)].
m	Das verwendete Schema (1, 2 oder 3)[a)].
n_i	Falls m gleich 2 ist, wird mit dieser Zahl festgelegt, in welchen regelmäßigen Abständen I-Werte zwischen die P-Werte geschrieben werden.
nb	Die Anzahl der möglichen Symbole im adaptiven Modell, kodiert als Anzahl von Bits.
p_{min}	Offset der P-Werte zu Symbolindizes.

Tabelle 3.4: Parameter für die Kodierung als Predictive MFField

[a)] Diese Parameter werden nur einmal zu Beginn des MF-Feldes kodiert. Ein Wechsel des Schemas m ist somit während der Kodierung des Feldes nicht möglich.

von Differenzen können negative P-Werte entstehen. Deshalb wird jeder P-Wert vor der Kodierung durch den arithmetischen Kodierer um p_{min} verschoben, um einen gültigen Symbolindex zu erhalten. Ob ein MF-Feld über Predictive MFField kodiert werden kann, hängt somit hauptsächlich davon ab, ob alle während der Kodierung auftretenden Differenzwerte nach der Verschiebung um p_{min} im Intervall $[0...2^{nb} - 1]$ liegen. Der Parameter nb ist durch den Standard auf einen Wert kleiner oder gleich 14 beschränkt, in der Praxis sind jedoch maximal nur 2^{13} Symbole möglich[1)]. Je kleiner nb gewählt werden kann, desto weniger Zeit benötigt der Algorithmus um die Häufigkeitsverteilung im Modell anzupassen.

3.4 Object Descriptor Framework

Wenn ein Empfangsterminal eine MPEG-4 Präsentation wiedergeben soll, muss ein Mechanismus existieren, der Informationen über die zur Verfügung stehenden Elementary Streams bereitstellen kann. Zu diesen Informationen gehören die Kodierungs- und Übertragungsparameter, der Zeitraum in dem der Inhalt des Elementary Streams dargestellt werden soll und wie die Elementary Streams geöffnet werden können. Einen großen Teil dieser Aufgabe übernimmt das *Object Descriptor Framework*.

Jedes Medienobjekt der Präsentation (Szenen-, Video- oder Audiodaten) wird in MPEG-4 mit einem Objektdeskriptor beschrieben. In diesem Objektdeskriptor befinden sich alle notwendigen Angaben, um Zugriff auf die Daten eines Medienobjekts zu bekommen. Die Menge aller Objektdeskriptoren kann sich über die Zeit verändern. Das Sendeterminal kann neue Medienobjekte hinzufügen, die Kodierung für bereits bekannte Medienobjekte ändern, oder Medienobjekte wieder entfernen. Um diese Informationen unabhängig zu den eigentlichen Mediendaten zu übertragen, wurde der *Object Descriptor Stream* (ODS) definiert. Über diesen Elementary Stream werden Access Units verschickt, die ein oder

[1)] siehe hierzu auch Anhang A

mehrere OD-Kommandos enthalten. Einige der möglichen OD-Kommandos sind in Tabelle 3.5 aufgeführt.

Inhalt eines Objektdeskriptors ist, neben einer Nummer zur Identifizierung, eine Liste der verfügbaren Elementary Streams. Diese Liste besteht aus den sog. *ES-Deskriptoren*. Jeder ES-Deskriptor beinhaltet ebenfalls eine Nummer zur Identifikation, sowie die notwendigen Parameter, um auf die Daten des jeweiligen Elementary Stream zuzugreifen. Über jeden Elementary Stream werden Access Units passend zum Typ des Medienobjekts übertragen. Für ein Szenenobjekt übertragen die Elementary Streams also entweder CommandFrames oder AnimationFrames.

Für ein Medienobjekt können mehrere Elementary Streams verfügbar sein. Wird ein Elementary Stream als abhängig von einem anderen Elementary Stream markiert, muss das Empfangsterminal zuerst den unabhängigen Elementary Stream dekodieren, bevor der abhängige Elementary Stream dekodiert werden kann. In Abbildung 3.6 ist die Abhängigkeit in Objektdeskriptor `OD 1` durch einen Pfeil angedeutet. Ein Anwendungsbeispiel für abhängige Elementary Streams sind Videoobjekte mit skalierbarer Bildqualität.

Sind die Elementary Streams nicht voneinander abhängig, bieten sie zwar den gleichen Inhalt, aber in unterschiedlichen Ausprägungen an. Das Anwendungsbeispiel hier sind Audioobjekte (`OD 2`), die aus unterschiedlichen Sprachen wählen können.

Die Elementary Streams der verschiedenen Medienobjekte dürfen nicht im gleichen Objektdeskriptor gemischt angeboten werden. Nur bei Objektdeskriptoren für Szenenobjekte dürfen zusätzlich zu den BIFS Elementary Streams auch ODS angegeben werden (`OD 3`).

Eine weitere Besonderheit sind die Upstreams. Über diese Rückkanäle kann ein Empfangsterminal Daten zurück zum Sendeterminal schicken. Upstreams hängen immer von einem Downstream ab (`OD 4`). Laut Standard gehören sie logisch zum Downstream und übertragen deshalb auch Informationen passend zu denen des Downstreams.

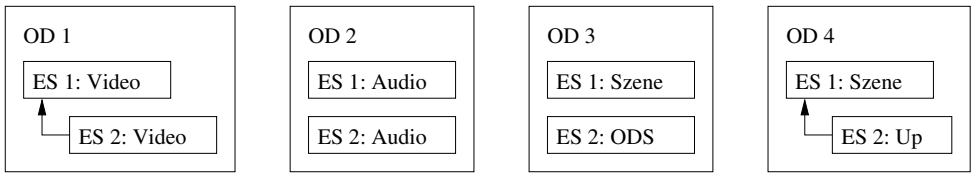


Bild 3.6: Beispiele für gültige Aggregationen von Elementary Streams

Ein Beispiel für eine Präsentation mit mehreren Medienobjekten und Elementary Streams ist in Abbildung 3.7 zu sehen. Der initiale Objektdeskriptor `OD 1` wird bereits während des Verbindungsaufbaus durch die Übertragungsschicht übermittelt. Siehe hierzu auch Abschnitt 3.6. Er beschreibt ein Szenenobjekt, für das ein einzelner BIFS Elementary Stream (`ES 1`) und ein ODS (`ES 2`) gehören. Der ODS überträgt mit Hilfe von OD-Kommandos eine Liste von zusätzlichen Objektdeskriptoren, die zum vollständigen Aufbau der Szenen benötigt werden. In `ES 1` werden BIFS-Update Kommandos zum Aufbau eines Szenengraphen übertragen. Dieser Szenengraph enthält unter anderem zwei sog. Medienknoten.

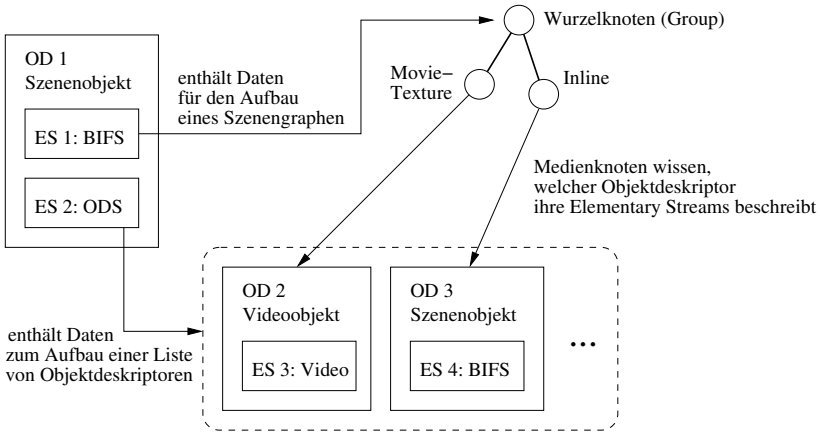


Bild 3.7: Beispiel für die Verwendung von Objektdeskriptoren und ODS anhand einer Präsentation mit zwei Szenen- und einem Videoobjekt

Diese Knoten repräsentieren weitere Medienobjekte im Szenengraphen und wissen, welcher Objektdeskriptor den Zugriff auf die entsprechenden Mediendaten beschreibt. Im Beispiel verweist der `MovieTexture` Knoten auf `OD 2`, der dem Knoten einen Video Elementary Stream `ES 3` anbietet. Dem `Inline` Knoten wird in `OD 3` ein BIFS Elementary Stream `ES 4` zur Verfügung gestellt, über den weitere BIFS-Update Kommandos zum Aufbau des eingebetteten Szenengraphen übertragen werden.

Kommt nun im Laufe der Präsentation ein neues Medienobjekt hinzu, wird beim Sendeterminal ein neuer Objektdeskriptor erzeugt. Über den ODS kann der neue Objektdeskriptor zum Empfangsterminal verschickt werden. Dieses reagiert entsprechend, wenn ein Medienknoten auf diesen Objektdeskriptor wartet. Im Standard sind bisher insgesamt acht OD-Kommandos definiert, die über einen ODS verschickt werden können. In Tabelle 3.5 werden vier dieser OD-Kommandos kurz erläutert.

Kommando	Erläuterung
`ObjectDescriptorUpdate`	Ein bestehender Objektdeskriptor wird verändert, oder ein neuer hinzugefügt. Bei einer Veränderung müssen alle bisher geöffneten Elementary Streams dieses Objektdeskriptors geschlossen, und bei Bedarf wieder geöffnet werden.
`ObjectDescriptorRemove`	Ein Objektdeskriptor wird vollständig entfernt.
`ES_DescrUpdate`	Ein neuer ES-Deskriptor wird zu einem Objektdeskriptor hinzugefügt. Eine Veränderung bereits eingefügter ES-Deskriptoren ist nicht möglich.
`ES_DescrRemove`	Ein ES-Deskriptor wird entfernt.

Tabelle 3.5: Eine Auswahl von OD-Kommandos, die über einen ODS verschickt werden können

3.4.1 Verwendung von URLs

Wie auf die Daten zugegriffen werden kann, wird mit Hilfe von URLs (Uniform Resource Locator) beschrieben. URLs werden im Kontext von MPEG-4 in vier Fällen verwendet:

1. Zugriff auf eine Präsentation, bereitgestellt durch ein Sendeterminal
2. Zugriff auf einen Objektdeskriptor ausserhalb der aktuellen Präsentation
3. Zugriff auf einen Elementary Stream ausserhalb der aktuellen Präsentation
4. Verweis in Medienknoten

Der erste Fall tritt obligatorisch immer auf, wenn ein Empfangsterminal eine Präsentation öffnen will. Die nächsten beiden Einsatzmöglichkeiten sind optional. Sie treten immer dann auf, wenn Daten von einem weiteren, zusätzlichen Sendeterminal zur Verfügung gestellt werden. Der letzte Fall gilt nur für Medienknoten. Eine dort angegebene URL soll auf einen, in der aktuelle Präsentation enthaltenen Objektdeskriptor verweisen.

Sowohl in Objektdeskriptoren, als auch in ES-Deskriptoren können anstatt den gewöhnlichen Deskriptorinformationen auch URLs eingetragen werden (Fälle 2 und 3). Entscheidet sich ein Empfangsterminal dazu, dass der Deskriptor für die Darstellung der Präsentation relevant ist, muss die URL gesondert gehandhabt werden:

- **Objektdeskriptor-URLs:** Anstatt der Liste von ES-Deskriptoren ist nur eine URL im Objektdeskriptor verfügbar. Für das Sendeterminal bedeutet das, dass eine weitere Präsentation unter der angegebenen URL geöffnet werden muss um Zugriff auf die Daten zu erhalten.
- **ES-Deskriptor-URLs:** Hier stehen zwar weiterhin alle für die Dekodierung des Elementary Stream relevanten Parameter im ES-Deskriptor, jedoch verweist die URL auf eine andere Datenquelle als die aktuelle Präsentation. Diese Datenquelle liefert auf Anfrage direkt die Daten des Elementary Stream in Form von SL-Paketen.

In Kapitel 5.2.4 wird das Konzept der Objektdeskriptor-URLs aufgegriffen und für Telepräsenz-Szenarien angewendet.

3.4.2 BIFSConfig

Bisher wurde über den Inhalt der ES-Deskriptoren nur gesagt, dass sie die notwendigen Informationen enthalten, um die SL-Pakete eines Elementary Streams zu dekodieren. Für BIFS Elementary Streams werden diese Parameter in einer *BIFSConfig* Datenstruktur abgelegt, die Teil des ES-Deskriptors ist. Tabelle 3.6 zeigt die relevanten Einträge.

Über den Parameter `nodeIDbits` wird festgelegt, mit wie vielen Bits die Knoten-IDs im Bitstrom kodiert werden. Dadurch ist automatisch festgelegt, wie viele Knoten in einem Szenengraphen mit einer Knoten-ID identifiziert werden können. Da die meisten

Eintrag	Erläuterung
`nodeIDbits`	Die Bitanzahl [0..31], mit der Knoten-IDs kodiert werden.
`isCommandStream`	Unterscheidung zwischen BIFS-Update und BIFS-Anim.
`animMask`	BIFS-Anim: Legt fest, welche Knoten des Szenengraphen kontinuierlich verändert werden.
`useNames`	Legt fest, ob Knoten zusätzlich zur Knoten-ID auch eine Zeichenkette als Namen zugewiesen bekommen.
`use3DMC`	Unterscheidung, ob `IFS` Knoten mit dem 3D Mesh Compression Werkzeug kodiert werden.
`usePredMFField`	Unterscheidung, ob MF-Felder prädiktiv kodiert werden können.

Tabelle 3.6: Einige Inhalte der BIFSConfig Datenstruktur

BIFS-Kommandos eine Knoten-ID enthalten um festzulegen, welcher Knoten manipuliert werden soll, bestimmt `nodeIDbits` damit die Anzahl der veränderbaren Objekte.

Der Parameter `useNames` dient dem Zweck zu unterscheiden, ob bei der Kodierung für einen Knoten eine alphanumerische Zeichenkette – der Name des Knotens – kodiert werden soll oder nicht. Die Zuweisung eines Namens ist dann sinnvoll, wenn auf ihn über MPEG-J oder ein Skript zugegriffen werden soll. MPEG-J definiert eine Java Schnittstelle und ein Framework, mit dem Java Bytecode empfangen und ausgeführt werden kann, um Einfluss auf die aktuelle Präsentation zu nehmen.

Mittels den Werten `use3DMC` und `usePredMFField` wird global die Verwendung von zwei Kompressionsverfahren gesteuert. Predictive MFField wurde bereits in 3.3.3 vorgestellt. Das 3D Mesh Compression Werkzeug wird in dieser Arbeit nicht betrachtet.

3.5 SyncLayer

Die *SyncLayer* ist für die zeitliche Synchronisation der Medieninhalte zuständig. Es geht dabei zum einen um die zeitgerechte Darstellung der Inhalte eines Elementary Stream, wodurch die korrekte Abspielgeschwindigkeit der übertragenen Inhalte erreicht wird. Zum anderen ist auch die Synchronisation mehrerer Elementary Streams zueinander möglich. Dadurch können zum Beispiel Video- und Audiodaten lippensynchron zueinander präsentiert werden.

Wie bereits am Anfang dieses Kapitels beschrieben, erhält die SyncLayer die Daten eines Elementary Stream in Form von SL-Paketen. In den Headern der SL-Pakete können – neben vielen weiteren Elementen – für jede Access Unit zwei Zeitstempel und eine fortlaufende Nummer eingetragen werden. Die fortlaufende Nummer bietet sich bei unsicheren Übertragungskanälen als Hinweis für verloren gegangene Datenpakete an. Die Zeitstempel drücken die vom Sendeterminal gewünschte Dekodier- und Darstellungszeit aus. Die Dekodierzeit bestimmt, zu welchem Zeitpunkt der Dekoder eine Access Unit zur Bearbeitung erhält, während der andere Zeitstempel den gewünschten Darstellungszeitpunkt

bestimmt. Eine Anwendung für unterschiedliche Dekodier- und Darstellungszeitstempel gibt es bei Videodaten, bei denen eine Dekodierung bidirektional prädizierter Zwischenbilder erst nach der Dekodierung des folgenden Intrabildes möglich ist.

Zur Bewertung der Zeitstempel soll nicht die lokale Uhr des Empfangsterminals verwendet werden, sondern die *Objektzeit*. Diese Objektzeit entspricht einer Uhr des Sendeterminals. In jedem SL-Paket Header kann ein weiterer Zeitstempel (*Object Clock Reference*, OCR) eingefügt werden, den das Sendeterminal auf seine aktuelle, lokale Uhrzeit setzt. Mit Hilfe jedes empfangenen OCR rekonstruiert das Empfangsterminal die Objektzeit. Alternativ kann ein separater Elementary Stream von einem Sendeterminal bereitgestellt werden, in dem ausschließlich SL-Pakete ohne Nutzdaten, jedoch mit OCR Zeitstempel verschickt werden. Auf diesen *Clock Reference Stream* verweisen dann andere Elementary Streams.

Die Entscheidung, ob eine Access Unit vom Sendeterminal auf mehrere SL-Pakete aufgeteilt wird, und welche maximale Bytegröße ein SL-Paket haben darf, hängt vom gewählten Transportmechanismus ab (siehe Kapitel 3.6). Im Header jedes SL-Pakets kann ein Paketzähler eingefügt werden. Auch dieser Zähler bietet sich, ähnlich wie der Access Unit Zähler, als Hinweis für verloren gegangene Daten an. In Anhang C.3 sind weitere Elemente des SL-Paket Headers beschrieben. Die Anzahl der tatsächlich verwendeten Elemente kann sehr flexibel für jeden einzelnen Elementary Stream über die *SLConfigDescriptor* Datenstruktur festgelegt werden. Diese Struktur ist, ebenso wie BIFSConfig, ein Teil des ES-Deskriptors.

3.6 Delivery Multimedia Integration Framework – DMIF

Mit dem *Delivery Multimedia Integration Framework* (DMIF) wird unter dem Namen *DMIF Application Interface* (DAI) eine Programmierschnittstelle beschrieben. Sie deckt in generischer Art alle Aspekte ab, die ein MPEG-4 System benötigt, um die Daten der Elementary Streams zu empfangen, oder zu verschicken. Die Schnittstelle ist dabei unabhängig von der verwendeten Transporttechnologie. Die DMIF Spezifikation ist Teil des MPEG-4 Standards, soll aber auch über MPEG-4 hinaus Verwendung finden (siehe [40], Kapitel 10.1).

In der Spezifikation werden drei allgemeine Zugriffsarten betrachtet:

- **Dateizugriff**
  Die benötigten Daten liegen in einer Datei vor.
- **Interaktive Netzwerktechnologie**
  Interaktive Netzwerktechnologie bezeichnet die Situation, wenn das Sende- mit dem Empfangsterminal über eine duplex-fähige Netzwerkverbindungen Nachrichten austauschen kann.
- **Broadcast Technologie**
  Dieser Fall gleicht dem der interaktiven Netzwerktechnologie, jedoch ist nur eine

simplex Datenübertragung vom Sende- zum Empfangsterminal möglich. Als Beispiel sei hier die Ausstrahlung von Fernsehprogrammen über Satellit genannt.

Vom Konzept existiert für jede Transporttechnologie ein Softwaremodul, das die DMIF Schnittstelle implementiert. Für den Fall der interaktiven Netzwerktechnologie gleichen sich die notwendigen Schritte, um Nachrichten über verschiedene Netzwerkprotokolle auszutauschen. Deshalb wurde noch eine weitere, interne Schnittstelle definiert. Das *DMIF Network Interface* (DNI) vereinfacht die Entwicklung von DMIF-Softwaremodulen für Netzwerkprotokolle.

Wie sich im Verlauf dieser Arbeit herausgestellt hat, sind für Telepräsenz-Szenarien alle drei Zugriffsarten interessant. Die Möglichkeit des Dateizugriffs wurde insbesondere während der Softwareentwicklung genutzt, aber auch für a-priori bekannte Umgebungsmodelle. Für die meisten Szenarien kam die interaktive Netzwerktechnologie zum Einsatz. In Einzelfällen wird auch die Broadcast Technologie benötigt. Zum Beispiel gibt es Szenarien, in denen Teleoperationen im Weltall stattfinden und durch technische Einschränkungen nur eine simplex Kommunikation möglich ist. Durch die generische Definition der DAI gestaltet sich eine Anpassung auf die Bedürfnisse des Szenarios einfach, da nur eines der erwähnten Softwaremodule modifiziert werden muss. Das MPEG-4 System bleibt davon unberührt.

Dem Elementary Stream entspricht in DMIF der *Kanal*. Ein Kanal kann SL-Pakete versenden oder empfangen, und ist Teil eines *Service*. Ein Service entspricht einer MPEG-4 Präsentation. Bevor Kanäle angefordert werden können, muss zuerst der entsprechende Service erfolgreich geöffnet werden. Rückgabewert dieser Operation ist der initiale Objektdeskriptor, aus dem sich der Rest der Präsentation dem MPEG-4 System erschließt (siehe Ablaufbeispiel in Abschnitt 3.4). Weiterhin kann für jeden Kanal eine Überwachung des gewünschten *Quality of Service* aktiviert werden, und über *Out of Band* Nachrichten können Statusinformationen zwischen Sende- und Empfangsterminals ausgetauscht werden.

4 BIFS Daten-Kompression

Eine wichtige gewünschte Eigenschaft für die Übertragung von Szenendaten in Telepräsenz-Szenarien ist die Einsparung von Bandbreite auf der Übertragungsstrecke zwischen Teleoperator und Operator. Dies ist auf zwei verschiedenen Ebenen möglich. Zum einen können die generierten Daten für die Übertragung komprimiert werden. Diese Möglichkeit wird in Abschnitt 4.2 untersucht. Zum anderen kann bereits während der Modellerstellung optimiert werden. In den Abschnitten 4.4 und 4.5 werden Verfahren vorgestellt, die effizienter komprimierbare Modelle generieren.

Um für die folgenden Untersuchungen eine gemeinsame Basis zu schaffen, wird im nächsten Abschnitt zuerst dargestellt, welche Sensordaten und Algorithmen zur Modellrekonstruktion verwendet wurden.

4.1 Modellrekonstruktion aus Sensordaten

Die hier vorgestellten Algorithmen zur Modellrekonstruktion verarbeiten Daten von Sensoren, die Tiefenbilder der Umgebung von einem bekannten Blickwinkel aus erzeugen. Zu den bekanntesten Vertretern dieser Klasse zählen kalibrierte Stereokamerasysteme. Eine Alternative dazu sind 3D-Kameras, so wie sie z. B. von der Firma PMD Technologies hergestellt werden. Die hier verwendete PMD-Kamera arbeitet nach dem Time of Flight Messprinzip [64]. Prinzipiell ist auch die Verwendung von Daten eines Laserscanners möglich. Dabei ist jedoch zu beachten, dass mit diesen Sensoren typischerweise keine dynamischen Szenen erfasst werden können. Für eine Rekonstruktion von 3D-Punkten im Kamera-Koordinatensystem müssen alle intrinsischen PMD-Kameraparameter passend zum jeweiligen Tiefenbild bekannt sein. Für eine korrekte Berechnung der 3D-Punkte im Welt-Koordinatensystem müssen zusätzlich die extrinsischen PMD-Kameraparameter bekannt sein. Für Stereokamerasysteme oder Laserscanner bestehen ebenfalls ähnliche Anforderungen bezüglich der Kalibrierung.

Es wird im Folgendem davon ausgegangen, dass die extrinsischen Parameter für jede Tiefenbildaufnahme bekannt sind, z. B. weil die Kamera auf ein Schwenk-Neige-Gelenk montiert ist. Aufgrund der gegebenen Möglichkeiten war es technisch nicht möglich, die intrinsischen Parameter in kontrollierter Weise während der Experimente zu verändern. Für die Algorithmen hat dies keinen Einfluss, solange für jedes Tiefenbild die intrinsischen Parameter bekannt sind.

4.1.1 Sensordaten

Für die Gewinnung der Sensordaten wurde die oben erwähnte PMD-Kamera auf einen Schwenk-Neige-Kopf (Amtec) montiert. Die Kamera war einerseits intrinsisch, als auch bezüglich ihrer Montagelage auf dem Schwenk-Neige-Kopf kalibriert (Hand-Auge-Kalibrierung). Da die Gelenkwinkel des Schwenk-Neige-Kopfes zum Zeitpunkt der Bildaufnahme bekannt sind, können die extrinsischen Kameraparameter für jede Aufnahme berechnet werden.

Die verwendete PMD-Kamera hat eine Auflösung von 160x120 Pixel und ist in der Lage bis zu 15 Aufnahmen pro Sekunde zu machen. Die Kamera stellt als Ergebnis jeder Aufnahme eine Matrix von Tiefenwerten zur Verfügung. Dabei entspricht der Tiefenwert der Distanz zwischen dem von einem Sensorpixel erfassten Raumpunkt und dem optischen Zentrum der Kamera.

Zur besseren Vergleichbarkeit wurde für die folgenden Untersuchungen ein einmalig aufgenommener Datensatz von Sensordaten verwendet. Dieser besteht aus 12 Tiefenbildern, die von der PMD-Kamera gemacht wurden. Die Kamera wurde für jedes Bild in eine neue Position gedreht. Dadurch wurde erreicht, dass ein relativ großer Bereich des Raumes vor der Kamera erfasst wurde. Bild 4.1 zeigt auf der linken Seite die reale Umgebung als Foto, auf der rechten Seite die Rekonstruktion als Polygonmodell. Für die Darstellung wurden die rekonstruierten Flächen jedes Tiefenbildes hier eingefärbt, wodurch der Aufbau des Gesamtmodells aus den einzelnen Tiefenbildern ersichtlich wird. Die Drehungen der Kamera wurden so gewählt, dass sich die Tiefenbilder leicht überlappen, um ein möglichst lückenloses Modell rekonstruieren zu können. Die rechts in Abbildung 4.1 sichtbaren Lücken kommen dadurch zustande, dass der Sensor an der entsprechenden Oberfläche keine Distanz messen konnte. Dies tritt vor allem bei Glasoberflächen auf. Oberflächen, die von der Kameraposition aus verdeckt sind, können ebenfalls nicht rekonstruiert werden.

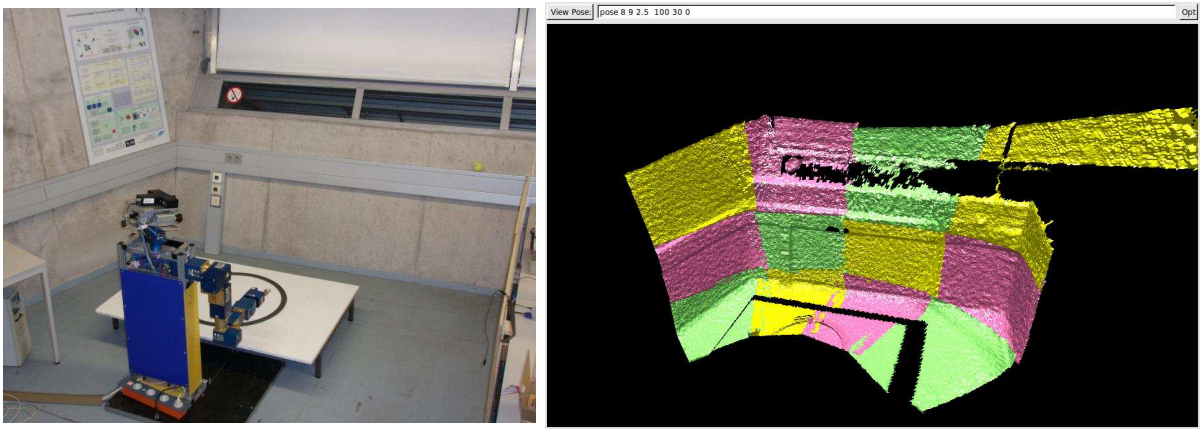


Bild 4.1: Bilder der Testszene. Links ein Foto der realen Szene, rechts die Rekonstruktion aus Tiefendaten (Tiefenbilder mit Falschfarben gekennzeichnet).

Weiterhin wurden für die Durchführung der Untersuchungen zwei exemplarische Werte für geforderte Rekonstruktionsgenauigkeit festgelegt. Für eine immersive Darstellung der entfernten Umgebung ist es oft wünschenswert, eine sehr präzise Rekonstruktion der Umgebung zu nutzen. Dieser Wunsch steht in Konkurrenz zur Anwendung von verlustbehafteten Kompressionsverfahren. Um eine möglichst große Abdeckung des gebräuchlichen Wertebereichs zu erreichen, wurde der erste Wert auf $2^{-9}\,\mathrm{m}$ ($\approx 2\,\mathrm{mm}$) gelegt. Die daraus resultierenden Modelle ermöglichen detailreiche, immersive Darstellungen. Der zweite Wert liegt bei $2^{-5}\,\mathrm{m}$ ($\approx 31\,\mathrm{mm}$). Er stellt die untere vertretbare Qualitätsgrenze dar.

4.1.2 Einzelbild-Rekonstruktion

Eine besonders einfache, und damit effizient programmierbare Möglichkeit der Rekonstruktion ist die Verwendung aller 3D-Punkte eines Tiefenbildes zur Generierung eines zusammenhängenden Polygonnetzes. Bei jeder neuen Aufnahme wird das alte Netz durch ein Neues ersetzt. Für die Erstellung eines Gesamtmodells der entfernten Umgebung ist diese Vorgehensweise ungeeignet. Sie kann jedoch trotzdem Verwendung im Szenario finden, solange sich alle relevanten Änderungen in der Szene innerhalb des Kamerasichtfeldes abspielen.

Der Algorithmus läuft zeilenweise über alle Pixel des Tiefenbildes. Für jeden Tiefenpixel mit gültigem Tiefenwert wird mit Hilfe der intrinsischen Kameraparameter ein 3D-Punkt im Kamerasystem berechnet. Ungültige Tiefenwerte ergeben sich, wenn die PMD-Kamera für einen Pixel nur unzureichende Messqualität feststellen konnte. Die entstehende Liste von 3D-Punkten wird im `point` Feld eines `Coordindate` Knoten abgelegt. Siehe hierzu auch Abbildung 4.3.

In einem zweiten Durchlauf über das Tiefenbild testet der Algorithmus, welche Polygone aus den 3D-Punkten gebildet werden können. Es werden hier ausschließlich Dreiecke verwendet. In Abbildung 4.2 ist ein Beispiel gegeben, in dem nur 12 Tiefenpixel betrachtet werden.

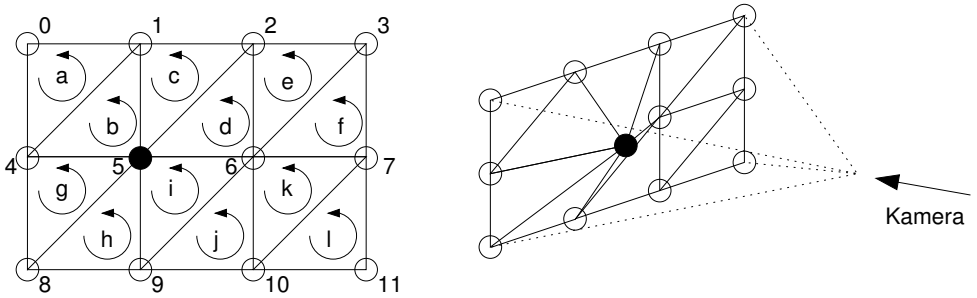


Bild 4.2: Beispiel für die Einzelbild-Rekonstruktion. Links: Tiefenpixel zusammen mit den Indizes der zugehörigen 3D-Punkte (0-11) sowie den rekonstruierbaren Dreiecken (*a*-*l*). Rechts: Schrägansicht des Dreiecksnetzes in Relation zur Kamera.

Der Algorithmus betrachtet immer vier benachbarte Tiefenpixel gemeinsam. Je nach Anzahl der ungültigen Tiefenpixel können kein, ein oder zwei Dreiecke gebildet werden. Im Beispiel sind die Dreiecke und ihr Umlaufsinn durch die Buchstaben a bis l gekennzeichnet. Eine weitere Bedingung für die Gültigkeit eines Dreiecks ist, dass der Winkel zwischen seinem Normalenvektor und dem Kamerablickrichtungsvektor einen festgelegten Schwellwert nicht unterschreitet. Diese Bedingung ist ein einfaches Mittel, um das Dreiecksnetz bei starken Sprüngen im Tiefenbild absichtlich aufzutrennen. Aus Erfahrung ist bekannt, dass solche Sprünge hauptsächlich dort auftreten, wo aus dem Blickwinkel der Kamera betrachtet ein Objekt im Vordergrund aufhört und ein anderes Objekt im Hintergrund beginnt. Im Beispiel sei angenommen, dass der schwarz gekennzeichnete 3D-Punkt mit dem Index 5 zu einem Objekt gehört, das näher an der Kamera liegt. Aufgrund der gewählten Heuristik sollten dann die Dreiecke b, c, d, g, h und i nicht rekonstruiert werden, da ansonsten dem Betrachter eine nicht-existente Verbindung zwischen Vorder- und Hintergrund suggeriert wird.

Sind alle Bedingungen für ein Dreieck erfüllt, werden die entsprechenden Indizes der 3D-Punkte zusammen mit der vorgeschriebenen -1 als Trennmarkierung in das `coordIndex` Feld eines `IFS` Knoten geschrieben. Für die Dreiecke im Beispiel ergibt dies folgende Liste: $[0, 4, 1, -1, 2, 6, 3, -1, 6, 7, 3, -1, ...]$.

Abbildung 4.3 zeigt den Szenengraphen für die Einzelbild-Rekonstruktion. Im Wurzelpunkt steht, wie vom Standard gefordert, ein `Group` Knoten. Als Kind ist ein `Transform` Knoten eingetragen, dessen Koordinatentransformation den extrinsischen PMD-Kameraparametern entspricht. Dadurch werden die 3D-Punkte, die oben für das Kamerasystem rekonstruiert wurden, bei der Darstellung in das Weltkoordinatensystem transformiert. Unterhalb des `Transform` Knotens befindet sich ein `Shape` Knoten, in dessen `geometry` Feld der bereits oben erwähnte `IFS` Knoten eingetragen ist. Der ebenfalls bereits erwähnte `Coordinate` Knoten muss im `coord` Feld des `IFS` Knoten stehen.

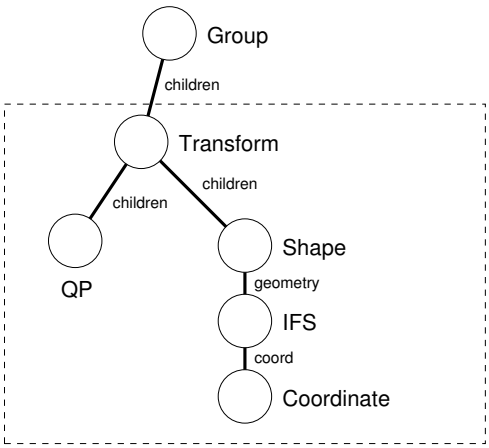


Bild 4.3: Knotenstruktur im Fall der Einzelbild-Rekonstruktion. Alle Knoten innerhalb des gestrichelten Bereiches werden für jedes neue Tiefenbild ersetzt.

Sobald ein BIFS Kompressionswerkzeug verwendet werden soll, muss zusätzlich ein `QP`

Knoten vorhanden sein. Die gewählte Position im Graphen ist sehr günstig, da ein `NodeReplace` Kommando für den `Transform` Knoten nicht nur diesen, sondern auch alle seine Kindknoten ersetzt. Folglich werden der `QP` Knoten und die zu komprimierenden Geometriedaten in einem Kommando übertragen, und nach Regel 2 aus Kapitel 3.3 wird das entsprechende Werkzeug aktiv. Andererseits wird kein einziges Feld des `Transform` Knoten durch verlustbehaftete Kompression übertragen, da der `QP` Knoten eine Ebene tiefer als der `Transform` Knoten liegt. Bereits kleine Kompressionsverluste im `rotation` Feld des `Transform` Knoten erzeugen eine deutliche Positionsverfälschung der rekonstruierten Dreiecksnetze.

Ein effizienteres Kommando als `NodeReplace` kann nicht verwendet werden. Zwar kann eine Veränderung in der Liste der 3D-Punkte als BIFS-Update Kommando verschickt werden, jedoch ist es nicht möglich, für eine Veränderung des `coordIndex` Feld im `IFS` Knoten ein Kommando zu generieren. Dieses Feld hat keine inID (siehe Kapitel 3.2.6). Einzig ein `IdxValueReplace` Kommando auf das `children` Feld des `Group` Knoten kann alternativ verwendet werden. Der Unterschied im Platzbedarf für die kodierten Kommandos ist marginal.

Neben dem `IFS` Knoten sind noch eine Reihe weiterer Knoten in MPEG-4 definiert, um Geometrie abzuspeichern. Der in ISO/IEC 14496-11 spezifizierte `ElevationGrid` Knoten kann hier jedoch nicht eingesetzt werden, da für die PMD-Kamera eine perspektivische Projektion verwendet werden muss. Die in ISO/IEC 14496-16 („Animation Framework eXtension“, AFX) spezifizierten Erweiterungen im Bereich „Depth Image-Based Representation“ erscheinen für diesen Zweck sehr interessant. Jedoch repräsentieren sie die Umgebung als Voxelmodell, welche im Rahmen dieser Arbeit nicht betrachtet werden.

Auf der in Anhang B beschriebenen Hardware benötigt der Algorithmus zur Erzeugung eines Netzes für ein Tiefenbild durchschnittlich 24 ms. Die Tiefenbilder liegen in einer Auflösung von 160x120 Pixel vor. Dabei werden maximal 19 200 3D-Punkte und 37 842 Dreiecke rekonstruiert, was 151 368 Indizes entspricht. Im Speicher belegt ein Dreiecksnetz maximal 835 872 Byte, abgesehen von ein paar wenigen Byte für die Beschreibung der Knotenstruktur. In Anhang C.2 sind ein paar Beispiele für den Platzverbrauch der Knotenkodierung abgedruckt. Eine unkomprimierte Übertragung sämtlicher Informationen würde bei einer Tiefenbildrate von 10 Bildern pro Sekunde eine Bandbreite von etwa 63 MBit/s erreichen.

4.1.3 Modifizierte Einzelbild-Rekonstruktion

In dem oben beschriebenen Algorithmus wird für jedes neue Tiefenbild das alte Dreiecksnetz gelöscht. Aus praktischen Gründen wurde für die Durchführung der Untersuchungen jedoch das Entfernen der alten Netze deaktiviert, sobald mit dem in Abschnitt 4.1.1 erwähnten Testdatensatz gearbeitet wurde. Das Resultat sind Modelle, die nicht nur ein Tiefenbild, sondern den ganzen Testdatensatz für die Rekonstruktion der Umgebung verwenden. Diese Modelle können in ihrem Umfang direkt mit anderen Rekonstruktionsalgorithmen verglichen werden, die ebenfalls das Gesamtmodell rekonstruieren. Diese

Vorgehensweise hat jedoch nur wenig Relevanz für echte Telepräsenz-Szenarien, da es nur eine Frage der Zeit ist, bis das Modell den gesamten Hauptspeicher belegt.

4.1.4 Raumaufteilung

Die oben beschriebene Einzelbild-Rekonstruktion eignet sich nur bedingt für den Einsatz in der Praxis, da mit jedem neuen Tiefenbild alle alten Informationen gelöscht werden. Somit liegt von einer größeren Szene immer nur ein kleiner Ausschnitt im Modell vor. Auch der modifizierte Einzelbild-Algorithmus kann nicht über längere Zeit eingesetzt werden. Wird der Einzelbild-Algorithmus daraufhin erweitert, dass eine Speicherung einer beschränkten Menge von Tiefenbilder zugelassen wird, treten neue Probleme auf. Zum Beispiel stellt sich die Frage, welche alte Tiefenbilder durch neue Tiefenbilder ersetzt werden dürfen.

Das grundlegende Problem hier ist, dass eigentlich für jedes neue generierte Dreiecksnetz eine Fusion mit den bisher vorhandenen Daten durchgeführt werden muss. Für die Lösung dieses Problems wird an dieser Stelle nur eine einfache, aber praktikable Herangehensweise vorgestellt.

Im Prinzip wird die entfernte Umgebung in kleinere Raumbereiche unterteilt. Diese Raumbereiche sind jedoch wesentlich größer als bei den üblichen Methoden der Spatial Occupancy Enumeration. Innerhalb der Raumbereiche werden wiederum Dreiecksnetze verwaltet. Die Dreiecke dürfen in die benachbarten Raumbereiche hineinragen. Dies ist ein Zugeständnis zugunsten der Laufzeitkomplexität. Die Form der Raumbereiche wurde zweckmäßigerweise als regelmäßige, gleichseitige Würfel festgelegt. Abbildung 4.4 zeigt auf der linken Seite ein Beispiel für die Anordnung dieser Raumwürfel.

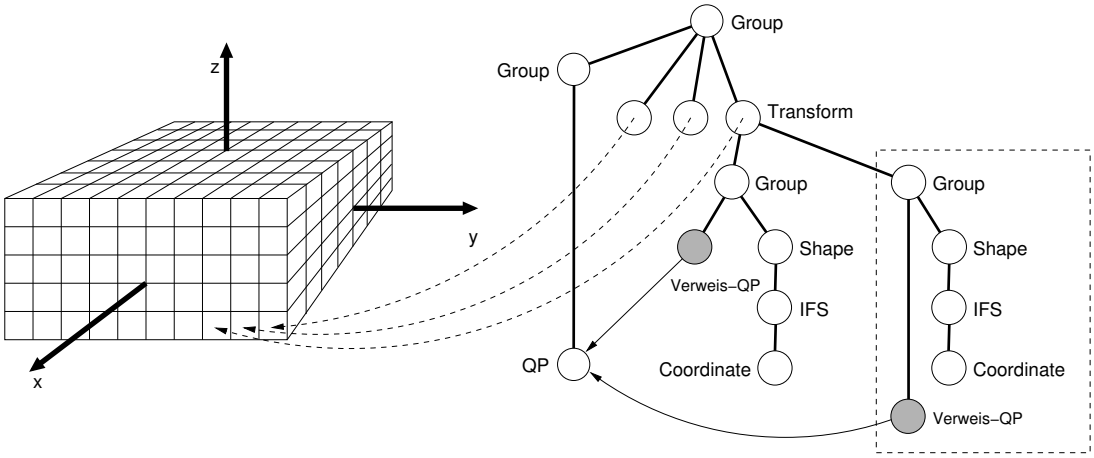


Bild 4.4: Würfelanordnung und Knotenstruktur für Raumaufteilung. Alle Knoten innerhalb des gestrichelten Rechtecks werden für jedes neue Tiefenbild hinzugefügt.

Die Würfel sind in einem festen Raster im Arbeitsraum des Teleoperators verteilt. Für jeden Würfel wird eine Knotenstruktur hinzugefügt, die mit einem `Transform` Knoten be-

ginnt (siehe auch Abbildung 4.4 rechts). Die Werte des `translation` Feldes im `Transform` Knoten sind durch die Position des jeweiligen Würfels im Raum fest vorgegeben. Das `rotation` Feld ist konstant für alle Würfel auf 0 initialisiert.

Da jeder Würfel die gleichen Abmessungen hat, gelten immer die gleichen Quantisierungsparameter. Deshalb reicht es, einen einzigen `QP` Knoten im Szenengraphen einzutragen, auf den an den notwendigen Stellen verwiesen wird. Da je nach Szenario und gewählter Würfelseitenlänge bis zu mehreren tausend Würfel verwendet werden, verringert die mehrfache Verwendung eines Knotens bereits hier die zu übertragene Datenmenge. Der `QP` Knoten wurde nicht direkt unter den Wurzelknoten eingetragen, um zu verhindern, dass die Felder der `Transform` Knoten quantisiert werden.

Die Dreiecksnetze sind wieder in der bereits vorgestellten `Shape`, `IFS` und `Coordinate` Knotenkombination abgelegt. Um später eine Fusion von alten und neuen Tiefenbilder durchführen zu können, dürfen hier jedoch mehrere Dreiecksnetze in einem Würfel gespeichert werden. Deshalb befindet sich zwischen `Transform` und `Shape` Knoten eine weitere Schicht von `Group` Knoten. Aus dem selben Grund wie bei der Einzelbild-Rekonstruktion werden die Verweise auf den `QP` Knoten als Kinder der `Group` Knoten eingetragen.

Jedes neue Tiefenbild muss nun daraufhin untersucht werden, in welchen Raumwürfel jeder berechnete 3D-Punkt einsortiert wird. Dieselbe Entscheidung muss für Dreiecke getroffen werden, deren Eckpunkte in mehreren Raumwürfeln einsortiert sind. Anders als bei der Einzelbild-Rekonstruktion werden hier 3D-Punkte und Dreiecke in einem Durchlauf erzeugt. Die Berechnung eines 3D-Punktes aus einem Tiefenwert erfolgt wie für die Einzelbild-Rekonstruktion. Anstatt das Tiefenbild Zeile für Zeile zu bearbeiten, werden immer Zeilenpaare untersucht. In Abbildung 4.5 ist ein Beispiel eines solchen Zeilenpaares skizziert, dessen 3D-Punkte von einem Würfel A bis in einen angrenzenden Würfel B reichen. Die folgende Beschreibung orientiert sich an diesem Beispiel.

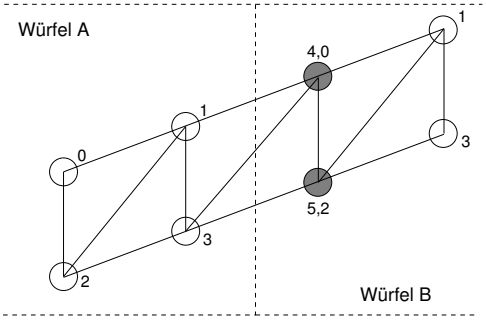


Bild 4.5: Tiefenbildzeilenpaare als Ausgangsbasis für die Einteilung in Würfel

Von jeder Zeile werden die ersten zwei 3D-Punkte betrachtet (von links nach rechts). Da alle vier Punkte innerhalb des Würfels A liegen, wird zuerst geprüft, ob und welche Dreiecke erzeugt werden sollen. Die Verfahren dazu sind identisch wie bei der Einzelbild-Rekonstruktion. Für jedes Dreieck werden die 3D-Punkte dem `point` Feld eines `Coordinate` Knotens von Würfel A hinzugefügt, solange dies nicht bereits für ein Dreieck vorher durchgeführt wurde. Die resultierenden Indizes der 3D-Punkte sind in Bild

4.5 angegeben. Die entsprechenden Indexfolgen für die Dreiecke werden wie im vorherigen Kapitel dem `coordIndex` Feld eines `IFS` Knoten von Würfel A hinzugefügt.

Danach wird die nächsten Vierergruppe von 3D-Punkten betrachtet. Da die neu hinzugekommenen 3D-Punkte nicht mehr in Würfel A liegen, jedoch noch Dreiecke mit den bereits zu Würfel A zugeordneten 3D-Punkten gebildet werden sollen, werden die neuen 3D-Punkte in beiden Würfeln hinzugefügt. In Abbildung 4.5 ist dies durch die Angabe beider Indexwerte angedeutet. Auf diese Weise treten keine Lücken am Übergang zwischen den Würfeln auf. Das Dreiecksnetz wird für den Betrachter nicht unterbrochen, solange die doppelt eingetragenen 3D-Punkte beim Betrachter wieder an identische Positionen dekodiert werden. Darauf ist vor allem bei der Wahl der Quantisierungsparameter zu achten.

Ist der Algorithmus über das gesamte Tiefenbild gelaufen, wird in einem abschließenden Schritt für alle Würfel geprüft, ob neue Dreiecke in Form eines `IFS` Knoten hinzugekommen sind. Ist dies der Fall, wird der neue `IFS` Knoten, wie in Abbildung 4.4 auf der rechten Seite im gestrichelten Rahmen dargestellt, in den Szenengraphen eingefügt. Das Einfügen wird mit einem `NodeInsert` Kommando übertragen.

Wenn in einem Würfel zum Zeitpunkt der Bearbeitung eines neuen Tiefenbildes bereits Geometrie aus vorherigen Tiefenbildern vorhanden ist, sollte eine Fusion der alten mit den neuen Daten erfolgen. Dies ist jedoch ein sehr rechenzeitintensiver Prozess, da alle alten Dreiecke auf Überschneidungen mit allen neuen Dreiecken überprüft werden müssen. Je nach Situation sind entsprechend viele Änderungen notwendig. Komplex wird die Fusion auch deshalb, weil für die Berechnung der Blickwinkel der PMD-Kamera entscheidend ist. Deswegen wurde an dieser Stelle eine weitaus einfachere Fusionsstrategie gewählt: es werden so lange neue Dreiecksnetze pro Würfel akzeptiert, bis eine festgelegte Insgesamtanzahl von 3D-Punkten im Würfel erreicht wird. Beim Überschreiten dieser Grenze werden solange die ältesten Dreiecksnetze entfernt, bis die Grenze wieder unterschritten wird. Diese Vorgehensweise lässt natürlich noch viel Raum für weitere Arbeiten offen, für einige Szenarien hat sie sich jedoch in der Praxis bewährt.

Für die Verwendung in der in Abbildung 4.1 gezeigten Umgebung wurden Raumwürfel mit einer Kantenlänge von 0,2 m bis 0,5 m verwendet. Bei einer erwarteten Größe des Gesamtmodells von ca. 7 m auf 5 m auf 3 m sind also zwischen 800 und 13 000 Würfel zu erwarten. Um nicht unnötige Bits mit der Übertragung leerer Knoten zur verwenden, werden die `Transform` Knoten der Würfel erst dann im Modell eingetragen, wenn aktuell Bedarf dazu besteht. Die Bearbeitung der in Abschnitt 4.1.1 vorgestellten Tiefenbildserie beansprucht durchschnittlich 32 ms bei einer Tiefenbildauflösung von 160x120 Pixel, gemessen für eine Würfelgröße von 0,3 m.

4.2 BIFS Kompressionswerkzeuge

In den folgenden drei Abschnitten wird untersucht, welche Auswirkungen die im Grundlagenkapitel in den Abschnitten 3.3.1, 3.3.2 und 3.3.3 beschriebenen BIFS Kompressions-

werkzeuge auf die generierten Modelle haben. Insbesondere wird hier die Kompressionsrate und das Laufzeitverhalten betrachtet.

4.2.1 EfficientFloat Enkoderstrategien

Durch den Standard ist der Algorithmus zur Dekodierung von EfficientFloat eindeutig festgelegt. Der Algorithmus für den Enkoder muss lediglich festlegen, mit wie vielen Bits sowohl die Mantisse, als auch der Exponent kodiert werden sollen. Für den Exponenten erscheint dazu nur eine günstige Wahl zu existieren. Es müssen gerade so viele Bits zur Kodierung verwendet werden, wie zur binären Darstellung des Exponenten e gerade notwendig sind. Das sind $\lceil log_2(e+1) \rceil$ Bits. Mehr Bits sind unnötig, und bei weniger Bits wird der Exponent nicht mehr vollständig kodiert.

Die Mantisse jedoch muss nicht immer vollständig übertragen werden. Je nach Anzahl der kodierten Mantissenbits steigt oder sinkt die Differenz zwischen en- und dekodiertem Wert. Die maximal mögliche Differenz wird im Folgenden als Maß für den Kompressionsverlust verwendet. In Abschnitt 4.1.1 wurden hierfür zwei Grenzwerte festgelegt.

Für die Wahl der Anzahl kodierter Mantissenbits wurden zwei Strategien entwickelt, die unterschiedliche Ziele verfolgen und in den nächsten beiden Abschnitten vorgestellt werden. Danach folgen Messergebnisse zu dem in 4.1 vorgestellten Testdatensatz.

Konservative Strategie

Bei dieser Strategie versucht der Enkoder von den bekannten 23 Mantissenbits so viele gesetzte Bits wie möglich zu kodieren. Da nicht gesetzte Bits am Ende der Mantisse den kodierten Wert nicht beeinflussen, können diese dort entfallen. Der Dekoder nimmt den Wert nicht kodierter Bits stets als 0 an. Da nicht mehr als 14 Mantissenbits über EfficientFloat kodiert werden können, beginnt die Suche immer beim 14ten Bit. In Abbildung 4.6 ist dies an einem Beispiel dargestellt.

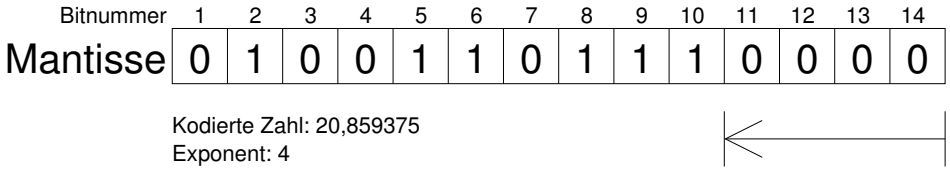


Bild 4.6: Beispiel für die konservative EfficientFloat Strategie

Beginnend bei Bit 14 wird solange gesucht, bis ein gesetztes Bit gefunden wird. In diesem Beispiel ist das bei Bit 10 der Fall. Hier würde die Mantisse mit einer Genauigkeit von 10 Bit abgespeichert werden.

Einsparungen ergeben sich bei dieser Strategie nur dann, wenn eine lange Folge von nicht gesetzten Bits ab Bit 14 in der zu kodierenden Mantisse steht. Der Name der Strategie

ergibt sich aus der Eigenschaft, so viel Information wie möglich, und so wenig wie notwendig in die Kodierung einzubeziehen. Der Vorteil dieser Strategie ist, dass keine Parameter festgelegt werden müssen.

Genauigkeits-Strategie

Bei der konservativen Strategie wird der Kompressionsverlust für jede zu kodierende Zahl auf das mögliche Minimum reduziert. Wenn jedoch ein maximal erlaubter Kompressionsverlust bekannt ist, können weniger Mantissenbits notwendig sein, um unter dem Grenzwert zu bleiben.

Die Genauigkeits-Strategie versucht, den Kompressionsverlust bis zur erlaubten Grenze zu maximieren, um weniger Mantissenbits zu verwenden. Dazu muss der Exponent der zu kodierenden Zahl betrachtet werden. Ist z. B. für die Kodierung der Position von 3D-Punkten ein maximaler Kompressionsverlust von 0,25 erlaubt, müssen bei einem Exponent von 0 nur zwei Mantissenbits kodiert werden. Alle nachfolgenden Mantissenbits tragen weniger als 0,25 zur dekodierten Zahl bei. Bei einem Exponent von -1 muss nur noch ein Mantissenbit kodiert werden.

Der Enkoder-Algorithmus funktioniert ähnlich wie oben, jedoch startet die Suche nach dem letzten 1er Mantissenbit nicht immer bei Bit 14. Die Nummer des Startbits variiert mit dem Grenzwert des Kompressionsverlusts und dem Exponenten der zu kodierenden Zahl. Wird für das Beispiel in Abbildung 4.6 ein maximaler Kompressionsverlust von 0,25 verlangt, startet die Suche bei Bit 6. Zwei Bit wegen dem Grenzwert plus vier Bit weil der Exponent den Wert 4 hat. Die Suche bricht gleich wieder ab, weil Bit 6 den Wert 1 hat. Der Empfänger würde die Zahl 20,75 dekodieren. Der Kompressionsverlust liegt mit $|20,859375 - 20,75| = 0,109375$ unterhalb der geforderten Grenze.

Wenn der erlaubte Kompressionsverlust nicht zu gering gewählt wird, ist bei dieser Strategie zu erwarten, dass mehr Bits gespart werden können. Die Einsparung wird dabei in direktem Zusammenhang mit dem erlaubten Kompressionsverlust stehen.

Messung der Kompressionsrate und Laufzeit

Mit Hilfe des modifizierten Einzelbild-Algorithmus wurde ein Modell aus dem in Abschnitt 4.1.1 beschriebenen Datensatz generiert und als BIFS-Update Kommandos kodiert. Die oben vorgestellten Enkoder-Strategien kamen für die Kodierung der Geometriedaten zum Einsatz. Die Versuche wurden auf der in Anhang B beschriebenen Hardware durchgeführt und die Messergebnisse sind in Tabelle 4.1 aufgeführt.

Da für die Verwendung von EfficientFloat ein `QP` Knoten im Szenengraphen vorhanden sein muss, wird automatisch das `CoordIndex` Verfahren für die Indizes verwendet[1)]. Die angegebenen Werte für das gesamte Modell in der obere Hälfte von Tabelle 4.1 werden

[1)] siehe Kapitel 3.3.2

Strategie	Unkomprimiert	Konservativ	Genauigkeit		
			2^{-9} m	2^{-7} m	2^{-5} m
Bitgröße gesamt [10^6 bit]	75.98	40.45	37.17	35.75	34.15
Kompressionsrate gesamt	1.0	1.88	2.04	2.12	2.22
Enkodierzeit gesamt [ms]	96.84	192.24	173.04	157.32	147.16
Dekodierzeit gesamt [ms]	82.36	127.68	126.6	128.8	126.92
Bitgröße 3D-Punkte [10^6 bit]	21.23	15.02	11.73	10.32	8.72
Kompressionsrate 3D-Punkte	1.0	1.41	1.81	2.06	2.43
Enkodierzeit 3D-Punkte [ms]	24.24	127.64	108.48	93.6	82.88
Dekodierzeit 3D-Punkte [ms]	18.36	51.4	49.48	51.3	48.88

Tabelle 4.1: EfficientFloat Kodierungsergebnisse

also durch zwei Effekte beeinflusst. Für eine aussagekräftige Beurteilung der EfficientFloat Kodierung alleine sind in der zweiten Hälfte der Tabelle die relevanten Zahlen zur Kodierung der 3D-Punkte aufgeführt. Diese werden nur durch EfficientFloat beeinflusst.

Wie erwartet werden 3D-Punkte mit EfficientFloat mit weniger Bits kodiert. Die konservative Strategie verhält sich wie die Genauigkeits-Strategie mit sehr hoher Anforderung an Kompressionsverluste. Die gewünschte Genauigkeit hat direkten Einfluss auf die Kompressionsrate, jedoch auch auf die Qualität der komprimierten Modelle. Zum subjektiven Vergleich sind für die hier aufgeführten Verfahren in Abbildung 4.7 Ansichten eines kleinen Objekts in der Testszene zu sehen. Während die konservative Strategie praktisch keine sichtbaren Unterschiede verursacht, wird das Objekt mit steigenden Kompressionsverlusten zunehmend deformiert. In der Praxis war der Grenzwert von 2^{-7}m für die meisten Szenarien angemessen.

Die benötigte Laufzeit zur Enkodierung sinkt mit steigenden Kompressionsverlusten. Dies ist in der Implementierung des Enkoders auf eine Schleife zurückzuführen, die in der Mantisse das letzte gesetzte Bit sucht. Diese Suche fällt durchschnittlich umso kürzer aus, je mehr Kompressionsverluste zugelassen werden. Da die Laufzeiten für die Dekodierung konstant sind, kann nicht davon ausgegangen werden, dass die Speicherbandbreite des Rechners ein limitierender Faktor ist. Insgesamt liegt die Laufzeit bei Verwendung des EfficientFloat Werkzeugs höher als bei unkomprimierter Übertragung, was bei der Enkodierung wieder auf die Suche zurückzuführen ist. Bei der Dekodierung benötigt die Berechnung der Fließkommazahl nach Formel 3.1 mehr Zeit als das reine Einlesen von 32 Bit in eine Fließkommavariable.

Für die Verwendung des EfficientFloat Werkzeugs müssen nur wenige Maßnahmen getroffen werden. Neben der Auswahl der Enkoder-Strategie muss nur ein `QP` Knoten mit gesetztem `useEfficientCoding` Feld an passender Stelle in das Modell eingefügt werden. Da von diesem Werkzeug die Kodierung aller Fließkommazahlen beeinflusst wird, muss dabei allerdings darauf geachtet werden, dass sensible Felder davon nicht betroffen sind. Als Beispiel für ein sensibles Feld wurde bereits in Abschnitt 4.1.2 auf das `rotation` Feld der `IFS` Knoten hingewiesen.

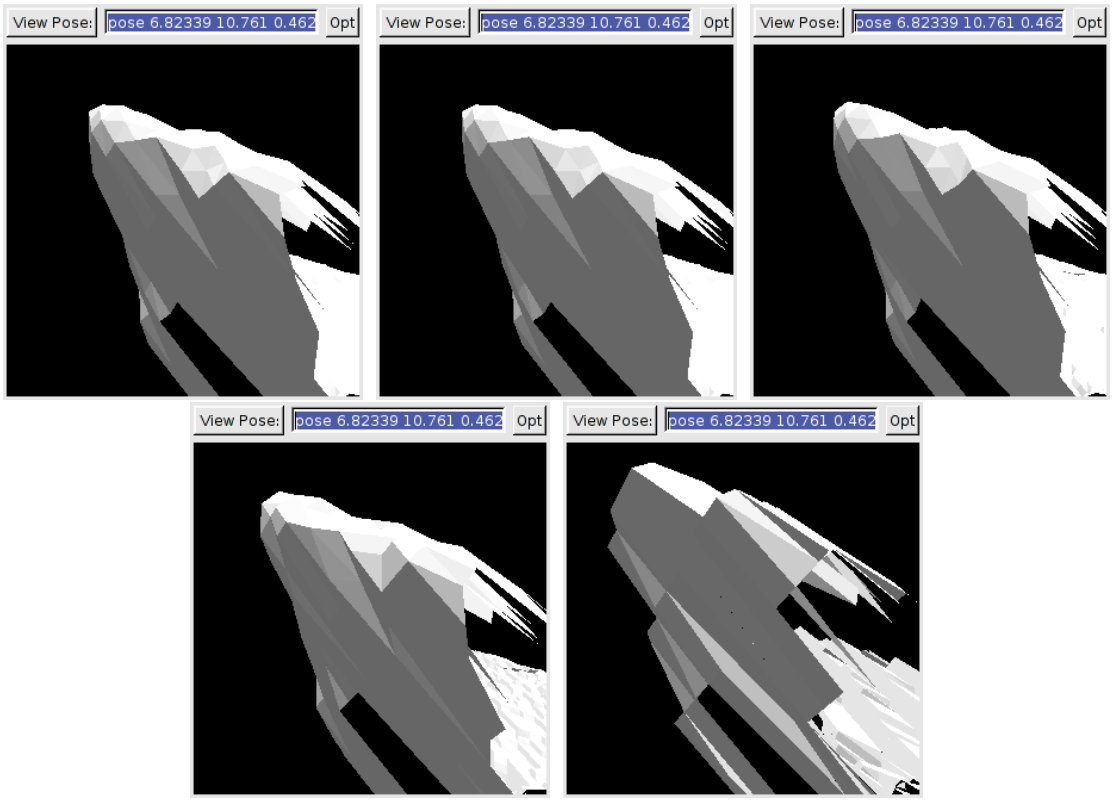


Bild 4.7: Detailansicht in der Testszene, v.l.n.r.: Unkomprimiert, Konservativ, Genauigkeit 2^{-9} m, 2^{-7} m, 2^{-5} m

4.2.2 Quantisierung und `CoordIndex`

Für das Quantisierungs-Werkzeug gibt es keine Wahl der Enkoder-Strategie, da die Vorschrift zur Dekodierung das Verfahren zur Enkodierung exakt vorschreibt. Eine Analyse der generierten Modelldaten des Testdatensatzes ergibt, dass 3D-Punkte mit über 27,9 % und Indizes mit 72 % den weitaus größten Teil der verwendeten Bits für ein unkomprimiertes Modell ausmachen. Deshalb konzentrieren sich die Untersuchungen hier auf `Position3D` und `CoordIndex`.

Während für `CoordIndex` keinerlei Parameter verfügbar sind, muss für `Position3D` sowohl der maximale Wertebereich, als auch die Anzahl *nb* der Bits für die Kodierung gewählt werden. Der maximale Wertebereich ergibt sich aus den minimalen und maximalen Vektorkomponenten aller von einem `QP` Knoten betroffenen 3D-Punkte. Für die Einzelbild-Rekonstruktion kann er während der Bearbeitung eines Tiefenbildes ermittelt werden. Beim Raumaufteilungs-Algorithmus ist dieser Wertebereich bereits durch die Einteilung in Raumbereiche festgelegt. Über *nb* kann, ähnlich wie bei der Genauigkeits-Strategie für EfficientFloat, der Verlust bei der Kompression gesteuert werden. Auch für `Position3D` wurde *nb*, in Abhängigkeit vom maximalen Wertebereich, immer derart festgelegt, dass die Kompressionsverluste unterhalb dem gewünschten Grenzwert bleiben.

Sowohl die Berechnung der quantisierten Werte nach Formel 3.2, als auch die Dekodierung mit Hilfe von Formel 3.3 konnten mittels SSE2 Befehlen optimiert werden. Mit Hilfe dieser *Single Instruction Multiple Data* (SIMD) Befehle konnten alle drei Komponenten eines 3D-Punktes vom Prozessor parallel bearbeitet werden.

Messung der Kompressionsrate und Laufzeit

Die Vorgehensweise zur Ermittlung der Messwerte war hier identisch wie im Abschnitt zu EfficientFloat. Anstatt EfficientFloat wurde hier jedoch zur Kodierung das Quantisierungs-Werkzeug verwendet. Tabelle 4.2 zeigt die ermittelten Messwerte.

Quantisierung	keine	2^{-9} m	2^{-7} m	2^{-5} m
Bitgröße gesamt [10^6Bit]	75.98	32.10	30.78	29.45
Kompressionsrate gesamt	1.0	2.37	2.47	2.57
Enkodierzeit gesamt [ms]	96.84	97.12	94.52	98.16
Dekodierzeit gesamt [ms]	82.36	118.64	122.2	120.08
Bitgröße 3D-Punkte [10^6Bit]	21.23	6.67	5.34	4.02
Kompressionsrate 3D-Punkte	1.0	3.18	3.97	5.29
Enkodierzeit 3D-Punkte [ms]	24.24	32.92	29.72	32.76
Dekodierzeit 3D-Punkte [ms]	18.68	42.4	43.8	41.52

Tabelle 4.2: Quantisierung Kodierungsergebnisse

Wie bei EfficientFloat werden durch das Einfügen des `QP` Knoten das `Position3D` und das `CoordIndex` Verfahren zugleich aktiviert. Deshalb befinden sich auch hier wieder in der oberen Hälfte der Tabelle die Werte für die gesamte Szene. In der unteren Hälfte sind die Messwerte für die Kodierung der 3D-Punkte über `Position3D` separat angegeben.

Im Vergleich zu EfficientFloat können hier annähernd doppelt so hohe Kompressionsraten für die 3D-Punkte erzielt werden. Da das `Position3D` Verfahren keine iterative Berechnungen durchführt, sind sowohl Enkodier- als auch Dekodierzeiten annähernd konstant über alle Quantisierungsstufen. Relativ zu EfficientFloat ist auch eine deutlich geringere Enkodierzeit festzustellen. Im Vergleich zur unkomprimierten Kodierung verbraucht die Berechnung der Quantisierung nur ein wenig mehr Rechenzeit.

Zum subjektiven Vergleich der Modellqualität wird in Abbildung 4.8 wieder das gleiche Objekt wie in Abbildung 4.7 dargestellt. Hier lässt sich feststellen, dass bereits bei einer Quantisierung auf 2^{-7}m die Form der kodierten Geometrie stärker als bei EfficientFloat verändert wird.

4.2.3 Predictive MFField

Das Predictive MFField Werkzeug baut auf einer vorhergehenden Quantisierung auf. Zusätzlich zu den im vorherigen Abschnitt beschriebenen Parametern muss für die PMF

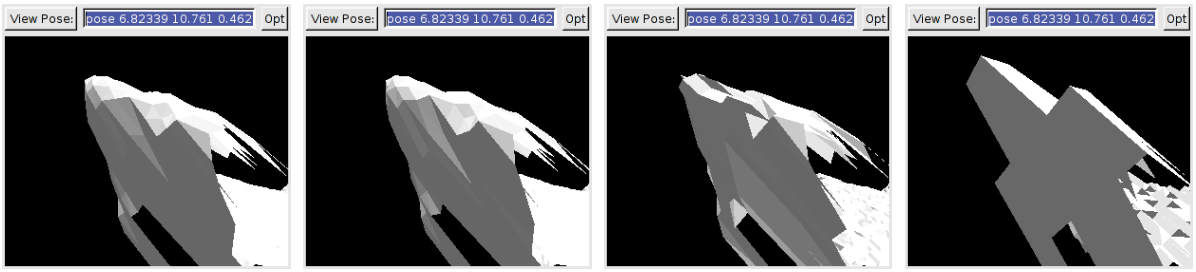


Bild 4.8: Detailansicht in der Testszene, v.l.n.r.: Unkomprimiert, Genauigkeit 2^{-9}m, 2^{-7}m, 2^{-5}m

Kodierung das Schema m[2)] festgelegt werden. Auch mit dem PMF Werkzeug sollen wieder die 3D-Punkte und die Indizes komprimiert kodiert werden.

In der folgenden Aufstellung ist für jedes Kodierschema m beschrieben, wie es für die generierten Modelle auf die Listen von 3D-Punkten und Indizes angewendet werden kann:

$m = 1$: Im IPPP-Modus werden alle Werte als Differenzen kodiert. Wird eine Quantisierung mit q Bit verwendet, können bei einer beliebigen Sortierung des MF-Feldes maximal $2^{q+1} - 1$ verschiedene Differenzwerte auftreten. Für jeden möglichen Differenzwert muss ein Symbol im Modell des arithmetischen Kodierers angelegt sein. Da die Größe dieser Modelle auf 2^{13} Symbole beschränkt ist, können beliebig sortierte MF-Felder nur dann mit diesem Schema kodiert werden, wenn sie mit $q \leq 12$ Bit quantisiert werden.

Wenn eine Sortierung des MF-Feldes angegeben werden kann, über die weniger als 2^{13} Symbole benötigt werden, entfällt die Beschränkung. Da die Positionen der 3D-Punkte durch Sensordaten bestimmt sind, kann jedoch nicht garantiert werden, dass eine solche Sortierung möglich ist. Ähnliches gilt für Indexlisten. Als Trennmarkierung wird hier der Wert -1 verwendet. Dieser steht folglich sehr oft in der Indexliste, umgeben von großen Zahlenwerten. Unter Umständen kann hier die Liste der 3D-Punkte, auf die die Indexliste referenziert, derart umsortiert werden, dass weniger Symbole notwendig sind. Auf eine Fortführung dieser Untersuchungen wurden für diese Arbeit allerdings verzichtet.

$m = 2$: Der IPPIPP-Modus ist dann vorteilhaft, wenn in regelmäßigen Abständen große Differenzen in der Folge auftreten, die restlichen Differenzen jedoch mit wenigen Symbolen dargestellt werden können.

Für beliebig sortierte 3D-Punktelisten oder Indexlisten bringt dieses Schema keinen Vorteil. Wieder können keine generellen Aussagen über die Differenzen zwischen den 3D-Punkte gemacht werden, ohne aufwendige Sortierverfahren anzuwenden. Für Indexlisten kann zwar der regelmäßige I-Wert auf die Trennmarkierung gelegt werden. Jedoch folgt auf jede Trennmarkierung ein möglicherweise hoher Indexwert. Für die Kodierung der Differenz -1 $\rightarrow$ *Index* werden 2^q Sym-

[2)] siehe Kapitel 3.3.3

bole benötigt. Für die restlichen Differenzen $Index \rightarrow Index$ kann die Menge der zusätzlich benötigten Symbole abgeschätzt werden. Insgesamt hält sich aber die erzielbare Einsparung an verwendeten Symbolen in Grenzen, da die Symbolanzahl nur in 2er Potenzen gewählt werden kann.

Später wird in Abschnitt 4.5 ein Verfahren vorgestellt, mit dem dieses Kodierschema für 3D-Punkte sehr effizient verwendet werden kann. Dazu wird jedoch eine a-priori bekannte Struktur der 3D-Punkte benötigt.

$m = 3$: In diesem Schema kann die Abfolge von I- und P-Werten beliebig festgelegt werden. Deshalb liegt es nahe, große Differenzen durch I-Werte zu kodieren, während die kleineren Differenzen über P-Werte kodiert werden.

Insbesondere für Indexlisten erscheint das folgende Muster günstig:
IPPIIPPIIPPI...
Die Trennmarkierung und der darauf folgende Indexwert werden über I-Werte kodiert. Für die restlichen Differenzen kann anhand des Rekonstruktionsalgorithmus die Anzahl der benötigten Symbole abgeschätzt werden. Für 3D-Punkte ist eine ähnliche Vorgehensweise möglich.

An dieser Stelle muss noch erwähnt werden, dass mindestens zwei zusätzliche Bits benötigt werden, um eine laufende arithmetische Kodierung abzuschließen. Bei jedem Wechsel von arithmetischer zu normaler Kodierung ist dies notwendig. Diese Wechsel treten bei allen drei Schemas auf:

$m = 1$: Einmal am Ende des MF-Feldes.
$m = 2$: Unmittelbar vor jedem I-Wert und am Ende des MF-Feldes.
$m = 3$: Vor jedem (!) kodierten Wert und am Ende des MF-Feldes.

Diese zusätzlichen, aber unbedingt notwendigen Bits verschlechtern die erwartete Kompressionsrate. Wenn ein Wert über die arithmetisch Kodierung dekodiert werden soll, muss in der kumulierten Symbolhäufigkeitsverteilung gesucht werden. Im MPEG-4 Standard wird als Suchmethode eine lineare Suche vorgeschlagen. Diese Suchmethode wurde für die hier verwendete Implementierung durch ein Bisektionsverfahren ersetzt, das insbesondere für größere Symbolmodelle einen deutlichen Laufzeitgewinn erzielt. Auch die Anpassung der kumulierten Symbolhäufigkeitsverteilung nach der Kodierung eines Symbols konnte mittels SSE2 oder alternativ MMX Befehlen beschleunigt werden.

Messung der Kompression und Laufzeit

Wie in den vorhergehenden zwei Kapiteln wurde wieder der Datensatz aus Abschnitt 4.1.1 kodiert. Auf eine Variation der Anzahl der Bits für die Quantisierung wurde verzichtet. Anstatt dessen werden Messergebnisse für zwei Kodierschemas gegeben. Tabelle 4.3 zeigt, jeweils getrennt für Gesamtmodell, Indizes und 3D-Punkte, die Anzahl der benötigten Bits, Kompressionsrate, sowie die erzielten Laufzeiten für Enkodierung und Dekodierung.

Verfahren	Nur Quantisierung 2^{-9} m	PMF Modus 1	PMF Modus 1+3
Bitgröße gesamt [10^6Bit]	32.10	27.67	22.39
Kompressionsrate gesamt	1.0	1.16	1.43
Enkodierzeit gesamt [ms]	97.12	569.40	911.16
Dekodierzeit gesamt [ms]	118.64	655.24	2310.20
Bitgröße Indizes [10^6Bit]	25.43	25.43	20.15
Kompressionsrate Indizes	1.0	1.0	1.26
Enkodierzeit Indizes [ms]	63.92	64.06	404.4
Dekodierzeit Indizes [ms]	72.88	73.04	1713.4
Bitgröße 3D-Punkte [10^6Bit]	6.67	2.23	2.23
Kompressionsrate 3D-Punkte	1.0	2.99	2.99
Enkodierzeit 3D-Punkte [ms]	32.92	504.16	505.56
Dekodierzeit 3D-Punkte [ms]	42.4	580.24	594.80

Tabelle 4.3: Predictive MFField Kodierungsergebnisse

Bei „PMF Modus 1“ wird versucht, sowohl 3D-Punkte als auch Indizes mit dem Schema $m = 1$ zu kodieren. In der Spalte „PMF Modus 1+3“ werden 3D-Punkte mit dem Schema $m = 1$ kodiert, Indizes mit dem Schema $m = 3$. Da durch PMF zu den Quantisierungsverlusten keine weiteren Kompressionsverluste hinzugefügt werden, wird hier auf eine subjektive Beurteilung der Modellqualität verzichtet.

Aus den Messwerten für die Laufzeiten wird deutlich, welchen algorithmischen Aufwand eine arithmetische Kodierung mit sich bringt. Der PMF Modus 1+3 kann sowohl für 3D-Punkte, als auch für Indizes eine Steigerung in der Kompressionsrate erzielen. Jedoch benötigt er allerdings auch ein Vielfaches der Laufzeit. Mit dem PMF Modus 1 ist es nicht möglich, die im Modell enthaltenen Indexlisten zu komprimieren. Da mehr als 2^{13} Symbole benötigt werden, verwendet die Implementierung hier anstatt dessen das normale `CoordIndex` Verfahren. Für 3D-Punkte erzielt der PMF Modus 1 eine ordentliche Steigerung der Kompression, jedoch auch hier auf Kosten der Laufzeit.

Der Gewinn an Kompressionsrate bei Indizes für den PMF-Modus 1+3 fällt gering aus. Dies ist auf die oben erwähnten zusätzliche Bits zurückzuführen, die nach jedem Wechsel von arithmetischer zu normaler Kodierung notwendig sind. Die auffallend starke Erhöhung der Dekodierzeit für Indizes im PMF-Modus 1+3 wird ebenfalls durch die Wechsel verursacht. Der Dekodieralgorithmus muss 16 Bit im Bitstrom vorauslesen, bevor das erste Symbol dekodiert werden kann. Bei jedem Umschalten zu arithmetischer Kodierung werden diese 16 Bit immer wieder erneut eingelesen. Da ebenfalls ein *Bit Stuffing* für den arithmetischen Bitstrom vorgesehen ist, kostet das Einlesen entsprechend viel Zeit.

4.3 Zwischenresümee

An dieser Stelle soll kurz zusammengefasst werden, welche Auswirkungen die vorgestellten Kompressionswerkzeuge auf kodierte Größe und benötigte Laufzeit haben. Dabei werden nur 3D-Punkte und Indizes betrachtet. Für den vorgestellten Datensatz stellen diese beiden Datentypen in Summe mit ca. 99,9 % den weitaus größten Anteil dar. Diese Prozentzahl gilt sowohl für die Laufzeit, als auch für den kodierten Platzbedarf. Andere Daten, wie z. B. die Kodierung der Knotenstruktur, werden in der BIFS Kodierung standardmäßig sehr effizient kodiert.

Sowohl EfficientFloat als auch Quantisierung reduzieren die benötigte Anzahl an Bits zur Kodierung von 3D-Punkten. Während EfficientFloat vor allem beim Enkodieren deutlich mehr Zeit benötigt, liegen die beiden Verfahren beim Dekodieren vom Zeitaufwand näher beieinander. Für dieselben Grenzen beim Kompressionsverlust war die Kompressionsrate für Quantisierung annähernd doppelt so hoch wie für EfficientFloat. Die subjektive Beurteilung der Modellqualität für die Grenzwerte 2^{-5}m und 2^{-7}m fällt bei EfficientFloat besser aus, als für Quantisierung. Dennoch erscheint letzteres Verfahren für die schritthaltende Verarbeitung von Sensordaten besser geeignet zu sein.

Das Predictive MFField Verfahren kann für beide Datentypen angewendet werden. Es hat sich jedoch gezeigt, dass für PMF einige Beschränkungen beachtet werden müssen. So können Indizes nicht für beliebig große Polygonnetze mit dem Kodierschema $m = 1$ kodiert werden. Das gleiche gilt für 3D-Punkte, wenn nur wenig quantisiert wird. Das Kodierschema $m = 3$ könnte Abhilfe schaffen, erreicht jedoch bei weitem nicht die gewünschte Kompressionsrate. PMF verbraucht für die bisher generierten Modelldaten deutlich mehr Rechenzeit. Auf aktueller Hardware konnte keine schritthaltende Verarbeitung mit anschließender Übertragung mit mehr als 7 Bildern pro Sekunde aus der PMD-Kamera realisiert werden. Mit den anderen Verfahren hingegen konnte die Kamera mit ihrer maximalen Bildrate betrieben werden.

Insgesamt betrachtet sind die Indizes schwieriger zu komprimieren, da hier keine Verluste auftreten dürfen. Das dafür vorgesehene `CoordIndex` Werkzeug arbeitet effizient und ohne zusätzliche Laufzeiteinbußen, jedoch übertreffen die Indizes auch nach der Kompression in ihrer Bitgröße stets die 3D-Punkte um ein Vielfaches.

Die folgenden Abschnitte gehen nun darauf ein, wie die BIFS Kompressionswerkzeuge besser ausgenutzt werden können, um höhere Kompressionsraten bei ähnlichen Laufzeiten zu erzielen.

4.4 IFS Splitting

In diesem Abschnitt wird eine Methode vorgestellt, die für das Gesamtmodell die Anzahl der notwendigen Bits zur Kodierung aller `coordIndex` Felder verringert. Diese Methode wird `IndexedFaceSet`-Splitting oder kurz IFS-Splitting genannt, da aus einem `IFS`

Knoten durch Teilungsoperationen mehrere erzeugt werden.

Die Anzahl der notwendigen Bits N_0 um die Indexliste in `coordIndex` zu kodieren beträgt

$$N_0 = n_{i0} \cdot b_0$$

Dabei ist n_{i0} die Anzahl der Elemente in der Indexliste. Im vorliegenden Fall entspricht dies der Anzahl der Dreiecke im `IFS` Knoten mal vier.

$$b_0 = \lceil log_2(n_{v0} + 1) \rceil$$

ist die Anzahl der Bits, die notwendig sind, um einen Wert in der Liste zu kodieren. Siehe hierzu auch Gleichung 3.4. b_0 hängt von der Länge der referenzierten 3D-Punkteliste n_{v0} ab. Der Logarithmus wird von $n_{v0} + 1$ berechnet, da die Trennmarkierung mit dem Indexwert -1 ebenfalls kodiert werden muss. Wird nun das Polygonnetz des `IFS` Knoten derart auf zwei `IFS` Knoten gleichmäßig aufgeteilt, dass in jedem neuen `IFS` Knoten nur die Hälfte der 3D-Punkte benötigt wird, ergibt sich b_1^t für jedes Teilnetz t zu

$$\begin{aligned} n_{v1}^t &= \frac{n_{v0}}{2}, \quad t \in \{1, 2\} \\ b_1^t &= \lceil log_2(n_{v1}^t + 1) \rceil \\ b_0 - 1 &\leq b_1^t \leq b_0 \end{aligned}$$

Nur für den Fall

$$n_{v0} = 2^x - 1, x \in \mathbb{N} \tag{4.1}$$

gilt $b_1^t = b_0$. Gleichzeitig ergibt sich

$$n_{i1}^t = \frac{n_{i0}}{2}, \quad t \in \{1, 2\}$$

da in jedem `IFS` Knoten nur noch die Hälfte der Polygone beschrieben wird. Da jedoch nun zwei `IFS` Knoten kodiert werden, ergeben sich hier insgesamt gesehen keine Ersparnisse. Angenommen Gleichung 4.1 trifft nicht zu, dann beträgt die nunmehr benötigte Bitanzahl N_1 zur Kodierung der Indexlisten:

$$\begin{aligned} N_1 &= \sum^{t \in \{1,2\}} n_{i1}^t \cdot b_1^t \\ &= 2 \cdot \frac{n_{i0}}{2} \cdot (b_0 - 1) \\ &= n_{i0} \cdot (b_0 - 1) \end{aligned}$$

Die Idee hinter IFS-Splitting ist nur dann realisierbar, wenn das Netz eines `IFS` Knoten in voneinander unabhängige Teilnetze zerlegt werden kann. Meistens müssen dazu einige 3D-Punkte in beiden `IFS` Knoten abgelegt werden. Dadurch werden allerdings wieder mehr Bits für die Kodierung der 3D-Punkteliste benötigt. Für den Fall der Einzelbild-Rekonstruktion ist eine Netzteilung bereits auf einfache Art durch die Teilung des Tiefenbildes möglich. Die Tiefenbildpunkte, die auf den Teilungslinien liegen, müssen jeweils

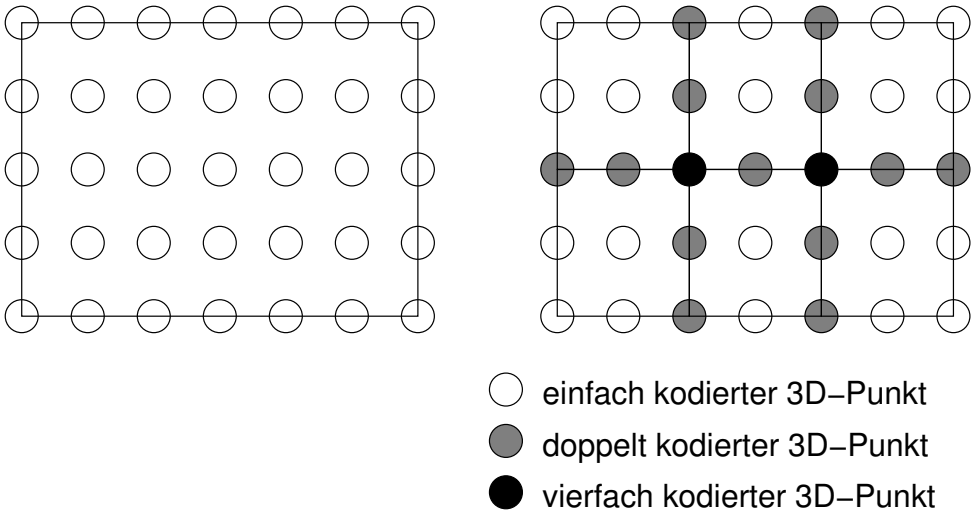


Bild 4.9: Beispiel der Aufteilung der 3D-Punkte für die Einzelbild-Rekonstruktion

in allen angrenzenden Teilnetzen kodiert werden. Für den Raumaufteilungs-Algorithmus ist IFS-Splitting prinzipiell genauso möglich, im Weiteren wird es jedoch im Kontext der Einzelbild-Rekonstruktion betrachtet.

Bild 4.9 zeigt die Aufteilung am Beispiel eines Tiefenbildes mit 7x5 Bildpunkten, das für die Erzeugung von $T = 6$ `IFS` Knoten zerlegt wird. Dadurch müssen 13 3D-Punkte in zwei `IFS` Knoten und zwei 3D-Punkte in vier `IFS` Knoten kodiert werden. Die Teilung der Netze hängt sehr stark von der Anordnung und Verwendung der 3D-Punkte ab, weshalb an dieser Stelle kein allgemeiner Algorithmus für die Unterteilung angegeben wird. Für die Einzelbild-Rekonstruktion gilt jedoch, dass möglichst quadratische Teilbereiche erzeugt werden sollten, um die Anzahl der mehrfach kodierten 3D-Punkte gering zu halten.

Bis zu welchem Punkt eine Unterteilung zu einem Kodierungsgewinn führt, hängt von vielen Faktoren ab. Zum Ersten sind das die Kosten, die durch die Kodierung eines zusätzlichen `IFS` Knoten entstehen. Diese Kosten liegen bei 61 Bit pro `IFS` Knoten[3)]. Für jeden neuen `Coordinate` Knoten werden zusätzlich 44 Bit benötigt. Da in einen `Shape` Knoten nicht mehrere `IFS` Knoten eingetragen werden können, muss ebenfalls ein zusätzlicher `Shape` Knoten mit 11 Bit kodiert werden. Die entstehenden Kosten pro mehrfach kodiertem 3D-Punkt hängen vom verwendeten Kompressionsverfahren ab. Ausgehend von einer Quantisierung mit 10 Bit verbraucht jeder kodierte 3D-Punkt 30 Bit. Bei diesen Annahmen entstehen folgende Kosten:

$$Kosten = (T-1)(11\,bit + 61\,bit + 44\,bit) + r \cdot 30\,bit = 1\,150\,bit$$

Hier ist T die Anzahl der Teilnetze und r gibt die Gesamtanzahl der mehrfach kodierten 3D-Punkte an. Im vorliegenden Beispiel ist $T = 6$ und $r = 19$. Demgegenüber steht der

[3)] 22 zusätzliche Bit für den Knoten plus maximal 39 zusätzliche Bit für sein `coordIndex` Feld. Für einen `Coordinate` Knoten: 5 Bit für den Knoten plus maximal 39 Bit für sein `point` Feld. In Anhang C.2 wird der Bitbedarf für die Kodierung einiger Beispiele ausführlicher dargestellt.

Bitgewinn durch kürzer kodierte Indexlisten in jedem `IFS` Knoten:

$$Gewinn = \sum_{t=1}^{T} n_{i1}^{t}(b_0 - b_1^t) = 6(32(6\,bit - 4\,bit)) = 384\,bit$$

Offensichtlich entstehen durch die Teilung mit $T = 6$ zu viele Kosten. Eine alternative Teilung, durch die das Netz horizontal in $T = 2$ Teilnetze zerlegt wird, führt zu anderen Zahlen:

$$\begin{aligned} Kosten &= 1 \cdot 116\,bit + 5 \cdot 30\,bit = 266\,bit \\ Gewinn &= 2(96(6\,bit - 5\,bit)) = 192\,bit \end{aligned}$$

Bei dieser Rechnung überwiegen zwar noch die Kosten, jedoch ist der Unterschied zum Gewinn nicht mehr so hoch. Wird dieselbe Rechnung für eine horizontale Zweiteilung nochmals für ein Tiefenbild der Größe 160x120 Pixel durchgeführt, wächst der Gewinn im Vergleich zu den Kosten stärker an:

$$\begin{aligned} Kosten &= 1 \cdot 116\,bit + 120 \cdot 30\,bit = 3\,716\,bit \\ Gewinn &= 2(75\,208(15\,bit - 14\,bit)) = 150\,416\,bit \end{aligned}$$

Für das große Tiefenbild führt eine Teilung mit $T = 6$ nach dem gleichen Schema wie in Abbildung 4.9 wiederum zu einem noch stärkeren Kodierungsgewinn:

$$\begin{aligned} Kosten &= 5 \cdot 116\,bit + 402 \cdot 30\,bit = 12\,640\,bit \\ Gewinn &= 4(24\,544(15\,bit - 12\,bit)) + \\ &\quad 2(25\,016(15\,bit - 12\,bit)) = 444\,624\,bit \end{aligned}$$

Hier zeigt sich bereits, dass eine uniforme Teilung über das gesamte Tiefenbild nicht erfolgen kann, da sich die Berechnung des Gewinns auf zwei Terme für unterschiedlich große Teilnetze aufteilt. Lässt man praktische Aspekte außer Acht, kann eine Funktion $f(n_{v0}, n_{v1}^t, q)$ für den erwarteten Bitbedarf zur Kodierung eines Dreiecksnetzes aufgestellt werden. Bekannte Ausgangsgröße ist die Anzahl n_{v0} der 3D-Punkte, auf denen ein gleichmäßiges, geschlossenes Dreiecksnetz rekonstruiert wird. Wenn n_{v1}^t die Anzahl an 3D-Punkten für jedes Teilnetz bezeichnet, so wird derjenige Wert von n_{v1}^t gesucht, der die Kodierungsgröße minimiert. Eine Quantisierung mit q Bit pro Vektorkomponente wird vorausgesetzt. Die Anzahl der gewünschten Teilnetze ergibt sich zu

$$T = \frac{n_{v0}}{n_{v1}^t}$$

Die Anzahl der mehrfach kodierten 3D-Punkte lässt sich über

$$r = \sqrt{n_{v0}} \cdot 2(\sqrt{T} - 1) + 2(\sqrt{T} - 1)^2$$

abschätzen. Ausgangspunkt für r ist die Annahme, das Tiefenbild hätte quadratische Größe und würde $\sqrt{T} - 1$ mal geteilt werden, sowohl jeweils horizontal als auch vertikal.

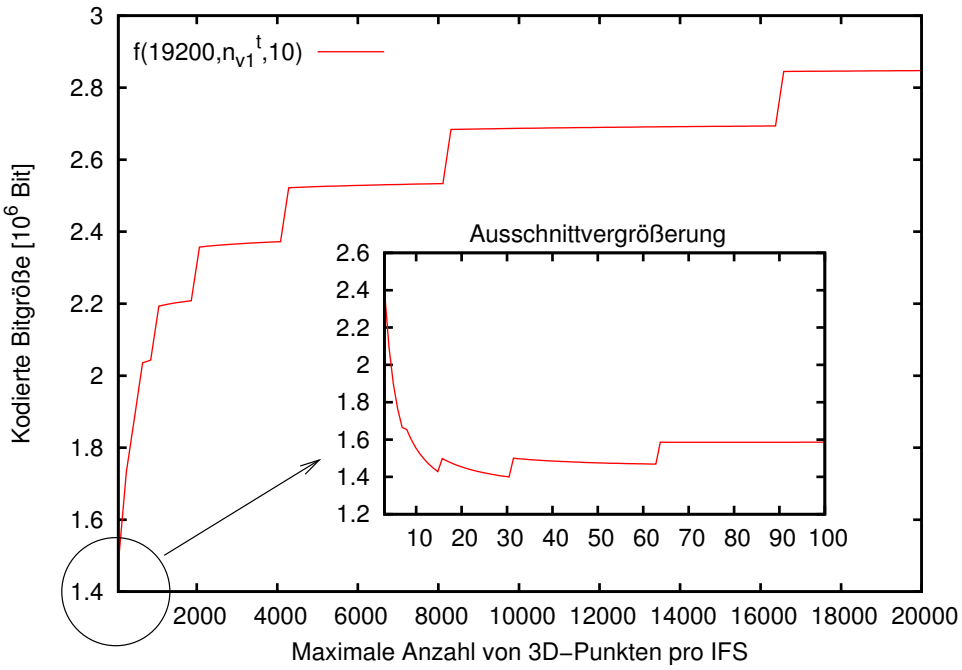


Bild 4.10: Geschätzter Bitaufwand für die Kodierung eines 160x120 Pixel Tiefenbildes über IFS-Splitting mit variierender Teilung, $q = 10$ Bit

Der erste Term der Summe modelliert dann die Anzahl der doppelt kodierten 3D-Punkte. Die vierfach kodierten 3D-Punkte werden über den zweiten Term hinzugefügt. Die gesuchte Funktion lautet dann:

$$\begin{aligned} f(n_{v0}, n_{v1}^t, q) \quad = \quad & T \cdot (11\,bit + 61\,bit + 44\,bit) + \\ & (n_{v0} + r) \cdot 3q + \\ & 8T \cdot (\sqrt{n_{v1}^t} - 1)^2 \cdot \lceil log_2(n_{v1}^t + 1) \rceil \end{aligned}$$

Abbildung 4.10 zeigt den Kurvenverlauf für $f(n_{v0} = 19200, n_{v1}^t, q = 10)$. Für sehr viele Teilungen wird das Konzept ad absurdum geführt. Bereits die obigen Rechenbeispiele haben einen Hinweis darauf gegeben. Wie in Abbildung 4.10 in der Ausschnittvergrößerung zu sehen ist, wird das IFS-Splitting Konzept laut der gewählten Modellierung bei sehr niedrigen Werten von n_{v1}^t ineffizient. Der folgende Abschnitt beschreibt, wie gut sich Theorie mit Praxis vereinen lässt.

Messergebnisse

Da sich IFS-Splitting auf die Anzahl der zu kodierenden 3D-Punkte, die Anzahl der verwendeten Knoten und auf die Kodierung der Indizes auswirkt, werden die folgenden Messergebnisse immer für die kodierte Größe des Gesamtmodells angegeben. Kodiert wurde wieder der in 4.1.1 beschriebene Datensatz von 12 Tiefenbildern.

Im folgenden Diagramm wird die reine Quantisierung mit PMF Modus 1 verglichen. Später folgt der Vergleich mit PMF Modus 1+3. Betrachtet wird jeweils die Größe der binären Kodierung, aufgetragen über verschieden viele Teilungen. Letztere werden durch die maximal zugelassene Anzahl von 3D-Punkten in einem `IFS` Knoten ausgedrückt. Anders als bei der theoretischen Modellierung müssen hier praktische Aspekte berücksichtigt werden. Deswegen können nicht immer Teilungen erzeugt werden, bei denen für alle `IFS` Knoten ein identischer Werte n_{v1}^t gilt. Beim Algorithmus für die Teilung wurde jedoch darauf geachtet, möglichst quadratische Teilnetze zu erzeugen, um die Anzahl der mehrfach kodierten 3D-Punkte niedrig zu halten. Ein Vergleich mit EfficientFloat wurde hier ausgelassen, da ähnliche Kurvenverläufe wie bei Quantisierung zu erwarten sind. Wenn für die folgenden Diagramme PMF gemessen wurde, hat die Implementierung für jedes MF-Feld erst überprüft, ob eine Kodierung mit dem gewünschten PMF Modus möglich ist. Traf dies nicht zu, wurde nur Quantisierung verwendet.

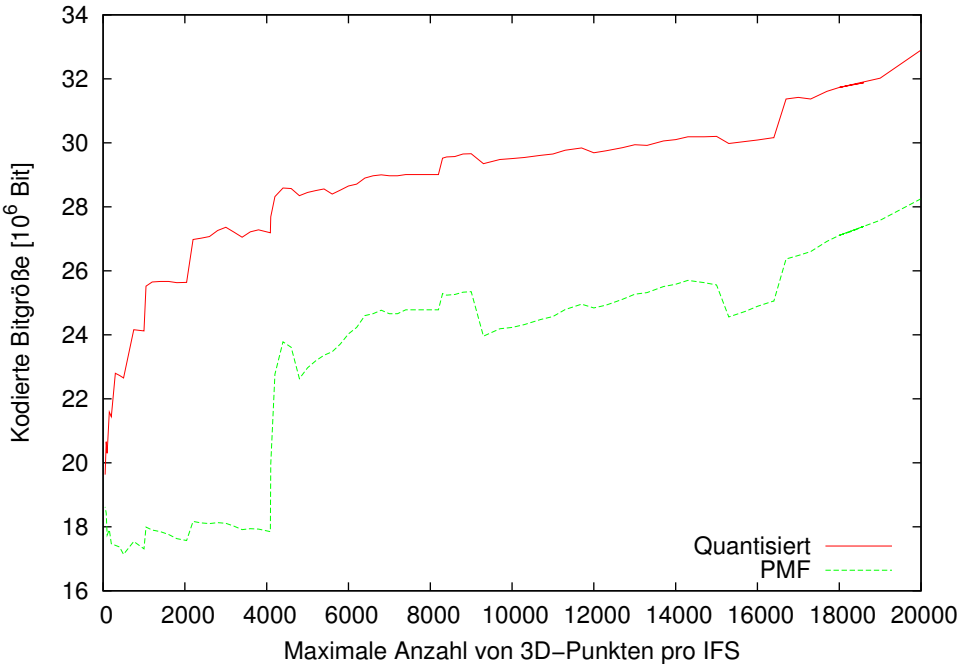


Bild 4.11: Auswirkung von IFS Splitting auf die kodierte Bitgröße, Quantisierung im Vergleich zu PMF Modus 1, Genauigkeit 2^{-9} m

Abbildung 4.11 zeigt die erzielten Bitgröße des Gesamtmodells bei verschiedenen Teilungen für Quantisierung und PMF Modus 1. Der Kurvenverlauf für Quantisierung stimmt sehr gut mit dem der theoretischen Modellierung überein. Die erwarteten Sprünge durch Biteinsparungen bei der Indexkodierung können nachvollzogen werden. Man sieht auch, dass der PMF Modus 1 bei gleicher Teilung des Tiefenbildes stets weniger Bits benötigt als das Quantisierungs-Werkzeug. Ein deutlicher Sprung bei PMF ist bei ca. 4000 3D-Punkten pro `IFS` zu sehen. Die Erklärung für diesen Effekt ist in Abbildung 4.12 zu sehen. Während für alle Teilungen immer alle 3D-Punkte mit PMF Modus 1 kodiert werden können, schwankt die Anzahl der mit PMF Modus 1 kodierten Indizes. Für die

3D-Punkte ist die Situation einfach, weil bei der gewählten, konstanten Quantisierung die maximal mögliche PMF Symbolmodellgröße von 2^{13} Bit ausreicht. Unterhalb von 4095 3D-Punkten pro `IFS` kann auch für jedes Indexfeld PMF Modus 1 verwendet werden, da nun auch hier ein Symbolmodell mit 2^{13} Symbolen ausreicht.

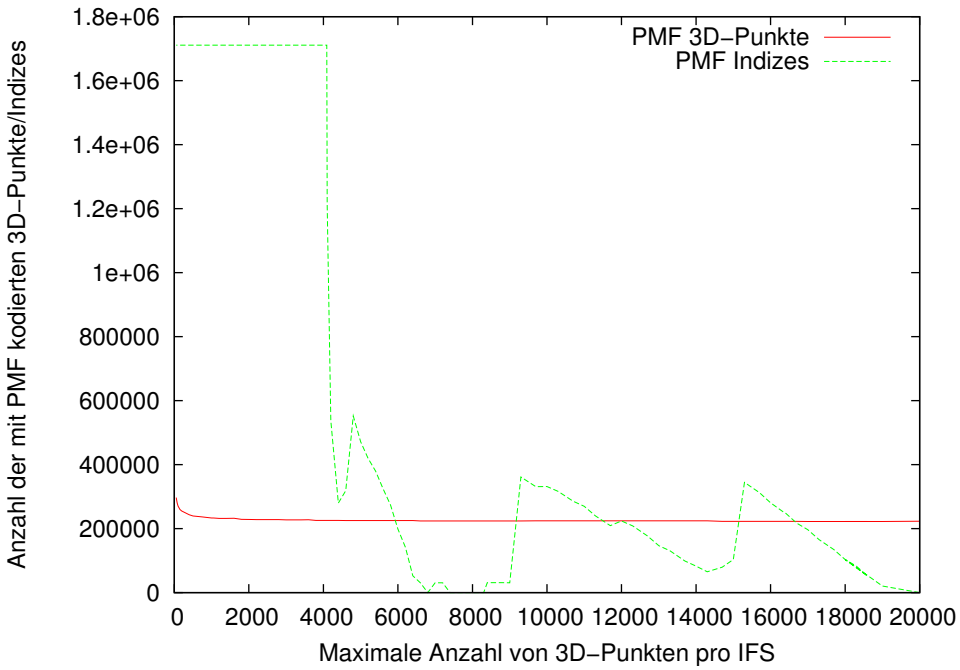


Bild 4.12: Anzahl der durch PMF Modus 1 kodierten Elemente

Werden mehr 3D-Punkte pro `IFS` Knoten zugelassen, kann ein großer Teil der Indexlisten nicht mehr über PMF kodiert werden. Nur noch die Randbereiche des Tiefenbildes, die vom Teilungsalgorithmus nicht mehr exakt auf die gewünschte Größe geschnitten werden konnten, werden mit PMF kodiert. Der Teilungsalgorithmus verursacht mit seiner Teilungsstrategie den sägezahnartigen Verlauf der Kurve in Abbildung 4.12. Die kleinen Spitzen in Abbildung 4.12 verursachen auch leichte Schwankungen in Abbildung 4.11.

Abbildung 4.13 zeigt das Laufzeitverhalten der Kompressionswerkzeuge. Sowohl Enkodierung als auch Dekodierung mit dem Quantisierungswerkzeug bewegt sich für alle Teilungen durchgängig im Größenbereich von 100 ms. Nur bei sehr hohen Teilungen, d. h. bei sehr vielen `IFS` Knoten steigt die Verarbeitungszeit deutlich an. Die PMF Kodierung ist durchgängig um ein Vielfaches langsamer. Da nicht immer alle Indizes über PMF kodiert werden, schwankt die Verarbeitungszeit mit der Anzahl der kodierten Indizes. Im Bereich unter 4 000 3D-Punkten pro `IFS` Knoten jedoch sinkt die Verarbeitungszeit wieder, was auf die Verwendung von kleineren Symbolmodellen zurückzuführen ist. Je weniger Symbole im Modell verwaltet werden, desto schneller arbeitet der adaptive arithmetische Kodierer.

Der PMF Modus 1+3 verhält sich wesentlich gutmütiger bei unterschiedlichen Teilungen. Da hier die Kodierbarkeit der Indizes über PMF nicht mehr von der Gesamtzahl der 3D-

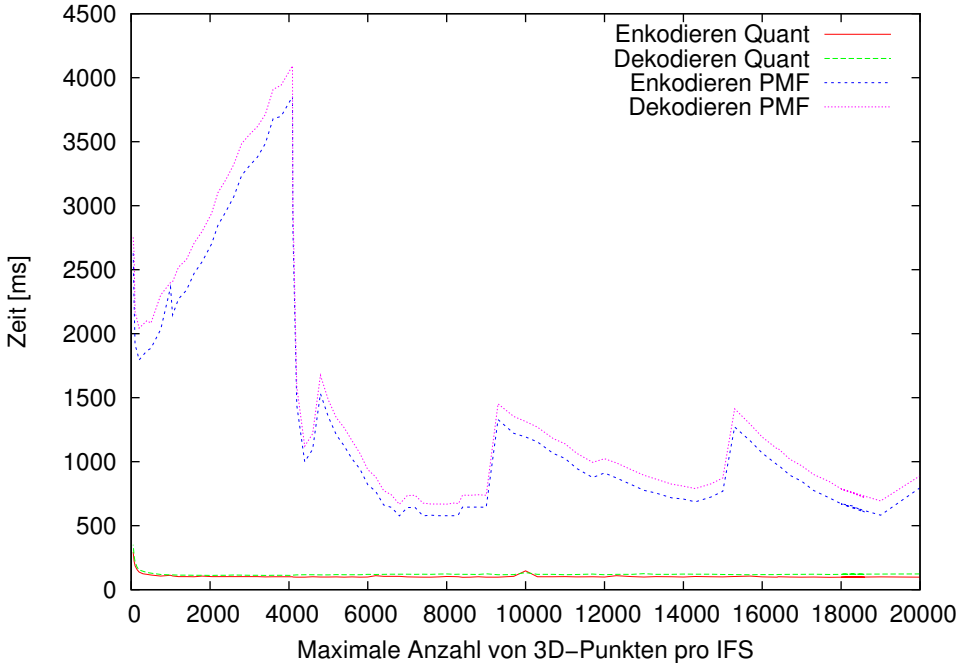


Bild 4.13: Auswirkung von IFS Splitting auf die Laufzeiten, Quantisierung im Vergleich zu PMF Modus 1, Genauigkeit 2^{-9} m

Punkte im `IFS` Knoten abhängt, werden hier für alle Teilungen immer alle Elemente über PMF Modus 1+3 kodiert.

Die Abbildungen 4.14 und 4.15 wiederholen den obigen Vergleich für Quantisierung gegen PMF Modus 1+3 bei unterschiedlichen Teilungen. Man sieht, dass die erreichbare Kompressionsrate bei PMF Modus 1+3 unter 4 000 3D-Punkten pro `IFS` geringer ausfällt. Bei höheren Teilungen bleibt sie jedoch trotz IFS Splitting relativ konstant und niedriger als bei PMF Modus 1. Bei den Laufzeiten fällt vor allem die stets deutlich höhere Dekodierzeit auf, die bereits in Abschnitt 4.2.3 erklärt wurde.

Insgesamt lässt sich behaupten, dass durch IFS Splitting die benötigte Bitanzahl zur Kodierung der Index-Felder verringert werden kann. Dieser Effekt überwiegt, zumindest in dem hier gewählten Rahmen, die zusätzlichen Kosten der mehrfach kodierten 3D-Punkte und zusätzlichen Knotenstrukturen. Die für die Kodierung benötigte Bitanzahl lässt sich über die vorgestellte Näherung recht gut schätzen. Für das PMF Kodierschema $m = 1$ ist IFS Splitting weiterhin eine wirkungsvolle Maßnahme, um den Einsatz dieses Verfahrens für alle `IFS` Knoten zu ermöglichen. Auf den PMF Modus 1+3 hat IFS Splitting wenig Einfluss. Bei sehr starker Teilung erreicht die normale Quantisierung sogar ähnliche Kompressionsraten, bei wesentlich geringeren Laufzeiten. Diese Beobachtung verstärkt den Eindruck, dass das PMF Kodierschema $m = 3$ für die Kompression von Geometrie nur wenig geeignet ist.

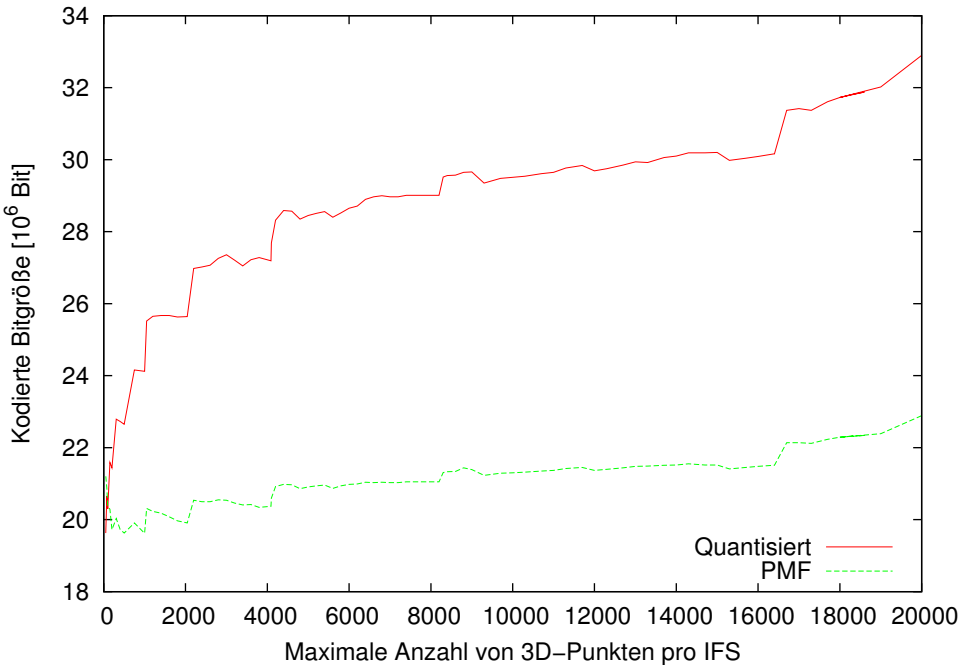


Bild 4.14: Auswirkung von IFS Splitting auf die kodierte Bitgröße, Quantisierung im Vergleich zu PMF Modus 1+3, Genauigkeit 2^{-9} m

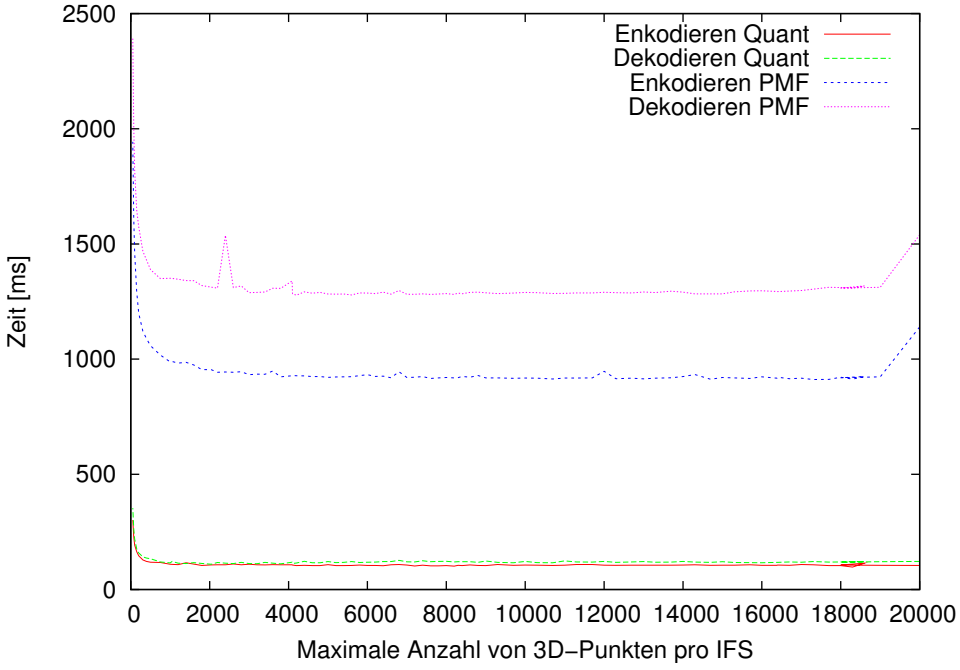


Bild 4.15: Auswirkung von IFS Splitting auf die Laufzeiten, Quantisierung im Vergleich zu PMF Modus 1+3, Genauigkeit 2^{-9} m

4.5 Modellgenerierung bei quantisierter Kodierung

Ist bereits bei der Modellrekonstruktion (siehe Kapitel 4.1) bekannt, dass für die Übertragung das Quantisierungswerkzeug verwendet wird, ergeben sich neue Möglichkeiten für Einsparungen. Wenn durch die Quantisierung mehrere berechnete 3D-Punkte auf identische Raumkoordinaten abgebildet werden, muss von diesen 3D-Punkten nur ein einziger tatsächlich kodiert werden. Diejenigen Dreiecke, deren Eckpunkte durch die Quantisierung zusammenfallen, nehmen degenerierte Formen an. Als degenerierte Form wird in diesem Zusammenhang bezeichnet, wenn das Dreieck nur noch als Linie oder Punkt wahrgenommen werden kann. Diese Dreiecke müssen ebenfalls nicht kodiert werden.

Die beiden in Abschnitt 4.1 vorgestellten Rekonstruktionsalgorithmen wurden um einen Modus erweitert, der die genannten Einsparung vornimmt. Für die Einzelbild-Rekonstruktion sind die Ergebnisse in den Abbildungen 4.16 und 4.17 zu sehen. Für hohe Genauigkeitsanforderungen erzeugt der modifizierte Algorithmus identische Modellstrukturen, da hier offensichtlich die örtliche Auflösung des Tiefenbildes zu gering ist. Sinkt die Genauigkeitsanforderung unter eine bestimmte Schwelle, beginnt der Algorithmus deutlich weniger 3D-Punkte und Indizes zu erzeugen. Für den Testdatensatz lag die Schwelle bei 8 mm. Die Abnahme der kodierten 3D-Punkte und Dreiecke macht sich sowohl bei der Anzahl der benötigen Bits, als auch bei der Zeitmessung zur Kodierung positiv bemerkbar. Der Algorithmus zur Einzelbild-Rekonstruktion im quantisierten Modus selber braucht bei hohen Genauigkeitsanforderungen nur unwesentlich länger zur Ausführung. Bei niedrigen Genauigkeitsanforderungen sinkt seine Ausführungszeit sogar etwas unter die des unmodifizierten Algorithmus.

Für den Raumaufteilungs-Algorithmus kommt durch die Quantisierung eine weitere wirkungsvolle Einsparmöglichkeit zustande, da nun innerhalb eines Raumwürfels nur noch eine endliche Menge von Positionen für 3D-Punkte möglich ist. Es ist daher ausreichend, einen einzigen `Coordinate` Knoten zu instanzieren, in dem alle möglichen 3D-Punkte hinterlegt sind. Jeder Raumwürfel kann über Verweisknoten diesen `Coordinate` Knoten verwenden, der nur ein einziges Mal übertragen werden muss. Für jeden neu angelegten Raumwürfel müssen nun anstatt eines eigenen `Coordinate` Knoten nur noch wenige Bits für einen Verweisknoten übertragen werden[4)].

Einziger Nachteil dieses Verfahrens ist, dass bei hohen Genauigkeitsanforderungen der `Coordinate` Knoten sehr viele 3D-Punkte beinhaltet und deshalb auch viele Bits zur Kodierung benötigt. Bei einer Genauigkeit von $2^{-9}\,\mathrm{m}$ und einer Würfelgröße von $0,3\,\mathrm{m}$ müssen mindestens

$$\left\lceil \frac{0,3\,m}{2^{-9}\,m} \right\rceil^3 = 154^3 = 3\,652\,264$$

3D-Punkte mit jeweils 8 Bit kodiert werden, was im Datenstrom zu einer Bitanzahl von $87,7 \cdot 10^6$ bit führt. Die Implementierung des Algorithmus für quantisierte Raumaufteilung benötigt sogar noch mehr 3D-Punkte, da nicht nur der Bereich innerhalb des eigentlichen

[4)] In Anhang C.2 wird gezeigt, dass dies maximal 32 Bit sind.

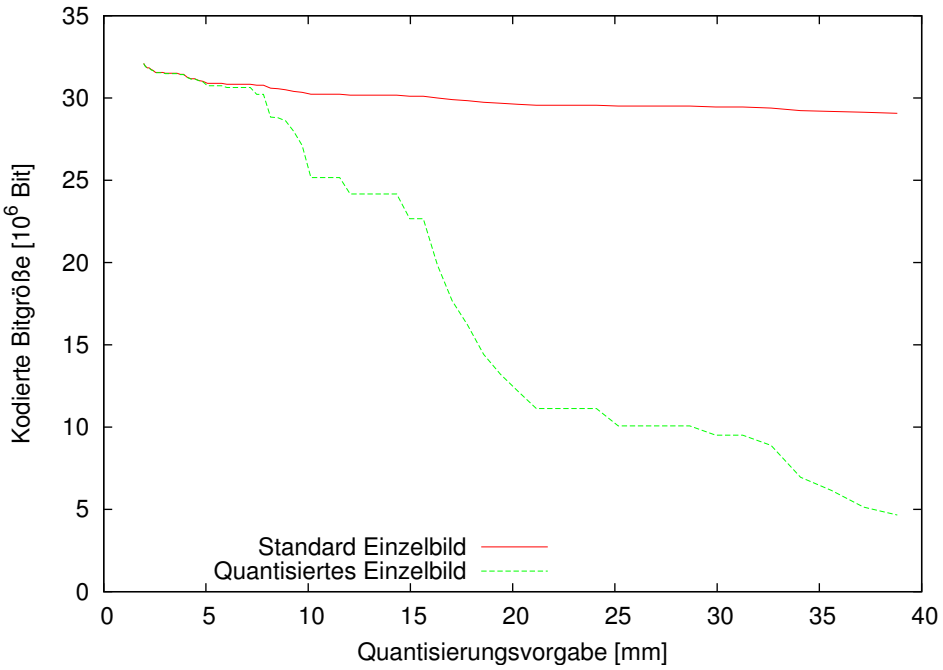


Bild 4.16: Kodierte Gesamtmodell-Bitgröße bei quantisierter Einzelbild-Rekonstruktion, Genauigkeit 2^{-9} m

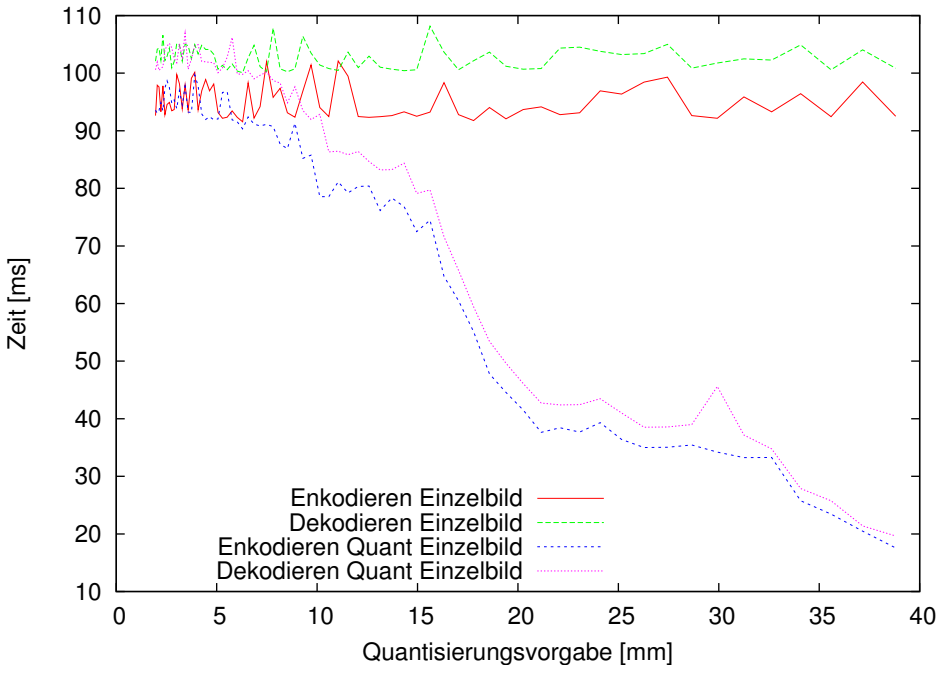


Bild 4.17: Dekodier- und Enkodierzeiten bei quantisierter Einzelbild-Rekonstruktion, Genauigkeit 2^{-9} m

Raumwürfels mit 3D-Punkten ausgefüllt sein muss, sondern auch ein paar 3D-Punkte außerhalb. Die 3D-Punkte außerhalb werden benötigt, damit Dreiecke, die den Raumwürfel verlassen, ebenfalls beschrieben werden können. In der Praxis werden für die oben angegebenen Zahlen $(154+16)^3 = 4\,913\,000$ 3D-Punkte verwendet. Das entspricht im Datenstrom $117,9 \cdot 10^6$ bit.

Für niedrigere geforderte Genauigkeiten fällt diese Zahl niedriger aus. Außerdem muss der `Coordinate` Knoten nur bei jedem `SceneReplace` BIFS-Update Kommando übertragen werden. Danach folgende Netzbeschreibungen in `IFS` Knoten erfordern nur noch wesentlich geringere Datenmengen. Für die Kodierung der 3D-Punkte kann hier außerdem das PMF Werkzeug sehr effizient und elegant eingesetzt werden.

Die Reihenfolge, mit der die Positionen der 3D-Punkte innerhalb des `Coordinate` Knoten beschrieben werden, kann beliebig gewählt werden. Dem Algorithmus muss nur bekannt sein, welcher Index zur Beschreibung einer bestimmten Raumposition im Raumwürfel benützt werden muss. Für die Kodierung des `Coordinate` Knoten mit dem Predictive MFField Verfahren bietet sich hier das IPPIPP-Schema an (m=2, siehe Kapitel 3.3.3), wenn die 3D-Punkte wie nach Abbildung 4.18 in regelmäßiger Reihenfolge angeordnet sind.

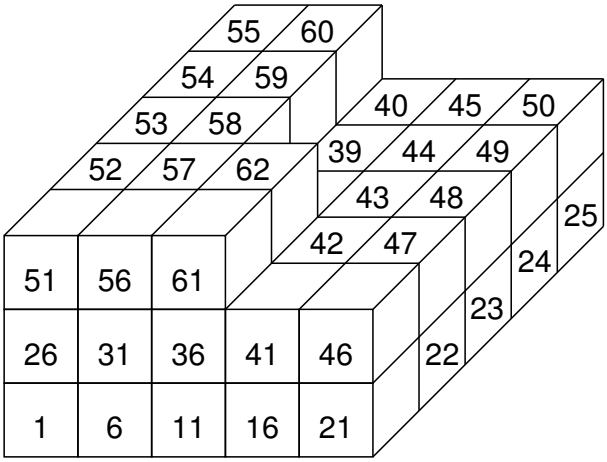


Bild 4.18: Für das PMF Kodierschema $m = 2$ günstige Anordnung und Reihenfolge der 3D-Punkte innerhalb eines `Coordinate` Knoten. Die 3D-Punkte sind als numerierte Quader symbolisiert.

Die 3D-Punkte werden dazu, einer nach dem anderen, entlang der ersten Koordinatensystemachse angeordnet. Wird die Begrenzung des Raumwürfels erreicht, wird entlang der zweiten Koordinatensystemachse, wieder am Anfang der ersten Koordinatensystemachse eine neue Reihe begonnen. Die Regel wiederholt sich mit der letzten Koordinatensystemachse, wenn die Begrenzung der zweiten Achse erreicht wird. Diese Folge wird solange fortgeführt, bis der Raumwürfel vollständig mit 3D-Punkten belegt ist. Durch diese Sortierung unterscheiden sich die meisten 3D-Punkte nur durch eine Änderung um den quantisierten Wert 1 voneinander. Jede neue Reihe verursacht eine größere Änderung,

jedoch treten diese größeren Änderungen nun in regelmäßigen Abständen auf. Die kleinen Änderungen innerhalb einer Reihe können mit dem kleinstmöglichen Symbolmodell (2^1 Symbole) sehr effizient als P-Werte arithmetisch kodiert werden. Die größeren Änderungen können in regelmäßigem Intervall durch I-Werte ausgedrückt werden. Genau diese Abfolge wird durch das PMF Kodierschema $m = 2$ in Kapitel 3.3.3 vorgegeben.

Durch Einsatz des PMF Kodierschema $m = 2$ werden im oben genannten Beispiel nur noch $0,78 \cdot 10^6$ bit für die Kodierung der 3D-Punkte benötigt. Das entspricht einer Kompressionsrate von 151,2. Anders gerechnet werden pro 3D-Punkt nur noch 0,159 Bit benötigt. Ein gewisser Teil dieser Ersparnis wird jedoch durch erhöhte Kosten für die Kodierung der Indizes wieder verbraucht. Aufgrund des `CoordIndex` Verfahrens sind wesentlich mehr Bits für die Kodierung eines Indexwerts notwendig, da eine sehr hohe Anzahl von 3D-Punkten im `Coordinate` Knoten referenziert werden kann. Im angegebenen Beispiel müssen 23 Bit pro Index verwendet werden, wohingegen beim Standardalgorithmus nur 11 Bit notwendig waren.

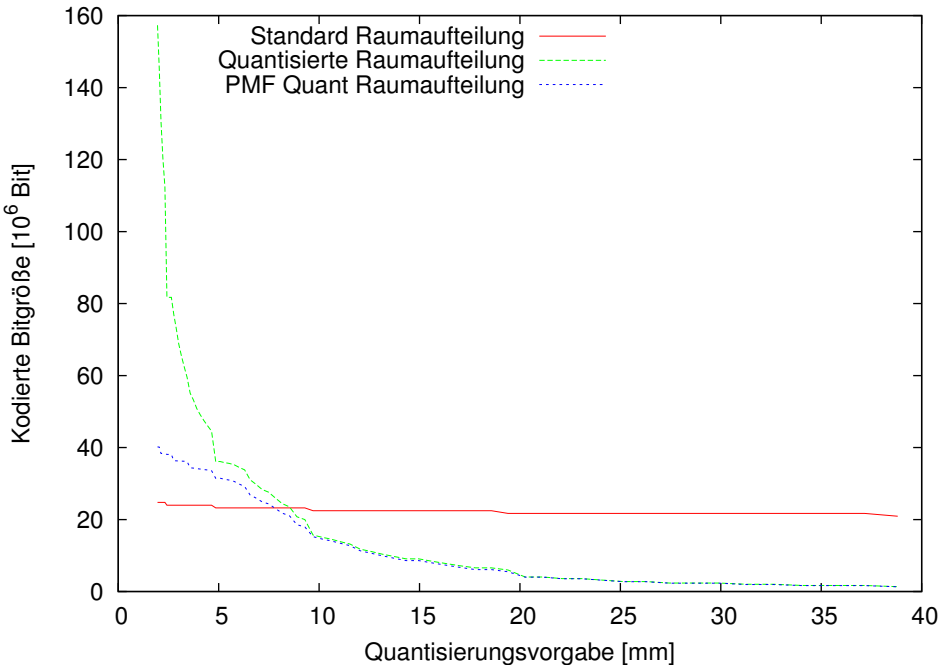


Bild 4.19: Kodierte Gesamtmodell-Bitgröße bei quantisierter Raumaufteilung (mit und ohne PMF) mit einer Würfelgröße von 0,3 m im Vergleich zur Standardraumaufteilung, Genauigkeit 2^{-9} m

In den Abbildungen 4.19 und 4.20 wird die modifizierte Raumaufteilung, mit und ohne PMF Kodierschema $m = 2$, für die Kodierung der 3D-Punkte gegen ihre ursprüngliche Implementierung verglichen. Die Würfelgröße wurde für alle Messungen auf 0,3 m festgelegt.

Es fällt auf, dass für hohe Genauigkeitsanforderungen die Modifikation des Raumaufteilungs-Algorithmus weder in Bezug auf Bitgröße, noch bei der Laufzeit Vorteile bringt.

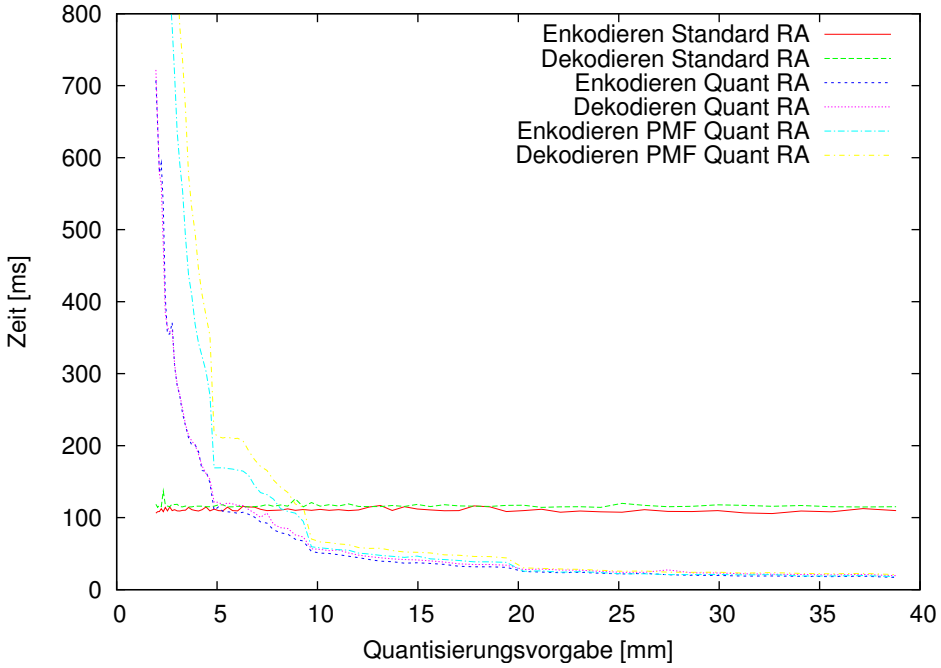


Bild 4.20: Dekodier- und Enkodierzeiten für quantisierte Raumaufteilung (mit und ohne PMF) mit einer Würfelgröße von 0,3 m im Vergleich zur Standardraumaufteilung, Genauigkeit 2^{-9} m

Auch die Verwendung des PMF Kodierschemas $m = 2$ ändert das nicht. Für die 3D-Punkte kann PMF zwar eine sehr hohe Kompressionsrate erzielen, die jedoch auch sehr lange Laufzeiten mit sich bringt. Dabei muss aber auch in Betracht gezogen werden, dass die 3D-Punkte nur für jedes `SceneReplace` BIFS-Update Kommando übertragen werden. Im weiteren Verlauf der Umgebungsrekonstruktion werden nur noch Indizes übertragen. In Abbildung 4.21 ist dargestellt, welche Datenmenge alleine für die Kodierung der Indizes benötigt wird. Der dort dargestellte Kurvenverlauf bestätigt, dass fast ausschließlich die Kodierung der Indizes den Verlauf der anderen Kurven bestimmt.

Je niedriger die Genauigkeitsanforderung gestellt wird, desto weniger 3D-Punkte müssen kodiert werden. Für den Testdatensatz ergibt das ab ca. 10 mm maximal zugelassenem Kompressionsverlust deutliche Vorteile für die quantisierte Raumaufteilung. Das PMF Werkzeug kann zwar die wenigen 3D-Punkte weiterhin sehr gut komprimieren, nur wirkt sich dieser Vorteil im Vergleich zur Kodierung der Indizes sehr schwach aus.

Das PMF Kodierschema $m = 2$ ist ein ausgezeichnetes Werkzeug zur Kodierung der 3D-Punkte für die quantisierte Raumaufteilung, jedoch bleibt nach wie vor das Hauptproblem die Kodierung der Indizes. Ähnlich wie bei IFS-Splitting werden nur dann weniger Bits für die Kodierung benötigt, wenn die Anzahl von 3D-Punkten pro Raumwürfel sinkt. Dies kann entweder, wie hier dargestellt, über eine gröbere Quantisierung, oder über eine Verkleinerung der Raumwürfel erreicht werden. Für die hier verwendete Raumwürfelgröße von 0,3 m und einen maximalen Kompressionsverlust von 2^{-5} m erreicht der quantisierte

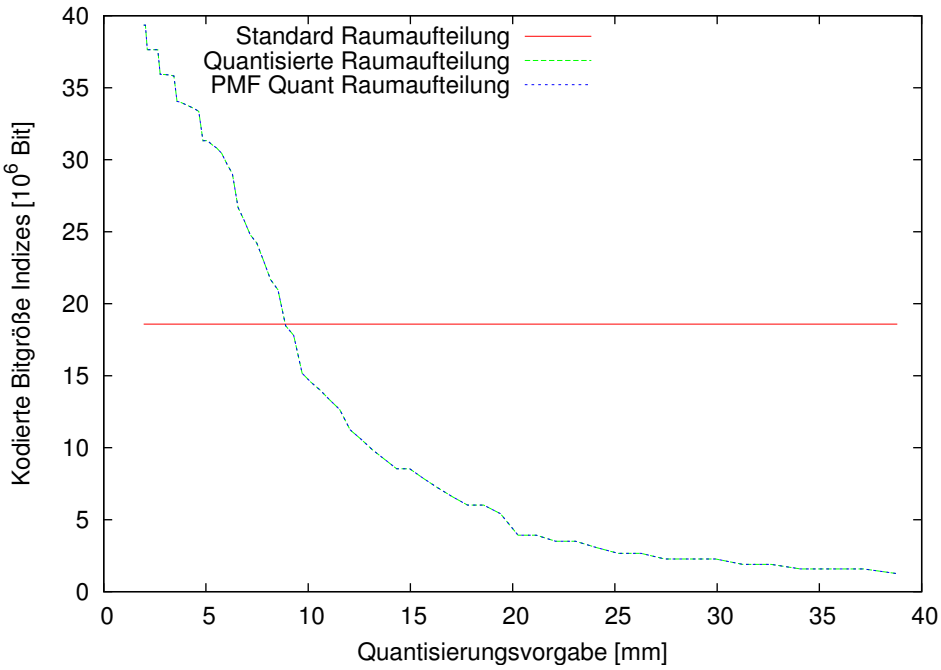


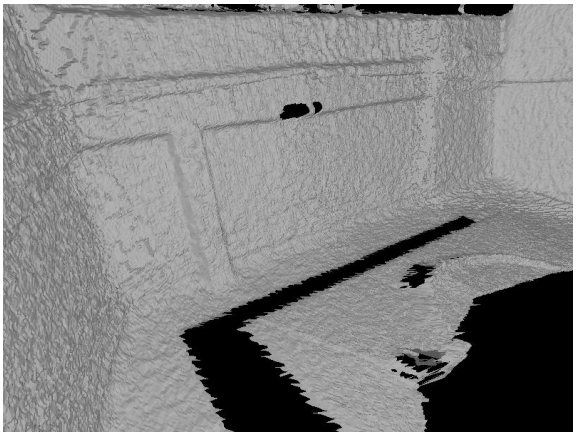
Bild 4.21: Bits die zur Kodierung von Indizes verwendet werden, Vergleich quantisierte zu Standard-Raumaufteilung

Raumaufteilungs-Algorithmus im Rahmen dieser Arbeit die besten Kompressionsergebnisse.

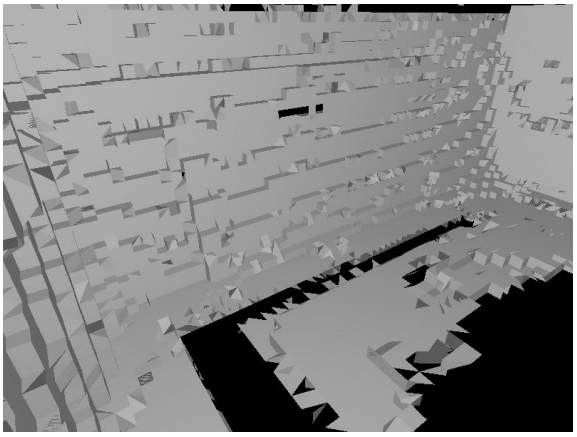
In Abbildung 4.22 wird eine Beispielansicht des Testdatensatzes gezeigt. Das unkomprimierte Modell (4.22(a)) zeigt wesentlich mehr Details als das sehr stark komprimierte Modell (4.22(b)). Die Kompressionsrate zwischen den beiden dargestellten Modellen beträgt allerdings auch $57,9 : 1$. Wird das Modell dem Operator mit Texturen angezeigt, so wie es in den Abbildungen 4.22(c) und 4.22(d) zu sehen ist, macht sich die wesentlich gröbere Modellierung nicht mehr so stark bemerkbar.

4.6 Zusammenfassung und Vergleich der Werkzeuge

In diesem Kapitel wurden einige Werkzeuge und Verfahren zur effizienten Kodierung von Geometriedaten vorgestellt. Für die Erstellung der Geometriedaten werden mit der Einzelbild-Rekonstruktion und dem Raumaufteilungs-Algorithmus zwei Alternativen beschrieben. Beide Verfahren können aus Tiefenbildern 3D-Modelle der entfernten Umgebung des Teleoperators erstellen. Anhand eines Testdatensatzes von 12 Tiefenbildern wurden verschiedene Kompressionswerkzeuge des MPEG-4 BIFS Standards verglichen. Um die Wirkung dieser Werkzeuge besser auszunutzen, wurde weiterhin vorgestellt, wie die genannten Algorithmen zur Modellerstellung modifiziert werden können.

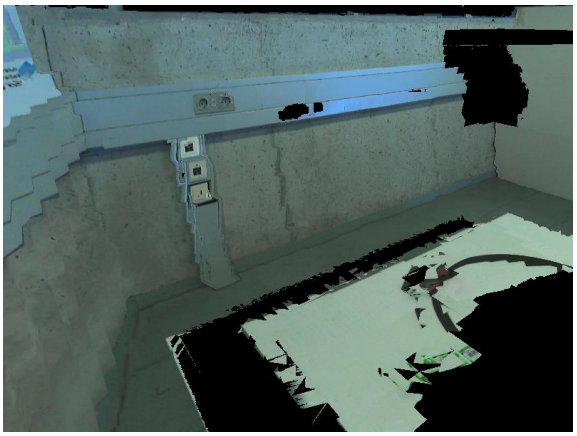


(a) Unkomprimiert: 9,06 MByte



(b) Raumaufteilung unter Berücksichtigung der Quantisierung auf 2^{-5} m und PMF: 160,2 KByte

(c) wie Abbildung 4.22(a), jedoch texturiert



(d) wie Abbildung 4.22(b), jedoch texturiert

Bild 4.22: Vergleich zwischen unkomprimiertem (links oben) und maximal komprimiertem Gesamtmodell (rechts oben). In der unteren Reihe dazu passend die texturierte Ansicht, die dem Operator dargestellt wird.

Für die komprimierte Übertragung von 3D-Punkten bietet der MPEG-4 Standard folgende Werkzeuge an: EfficientFloat, Quantisierung und Predictive MFField. Für EfficientFloat wurden zwei Strategien für den Enkoder-Algorithmus vorgestellt. Es hat sich jedoch herausgestellt, dass es das schwächste Kompressionswerkzeug ist. Zwar gestaltet sich seine Verwendung als sehr einfach, aber es erreicht weder in Bezug auf Kompressionsrate, noch für die benötigte Laufzeit die Leistung der anderen Verfahren. Diese Aussagen gelten für den verwendeten Testdatensatz. Die zu kodierenden Daten enthalten aufgrund von Sensorrauschen viele gesetzte Bits in ihrer Mantisse. Sobald andere, z. B. synthetisch generierte Geometriemodelle verwendet werden, kann das Verfahren wahrscheinlich mehr Qualitäten zeigen.

Das Quantisierungswerkzeug hat den besten Gesamteindruck hinterlassen. Durch die ein-

fache Berechnungsvorschrift ist es sehr effizient implementierbar und komprimiert mit guten Raten die 3D-Punkte. Das Predictive MFField Werkzeug erweitert die Quantisierung um eine arithmetische Kodierung. Zu einer generellen Verwendung von PMF für Geometriedaten kann an dieser Stelle nicht geraten werden. Seine Anwendung ist komplex und mit einigen Bedingungen verbunden. Die Messergebnisse zu Kompressionsraten und Laufzeit lassen sich leider nicht in wenigen Worten zusammenfassen, da es günstige und ungünstige Einsatzbeispiele gibt. Unter speziellen Voraussetzungen kann Predictive MFField für 3D-Punkte jedoch seine Vorteile sehr deutlich ausspielen (siehe Abschnitt 4.5).

Letztendlich können die 3D-Punkte sehr effizient kodiert werden. Aus einer Liste von 3D-Punkten werden Polygone gebildet, indem eine Reihenfolge von Indizes in diese 3D-Punktliste abgespeichert werden. Die komprimierte Kodierung dieser Indizes gestaltet sich schwieriger, da hier keine verlustbehaftete Kompression verwendet werden darf. Für diesen Datentyp stellt MPEG-4 BIFS das `CoordIndex` Werkzeug zur Verfügung. Dieses kann umso effizienter kodieren, je kürzer die referenzierte Liste von 3D-Punkten ist. Das hier vorgestellte IFS-Splitting ist ein Vorgehen, mit dem kürzere 3D-Punktelisten kodiert werden, wodurch die Effizienz von `CoordIndex` erheblich gesteigert werden kann. Ebenso wie beim Quantisierungswerkzeug kann das Predictive MFField Verfahren auf die durch `CoordIndex` erzielten Werte aufbauen. Jedoch erscheint keines der möglichen PMF Kodierschemas für die Anwendung auf Indizes geeignet zu sein. Nur unter erheblichen Einbußen bei der Laufzeit werden bessere Kompressionsraten erreicht.

5 MPEG-4 angewendet im Haptisch-Visuell-Auditorischen Arbeitsraum (HVA)

Für eine erfolgreiche Teleoperation steht vor der Übertragung der komprimierten Daten der Verbindungsaufbau. Im einfachsten Fall findet er während der Initialisierungsphase des Szenarios statt und betrifft nur eine einzige Datenverbindung. Für komplexere Telepräsenz-Szenarien, zu denen kooperative Manipulationen mit mehreren Teleoperatoren und vielen Datenströmen zählen, ist hier das Ziel, dass die vorhandenen Datenquellen automatisch und bedarfsgerecht für alle Teilnehmer am Szenario gemeinsam zur Verfügung stehen.

In diesem Kapitel wird beschrieben, wie durch weitere Komponenten eines MPEG-4 Systems die Signalisierung der verfügbaren Datenquellen, sowie der Verbindungsaufbau zu diesen auch in komplexen Szenarien realisiert werden kann. Die zusätzlichen Komponenten sind das Object Descriptor Framework, die Synchronisations- und die Übertragungsschicht (siehe Kapitel 3.4, 3.5 und 3.6). An ein Gesamtsystem in einem Telepräsenz-Szenario werden andere Anforderungen gestellt, als an eine reine Präsentation von Multimedia-Inhalten. Letztere entspricht dem eigentlichen Ziel von MPEG-4. Deshalb geht der folgende Abschnitt zuerst auf die besonderen Anforderungen für Telepräsenz ein.

5.1 Anforderungen des HVA

Im Sonderforschungsbereich 453 „Wirklichkeitsnahe Telepräsenz und Teleaktion“ wird mit dem *Haptisch-Visuell-Auditorischen Arbeitsraum* (HVA) die Gesamtheit aller auf der Teleoperatorseite aufgenommenen oder rekonstruierten Modalitäten bezeichnet. Die erfassten Daten sollen allen Operatoren, die an einer gemeinsamen, kooperativen Teleoperation teilnehmen, gleichermaßen zur Verfügung gestellt werden. Ziel ist, eine identische Wahrnehmung der entfernten Szene für alle Operatoren zu gewährleisten. In Abbildung 5.1 ist dieser Arbeitsraum auf der linken Seite der Barriere eingezeichnet. In den Kontext von MPEG-4 übertragen heißt das, dass eine multimediale Präsentation der entfernten Szene zur Verfügung gestellt werden soll, in der alle von den Teleoperatoren erstellten Elementary Streams gemeinsam verfügbar sind.

Zur Durchführung einer kooperativen Telemanipulation sind üblicherweise weitere Infor-

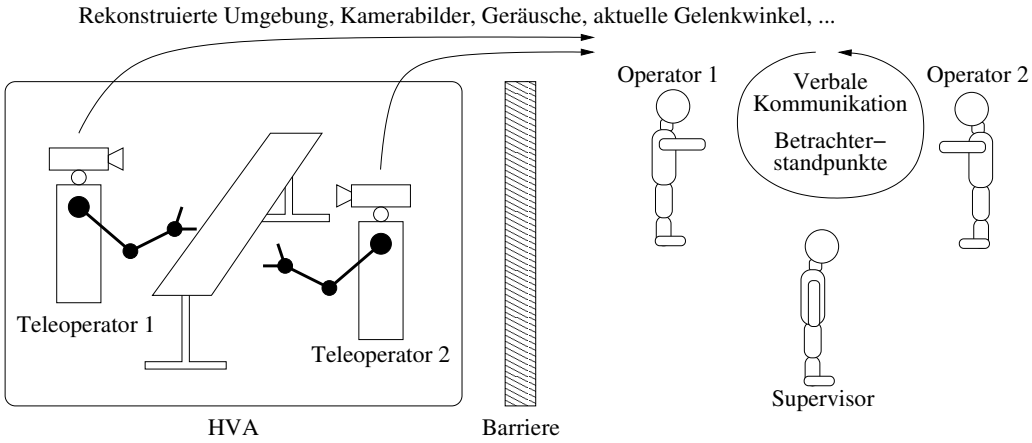


Bild 5.1: Elementary Streams im HVA

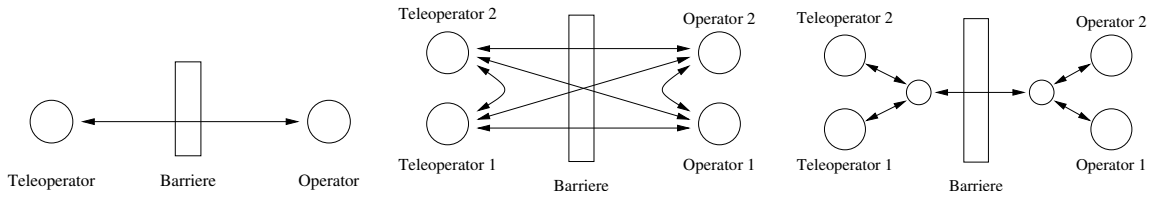
mationsströme zur Verständigung zwischen den Operatoren notwendig, die die Elementary Streams aus dem HVA ergänzen. Als Beispiele hierzu sind in Abbildung 5.1 auf der rechten Seite verbale Kommunikation zwischen den Operatoren und die Betrachterstandpunkte aufgeführt. Letztere teilen jeweils den anderen Operatoren mit, aus welchem Blickwinkel die virtuelle Welt betrachtet wird. Diese zusätzliche Information ist für die verbale Kommunikation hilfreich.

Die weiteren Ausführungen orientieren sich an den folgenden drei Kommunikationsstrukturen, nach denen die Teilnehmer in einem HVA-Telepräsenz-Szenario untereinander Datenverbindungen aufbauen können:

- **Punkt-zu-Punkt Struktur** (siehe Abbildung 5.2(a))
  Diese Struktur ist die Minimalanforderung für Telepräsenz-Systeme, in der nur ein Teleoperator mit genau einem Operator über die Barriere kommuniziert.
- **Sternstruktur**[1)] (siehe Abbildung 5.2(b)) Jeder Teleoperator bietet bei dieser Anordnung seine Elementary Streams jedem Operator einzeln an. Folglich muss jeder Operator mit allen Teleoperatoren getrennt kommunizieren, um alle Elementary Streams des HVA zu erhalten. Der offensichtliche Nachteil dieser Struktur ist die zusätzlich benötigte Bandbreite über die Barriere, sobald mehr als ein Operator beteiligt ist. Ein weiterer potenzieller Nachteil ist, dass die einzelnen Elementary Streams erst bei jedem Operator zusammengeführt werden. Dadurch liegt erst im Operator eine Gesamtmodell der entfernten Szene vor, wodurch eine eventuell notwendige Datenfusion deshalb bei jedem Operator einzeln vorgenommen werden muss. Eine Sternstruktur wird realisiert, indem jeder Teleoperator als Server tätig ist, auf den die Operatoren als Clients zugreifen.
- **Baumstruktur** (siehe Abbildung 5.2(c))
  Bei dieser Struktur werden die Elementary Streams sowohl auf Teleoperator- als

[1)] Die Bezeichnung der Struktur wurde gewählt, da aus der Perspektive eines Teilnehmers alle seine zur Verfügung gestellten Daten sternförmig von allen anderen Teilnehmern angefordert werden. Betrachtet man Abbildung 5.2(b), wäre auch die Bezeichnung „jeder mit jedem“ treffend.

auch auf Operatorseite an einem zentralen Sammelpunkt zusammengeführt und von dort aus auf die andere Seite der Barriere übertragen. Die Baumstruktur beinhaltet sowohl die Punkt-zu-Punkt, als auch die Sternstruktur als Sonderfälle. Da nur die Sammelpunkte miteinander über die Barriere kommunizieren, erreicht diese Struktur das Minimum der benötigten Bandbreite. Aus der Anordnung der Teilnehmer in Abbildung 5.2(c) ist nicht direkt die Baumstruktur ersichtlich. Oben rechts in Abbildung 5.5 wird eine alternative Anordnung zu diesem Beispiel gezeigt. Als Nachteil dieser Struktur wird erwartet, dass das Zusammenführen und Weiterleiten der Daten in den Sammelpunkten zu längeren Verzögerungszeiten führt. Die Funktionalität der Sammelpunkte kann auch durch einen Teleoperator/Operator übernommen werden. Dadurch wird für genau diesen Teleoperator/Operator die zusätzliche Verzögerung vermieden. Diese Optimierung ist jedoch stark vom Szenario abhängig und wird im Folgenden nicht weiter betrachtet.



(a) Punkt-zu-Punkt (b) Stern (c) Baum

Bild 5.2: Drei Beispiele für mögliche HVA-Kommunikationsstrukturen

Da reale Objekte in der entfernten Umgebung manipuliert werden sollen, ist eine möglichst geringe Verzögerung zwischen der Erfassung einer Veränderung in der realen Szene und der Darstellung dieser Veränderung beim Operator von hohem Interesse. Gleichzeitig muss in Betracht gezogen werden, dass für einige Telepräsenz-Szenarien die zur Verfügung stehende Bandbreite durch die Barriere limitiert ist. Um beide Forderungen gleichzeitig zu erfüllen ist weder die Stern-, noch die Baumstruktur ideal. Im Laufe des Kapitels wird in den Abschnitten 5.2.2 und 5.2.3 beschrieben, wie mit MPEG-4 eine Baumstruktur realisiert werden kann, deren Verzögerung durch die Sammelpunkte minimiert wird.

Der größte Anteil der visuellen, auditorischen und auf Geometrie bezogenen Daten wird von den Teleoperatoren zu den Operatoren geschickt und dort dargestellt. In einigen Fällen wird jedoch, wie oben bereits erwähnt, auch ein Austausch von Video-, Audio- oder Geometriedaten zwischen den Operatoren oder zu den Teleoperatoren benötigt. Unabhängig von der Struktur des HVA wird dazu eine Möglichkeit für den Datenaustausch in beide Richtungen gefordert. In Abschnitt 5.2.1 wird beschrieben, wie dies für Geometriedaten realisiert werden kann.

Für die folgenden Betrachtungen wird nicht weiter auf Multicast eingegangen. Multicast bezeichnet eine Netzwerktechnologie, die es erlaubt, Datenpakete von einer Quelle an viele Empfänger gleichzeitig zu übermitteln. Der Grund hierfür liegt darin, dass die für Multicast notwendigen Voraussetzungen nicht für alle Telepräsenz-Szenarien garantiert werden können. Dies gilt insbesondere für die Übertragung der Elementary Streams über

die Barriere. So werden z. B. im Bereich der Raumfahrt sehr spezielle Protokolle für die Datenübertragung ins Weltall benützt, die nicht für Multicast konzipiert sind. Die hier vorgestellten Konzepte behandeln deshalb ausschließlich Unicast Datenübertragungen. In einem lokalen Netzwerk kann natürlich Multicast wieder eingesetzt werden, z. B. für die Verteilung der Elementary Streams vom Sammelpunkt auf der Operatorseite zu den Operatoren. Diese Verbindung von Uni- mit Multicast-Netzwerken wurde bereits in [23] mit dem DIVEBONE für die virtuelle Welt DIVE (distributed interactive virtual environment) vorgestellt.

5.2 Grundlegende Vorgehensweise zur Einbindung von MPEG-4

Die im vorhergehenden Abschnitt aufgeführten Anforderungen wurden auf ihre Realisierbarkeit über Mittel des MPEG-4 Standards untersucht. Die entstandenen Konzepte werden nun in den folgenden Abschnitten präsentiert. Diese Konzepte bilden die Grundlage für den Einsatz von MPEG-4 Systemen in einem HVA. Darauf aufbauend bieten MPEG-4 Systeme weitere Möglichkeiten, für die in Abschnitt 5.3 ein Beispiel vorgestellt wird.

5.2.1 Duplex Übertragung für BIFS-Update Kommandos

Obwohl sich der MPEG-4 Standard sehr stark an der Datenübertragung *von* einem Sendeterminal *zu* einem oder mehreren Empfangsterminals orientieren, wurde dort auch die Verwendung eines oder mehrere Rückkanäle spezifiziert (siehe auch Kapitel 3.1). Zur Anwendung kommen sie im Standard jedoch für ganz andere Zwecke, als wie für den geplanten Einsatz hier im HVA. Der Standard differenziert für die Spezifikation der Rückkanäle zwischen den übertragenen Datentypen:

- **Video**
  Für Videodaten werden zwei Rückkanalprotokolle definiert: `NEWPRED` und `SNHC_QoS`. Über beide Protokolle kann ein Empfangsterminal die Kodierung eines Video Elementary Streams beim Sendeterminal beeinflussen.
- **Audio**
  Für Audio Elementary Streams ist ebenfalls ein Rückkanalprotokoll definiert, mit dem der Enkoder beeinflusst werden kann. So ist z. B. für das BSAC Audiokodierverfahren eine Steuerung des „frame interleave“ Parameters möglich.
- **Szenendaten**
  Für BIFS Elementary Streams wird unter dem Begriff `ServerCommandRequest` ein Rückkanalprotokoll spezifiziert. Der Aufbau des Protokolls ist in Tabelle 5.1 dargestellt und wird im Folgenden weiter erläutert.

- **Object Descriptor Streams**
  Für ODS wurden bisher keine Rückkanäle definiert.

Feld	Bedeutung
`nodeID`	Knoten-ID des Knotens, der das Kommando verschickt
`SFString command`	eine beliebige Zeichenkette, die das Kommando enthält

Tabelle 5.1: Aufbau des `ServerCommandRequest` Protokolls

Im Folgenden werden nur Rückkanäle für Szenendaten betrachtet. Konzipiert wurde das `ServerCommandRequest` Protokoll für die Verwendung in einem `ServerCommand` Knoten:

ServerCommand	

`SFBool trigger`
`SFBool enable`
`MFString url`
`SFString command`
	Wird über Anwenderinteraktion, entweder direkt oder über eine ROUTE, das Feld `trigger` auf wahr gesetzt und ist `enable` ebenfalls wahr, dann verschickt ein `ServerCommand` Knoten das in `command` definierte Kommando (eine beliebige Zeichenkette) über einen Rückkanal, der in `url` angegeben ist.

Wenn ein `ServerCommand` Knoten eine `ServerCommandRequest`-Nachricht verschickt, dann nimmt das Feld `nodeID` (siehe Tabelle 5.1) die Knoten-ID des `ServerCommand` Knotens an. Das Feld `command` in der `ServerCommandRequest`-Struktur entspricht dem Feld `command` des `ServerCommand` Knotens.

Syntax und Semantik der Zeichenkette im `command` Feld werden nur durch die Applikation festgelegt [43]. Die Bytes einer `ServerCommandRequest`-Nachricht sollen als Access Unit behandelt werden und in gleicher Weise wie andere Access Units zum Sendeterminal geschickt werden. Für das Versenden soll ein als *Upstream* gekennzeichneter Elementary Stream verwendet werden (siehe Kapitel 3.4).

Da das `command` Feld beliebige Daten enthalten darf, spricht nichts gegen ein Versenden von kodierten BIFS-Update Kommandos in diesem Feld. Das Sendeterminal kennt die Konvention und führt das empfangene Kommando für seinen Szenengraphen aus. Das `nodeID` Feld der `ServerCommandRequest`-Nachricht wird beim Sendeterminal für die Ausführung des Kommandos nicht benötigt und kann deshalb beliebige Werte annehmen.

Abbildung 5.3 stellt die Elementary Streams, sowie die an der Kommunikation beteiligten Systemelemente für den Fall einer Punkt-zu-Punkt Struktur dar. Wie in Kapitel 3.6 beschrieben, signalisiert das Sendeterminal während dem Verbindungsaufbau dem Empfangsterminal den dargestellten initialen Objektdeskriptor `OD 1`. In diesem Beispiel enthält `OD 1` ES-Deskriptoren für zwei Elementary Streams, die BIFS-Update Kommandos übertragen. Der Elementary Stream `ESD 2` ist als Upstream gekennzeichnet und muss deshalb laut MPEG-4 Standard von Elementary Stream `ESD 1` abhängen. Beide Teilnehmer können aufgrund der Informationen in den ES-Deskriptoren die notwendigen Systemelemente instanzieren:

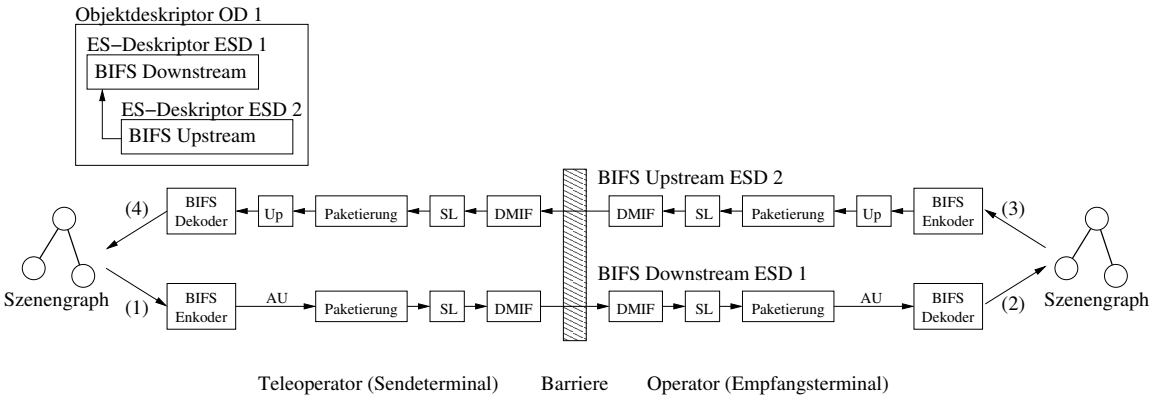


Bild 5.3: Beteiligte Komponenten für die Realisierung eines BIFS-Rückkanals

- Für jeden BIFS Elementary Stream muss entweder ein BIFS-Dekoder oder ein BIFS-Enkoder verfügbar sein.
- Für den Rückkanal muss ein Element verfügbar sein, dass ein BIFS-Update Kommando in eine `ServerCommandRequest`-Nachricht oder zurück wandelt (Up).
- Auf der sendenden Seite muss ein Element erzeugt werden, dass lange Access Units (AU) in kleinere Pakete zerteilt (Paketierung). Auf der empfangenden Seite muss entsprechend ein Element erzeugt werden, das diese Pakete wieder zu einer AU zusammensetzt.
- Um den Datenpaketen Informationen zur Synchronisierung in Form eines SL-Headers hinzuzufügen sind Elemente der Synchronisationsschicht notwendig (SL). Auf der empfangenden Seite ist ebenfalls ein solches Element notwendig, um den SL-Header wieder auszulesen.
- Für das tatsächliche Verschicken und Empfangen von Paketen sind Elemente aus der Übertragungsschicht (DMIF) notwendig.

Wird nun auf Teleoperatorseite eine Änderung im Szenengraphen vorgenommen (Pfeil 1), wird das entsprechende BIFS-Update Kommando als Access Unit kodiert und in mehrere kleinere Pakete zerlegt. Die Pakete werden mit Synchronisationsinformation versehen und von der Übertragungsschicht verschickt. Auf der Operatorseite empfängt wiederum die Übertragungsschicht und die Synchronisationsinformation wird ausgelesen. Die Pakete werden zu einer Access Unit zusammengesetzt, die durch einen BIFS-Dekoder wieder zum ursprünglichen BIFS-Update Kommando dekodiert wird. Die im Kommando beschriebenen Veränderungen werden auf dem auf Operatorseite vorhandenen Szenengraphen ausgeführt (Pfeil 2).

Ergeben sich Änderungen im Szenengraphen der Operatorseite (Pfeil 3), wird ein ähnlicher Prozess durchlaufen, jedoch muss hier noch zusätzlich eine Umwandlung in eine `ServerCommandRequest`-Nachricht vollzogen werden. Auf der Teleoperatorseite muss diese Umwandlung wieder symmetrisch rückgängig gemacht werden. Danach kann das BIFS-Update Kommando wieder dekodiert und für den Szenengraphen auf Teleoperator-

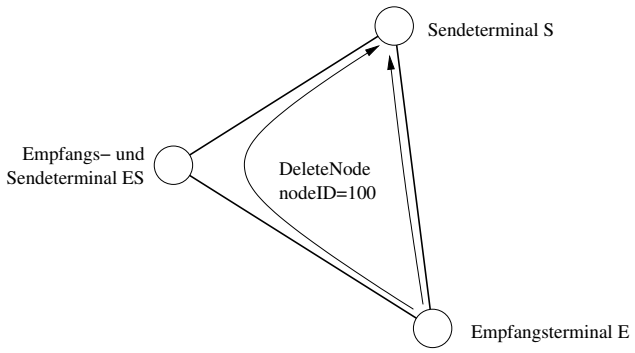


Bild 5.4: Komplikationen im Fall mehrerer Verbindungen zu Sendeterminals

seite vollzogen werden (Pfeil 4).

Mit Hilfe der soeben beschriebenen Abläufe können die beiden Szenengraphen auf Teleoperator- und Operatorseite auf gleichem Stand gehalten werden. Für komplexere HVA-Strukturen sind weitere Maßnahmen notwendig, wenn mehr als nur zwei Szenengraphen betroffen sind. Diese werden im folgenden Abschnitt beschrieben.

5.2.2 Weiterleitung von Elementary Streams

Betrachtet man sowohl die Stern-, als auch die Baumstruktur, muss eine Methode entwickelt werden, mit der BIFS-Update Kommandos von einem Teilnehmer zu allen anderen Teilnehmern weitergeleitet werden. Die im vorhergehenden Abschnitt vorgestellte Verwendung der Rückkanäle ist dazu unbedingt notwendig.

Allgemein gilt für die weiteren Betrachtungen, dass kein Teilnehmer sich als Empfangsterminal für einen bestimmten Elementary Stream zu mehr als einem Sendeterminal verbinden darf. Jeder Teilnehmer kann jedoch die Rolle des Sendeterminal für beliebig viele Empfangsterminals einnehmen. Als Begründung soll hier der Fall in Abbildung 5.4 betrachtet werden. Das Empfangsterminal E sei hier sowohl mit dem Sendeterminal S, als auch mit dem Sendeterminal ES verbunden. ES nimmt hier zwei Rollen ein. Einerseits ist es Sendeterminal für E, andererseits erhält es seine Daten vom Sendeterminal S. In E wird nun ein `NodeDelete` BIFS-Update Kommando generiert, um den Knoten mit der `nodeID`=100 zu löschen. Dieses Kommando wird nun an alle für E bekannten Sendeterminals geschickt und dort ausgeführt. ES wird dieses Kommando ebenfalls an S weiterleiten, wodurch es ein zweites Mal bei S ausgeführt werden soll. Da der Knoten aber bereits durch das direkt übertragene Kommando von E entfernt wurde, muss S annehmen, dass es ein Kommando zum Erzeugen des Knotens mit der `nodeID`=100 nicht empfangen hat und dass ein Übertragungsfehler aufgetreten ist. Daher kann S nicht mehr davon ausgehen, dass sein Szenengraph auf dem gleichen Stand wie die anderen Szenengraphen ist. Folglich muss S nun auf ein `SceneReplace` BIFS-Update Kommando warten, obwohl in Wirklichkeit gar kein Übertragungsfehler aufgetreten ist. Wird die Verbindungsstruktur zwischen den Teilnehmern als Graph betrachtet, trifft also die Aussage zu, dass der Graph

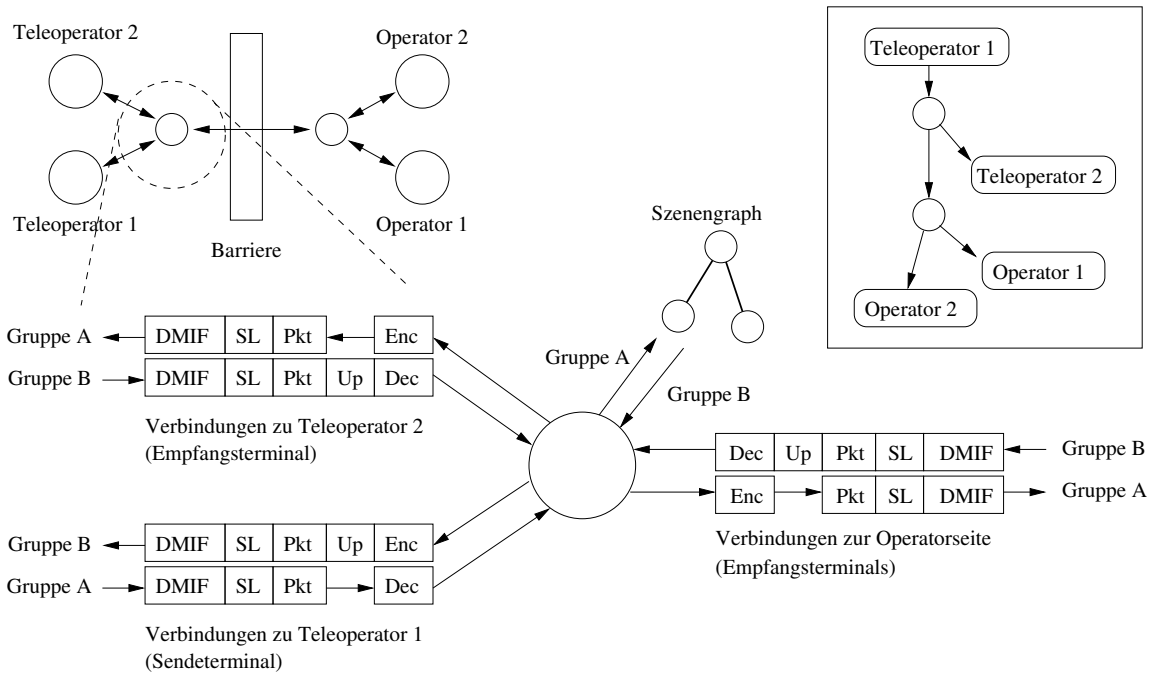


Bild 5.5: Verbundene Elementary Streams für den Sammelpunkt 1

keine Zyklen haben darf.

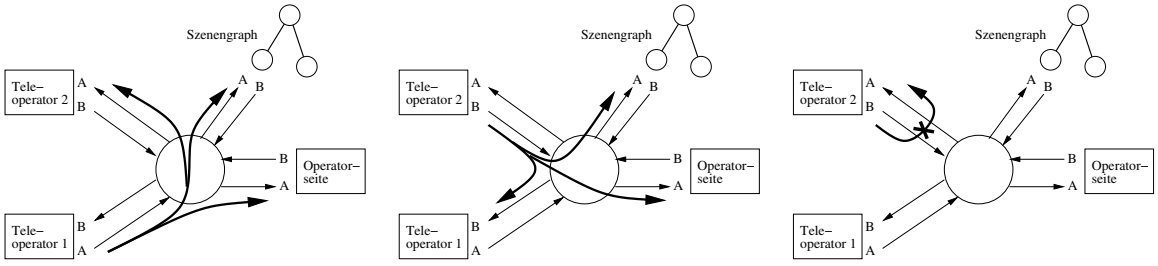
In Abbildung 5.5 ist nochmals die Baumstruktur dargestellt. Die internen Elemente im Sammelpunkt auf Teleoperatorseite sind hier vergrößert dargestellt, um auf die notwendigen Abläufe für eine Weiterleitung eingehen zu können. Dieser Sammelpunkt soll sich nun zu Teleoperator 1 wie ein Empfangsterminal verhalten, ist seinerseits jedoch Sendeterminal sowohl für Teleoperator 2, als auch für alle Teilnehmer auf Operatorseite. Rechts oben ist in Abbildung 5.5 die Baumstruktur in anderer Weise dargestellt, aus der diese Rollenzuordnung direkt ersichtlich wird. Der Sammelpunkt enthält ebenfalls einen Szenengraphen, der notwendig ist, um die empfangenen BIFS-Update Kommandos dekodieren zu können. Damit der Szenengraph auf aktuellem Stand ist, werden alle empfangenen Kommandos ebenfalls für diesen Szenengraphen ausgeführt.

Das Ziel für den Sammelpunkt ist nun, jedes empfangene BIFS-Update Kommando an die anderen Teilnehmer des Szenarios weiterzuleiten. Dabei ist egal, ob sie von Teleoperator 1, 2 oder der Operatorseite empfangen werden. Um dies zu erreichen, soll an dieser Stelle der Begriff einer Elementary Stream Gruppe eingeführt werden. Zur Gruppe A in Abbildung 5.5 gehören alle BIFS Elementary Streams, die von Sendeterminals zu Empfangsterminals weitergeleitet werden sollen. Zur Gruppe B gehören alle BIFS Elementary Streams, die von den Empfangsterminals an das Sendeterminal weitergeleitet werden.

In der alternativen Darstellung der Baumstruktur oben rechts in Abbildung 5.5 repräsentiert Gruppe A also die Weiterleitung von BIFS-Update Kommandos „nach unten“, während Gruppe B „nach oben“ weiterleitet. Im dargestellten Sammelpunkt müssen nun Kommandos, die über eine Verbindung der Gruppe A empfangen werden, an alle weite-

ren Verbindungen der Gruppe A weitergeleitet werden (siehe auch Abbildung 5.6(a)). Da jeder Teilnehmer für eine Gruppe nur mit einem Sendeterminal verbunden sein darf, ist damit auch bereits alles Notwendige geschehen.

Werden Kommandos über eine Verbindung der Gruppe B empfangen, so müssen sie zum einen zum Sendeterminal weitergereicht werden. Dies geschieht über eine Verbindung der Gruppe B. Zum anderen werden die Kommandos über Verbindungen der Gruppe A an alle Empfangsterminals weitergeleitet (siehe Abbildung 5.6(b)), jedoch *nicht* zu dem Empfangsterminal, von dem das Kommando empfangen wurde (siehe Abbildung 5.6(c)).



(a) Weiterleitung Gruppe A (b) Weiterleitung Gruppe B (c) Ausnahme für Gruppe B

Bild 5.6: Weiterleitungsregeln für empfangene BIFS-Update Kommandos

Für die Punkt-zu-Punkt und Sternstruktur gelten die gleichen Weiterleitungsregeln, da diese nur Sonderfälle der Baumstruktur sind.

Die erwartete Verzögerung für eine Weiterleitung eines einzelnen BIFS-Update Kommandos setzt sich folgendermaßen zusammen:

- Alle SL-Pakete, die zu einer Access Unit gehören, müssen empfangen, interpretiert (SyncLayer) und zu einer Access Unit zusammengesetzt werden.
- Die vollständig empfangene Access Unit muss zu einem BIFS-Update Kommando dekodiert werden (Compression Layer).
- Anschließend folgte ein Enkodierung des BIFS-Update Kommandos wieder zu einer Access Unit. Im nächsten Abschnitt wird genauer betrachtet, unter welchen Bedingungen dieser Schritt entfallen kann.
- Die neue Access Unit wird wieder in SL-Pakete zerlegt und diese einzeln verschickt.

Es ist nur mit hohem Aufwand für die Implementierung möglich, eine Access Unit bereits während des Empfangs zu dekodieren. Für jeden Zugriff auf den Bitstrom müsste der Dekoder erst prüfen, ob die benötigte Menge an Daten bereits empfangen wurde. Die in Kapitel 4 erzielten Zeiten für die Dekodierung würden viel schlechter ausfallen. Ein potenzieller Vorteil einer Dekodierung während des Empfangs entsteht, wenn mehrere BIFS-Update Kommandos in einer Access Unit kodiert sind. Sobald ein einzelnes Kommando vollständig dekodiert wurde, kann es bereits in einer kleineren Access Unit weitergeleitet werden. Diese Möglichkeit wurde jedoch nicht weiterverfolgt. Im folgen-

den Abschnitt wird beschrieben, wie die Weiterleitung optimiert werden kann, ohne die Komplexität des Dekoders zu erhöhen.

5.2.3 Optimierung der Weiterleitung

Die Methode zur Weiterleitung wurde bisher auf Basis von BIFS-Update Kommandos beschrieben. Die dazu notwendige Dekodierung, gefolgt von einer erneuten Enkodierung, ist jedoch ein aufwendiger Arbeitsschritt, weshalb eine Optimierung des Verfahrens notwendig erscheint.

An dieser Stelle werden nun zuerst die verfügbaren Kodierungs- und Übertragungsparameter genauer betrachtet. Die für die Kodierung relevanten Parameter sind in den `BIFSConfig` Strukturen hinterlegt (siehe 3.4.2), die wiederum Teil der ES-Deskriptoren sind. Für jeden Elementary Stream zwischen zwei Teilnehmern kann ein unterschiedlicher ES-Deskriptor verwendet werden. Der Inhalt eines ES-Deskriptors wird immer durch das jeweilige Sendeterminal für jede Verbindung bestimmt. Im Einzelnen ergibt sich für die Kodierungsparameter folgendes:

- **nodeIDbits:** Für ein konkretes Szenario muss für `nodeIDbits` a-priori eine minimale Grenze festgelegt werden, da dieser Wert die maximale Anzahl von kodierbaren Knoten-IDs bestimmt. Werden Knoten-IDs außerhalb des Intervalls $[0 \ldots 2^{nodeIDbits} - 2]$ im Modell verwendet, kann das Sendeterminal nicht alle Knoten-IDs korrekt enkodieren. Spätestens bei der Verwendung eines `SceneReplace` Kommandos werden alle verwendeten Knoten-IDs kodiert. Da das `SceneReplace` Kommando mindestens einmal zu jedem Empfangsterminal geschickt werden muss, darf der Parameter `nodeIDbits` nicht zu niedrig gewählt werden. Dies gilt für alle Verbindungen. Es ist jedoch durchaus möglich, `nodeIDbits` für eine bestimmte Verbindung größer als notwendig zu wählen. Auch der Versuch, `nodeIDbits` nur zeitweise für eine Verbindung geringer zu wählen kann nicht funktionieren. Eine Parameterveränderung ist laut Standard nur durch einen Neuaufbau der Verbindung erreichbar. Da bei jedem Aufbau einer Verbindung zuerst ein `SceneReplace` Kommando zum Empfangsterminal geschickt wird, kann dieser Ansatz nicht funktionieren. Die gleichen Überlegungen gelten für die Parameter `protoIDbits` und `routeIDbits`, die zwar nicht in Tabelle 3.6 aufgeführt sind, jedoch die Anzahl der Bits für die Kodierung anderer Identifizierungsnummern bestimmen.

- **isCommandStream:** Da es keinerlei Umwandlungsmöglichkeit zwischen BIFS-Update und BIFS-Anim Kommandos gibt, muss der Parameter `isCommandStream` für jede Verbindung identisch sein. Der Parameter `animMask` wird hier nicht betrachtet, da er ausschließlich für BIFS-Anim Elementary Streams verwendet wird.

- **useNames:** Für eine Weiterleitung ist ein Wechsel des Parameters `useNames` vom Wert `wahr` auf `falsch` möglich, da zusätzliche Information immer weggelassen werden kann. Im anderen Fall eines Wechsel von `falsch` auf `wahr` kann die benötigte Information nicht ohne Weiteres rekonstruiert werden. Wenn im Szenario weder

Rückkanäle, JavaScripts oder MPEG-J verwendet werden, kann dieser Parameter folglich verändert werden.

- **use3DMC:** Dieser Parameter kann für jede Verbindung unterschiedlich gewählt werden. 3DMC ist ein Kompressionswerkzeug, mit dem inhaltlich nicht mehr oder weniger Informationen übertragen werden. Eine Konversion ist in beide Richtungen möglich.
- **usePredMFField:** Mit der gleichen Begründung wie bei `use3DMC` kann auch das Predictive MFField Verfahren für jede Datenverbindung wahlweise ein- oder ausgeschaltet werden.

Wie sich zeigt ist es durchaus möglich, einige Kodierungsparameter für einzelne Verbindungen nach den gewünschten Vorstellungen anzupassen. Dies mag auf den ersten Blick nicht notwendig erscheinen, jedoch ist es durchaus möglich, dass z. B. ein einzelnes Empfangsterminal in einem Szenario keine über PMF kodierten Daten verarbeiten kann. Durch einen Wechsel der Kodierungsparameter können die restlichen Terminals jedoch durchaus PMF kodierte Daten untereinander austauschen.

Die für die Übertragung relevanten Parameter werden durch die verwendete DMIF-Transporttechnologie festgelegt. Im Weiteren wird nur der Parameter `MAX_PDU_SIZE` (Maximum Payload Data Unit Size) betrachtet, der die maximale Größe der SL-Pakete bestimmt. Je nach gewählter Transporttechnologie muss sich dieser Wert innerhalb systemgegebener Grenzen bewegen. Beispielsweise für eine Datenverbindung über Ethernet sollte `MAX_PDU_SIZE` nicht größer als 1 500 Bytes gewählt werden, da ansonsten die Netzwerkschicht die Datenpakete fragmentiert überträgt. Da nicht alle Datenverbindungen zwangsläufig die gleiche Transporttechnologie verwenden müssen, kann auch ein Wechsel der Übertragungsparameter notwendig sein.

Die Parametern der SyncLayer werden durch die `SLConfigDescriptor` Struktur bestimmt. Wie später erläutert wird, haben diese Parameter jedoch kaum Einfluss auf die optimierte Weiterleitung, weshalb sie an dieser Stelle auch nicht weiter betrachtet werden. Welche Auswirkungen unterschiedliche Parameter für eine optimierte Weiterleitung haben, wird in den folgenden Absätzen genauer betrachtet.

Unterschiede in den Kodierungsparametern

Unabhängig von der Wahl der Übertragungsparameter schränken unterschiedliche Kodierungsparameter jegliche optimierte Weiterleitung von BIFS-Update Kommandos sehr stark ein. Der Aufwand, um die notwendigen Veränderungen in der Abfolge der kodierten Bits zu berechnen ist beträchtlich. Zuerst müsste eine vollständige Dekodierung erfolgen, um die im Kommando enthaltenen Daten identifizieren zu können. Für eine anschließende Veränderung der Bitabfolge müsste zusätzlich bekannt sein, an welcher Bitposition die einzelnen Daten beginnen. Da mit großer Sicherheit an einigen Stellen Bits hinzuzufügen oder zu entnehmen sind, werden umfangreiche Kopiervorgänge im Speicher notwendig. Jeder dieser Kopiervorgänge ist aufwendig, da im Allgemeinen nicht Byte für Byte, sondern

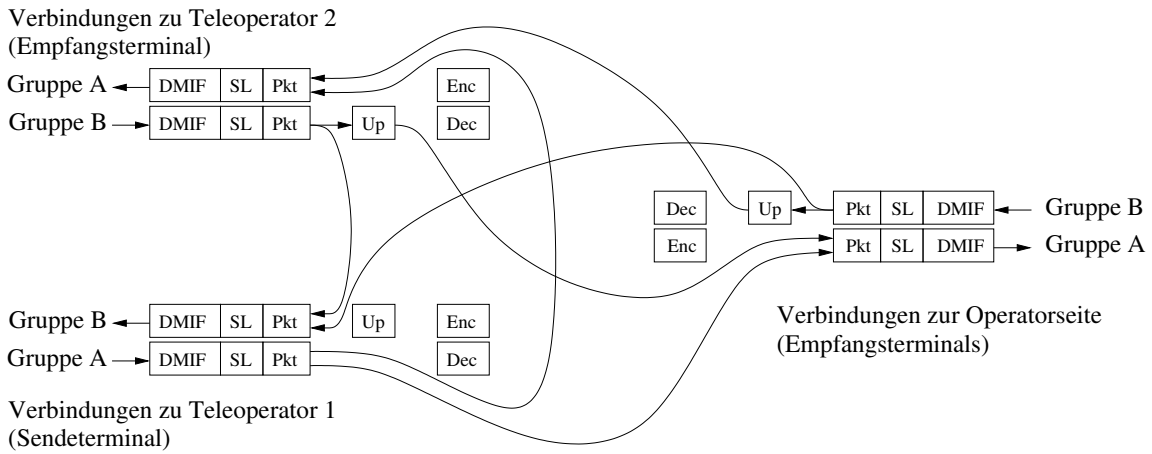


Bild 5.7: Optimierte Weiterleitung von Access Units bei identischer Kodierung, jedoch unterschiedlicher Datenpaketgröße

Bit für Bit kopiert werden muss.

Alleine aufgrund dieser Überlegungen wurde der Fragestellung, ob dieses Vorgehen Vorteile gegenüber einer erneuten, vollständigen Enkodierung mit sich bringt, hier nicht nachgegangen. Bei Unterschieden in den Kodierungsparametern wird in dieser Arbeit deshalb auf jegliche Optimierung für die Weiterleitung verzichtet.

Identische Kodierungsparameter, unterschiedliche Übertragungsparameter

Für den Fall, dass identische Kodierungsparameter, jedoch unterschiedliche Werte für `MAX_PDU_SIZE` für zwei Verbindungen gewählt werden, kann auf die aufwendige Dekodierung in BIFS-Update Kommandos verzichtet werden. Anstatt dessen reicht es aus, die empfangenen Access Units passend für den neuen Wert von `MAX_PDU_SIZE` wieder in kleinere Datenpakete zu teilen.

Abbildung 5.7 zeigt, wie nun die Weiterleitung der Elementary Streams optimiert werden kann. Nachdem die eintreffenden Datenpakete zu Access Units (AU) zusammengesetzt wurden, werden diese nicht mehr zwangsläufig zu BIFS-Update Kommandos dekodiert, sondern an den Dekodern vorbei zu den anderen Teilnehmern weitergeleitet. Eine Dekodierung kann trotzdem erforderlich sein, falls der weiterleitende Teilnehmer selber die BIFS-Update Kommandos ausführen will. Aus Gründen der Übersichtlichkeit wurden die entsprechenden Pfeile jedoch in Abbildung 5.7 nicht eingezeichnet. Bei den Rückkanälen ist zu beachten, dass bei einem Wechsel der Gruppe eine Wandlung zwischen Downstream-AUs und Upstream-AUs durchgeführt werden muss.

Die erwartete Verzögerung für die Weiterleitung eines BIFS-Update Kommandos reduziert sich in diesem Fall auf das Empfangen, SL-Header auslesen und Zusammensetzen zu einer Access Unit, gefolgt von einem erneuten Zerlegen und Verschicken der neuen SL-Pakete. Die aufwendigen Kodierungsschritte verzögern die Weiterleitung der BIFS-Update

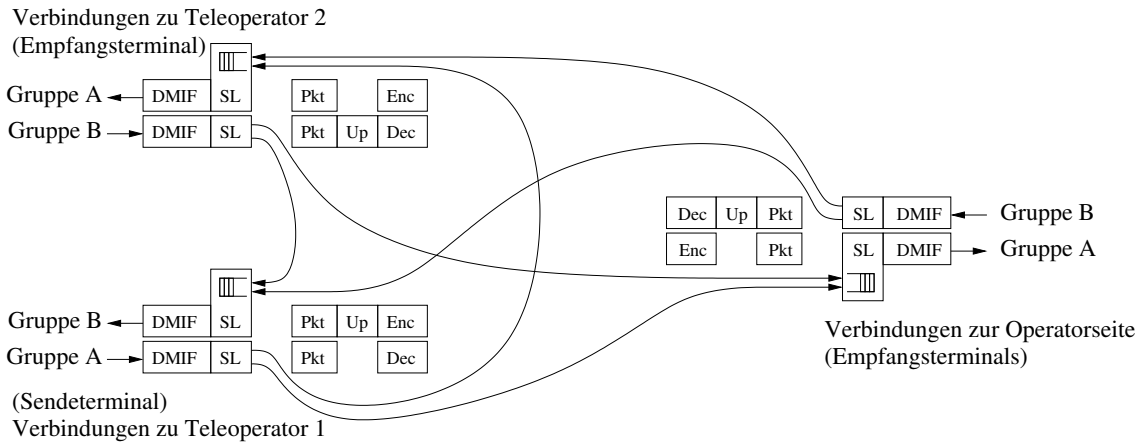


Bild 5.8: Optimierte Weiterleitung von SL-Paketen bei identischer Kodierung und Datenpaketgröße

Kommandos nun nicht mehr.

Identische Kodierungs- und Übertragungsparameter

In diesem Fall muss weder umkodiert, noch die Datenpaketgröße geändert werden. Empfangene SL-Pakete können direkt nach dem Auslesen des SL-Headers weitergegeben werden. Abbildung 5.8 zeigt die dazu notwendigen Datenflüsse. Hierbei sind jedoch zwei Einschränkungen zu beachten:

- **Ausschluss parallelen Sendens**
  Wird eine Access Unit in mehrere SL-Pakete aufgeteilt und verschickt, erwartet der Empfänger jedes SL-Paket dieser AU in der richtigen Reihenfolge und ohne Unterbrechung durch SL-Pakete anderer AUs zu empfangen. Es ist deshalb unbedingt notwendig, eine Access Unit vollständig als SL-Paketfolge zu verschicken, bevor SL-Pakete einer anderen Access Unit über die gleiche Verbindung geschickt werden. Dazu müssen in die versendenden SL-Elemente Puffer eingebaut werden, die SL-Pakete so lange zurückhalten können, bis eine laufende Übertragung von SL-Paketen einer anderen AU abgeschlossen ist. Dazu muss aus den SL-Headern ersichtlich sein, wann eine AU abgeschlossen ist. In Anhang C.3 werden dazu verschiedene Konfigurationsmöglichkeiten angegeben.

- **Up/Downstream Wandlung**
  Ebenfalls in den versendenden SL-Elementen muss bei einem Wechsel der Gruppe eine Wandlung von Upstream-Paketen in Downstream-Pakete notwendig sein. Davon ist nur das erste SL-Paket einer AU betroffen, da nur dieses Paket die zusätzlichen Daten einer `ServerCommandRequest`-Nachricht transportiert. Für die Konfiguration des SL-Headers ist deshalb wiederum darauf zu achten, dass das erste SL-Paket einer AU identifizieren werden kann.

Die erwartete Verzögerung für die Weiterleitung eines BIFS-Update Kommandos reduziert sich in diesem Fall auf die Zeit, die für ein Paket zum Empfangen, SL-Header auslesen, SL-Header verändern und schließlich erneut Verschicken notwendig ist. Durch die Serialisierung kommen weitere, zusätzliche Verzögerungen hinzu, die von der Häufigkeit und Größe der parallel zu verarbeitenden Access Units abhängen und damit nur schwer deterministisch bestimmbar sind.

Identische Parameter der SyncLayer

Eine noch weitergehende Optimierung soll hier kurz betrachtet werden, die identische Parameter der SyncLayer voraussetzt. Im Prinzip könnten die empfangenen Datenpakete direkt nach den DMIF-Elementen weitergeleitet werden. Da jedoch für den praktischen Einsatz zur Detektion von Übertragungsfehlern im SL-Header sowohl Paket-, als auch AU-Zähler verwendet werden sollten (siehe 3.5), müssen die jeweiligen Zählerwerte vor dem Verschicken modifiziert werden. Der Aufwand zum Modifizieren eines SL-Headers unterscheidet sich nur sehr geringfügig vom Aufwand des Auslesens und erneut Schreibens eines SL-Headers, da dieser typischerweise nur wenige Bytes groß ist. Für die Weiterleitung auf Basis von DMIF-Paketen würden dieselben Einschränkungen gelten, die bereits oben für die Weiterleitung auf Basis von SL-Paketen beschrieben wurden. Für eine Implementierung dieser Einschränkungen sind die DMIF-Elemente jedoch eine ungünstige Stelle, da DMIF-Elemente an und für sich kein Wissen über die Parameter der SyncLayer besitzen. Aus diesen Gründen wurde auf eine weitere Untersuchung dieses Falles verzichtet.

5.2.4 Verteilte Modellrekonstruktion über Medienknoten

In Kapitel 3.4.1 wurde beschrieben, dass bei Einsatz einer URL im Objektdeskriptor eine weitere Präsentation geöffnet wird. Diese Präsentation kann von einem beliebigen anderen Rechner erzeugt werden. Bei komplexen entfernten Umgebungen kann dieser Mechanismus dazu verwendet werden, die Rekonstruktionsaufgabe auf mehrere Teleoperatoren aufzuteilen.

Die dazu notwendigen Schritte sollen an dieser Stelle erläutert werden. Im nächsten Abschnitt wird zuerst prinzipiell dargestellt, wie eine Anwendung der Objektdeskriptor-URLs gemäß dem MPEG-4 Standard in einem HVA-Telepräsenz-Szenario aussehen kann. Der darauf folgende Abschnitt geht näher auf eine besondere Maßnahmen ein, die durch Bündelung der Elementary Streams im Sammelpunkt unnötige Datenübertragungen über die Barriere verhindert.

Verwendung von Objektdeskriptor-URLs

Es wird angenommen, dass jeder Teleoperator einen Sensor hat, der die 3D-Geometrie seiner Umgebung rekonstruiert und in einem Szenengraphen ablegt. Da anzunehmen ist,

dass diese Rekonstruktion pro Teleoperator durch einen eigenen Rechner ausgeführt werden kann, wird die benötigte Rechenzeit dadurch auf mehrere Rechner aufgeteilt. Aus mehreren Gründen erscheint dann ein Mechanismus sinnvoll, der die einzelnen Szenengraphen zueinander in Relation setzt, indem die einzelnen Präsentationen in eine gemeinsame Präsentation integriert werden:

- Damit ein Teilnehmer Zugriff auf alle Teilszenengraphen haben kann, muss ihm nur bekannt sein, unter welcher URL die Gesamtpräsentation verfügbar ist. Es müssen zwar immer noch mehrere Datenverbindungen aufgebaut werden, jedoch läuft dieser Mechanismus automatisiert ab.
- Kommen neue Teleoperatoren zum Szenario hinzu, wird diese Information ebenso automatisch über das MPEG-4 System an alle Teilnehmer verteilt. Gleiches gilt, wenn Teleoperatoren das Szenario verlassen.
- Allen Teilszenengraphen liegt ein gemeinsames Weltkoordinatensystem zugrunde. Bereits bei der Integration in die Gesamtpräsentationen muss bekannt sein, in welchem Bezug zum Weltkoordinatensystem die rekonstruierten Daten liegen. Den Operatoren werden, ganz im Sinne des HVA, die gesammelten Informationen der entfernten Umgebung in einem Stück und für alle Operatoren gleichsam wahrnehmbar präsentiert.
- Die Gesamtpräsentation kann, auch in ihrem Verlauf über die Zeit, als eine einzige Datei abgespeichert werden. Insbesondere zum Zweck der Versuchsauswertung können mit Hilfe einer solchen Datei sämtliche Elementary Streams zeitsynchron zueinander nochmals wiedergegeben werden.

Anhand von Abbildung 5.9 soll dazu das Funktionsprinzip von Objektdeskriptor-URLs in einem konkreten Anwendungsfall erläutert werden. Beide Teleoperatoren erstellen, jeder für sich, ein Teilmodell der entfernten Umgebung. Hier wird angenommen, dass das Teilmodell von Teleoperator 1 zugleich die Grundlage für die Gesamtpräsentation ist. Ein beliebiger Teilnehmer X will nun auf die gesammelten Daten zugreifen. Dazu öffnet er eine Verbindung zu Teleoperator 1, der ihm daraufhin einen initialen Objektdeskriptor `OD 1` zurückschickt. Über `OD 1` erfährt der Teilnehmer, wie er Zugriff auf den Elementary Stream `BIFS 1` für den Szenengraphen von Teleoperator 1 erhält. In diesem Szenengraphen ist ein `Inline` Knoten enthalten, der als Stellvertreter für die Sensordaten von Teleoperator 2 dient. Um nun tatsächlich Zugriff auf diese Sensordaten zu bekommen, sind mehrere zusätzliche Schritte notwendig:

1. Das `url`-Feld des `Inline` Knoten verweist auf den bisher unbekannten Objektdeskriptor `OD 2`.
2. Um `OD 2` zu erhalten, öffnet Teilnehmer X den Elementary Stream `ODS 1`. Die im Elementary Stream übertragenen OD-Kommandos beschreiben `OD 2`.
3. `OD 2` enthält nun anstatt einer Liste von ES-Deskriptoren eine Objektdeskriptor-URL. Diese verweist Teilnehmer X auf die Präsentation von Teleoperator 2.
4. Teilnehmer X öffnet daraufhin eine neue Verbindung zu Teleoperator 2. Nachdem er den initialen Objektdeskriptor `OD 3` von Teleoperator 2 empfangen hat, kann er

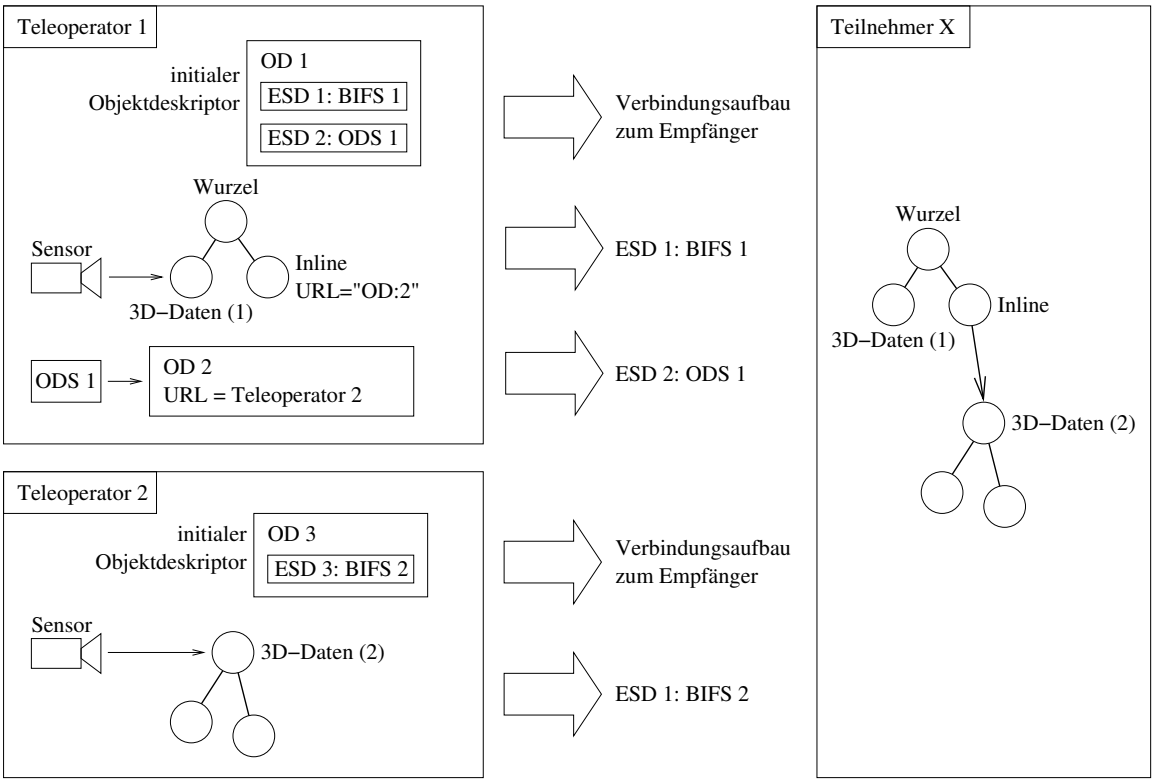


Bild 5.9: Beispiel für die Verteilung der Szenenrekonstruktion auf zwei Teleoperatoren

über `ESD 3` Zugriff auf die Szenendaten von Teleoperator 2 erhalten.

Der Gesamtszenengraph, wie ihn Teilnehmer X aus den dekodierten BIFS-Update Kommandos aus den Elementary Streams `BIFS 1` und `BIFS 2` zusammenstellen kann, enthält nun die gesammelten Sensordaten von beiden Teleoperatoren.

Weiterleitung von Medienknoten über die Barriere

Medienknoten, wie z. B. der `Inline` Knoten, verweisen auf weitere Inhalte, die über neue Verbindungen angefordert werden. Solange diese neuen Verbindungen nur zwischen Rechnern auf der Teleoperator- bzw. Operatorseite auftreten, werden nicht unnötig Daten über die Barriere übertragen.

Sobald sich jedoch die Datenquelle für einen Medienknoten auf der einen Seite der Barriere befindet und von mehreren Teilnehmern auf der anderen Seite benötigt wird, baut jeder Teilnehmer seine eigene Datenverbindung zur Quelle auf. Dass dabei dieselben Daten mehrfach über die Barriere übertragen werden, ist für den einzelnen Teilnehmer nicht ersichtlich. In Abbildung 5.10 ist zu sehen, welche Datenverbindungen für eine durch zwei Rechner erstellte Präsentation über die Barriere geleitet werden müssen.

Wie im vorangegangenen Beispiel stellt Teleoperator 1 über den Elementary Stream `BIFS 1` den Hauptszenengraph des Gesamtmodells zur Verfügung. Teleoperator 2 bindet sei-

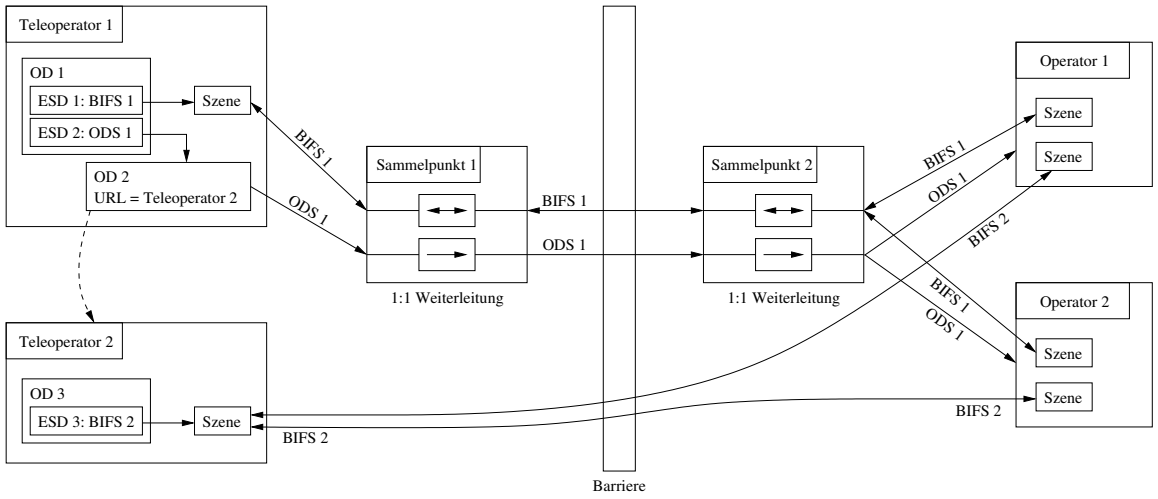


Bild 5.10: Von zwei Teleoperatoren erzeugte Präsentation ohne Maßnahmen zur Bandbreitenbegrenzung

nen Teilszenengraphen wieder über einen `Inline` Knoten ein. Auf Operatorseite erhalten beide Operatoren vom Sammelpunkt 2 genau dieselben Elementary Streams, wie sie bei Sammelpunkt 1 ankommen. Wollen die Operatoren nun den Inhalt des `Inline` Knoten darstellen, öffnen sie direkte Verbindungen zu Teleoperator 2, wodurch der Elementary Stream `BIFS 2` doppelt über die Barriere übertragen wird.

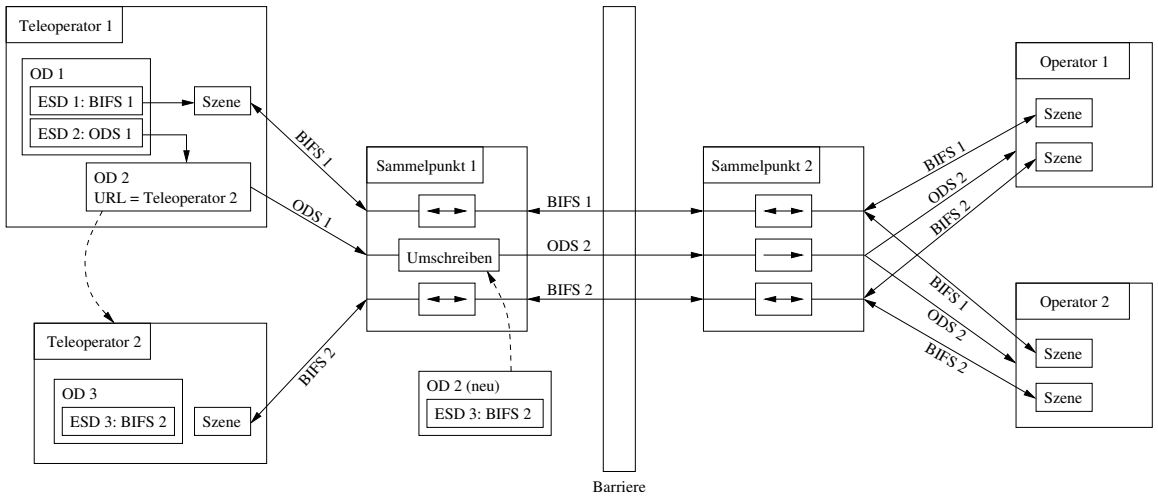


Bild 5.11: Von zwei Teleoperatoren erzeugte Präsentation mit geeigneten Maßnahmen zur Bandbreitenbegrenzung

Um die Daten in `BIFS 2` nur einmal über die Barriere zu übertragen, müssen mehrere Maßnahmen getroffen werden. Die Ursache für die mehrfache Übertragung liegt darin, dass im Objektdeskriptor `OD 2` die URL zu Teleoperator 2 eingetragen ist. Die Teilnehmer auf der Operatorseite bauen deshalb Verbindungen über die Barriere auf. Die vor-

geschlagene Lösung sieht eine Veränderung der OD-Kommandos bei der Weiterleitung in Sammelpunkt 1 vor. Weiterhin muss Sammelpunkt 1 beide BIFS Elementary Streams selber öffnen und für die Operatorseite in einer Präsentation anbieten. Der veränderte `OD 2` enthält nun keine URL mehr, sondern bietet `BIFS 2` als ganz normalen Elementary Stream an. Diese Maßnahmen sind in Abbildung 5.11 dargestellt.

Die Veränderung der OD-Kommandos muss in den Sammelpunkten stattfinden. Das Prinzip kann übrigens auch angewendet werden, wenn sich die Datenquelle auf der Operatorseite befindet. Sammelpunkt 1 und 2 tauschen dann ihre Rollen. In Abbildung 5.12 ist der zeitlichen Ablauf des Verbindungsaufbaus durch Nummern für das angeführte Beispiel dargestellt.

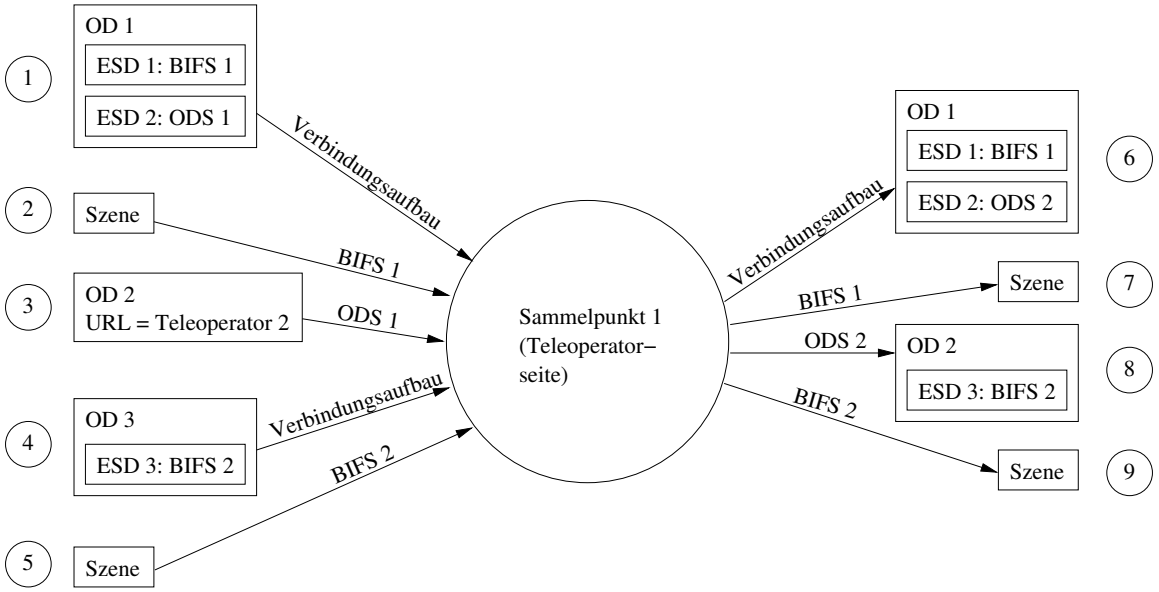


Bild 5.12: Zeitlicher Ablauf aller Vorgänge in Sammelpunkt 1

Zuerst verbindet sich Sammelpunkt 1 zu Teleoperator 1 und erhält daraufhin den initialen Objektdeskriptor `OD 1` (Schritt 1). Der Hauptszenengraph wird an Sammelpunkt 1 in Schritt 2 übertragen. Sammelpunkt 1 durchsucht den Szenengraphen nach Medienknoten, und verfährt wie bereits weiter oben geschildert: `ODS 1` anfordern, um die Daten von `OD 2` zu bekommen (Schritt 3), Verbindung mit Teleoperator 2 aufbauen, um `OD 3` zu erhalten (Schritt 4). Anschließen wird der Elementary Stream `BIFS 2` geöffnet, um den Teilszenengraph von Teleoperator 2 ebenfalls nach Medienknoten zu durchsuchen (Schritt 5). Diese Prozedur läuft solange weiter, bis Sammelpunkt 1 alle Elementary Streams für die Gesamtpräsentation geöffnet hat.

Verbindet sich nun Sammelpunkt 2 auf der Operatorseite zu Sammelpunkt 1, schickt ihm dieser den unveränderten initialen Objektdeskriptor `OD 1` von Teleoperator 1 (Schritt 6). Auch hier wird zuerst der BIFS Elementary Stream `BIFS 1` geöffnet (Schritt 7). Um auf die Daten des `Inline` Knoten zuzugreifen, muss Sammelpunkt 2 `ODS 2` öffnen (Schritt 8). `ODS 2` enthält prinzipiell dieselben Daten wie `ODS 1`, jedoch verändert Sammelpunkt 1 einige der OD-Kommandos. Jeder dort beschriebene Objektdeskriptor mit URL wird durch

einen Objektdeskriptor ersetzt, der ES-Deskriptoren für Elementary Streams enthält, die Sammelpunkt 1 selber anbieten kann.

Deshalb erhält Sammelpunkt 2 in Schritt 9 den `BIFS 2` Elementary Stream aus einer Verbindung mit Sammelpunkt 1. Alle Operatoren verbinden sich ihrerseits mit Sammelpunkt 2 und erhalten von diesem die unveränderten BIFS-Update und OD-Kommandos von Sammelpunkt 1. Da sich in den OD-Kommandos nun keine URLs mehr befinden, werden alle benötigten Daten nur noch über Sammelpunkt 2 bezogen.

5.2.5 Validierung der Vorgehensweise und Messergebnisse

Zur Validierung der in diesem Kapitel beschriebenen Konzepte:

- Rückkanäle für BIFS-Update Kommandos
- Optimierte Weiterleitung von Elementary Streams
- Weiterleitung von Medienknoten über die Barriere

wurde eine Implementierung eines MPEG-4 Systems erstellt, mit der die beschriebene Funktionalität der Konzepte überprüft werden konnte. Im nächsten Abschnitt werden Testszenario und Messmethode beschrieben, danach folgen Messergebnisse zu den erzielten Verzögerungszeiten.

Testszenario und Messmethode

Das den Experimenten zugrunde liegende Szenario stellt ein mögliches Telepräsenz-Szenario mit zwei Teleoperatoren und zwei Operatoren nach. Abbildung 5.13 zeigt schematisch die Teilnehmer und die übertragenen Daten. Teleoperator 1 erstellt hier nur Oberflächenmodelle der entfernten Umgebung, wofür die Daten eines PMD-Sensors verwendet werden. Es kam dieselbe Methodik zur Modellrekonstruktion zum Einsatz, wie sie bereits in Kapitel 4.1.2 beschrieben wurde. Der Szenengraph des rekonstruierten Modells ist der Grundbaustein der Gesamtpräsentation. Um Änderungen in der realen Umgebung im Modell abbilden zu können, erfolgt in periodischen Abständen eine Aktualisierung des Szenengraphen, bei der neue Oberflächen hinzukommen und alte Oberflächen gelöscht werden können.

Für Teleoperator 2 ist zur Manipulation von Werkstücken ein Roboterarm vorgesehen. Ein geometrisches Modell des Roboterarms wird hier in Form eines zweiten Szenengraphen angeboten. Dieser Szenengraph wird, ebenfalls periodisch, passend zur aktuellen Armposition des realen Roboters verändert.

Identisch zu den vorhergehenden Abschnitten ist Teleoperator 1 für den initialen Aufbau der Gesamtpräsentation verantwortlich. Er erzeugt den Hauptszenengraphen, in den über einen `Inline` Knoten der Szenengraph von Teleoperator 2 eingebunden ist. Der dazu notwendige ODS wird deshalb ebenfalls von Teleoperator 1 erzeugt. Verzögerungen bei der Übertragung auf dem ODS werden im Weiteren aufgrund des geringen und sporadischen

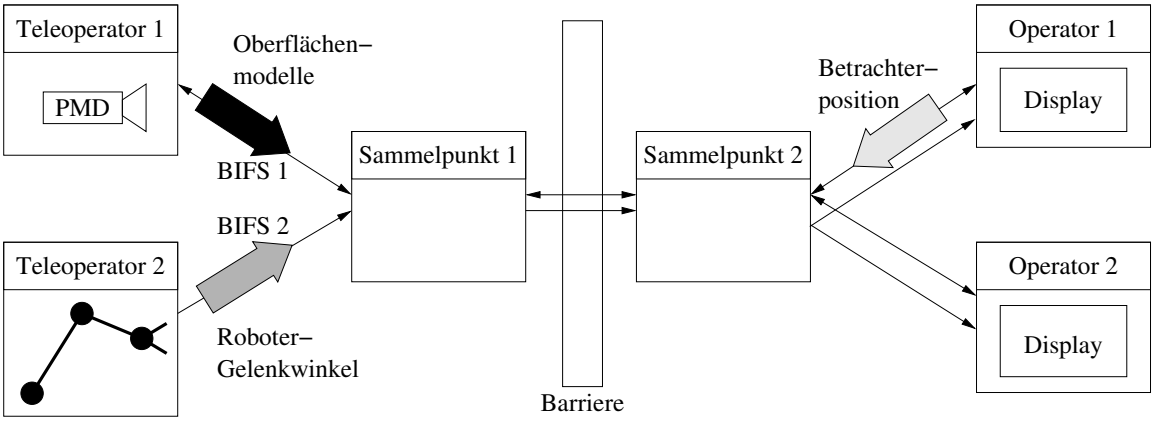


Bild 5.13: Versuchsszenario für die Messung von Verzögerungszeiten

Transfervolumens jedoch nicht betrachtet. In Sammelpunkt 1 wurden, wie in Abschnitt 5.2.4 beschrieben, die OD-Kommandos umgeschrieben. Die gewünschte Wirkung des vorgestellten Konzepts konnte verifiziert werden, da über die Barriere zwischen Sammelpunkt 1 und Sammelpunkt 2 nur drei Elementary Streams übertragen wurden: `BIFS 1`, `BIFS 2` und der ODS.

Auf der Operatorseite sind zwei Displays vorhanden. Auf beiden Displays sollen die Inhalte der Gesamtpräsentation dargestellt werden, um den Fortschritt der Telemanipulation verfolgen zu können. Damit Operator 2 stets weiß, von welcher Perspektive aus Operator 1 die Szene betrachtet, verschickt Operator 1 mit einer vorgegebenen Rate seine Betrachterposition. Er fügt dazu in den Szenengraphen von Teleoperator 1 einen zusätzlichen Knoten ein, der diese Information enthält. Auch diese Information wird mit einer vorgegebenen Rate ständig aktualisiert. Damit Operator 1 BIFS-Update Kommandos für den Szenengraphen von Teleoperator 1 verschicken darf, war hierfür ein Rückkanal vorgesehen.

Die Messungen im folgenden Abschnitt untersuchen nun die Zeit, die notwendig ist, um Daten von einer der drei möglichen Quellen zu Operator 2 zu übertragen:

- Änderungen im Szenengraphen von Teleoperator 1 sind mit einem hohen Datenvolumen von 816,3 KByte pro Tiefenbild verbunden. Die Rate des PMD-Sensors für neue Tiefenbilder wurde auf 5 Hz festgelegt.
- Um neue Gelenkwinkel von Teleoperator 2 zu übertragen werden nur 175 Byte benötigt. Jedoch werden die Gelenkwinkel hier mit einer wesentlich höheren Frequenz von 100 Hz übertragen. Die Frequenz spiegelt die real verwendete Regelfrequenz des Roboterarms wieder.
- Noch weniger Datenvolumen, als für die Übertragung der Gelenkwinkel, wird mit 38 Byte für die Übertragung der Betrachterpositionen von Operator 1 benötigt. Da viele Displays mit einer Bildwiederholrate im Bereich von 30 Hz arbeiten, wurde diese Frequenz hier verwendet. Anders als bei den Gelenkwinkeln erfolgt die Änderung der Betrachterposition im gleichen Szenengraphen wie die Änderungen des Umgebungsmodells. Die daraus resultierenden Auswirkungen werden später diskutiert.

Die Teilnehmer waren für die Experimente über Gigabit Ethernet miteinander verbunden. Jede Datenverbindung war in ihrer Rate auf jeweils $100 \cdot 10^6$ bit/s limitiert. Diese Limitierung war notwendig, da ansonsten Paketverluste auftraten. Für einzelne Datenübertragungen konnten mit der eingesetzten Hard- und Software maximal Raten von bis zu $800 \cdot 10^6$ bit/s erzielt werden, jedoch nicht wenn alle Teilnehmer gleichzeitig aktiv waren. Andererseits ist gerade für die Übertragung über die Barriere in den meisten Szenarien ebenfalls eine Bandbreitenlimitierung gegeben.

Gemessen wurde die Zeitdifferenz zwischen den Zeitpunkten genau vor der Enkodierung beim Sender und genau nach der Dekodierung beim Empfänger. Realisiert wurde die Zeitmessung über den *Composition Time Stamp* (CTS), der für jede Access Unit als Teil des SL-Headers übertragen werden kann. In diesen Zeitstempel wurde die Systemzeit der Datenquelle genau vor der Enkodierung abgelegt. Direkt nach der Dekodierung wurde dann die Differenz zwischen der aktuellen Systemzeit bei Operator 2 und dem mitgeschickten Zeitstempel protokolliert. Diese Vorgehensweise setzt eine präzise Synchronisierung der Systemzeiten auf den eingesetzten Rechnern voraus. Für diesen Zweck wurde der `ntp`-Dienst (Network Time Protocol) verwendet. Da mittels `ntp` in diesem Aufbau nur eine Genauigkeit der Synchronisierung bis knapp unter 1 ms erreicht werden konnte, sind die folgenden Messungen auch nur bis zu dieser Genauigkeit interpretierbar.

Erzielte Verzögerungszeiten

Verglichen werden im Folgenden vier verschiedene Vorgehensweisen zur Übertragung der Datenströme:

- **Direkte Übertragung:** Ganz ohne die Verwendung von Sammelpunkten ist es über eine Sternstruktur (siehe Abschnitt 5.1) möglich, die Elementary Streams von jedem Teleoperator direkt zu jedem Operator zu übertragen. Im gewählten Versuchsszenario werden dann sowohl Umgebungs- und Gelenkwinkeldaten, als auch die Betrachterpositionen doppelt über die Barriere übertragen. Im ersten Fall um die Daten an beide Operatoren zu schicken, im letzteren Fall werden die BIFS-Update Kommandos erst an Teleoperator 1 geschickt, um dann an Operator 2 weitergeleitet zu werden. Über dieses Vorgehen sollten die niedrigsten Verzögerungszeiten messbar sein.

- **Unoptimierte Weiterleitung:** Ebenso wie bei den zwei folgenden Vorgehensweisen werden hier die Sammelpunkte, wie sie in Abbildung 5.13 eingezeichnet sind, eingesetzt. Es wird hier eine Weiterleitung der Datenströme auf Basis von BIFS-Update Kommandos eingesetzt, wie sie in Abschnitt 5.2.2 beschrieben wurde.

- **Weiterleitung auf AU-Ebene:** Die Weiterleitung wird hier, wie in Abbildung 5.7 dargestellt, auf Basis von Access Units durchgeführt.

- **Weiterleitung auf SL-Ebene:** Hier kommt die in Abbildung 5.8 beschriebene Optimierung der Weiterleitung auf Basis von SL-Paketen zum Einsatz.

Zuerst wurden die Verzögerungszeiten für jeden Datentyp einzeln gemessen. Nur eine Datenquelle verschickte während der Messzeit Daten. Danach wurde die Messung wiederholt, wobei jedoch alle drei Quellen gleichzeitig aktiv waren. In den folgenden Diagrammen wird jeweils die relative Häufigkeit des Auftretens einer Verzögerungszeit dargestellt. Die verschiedenen Farben kodieren die vier Vorgehensweisen.

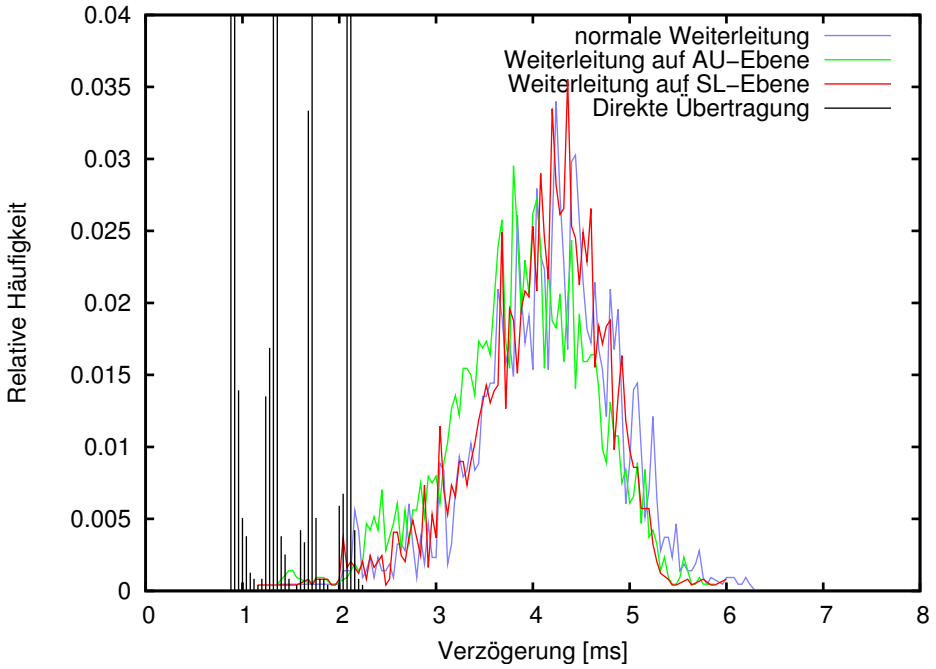


Bild 5.14: Verzögerung von exklusiv versendeten Gelenkwinkeldaten

Die Ergebnisse für die Übertragung von Gelenkwinkeländerungen sind in den Abbildungen 5.14 und 5.15 dargestellt. Aufgrund der geringen Datenmenge von 175 Byte pro Gelenkwinkeländerung wird das entsprechende BIFS-Update Kommando in einem einzigen Netzwerkpaket übertragen. Weiterhin können die Kommandos innerhalb sehr kurzer Zeit kodiert werden, weshalb sich die Verteilungen bei Vorgehensweisen mit Weiterleitung über Sammelpunkte nur sehr wenig voneinander unterscheiden. Deutlich sichtbar ist jedoch der Unterschied zur direkten Übertragung. Durch jeden Sammelpunkt wird eine zusätzliche Verzögerung von etwa 1 ms verursacht, was durch die Taktung der Betriebssystem-Kernels von 1 kHz erklärt werden kann. Für die direkte Übertragung liegt die wahrscheinlichste Übertragungszeit bei 2 ms (mit 32 % relativer Häufigkeit im Diagramm nicht mehr abgebildet). Die drei deutlichen Nebenmaxima bei 1,7 ms, 1,3 ms und 0,9 ms ergeben sich durch die Uhrensynchronisierung mittels `ntp`: während der Messung wurden die lokalen Uhren auf beiden Rechnern mehrfach in kleinen Schritten aktualisiert.

Für die Übertragung der Betrachterpositionen konnten ähnliche Verzögerungen gemessen werden. Hier werden bei der direkten Übertragung die BIFS-Update Kommandos erst zu Teleoperator 1 geschickt und erst anschließend an Operator 2. Deshalb verschlechtern sich die Verzögerungszeiten der direkt übertragenen Betrachterpositionen im Vergleich zu den direkt übertragenen Gelenkwinkeldaten um etwa 1 ms. Aus diesem Grund wurde hier

auch eine leicht andere Darstellung in den Abbildungen 5.16 und 5.17 gewählt, in der für die direkte Übertragung die Ordinate auf der rechten Seite des Diagramms gültig ist.

Die bisherigen Diagramme stellen nur den wahrscheinlichsten Bereich für Verzögerungen dar. In Abbildung 5.18 sind alle bisher gezeigten Messdaten nochmals für den vollständigen Wertebereich der Verzögerungen für den Fall abgebildet, dass alle Quellen aktiv waren. Neben den bereits betrachteten wahrscheinlichsten Verzögerungen treten für Gelenkwinkeldaten, vor allem jedoch für die unoptimierte Weiterleitung, einzelne Verzögerungen bis in den Bereich von etwa 45 ms auf. Diese werden durch Sammelpunkt 1 verursacht, der in etwa diese Zeit zum De- und Enkodieren eines BIFS-Update Kommandos für Umgebungsdaten benötigt (34 ms). Da alle Quellen gleichzeitig aktiv sind und nicht synchron zueinander laufen, kann der Fall auftreten, dass während ein BIFS-Update Kommando für Umgebungsdaten umkodiert wird ein SL-Paket eines BIFS-Update Kommandos für Gelenkwinkeldaten zur Bearbeitung bereit steht, jedoch keine CPU-Zeit vom Scheduler zugewiesen werden kann.

Für die Betrachterpositionen gibt es in Sammelpunkt 2 noch eine Situation, in der weitere Verzögerungen der Weiterleitung auftreten können. Wird in Sammelpunkt 2 ein BIFS-Update Kommando für Umgebungsdaten als eine Folge von SL-Pakete verschickt, kann kein SL-Paket eines BIFS-Update Kommandos für Betrachterpositionen in diese Folge eingefügt werden (Begründung siehe „Ausschluss parallelen Sendens“ in Abschnitt 5.2.3). Bei der gegebenen Bandbreitenlimitierung von $100 \cdot 10^6$ bit/s kann das nächste SL-Paket

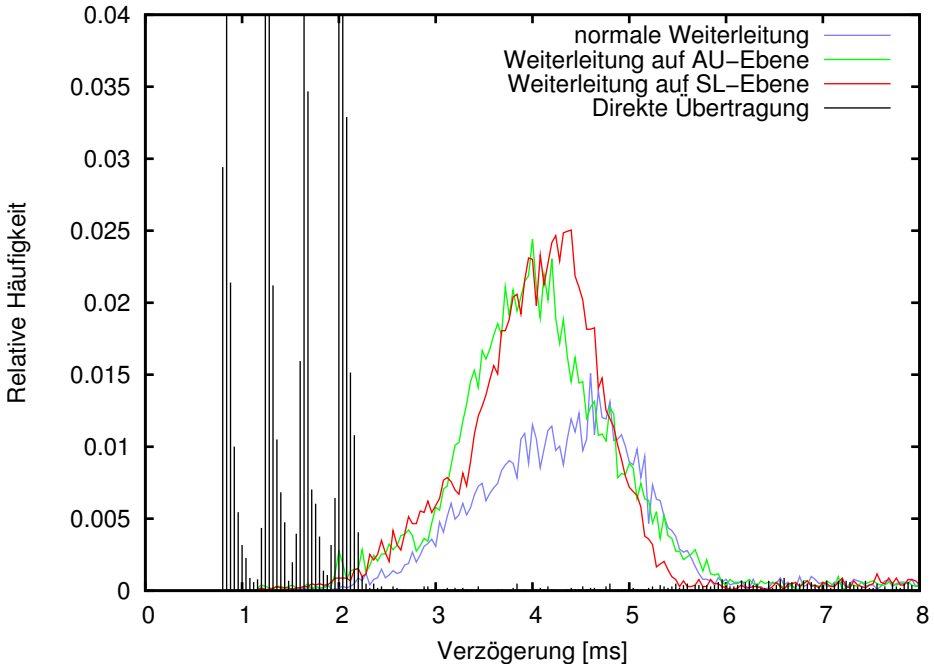


Bild 5.15: Verzögerung von Gelenkwinkeldaten, wenn alle Quellen aktiv sind

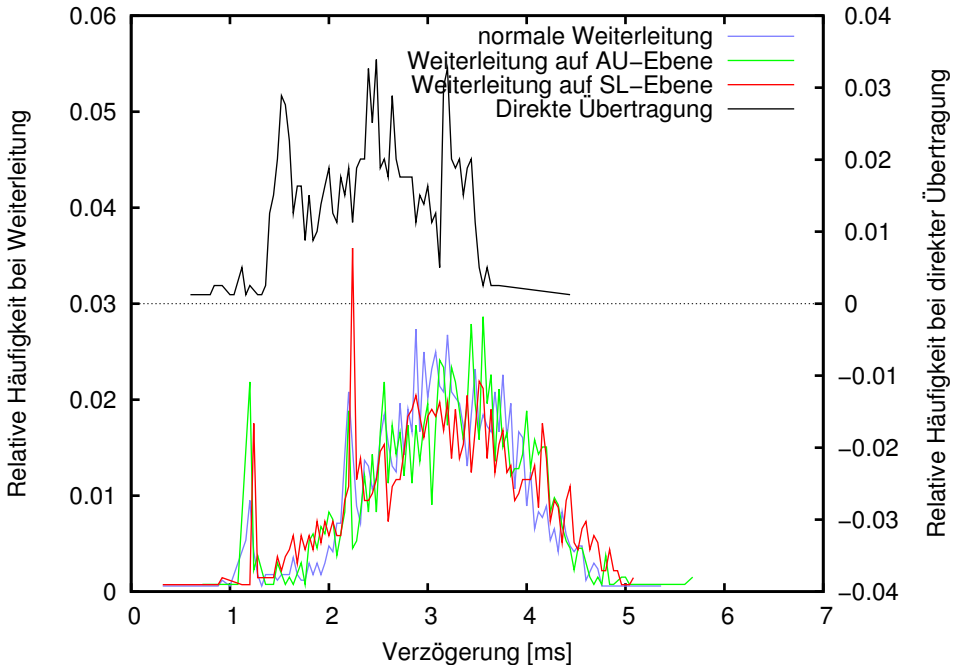


Bild 5.16: Verzögerung von exklusiv versendeten Betrachterpositionen

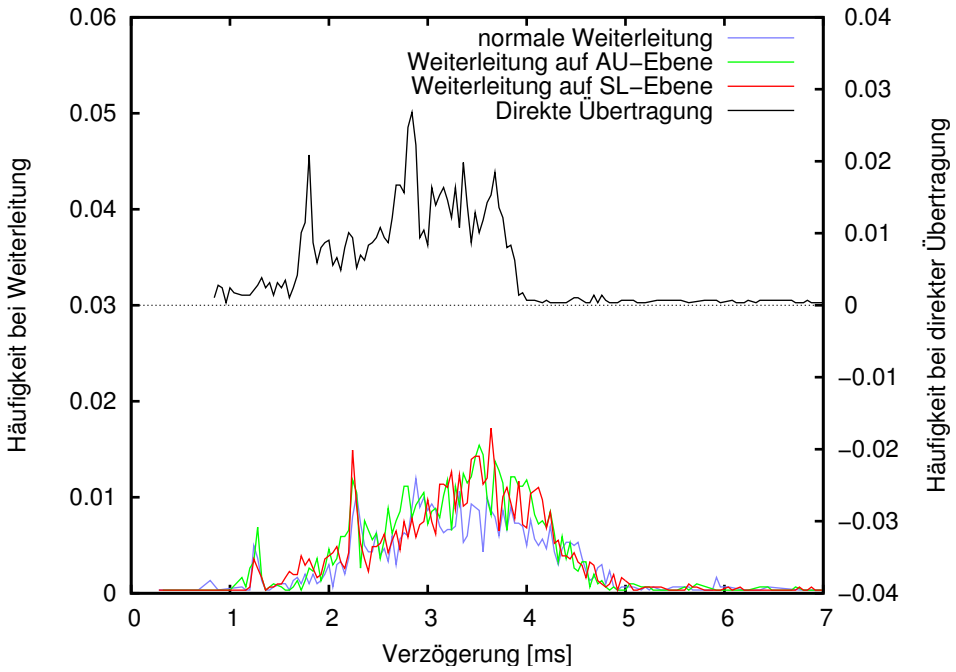


Bild 5.17: Verzögerung von Betrachterpositionen, wenn alle Quellen aktiv sind

in ungünstigen Fällen erst nach

$$\frac{816,3\,KByte}{100 \cdot 10^6\,\mathrm{bit/s}} = 66,9\,ms$$

verschickt werden. Diese Situation betrifft alle Vorgehensweisen, wie auch in Abbildung 5.18 im rechten Diagramm als zusätzliche Verzögerung zu beobachten ist.

Für die Übertragung von Umgebungsdaten (siehe Abbildungen 5.19 und 5.20) unterscheiden sich die Verzögerungszeiten für exklusives oder gleichzeitiges Senden wieder nur unmerklich. Da die anderen Access Units sehr klein sind, müssen Access Units für Umgebungsdaten im Verhältnis zu ihrer absoluten Verzögerung nur sehr kurz auf eine Bearbeitung durch den Sammelpunkt oder Teleoperator warten. Wie zu erwarten war, liegt hier jedoch ein großer Unterschied zwischen den Verzögerungszeiten der direkten Übertragung und dem unoptimierten Weiterleiten vor. Verursacht wird dies durch die zusätzlichen Kodier- und Paketiervorgänge in beiden Sammelpunkten. Da bei der Weiterleitung auf AU-Ebene zumindest die zusätzliche Kodierung entfällt, tritt eine geringere Verzögerung auf. Jedoch kann eine Weiterleitung auch hier erst dann erfolgen, wenn die vollständige Access Unit empfangen wurde. Weiter oben wurde bereits berechnet, dass durch die Bandbreitenlimitierung das Versenden einer Access Unit für Umgebungsdaten 66,9 ms dauert. Da beide Sammelpunkte diese Zeit abwarten müssen, erklärt dies den größten Teil der gemessene Differenz von 144 ms zwischen direkter Übertragung und Weiterleitung auf AU-Ebene. Die Differenz von 68 ms zwischen Weiterleitung auf AU-Ebene und unoptimierter Weiterleitung entspricht der Zeit, die in beiden Sammelpunkten für die De- und Enkodierung benötigt wird.

Die optimierte Weiterleitung auf SL-Ebene kommt der direkten Übertragung sehr nahe. Die Verteilungen der gemessenen Verzögerungen sind nur um wenige Millisekunden gegeneinander verschoben, was in den eingebetteten Diagrammen in den Abbildungen 5.19 und 5.20 vergrößert dargestellt ist.

Dass die gemessenen Verzögerungen ohne Optimierung und auf AU-Ebene sich direkt proportional zum Kodier- und Paketieraufwand verhalten, belegt Abbildung 5.21. Für diese Messung wurden ausschließlich Umgebungsdaten von Teleoperator 1 verschickt. Die Variation der Größe der Access Units wurde durch eine zufällig gewählte, künstliche Beschränkung der Auflösung des PMD-Tiefenbildes realisiert. Die sporadischen Ausreißer,

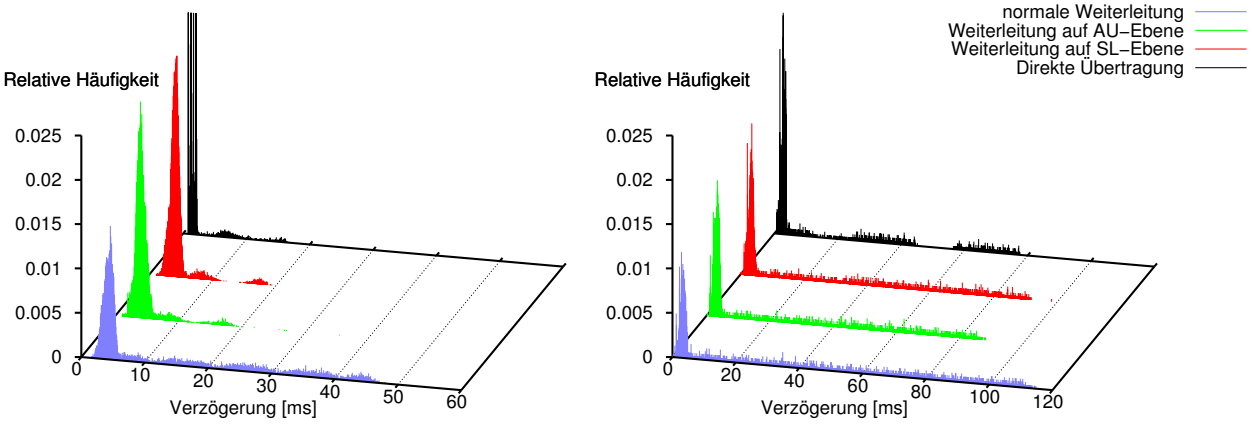


Bild 5.18: Vollständiger Wertebereich der gemessenen Verzögerungszeiten. Links für Gelenkwinkeldaten, rechts für Betrachterpositionen. Alle Quellen waren aktiv.

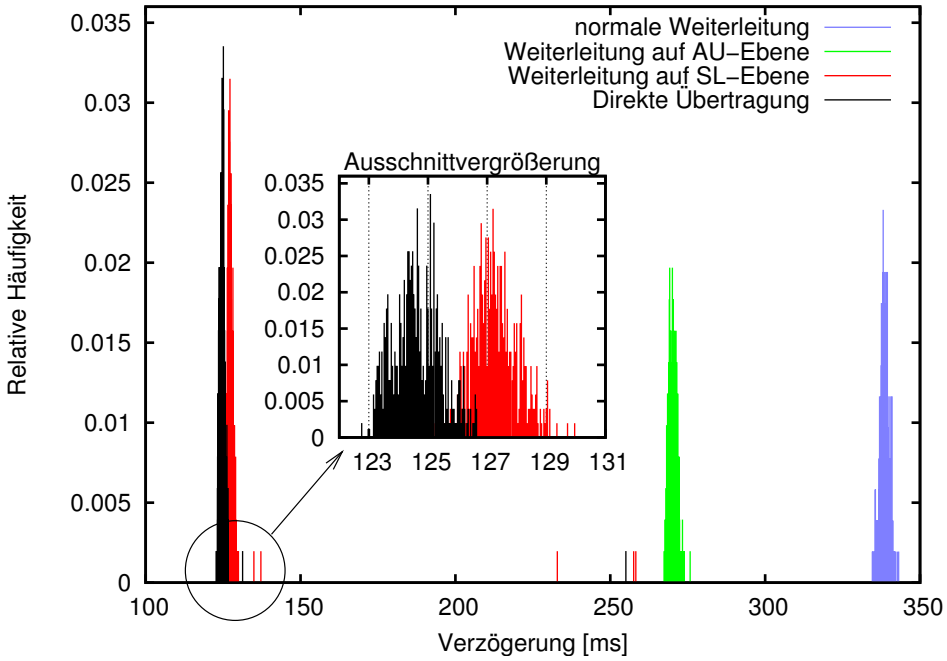


Bild 5.19: Verzögerung von exklusiv versendeten Umgebungsdaten

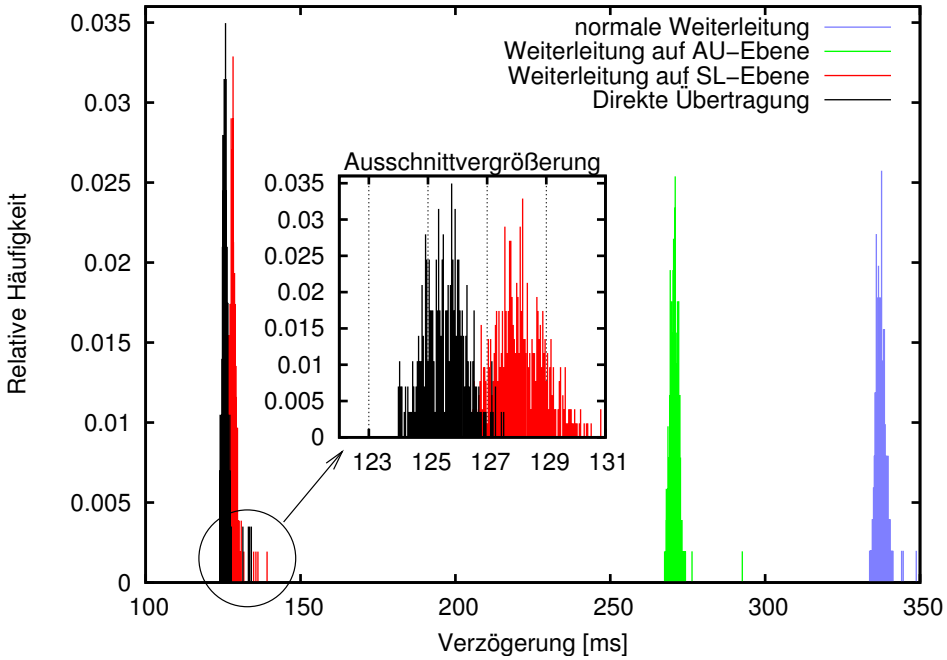


Bild 5.20: Verzögerung von Umgebungsdaten, wenn alle Quellen aktiv sind

die bei jeder Vorgehensweise auftreten, sind durch andere laufende Betriebssystemprozesse auf den Rechnern zu erklären. Als Anhaltspunkt ist in Abbildung 5.21 noch die, allein durch die Bandbreitenlimitierung verursachte, theoretisch minimale Verzögerungszeit ein-

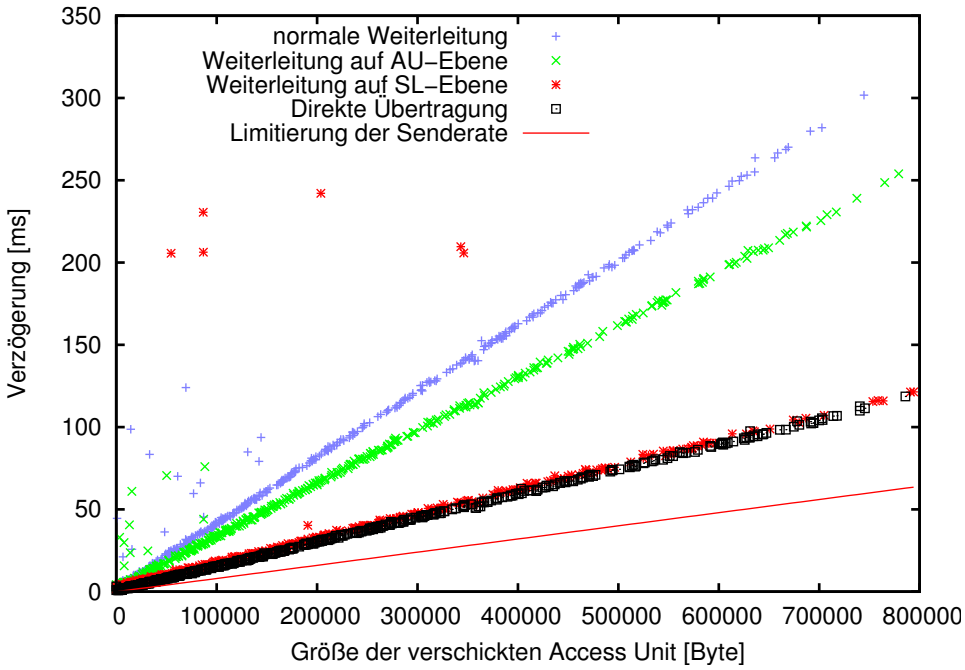


Bild 5.21: Verzögerung von Umgebungsdaten bei variabler Größe der Access Units

gezeichnet. Der Abstand zwischen dieser unteren Grenze und der direkten Übertragung bzw. Weiterleitung auf SL-Ebene resultiert aus der für die Kodierung in der Quelle und Dekodierung in Teleoperator 2 notwendigen Rechenzeit.

Zusammenfassend kann gesagt werden, dass durch die Weiterleitung über zwei Sammelpunkte nur eine kleine zusätzliche Verzögerungszeit im Bereich weniger Millisekunden in Kauf genommen werden muss. Dies gilt jedoch nur für die Weiterleitung auf SL-Ebene. Es ist deshalb ratsam, für alle Verbindungen stets identische Kodier- und Übertragungsparameter zu wählen. Für zeitkritische Informationsübertragungen sollten eigene Elementary Streams verwendet werden, damit große Access Units nicht die Weiterleitung der kleineren Access Units verzögern. Am Beispiel der Umgebungsdaten und Betrachterpositionen war dieser Effekt zu beobachten.

5.3 Erweiterung: Level of Detail mit blickpunktabhängigem Bandbreitenbedarf

In den vorhergehenden Abschnitten wurde beschrieben, wie durch den Einsatz des Object Descriptor Frameworks in HVA-Telepräsenz-Szenarien der Aufwand zur Rekonstruktion und Kodierung der Raumgeometrie der entfernten Umgebung auf mehrere Teleoperatoren verteilt werden kann. Dadurch können weiträumigere oder komplexere Umgebungen im gleichen Zeitraum rekonstruiert werden.

Diese Möglichkeit der Skalierung gilt nur für die Teleoperatorseite. Auf Operatorseite muss bei jedem Operator weiterhin das gesamte rekonstruierte Umgebungsmodell visualisiert werden. Im Folgenden soll nun betrachtet werden, wie mit Mitteln des MPEG-4 Standards weitere Möglichkeiten der Kodierung genutzt werden können, um ausgedehnte oder komplexe Raumgeometrien mit geringerem Rechenaufwand beim Operator darzustellen.

5.3.1 Vorgehensweise

Die unter dem Begriff *Level of Detail* (LOD) weitverbreitete Methode zur situationsangepassten Reduktion der Geometriekomplexität dient hier als Grundlage. Hierbei werden für komplexe Objekte mehrere 3D-Modelle mit jeweils unterschiedlichem Detaillierungsgrad erzeugt. Abhängig von der Betrachterposition wird genau eines dieser Modell tatsächlich gezeichnet. Je weiter entfernt das Objekt vom Betrachter liegt, desto kleiner wird der Detaillierungsgrad gewählt. Für dieses Verfahren wurde eigentlich in BIFS der `LOD` Knoten spezifiziert, der jedoch hier nicht verwendet wird.

Wie bereits in Kapitel 3.4 erwähnt wurde, können in einem Objektdeskriptor mehrere Elementary Streams angegeben werden, die alternative Kodierungen desselben Inhalts anbieten. Anwendung findet dies normalerweise bei der mehrsprachigen Vertonung von Videofilmen. Übertragen auf Szenendaten bedeutet dies, dass alternative Kodierungen derselben Umgebung in einem Objektdeskriptor als BIFS Elementary Streams angeboten werden können. Es wird im Folgenden angenommen, dass auf der Teleoperatorseite mehrere dieser verschiedene Kodierungen gleichzeitig generiert werden können, die sich in Detaillierungsgrad und Bitrate voneinander unterscheiden. Die einfachste Kodierung beschreibt die entfernte Umgebung mit sehr wenigen Geometrieelementen. Die komplexeste Kodierung beinhaltet hingegen alle rekonstruierbaren Details.

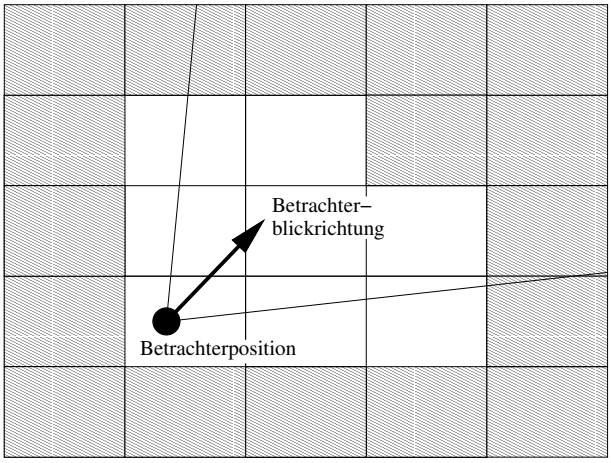


Bild 5.22: Bereiche mit einfacher (schraffiert) und komplexer Kodierung. Die Auswahl ist abhängig von der Betrachterperspektive.

Weiterhin wird angenommen, dass die entfernte Umgebung in mehrere Bereiche einge-

teilt werden kann, wie z. B. in Abbildung 5.22 dargestellt. Genau wie bei der klassischen LOD Methode kann nun der Operator auswählen, welche der alternativen Elementary Streams für eine bestimmte Betrachterposition benötigt werden. Im Beispiel in Abbildung 5.22 dienen als Kriterien für die Auswahl sowohl Position als auch Blickrichtung des Betrachters.

Werden für viele Bereiche die einfacheren Kodierungen verwendet, reduziert sich auf der Operatorseite sowohl der Rechenaufwand für die Generierung einer 3D-Ansicht, als auch der benötigte Speicherplatz für das Gesamtmodell. Bei Verwendung des `LOD` Knotens hingegen wird mehr Speicher für das Gesamtmodell benötigt, da immer alle Detailstufen in Form von 3D-Modellen vorliegen. Bei einer Änderung in der Umgebung müssen auch wieder alle Detailstufen für den betroffenen Bereich erneut übertragen werden.

Bei einer Veränderung der Betrachterposition hingegen kann ein Wechsel der Elementary Streams für einen oder mehrere Bereiche notwendig werden. In diesem Fall zeigen sich die Schwächen des vorgestellten Verfahrens, da bei jedem Elementary Stream Wechsel der vollständige Datensatz des neuen Elementary Stream über ein `SceneReplace` BIFS-Update Kommando verschickt werden muss.

5.3.2 Das Testszenario

Für das beschriebene LOD Verfahren mit alternativen Elementary Streams wurde ein Testszenario implementiert. Abbildung 5.23 zeigt, wie der Szenengraph aussieht, mit dem die vorgestellte Vorgehensweise realisiert wurde.

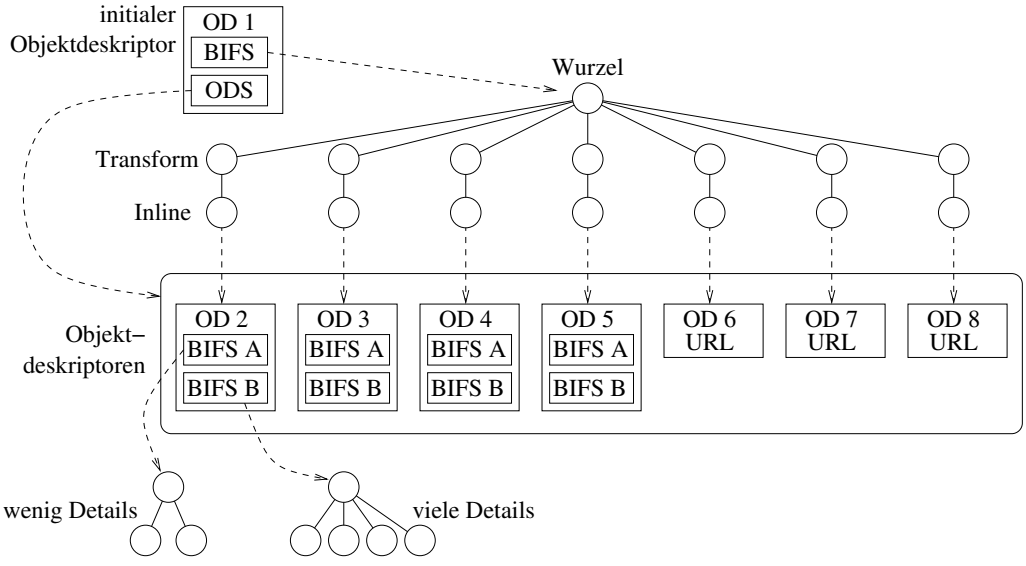


Bild 5.23: Aufbau des Gesamtszenengraphen und der Objektdeskriptoren für die Raumbereiche

Ähnlich wie bei der Zerlegung in Raumwürfel in Kapitel 4.1.4 modellieren die `Transform`

Knoten direkt unterhalb dem Wurzelknoten die Einteilung der entfernten Szene in einzelne Bereiche. Unterhalb jedes `Transform` Knotens beginnen über `Inline` Knoten neue Teilszenengraphen. Jeder dieser `Inline` Knoten verweist auf einen Objektdeskriptor, in dem alle verfügbaren alternativen Elementary Streams für den Inhalt des Raumbereichs beschrieben sind. Über Objektdeskriptor-URLs kann zusätzlich, wie bereits in den vorhergehenden Abschnitten beschrieben, die Erzeugung der einzelnen Datenströme auf der Teleoperatorseite auf mehrere Rechner verteilt werden.

Die Sensordaten für die Rekonstruktion der Geometrie wurden wieder mit Hilfe eines PMD-Sensors gewonnen. Dieser Sensor wurde mit einer Frequenz von 5 Hz abgefragt und mit einer bekannten Winkelgeschwindigkeit um seinen Mittelpunkt gedreht, wie in Abbildung 5.24 auf der linken Seite dargestellt. Durch die Drehung und das periodische Auswerten der Sensordaten wurde erreicht, dass sich Modelländerungen in vielen verschiedenen Raumbereichen während eines Messdurchgangs ergeben. Die vom Sensor erfasste Umgebung wurde durch Raumbereiche modelliert, die insgesamt eine Fläche von 5,5 m auf 5,5 m abdecken. Für die nachfolgenden Messungen wurden anhand der Sensordaten zwei alternative Kodierungen des Modells erzeugt. Während für eine Kodierung sämtliche verfügbaren Tiefenwerte des Sensors verwendet wurden, standen für die zweite Kodierung nur $\frac{1}{64}$ der Tiefenwerte zur Verfügung.

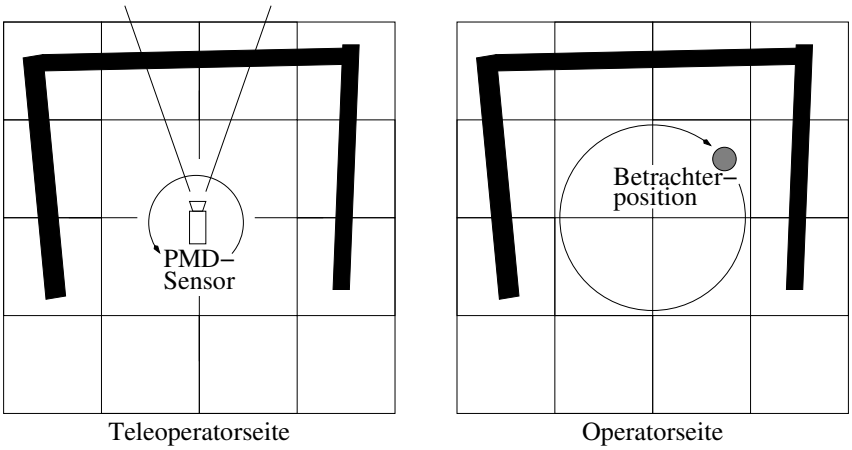
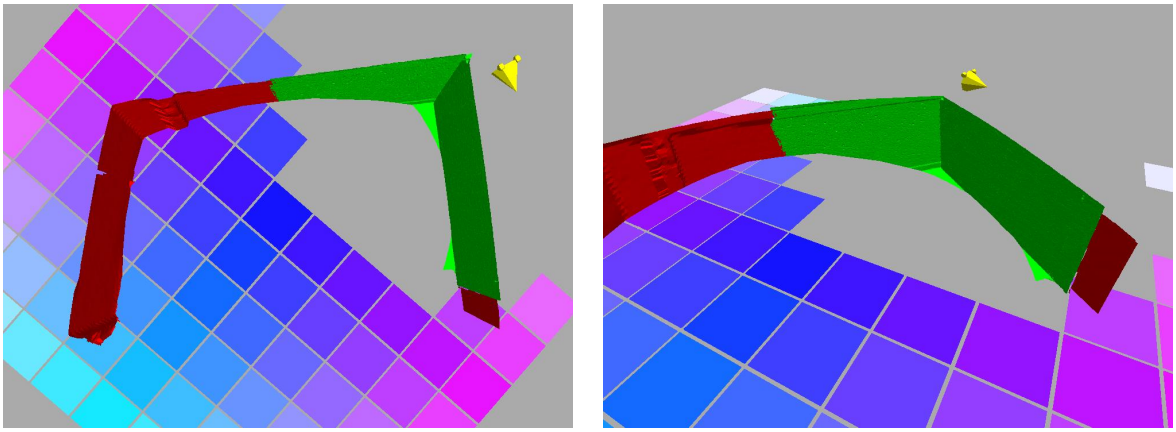


Bild 5.24: Der Messaufbau: Links ist der sich drehende PMD-Sensor in der realen Umgebung zu sehen, rechts befindet sich der sich ebenfalls bewegende Betrachter in der rekonstruierten, virtuellen Umgebung.

Auch die Operatorposition wurde mit bekannter Winkelgeschwindigkeit und Radius um den Mittelpunkt des rekonstruierten 3D-Modells gedreht. Die Auswahl zwischen den beiden verfügbaren Kodierungen erfolgte alleine Aufgrund des Abstandes zwischen Betrachterposition und Zentrum jedes Raumbereichs. Je nach gewählter Winkelgeschwindigkeit des Operators und Größe der Raumbereiche ergab sich daraus unterschiedlich oft die Notwendigkeit, die Elementary Streams zu wechseln.

Abbildung 5.25 zeigt eine 3D-Ansicht der rekonstruierten Umgebung. Der Betrachterstandpunkt des Operators ist durch den kleinen gelben Kegel gekennzeichnet. Auf dem



(a) Draufsicht (b) Schrägansicht

Bild 5.25: Beispielansichten des Messszenarios

Boden sind die Raumbereiche durch Quadrate visualisiert. Wenn kein Quadrat sichtbar ist, wurde für diesen Raumbereich die detailreiche Kodierung ausgewählt. Die Modelldaten sind entweder Grün oder Rot dargestellt. Grün steht für die detailreiche, Rot für die wenig detaillierte Kodierung. Die dargestellte Betrachterposition entspricht ungefähr der Position, die in Abbildung 5.24 rechts eingezeichnet ist.

5.3.3 Messergebnisse

Um das vorgeschlagene Verfahren mit dem klassischen LOD vergleichen zu können, wurden auch hierzu Messungen durchgeführt. In Tabelle 5.2 ist festgehalten, welche Datenraten für die gemeinsame Übertragung beider Kodierungen zusammen auftraten. Dies entspricht der klassischen Herangehensweise für LOD. Die sehr geringen Unterschiede in der genutzten Bandbreite bei unterschiedlicher Anzahl und Größe der Raumbereiche lassen darauf schließen, dass diese Variation das klassische LOD nur wenig beeinflusst.

Seitenlänge eines Raumbereichs [m]	5,5	1,1	0,5	0,22
Anzahl Raumbereiche	1	25	121	625
genutzte Bandbreite [MByte/s]	6.6	6.5	6.4	6.5

Tabelle 5.2: Benötigte Bandbreite für das klassische LOD-Verfahren

Tabelle 5.3 zeigt im Vergleich dazu die gemessenen Datenraten für das vorgeschlagenen Verfahren. Bei günstiger Wahl von Winkelgeschwindigkeit des Operators und der Bereichsgröße wird weniger Bandbreite als für das klassische LOD Verfahren benötigt. Andernfalls können jedoch auch weitaus höhere Bandbreiten notwendig sein. Je schneller sich der Operator bewegt, desto mehr Wechsel der Elementary Streams werden notwendig. Die Größe der Raumbereiche hat ebenfalls einen Einfluss auf die benötigte Bandbreite. Je größer die einzelnen Raumbereiche sind, desto mehr Daten werden bei einem Elementary

Winkelgeschwindigkeit des Operators [rad/s]	Seitenlänge [m]		
	1,1	0,5	0,22
0,05	5.1	3.1	4.2
0,1	6.8	3.8	5.2
0,2	11.3	5.2	8.5
0,5	18.1	8.2	14.7
1,0	24.4	13.2	19.0

Tabelle 5.3: Benötigte Bandbreite in MByte/s für verschiedene Bereichsgrößen und Winkelgeschwindigkeiten des Operators

Stream Wechsel über das `SceneReplace` BIFS-Update Kommando übertragen. Für die im Versuch verwendete Umgebung erscheint eine Bereichsgröße mit 0,5 m Seitenlänge am günstigsten.

Es zeigt sich, dass die Wahl der Raumbereiche passend zur Bewegung des Operators ausgelegt werden müsste. Für reale Telepräsenz-Szenarien kann z. B. ein Bereich ein Zimmer eines Hauses darstellen. Der Grund für die Verwendung dieser LOD Realisierung muss jedoch nicht unbedingt nur die benötigte Bandbreite sein. Das detailreiche Kodierungsverfahren erzeugt für das gesamte 3D-Modell der entfernten Umgebung ca. 1,38 Millionen Dreiecke, die wenig detaillierte Kodierung etwa 20,1 tausend Dreiecke. Während den Messungen zu dem hier beschriebenen Verfahrens wurden für eine 3D-Ansicht maximal 680 tausend Dreiecke gezeichnet und im Speicher vorgehalten. Die klassische LOD Implementierung jedoch würde stets 1,4 Millionen Dreiecke speichern müssen.

5.4 Zusammenfassung

In diesem Kapitel wurde dargestellt, wie das MPEG-4 Object Descriptor Framework in Telepräsenz-Szenarien eingesetzt werden kann. Dazu wurde vorgestellt, wie über Rückkanäle ein wechselseitiger Austausch von Szenendaten realisiert werden kann. Dieser wird benötigt, wenn mehr als nur ein Teilnehmer Veränderungen am Modell vornehmen will. Greifen mehr als zwei Teilnehmer auf ein gemeinsames Modell zu, müssen die Inhalte der Elementary Streams an alle Teilnehmer weitergeleitet werden. Auch hierzu wurde ein Konzept vorgestellt, das unter der Voraussetzung gleicher Parametrierung der Elementary Streams auch gut optimiert werden kann. Das Objekt Descriptor Framework bietet die Möglichkeit an, dass mehrere Teleoperatoren gemeinsam ein Gesamtmodell der entfernten Umgebung erstellen. Eine Methode wurde beschrieben, mit der Elementary Streams nicht unnötig mehrfach über die Barriere übertragen werden müssen. Durch Messungen konnte zum einen belegt werden, dass die vorgestellten Konzepte in der Praxis realisierbar sind, und zum anderen, dass die Weiterleitung der Elementary Streams auf Basis von SL-Paketen nur zusätzliche Verzögerungen Bereich weniger Millisekunden benötigt.

Wird das Object Descriptor Framework in einem Telepräsenz-Szenario eingesetzt, eröffnen

sich dadurch auf einfache Weise neue Möglichkeiten. Als Beispiel wurde eine Level of Detail Realisierung beschrieben, die im Vergleich zur klassischen Level of Detail Implementierung auf Seite der Operatoren weniger Speicherplatz benötigt. Unter günstigen Vorraussetzungen benötigt sie ebenfalls weniger Bandbreite zur Übertragung der Geometriedaten.

6 Zusammenfassung und Ausblick

In dieser Arbeit wurde vorgestellt, wie der MPEG-4 Standard in Telepräsenz-Szenarien eingesetzt werden kann. Insbesondere die Übertragung einer geometrischen Beschreibung der entfernten Umgebung zur Operatorseite steht hier im Vordergrund.

Die Einsatzmöglichkeiten von MPEG-4 wurden im Hinblick auf folgende Ziele untersucht:

- Reduzierung der benötigten Bandbreite für die Geometriedatenübertragung über die Barriere zwischen Teleoperator- und Operatorseite.
- Bereitstellung der Daten der entfernten Umgebung gleichermaßen für alle Operatoren in einer gemeinsamen Repräsentation. Dies entspricht dem Haptisch-Visuell-Auditorischen Arbeitsraum (HVA).
- Minimierung der Zeitdifferenz zwischen Datenerfassung auf Teleoperatorseite und Darstellung auf Operatorseite.

Die zur Verfügung stehende Bandbreite für Datenübertragungen zwischen Teleoperator- und Operatorseite ist für viele Szenarien begrenzt. Um den Verbrauch dieser Ressource für die Übertragung der Szenendaten in Grenzen zu halten wurde untersucht, welche Möglichkeiten zur Kompression der Szenendaten in MPEG-4 BIFS gegeben sind. Die grundlegenden BIFS Kompressionswerkzeuge wurden anhand von Daten aus einer realen Umgebung bewertet. Für das EfficientFloat Werkzeug wurden dazu zwei Strategien für die Enkodierung entwickelt. Weiterhin wurde vorgestellt, wie die BIFS Werkzeuge effizienter eingesetzt werden können. Unter günstigen Voraussetzung werden Kompressionsraten von $57,9 : 1$ erreicht. Die dafür benötigte Rechenzeit lässt eine schritthaltende Verarbeitung der Daten des eingesetzten Sensors zu.

Wenn mehrere Teleoperatoren und Operatoren kooperativ im Szenario eine gestellte Aufgabe bearbeiten, definiert der Haptisch-Visuell-Auditorische Arbeitsraum (HVA) die Gemeinsamkeit aller von den Teleoperatoren erfassten Daten der entfernten Umgebung. Mit Hilfe des MPEG-4 Object Descriptor Framework wurde eine Vorgehensweise vorgestellt, um den HVA als eine MPEG-4 Präsentation darzustellen. Alle Operatoren haben Zugriff auf diese Präsentation und damit Zugang zu allen Daten der entfernten Umgebung.

Damit die von den verschiedenen Teleoperatoren gesammelten HVA-Daten nicht zu jedem Operator einzeln über die Barriere übertragen werden, wurde weiterhin eine Kommunikationsstruktur entwickelt, bei der jeder Elementary Stream nur einmal über die Barriere übertragen wird. Für diese Kommunikationsstruktur wurde untersucht, welche zusätzlichen Verzögerungen zwischen Datenerfassung und Darstellung auftreten. Durch

eine Optimierung in der Datenpaketverarbeitung konnten die zusätzlichen Verzögerungen auf wenige Millisekunden reduziert werden.

Wird das MPEG-4 Object Descriptor Framework in einem Telepräsenz-Szenario verwendet, eröffnen sich weitere Anwendungsmöglichkeiten des MPEG-4 Standards. Am Beispiel einer Level of Detail Kodierung mit blickpunktabhängigem Bandbreitenbedarf wurde dies exemplarisch gezeigt.

Für alle vorgestellten Konzepte und Methoden wurde eine Implementierung erstellt, die zum einen die Realisierbarkeit der gewünschten Funktion belegt, und zum anderen die Durchführung von Messungen ermöglicht hat. Insbesondere die Messung der benötigen Ausführungszeiten, Übertragungszeiten und der erzielbaren Kompressionsraten für Daten einer realen Umgebung standen hier im Vordergrund. Die Implementierung konnte ebenfalls erfolgreich in mehreren Demonstratoren des Sonderforschungsbereichs 453 „Wirklichkeitsnahe Telepräsenz und Teleaktion“ eingesetzt werden.

Ausblick MPEG-4 Systeme

In dieser Arbeit werden nur Elementary Streams berücksichtigt, die Szenendaten enthalten. Der Schritt zu einer multimedialen Datenverarbeitung ist nicht groß, da der MPEG-4 Standard von sich aus die gemeinsame Übertragung von Szenen-, Video- und Audiodaten unterstützt. Jedoch können nicht alle in dieser Arbeit vorgestellten Konzepte direkt für Video- und Audiodaten übernommen werden. Da diese beiden Elementary Stream Typen nur Daten enthalten, die für genau einen Zeitpunkt gültig sind, macht es keinen Sinn Rückkanäle wie für Szenendaten einzuführen. Die bereits durch MPEG-4 definierten Rückkanalprotokolle spiegeln diesen Sachverhalt wieder, da über diese Protokolle nur die Enkodierung im Sendeterminal beeinflusst werden kann. Sollen über Video und Audio Elementary Streams trotzdem Daten zwischen den Operatoren ausgetauscht werden, muss jeder Operator in diesem Fall die Rolle des Sendeterminals für seine eigenen Daten übernehmen.

Wenn neue Teilnehmer zu einem bestehenden Szenario hinzukommen und ihre Daten der HVA-Präsentation hinzufügen wollen, muss dieser Fall bereits im Voraus eingeplant sein. Der Grund dafür liegt in der Verwaltung der Objektdeskriptoren im Sendeterminal der Gesamtpräsentation. Über ein Rückkanalprotokoll für OD-Kommandos könnte hier auf einfache Weise ein automatisiertes Verfahren für solche Fälle realisiert werden.

Schwieriger wird sich die Integration von Haptikdaten gestalten. Da diese Modalität in MPEG-4 bisher überhaupt nicht berücksichtigt wird, müsste hier ein eigenes Elementary Stream Format definiert werden. Laut Standard ist dies als „user private“ Elementary Stream möglich. Im Hinblick auf die Anforderungen für die Erfassung und Darstellung der haptischen Modalität, insbesondere deren typischerweise hohe Abtastfrequenz im Kilohertzbereich, können an dieser Stelle jedoch nur sehr unsichere Aussagen über eine praxistaugliche Realisierbarkeit gemacht werden. Sollte dies gelingen, könnten alle HVA-Daten in einer MPEG-4 Präsentation angeboten werden.

Ausblick MPEG-4 BIFS

Gerade im Hinblick auf die BIFS Kompressionswerkzeuge wird in dieser Arbeit das 3D Mesh Coding Werkzeug (3DMC) nicht behandelt. Der Grund dafür liegt in den Voraussetzungen, die ein Polygonnetz für 3DMC erfüllen muss: jede Polygonkante muss in genau zwei Polygonen enthalten sein. Zwar bietet der 3DMC Algorithmus Erweiterungen für Polygonnetze mit Rändern und Löchern, jedoch muss auf jeden Fall ein aufwendiger Vorverarbeitungsschritt über jedes neue Polygonnetz angewendet werden. Dies ist notwendig, da über die erfassten Sensordaten im Voraus keine Aussagen getroffen werden können. Für weitere Arbeiten ist die Untersuchung von 3DMC eines der wichtigsten Themengebiete.

Der MPEG-4 Standardisierungsprozess ist noch nicht abgeschlossen. In den Animation Framework eXtensions (AFX) [44] wird eine Vielzahl neuer Knoten definiert, die auch im Rahmen von Telepräsenz-Szenarien gut eingesetzt werden können. Insbesondere eine Untersuchung der AFX Konzepte im Bereich „Depth Image Based Representation“ bietet hier nahe liegende Ansatzpunkte. Hier kann die entfernte Umgebung als Voxelmodell übertragen und in Octree-Strukturen abgespeichert werden.

Bereits in Kapitel 4.2.3 wurde darauf hingewiesen, dass über eine besondere Sortierung der 3D-Punkte und Indizes das Predictive MFField Werkzeug wesentlich effizienter eingesetzt werden könnte. Für die Kodierung der 3D-Punkte konnte eine solche Sortierung, wie in Kapitel 4.5 beschrieben, gefunden werden. Eine günstige Sortierung der Indizes ist bisher nicht bekannt. Dies kann entweder über entsprechende Anpassungen bei der Modellerstellung oder durch einen Verarbeitungsschritt erst kurz vor der Kodierung erfolgen.

Weiterhin wurden ein paar Möglichkeiten von BIFS hier nicht untersucht, die zwar interessante Verwendungsmöglichkeiten in Telepräsenz-Szenarien hätten, jedoch keinen großen Einfluss auf die erzielte Kompressionsrate haben. Dazu gehört der ROUTE Mechanismus, die Verwendung von PROTO Knoten und das Kodieren von BIFS-Anim Elementary Streams. Über ROUTEs könnte der Präsentation, ähnlich wie bei handelsüblichen DVDs, ein Menü hinzugefügt werden, über das dem Operator erweiterte Kontrollfunktionen für den Teleoperator eingeblendet werden. Da insbesondere Gelenkwinkeldaten von Roboterarmen in regelmäßigen Zeitabständen erfasst werden, bieten sie sich für die Übertragung mit BIFS-Anim an. Die in dieser Arbeit nur kurz erwähnten PROTO Knoten könnten die Kodierung der entstehenden BIFS-Anim Elementary Streams nochmals effizienter gestalten.

A Anmerkungen zum MPEG-4 Standard

Im Laufe der Implementierung einzelner Komponenten des MPEG-4 Standards sind an ein paar Stellen Probleme aufgetreten, die in diesem Abschnitt angesprochen werden. Dazu kommen noch ein paar Situationen, auf die im Standard ausführlicher eingegangen werden könnte.

Verweisknoten referenzieren bisher nicht bekannte Knoten

Bei der Dekodierung eines Knotens wird als erstes bestimmt, ob der Knoten auf einen anderen verweist. Trifft dies zu, enthält der Bitstrom die Knoten-ID des Knotens, auf den verwiesen wird. Solange dieser andere Knoten bereits bekannt ist, treten keine Probleme auf. Andernfalls misslingt die Dekodierung, wenn der andere Knoten vom Typ `Coordinate` oder `Coordinate2D` bei eingeschaltetem `CoordIndex` Werkzeug ist, oder wenn der andere Knoten vom Typ `QP` ist.

Der Dekoder muss für das `CoordIndex` Werkzeug immer wissen, wie viele Elemente der zuletzt dekodierte `Coordinate` oder `Coordinate2D` Knoten enthalten hat. Das gilt auch, wenn der Knoten mehrfach verwendet wird. Ist der `Coordinate` Knoten noch nicht bekannt, ist die Anzahl der in ihm enthaltenen Elemente nicht bestimmbar. Die weitere Dekodierung wird mit hoher Sicherheit abbrechen müssen.

Die Mehrfachverwendung von `QP` Knoten wird explizit im Standard erlaubt. Wird auf einen bisher nicht bekannten `QP` Knoten verwiesen, können die darin enthaltenen Parameter nicht verwendet werden. Auch hier wird die weitere Dekodierung sehr bald abbrechen müssen.

Die Umgehung dieser Probleme liegt beim Enkoder. Dieser muss sicherstellen, dass erst die mehrfach verwendeten Knoten kodiert werden, und erst anschließend die Verweisknoten. Ein Hinweis auf die Problematik im Standard wäre nützlich gewesen. Die Referenzimplementierung [41] bricht die Dekodierung bereits ab, wenn auf einen unbekannten Knoten verwiesen wird.

Kodierreihenfolge der Felder bei `ListNodeDescription`

Die Felder eines Knotens können auf zwei Arten kodiert werden: `MaskNodeDescription` oder `ListNodeDescription`. Im zweiten Fall wird über ein Bit bestimmt, ob ein weiteres Feld kodiert wird, oder nicht. Im positiven Fall wird über eine festgelegte Anzahl von

Bits die Nummer des nächsten Feldes kodiert. Dadurch ist eine beliebige Abfolge für die Kodierung der Felder möglich. Wird das `CoordIndex` Werkzeug benützt, muss jedoch der `Coordinate` Knoten stets vor dem Indexfeld kodiert werden. Wieder liegt es am Enkoder, die Reihenfolge passend zu wählen.

Rangordnung EfficientFloat und Quantisierung

Im Text des Standards [43] wird festgelegt, dass das Quantisierungswerkzeug angewendet werden soll, wenn sowohl das EfficientFloat als auch das Quantisierungswerkzeug aktiviert sind. In der Referenzimplementierung des Standards [41] ist dies jedoch nicht der Fall. In der Datei `MQuantize.cpp` wird für alle Fließkommazahlen erst geprüft, ob sie mit EfficientFloat kodiert werden sollen. Nur wenn dieser Test negativ ausfällt, wird das Quantisierungswerkzeug verwendet. Die GPAC-Implementierung [51] hält sich, genauso wie die hier entwickelte Implementierung, an den Text des Standards.

Knoten und deren Namen

Im Text des Standards [43] steht, dass ein Knoten neben seiner Knoten-ID zusätzlich einen Namen in Form einer Zeichenkette haben kann. Die Referenzimplementierung berücksichtigt dies nicht. In der Datei `NodeFld.cpp` wird nur die Knoten-ID eingelesen, während sich die GPAC-Implementierung [51] wiederum an den Text im Standard hält.

Semantik für `FieldReplace` bei `SFNode` oder `MFNode` Feldern

Wird ein Knoten über ein `NodeDelete` oder ein `NodeReplace` BIFS-Update Kommando gelöscht, legt der Standard fest, dass alle Verweisknoten auf den gelöschten Knoten ebenfalls entfernt werden sollen. Wird ein Knoten über ein `IdxValueDelete` oder ein `IdxValueReplace` BIFS-Update Kommando entfernt, soll nur diese eine Instanz davon betroffen sein. Für die `FieldReplace` und `MultipleFieldReplace` BIFS-Update Kommandos fehlt eine Festlegung der Semantik, für den Fall dass ein `SFNode` oder ein `MFNode` Feld vollständig ersetzt wird.

Das `GlobalQuantizationConfiguration` BIFS-Update Kommando

Mit diesem Kommando kann ein global gültiger `QP` Knoten definiert werden. Dadurch wird eine in Kapitel 3.3 beschriebene Beschränkung für die Gültigkeit der `QP` Knoten umgangen. Das funktioniert in einem Szenario ohne Übertragungsfehler und mit nur einem Empfangsterminal auch ohne Komplikationen. Andernfalls müssen zwischendurch `SceneReplace` BIFS-Update Kommandos als sog. *Random Access Points* (RAP) verschickt werden. Nach dem Standard müsste für diese RAP ebenfalls der globale `QP` Knoten gelten, wodurch aber z. B. ein neu hinzugekommener Teilnehmer erst Wissen über den globalen `QP` Knoten erlangen muss. Folglich kann ein `SceneReplace` Kommando nicht mehr alleine ein RAP sein, sondern nur in Verbindung mit einem `GlobalQuantizationConfiguration` Kommando. Im Standard wird auf diese Situation jedoch nur unzureichend eingegangen. Eine Lösung könnte sein, den globalen `QP` Knoten bei Bedarf als Teil des `SceneReplace` Kommandos zu kodieren.

Das Predictive MFField Werkzeug

Der Standard enthält eine normative Beschreibung für den Algorithmus des arithmetischen Dekoders. Jedoch kann die dort enthaltene Beschreibung nicht direkt in eine Implementierung übernommen werden:

- Die Funktion `model_reset` initialisiert das Symbolmodell. Die Schleife

  ```
  for(i=1;i<=nbValues;++i) { ... }
  ```

  jedoch schreibt einen Wert zuviel. Sowohl die Referenzimplementierung als auch GPAC enthalten jedoch korrekte Grenzwerte für die Schleife.

- In der Funktion `update_model` wird bei Bedarf eine Neuskalierung der kumulativen Symbolhäufigkeitsstatistik auf die Hälfte vorgenommen. Dies ist notwendig, um Überläufe zu vermeiden. Die im Standard abgedruckte Schleife

  ```
  for(int i=nb_of_symbols-1; i>=0; i--) {
    cum += (cumul_freq[i]-cumul_freq[i+1]+1)/2;
    cumul_freq[i] = cum;
  }
  ```

  jedoch berechnet den Wert `cum` mit Hilfe bereits halbierter Werte.

- Der Standard erlaubt bis zu 2^{14} verschiedene Symbole. Es ist jedoch nicht möglich, mehr als 2^{13} Symbole zu verwenden. Die kumulative Symbolhäufigkeitsstatistik erreicht bei 2^{14} Symbolen bereits Werte, die über dem Schwellwert `q1`$=2^{14}$ in der Funktion `update_model` liegen. Als Folge davon müsste bereits eine unbenutzte Symbolhäufigkeitsstatistik halbiert werden. Da der Algorithmus zur Halbierung in `update_model` dies nicht erreichen kann, funktioniert der ganze Kodieralgorithmus nicht mehr.

- Der arithmetische Kodierer darf laut Standard nicht mehr als 22 aufeinanderfolgende nichtgesetzte Bits schreiben: nach einer Folge von 22 nicht gesetzten Bits muss ein gesetztes Bit eingefügt werden (Bitstuffing). GPAC verwendet eben diese Anzahl 22, während die Referenzimplementierung die Anzahl 40 verwendet.

- Trotz des bereits erwähnten Bitstuffing können durchaus längere Folgen von nichtgesetzten Bits auftreten. Auf das Ende eines arithmetisch kodierten Abschnitts im Bitstrom folgen weitere Bits durch andere kodierte Elemente. Auf die folgenden Bits hat der arithmetische Kodierer keinen Einfluss. In ungünstigen Fällen folgen auf einige nichtgesetzte Bits am Ende der arithmetischen Kodierung viele weitere nichtgesetzte Bits. Da der Dekoder immer 16 Bit des Bitstrom vorausliest, vermutet er an dieser Stelle eine Verletzung der oben erwähnten 22 Bit Regel. Somit darf eine Verletzung nicht unbedingt zum Abbruch der Dekodierung führen. Das Einfügen eines einzelnen gesetzten Bits zum Ende der arithmetischen Kodierung würde diesen Umstand beheben.

B Eingesetzte Hard- und Software

B.1 Kapitel 4: BIFS-Kompression

Für die in Kapitel 4 dargestellten Ergebnisse wurde für die Kodierung und Rekonstruktion der Modelldaten stets der folgende Rechner verwendet:

Prozessor:	Pentium-M
Taktfrequenz:	1,3 GHz
Hauptspeicher:	512 MByte
Betriebssystem:	Debian 4.0 „Etch“

Da der Prozessor zu Energiesparzwecken auch niedrigere Taktfrequenzen unterstützt, wurde sichergestellt, dass immer die maximal möglichen 1,3 GHz verwendet wurden. Weiterhin unterstützt der Prozessor SSE2 Befehle, was an zwei Stellen genutzt wurde. Einerseits um die Berechnung der linearen Quantisierung für alle drei Koordinaten eines 3D-Punktes parallel mit einfacher Fließkommagenauigkeit durchzuführen, andererseits um bei der Aktualisierung der kumulativen Symbolhäufigkeit im arithmetischen Kodierer acht kumulative Häufigkeiten parallel zu inkrementieren. Für Rechner, die nur die MMX Befehle unterstützen, konnte eine Implementierung entwickelt werden, die immerhin noch vier kumulative Häufigkeiten parallel inkrementiert.

Die Zeitmessungen wurden mit Hilfe des Systemaufrufs `gettimeofday()` realisiert. Da die ausgeführten Algorithmen im Userspace laufen und das eingesetzte Betriebssystem keinen Mechanismus bietet, um Unterbrechungen im Userspace zu vermeiden, sind alle Messungen nur Schätzungen der tatsächlichen Laufzeit. Um Beeinflussungen soweit wie möglich zu unterdrücken wurden die Messungen mit möglichst wenig parallel laufenden Anwendungen durchgeführt.

B.2 Kapitel 5: MPEG-4 angewendet im HVA

Für die Messungen in Kapitel 5 wurden mehrere Rechner eingesetzt. Jeder Rechner hat dabei die Rolle eines Teilnehmers im Telepräsenz-Szenario übernommen:

Rolle	CPU(s)	Speicher	Betriebssystem
Teleoperator 1	1 Athlon XP 1800+	512 MByte	Fedora Core 6 (32Bit)
Teleoperator 2	2 Pentium III 733MHz	512 MByte	Fedora Core 6 (32Bit)
Sammelpunkt 1	2 Xeon 3,2GHz	1 GByte	Fedora Core 6 (32Bit)
Sammelpunkt 2	1 Core2 Quad 2.66GHz	2 GByte	Fedora Core 6 (64Bit)
Operator 1	1 Athlon64 X2 5200+	2 GByte	Fedora Core 6 (64Bit)
Operator 2	1 Athlon64 3500+	1 GByte	Fedora Core 6 (64Bit)

Die schnellsten Systeme mit mehreren CPUs wurden bewusst als Sammelpunkte verwendet, um die erreichbaren Verzögerungszeiten möglichst gering zu halten. Dazu wurde die Software auch entsprechend mit mehreren Threads zum Senden, Empfangen und Kodieren ausgelegt. Der Rechner für Teleoperator 2 mag im Vergleich unangemessen erscheinen, jedoch war er seiner kleinen Aufgabe, die Gelenkwinkeländerungen alle 10 ms zu verschicken, durchaus gewachsen.

C MPEG-4 Details

Einige Details des Standards wurden in den Hauptkapiteln nicht ausführlich beschrieben. Für interessierte Leser werden hier mehr Informationen dazu gegeben.

C.1 Datenfelder der verwendeten Knoten

Die folgenden Tabellen enthalten alle datenbezogenen Felder für die in dieser Arbeit verwendeten Knoten. Felder, die nur für den ROUTE-Mechanismus relevant sind, erscheinen hier nicht.

Feldtyp	Feldname	Kurzerklärung
`MFNode`	`children`	Eine Liste von Knoten, die im Szenengraphen direkt unterhalb des `Group` Knotens angeordnet werden.

Tabelle C.1: Datenfelder eines `Group` Knotens

Feldtyp	Feldname	Kurzerklärung
`SFVec3f`	`center`	Eine Verschiebung des lokalen Koordinatensystems, die für Rotation und Skalierung angewendet wird.
`MFNode`	`children`	siehe `children` Feld im `Group` Knoten
`SFRotation`	`rotation`	Rotationsanteil der Transformation
`SFVec3f`	`scale`	Skalierungsanteil der Transformation
`SFRotation`	`scaleOrientation`	Bestimmt die Lage der Achsen, entlang denen skaliert wird.
`SFVec3f`	`translation`	Translationsanteil der Transformation

Tabelle C.2: Datenfelder eines `Transform` Knotens

Feldtyp	Feldname	Kurzerklärung
`SFNode`	`appearance`	Enthält einen Knoten, der die visuellen Attribute des Objekts beschreibt (Farbe, Transparenz,...).

Tabelle C.3: Datenfelder eines `Shape` Knotens (wird fortgesetzt ...)

Feldtyp	Feldname	Kurzerklärung
SFNode	geometry	Enthält einen Knoten, der die Geometrie des Objekts beschreibt.

Tabelle C.3: Datenfelder eines Shape Knotens

Feldtyp	Feldname	Kurzerklärung
SFNode	color	Enthält einen Knoten, in dem alle verwendeten Farben definiert sind.
SFNode	coord	Enthält einen Knoten, in dem alle verwendeten 3D-Punkte definiert sind.
SFNode	normal	Enthält einen Knoten, in dem alle verwendeten Normalenvektoren definiert sind.
SFNode	texCoord	Enthält einen Knoten, in dem alle verwendeten Texturkoordinaten definiert sind.
SFBool	ccw	Gibt an, ob die Polygoneckpunkte gegen den Uhrzeigersinn angegeben werden (counter clock wise).
MFInt32	colorIndex	Liste von Indizes, die Farben im color Feld referenzieren.
SFBool	colorPerVertex	Bestimmt, ob für jeden 3D-Punkt eine eigene Farbe in colorIndex angegeben ist. Ansonsten gilt: eine Farbe pro Polygon.
SFBool	convex	Gibt an, ob alle Polygone konvex sind.
MFInt32	coordIndex	Liste von Indizes, die 3D-Punkte im coord Feld referenzieren.
SFFloat	creaseAngle	Der Grenzwinkel, ab dem bei automatisch berechneten Normalen aneinander angrenzende Polygone mit weichen Übergängen gezeichnet werden sollen.
MFInt32	normalIndex	Liste von Indizes, die Normalenvektoren im normal Feld referenzieren.
SFBool	normalPerVertex	Bestimmt, ob für jeden 3D-Punkt ein eigener Normalenvektor in normalIndex angegeben ist. Ansonsten gilt: ein Normalenvektor pro Polygon.
SFBool	solid	Bestimmt, ob beide Polygonseiten gezeichnet werden sollen.
MFInt32	texCoordIndex	Liste von Indizes, die Texturkoordinaten im texCoord Feld referenzieren.

Tabelle C.4: Datenfelder eines IndexedFaceSet Knotens

Feldtyp	Feldname	Kurzerklärung
MFVec3f	point	Liste von 3D-Punkten

Tabelle C.5: Datenfelder eines Coordinate Knotens

Feldtyp	Feldname	Kurzerklärung
SFBool	isLocal	Gültigkeitsbereich der Quantisierungsparameter, siehe Abbildung 3.2
SFBool	position3DQuant	Aktiviert das Position3D Verfahren.
SFVec3f	position3DMin	minimale Position3D Wertgrenzen
SFVec3f	position3DMax	maximale Position3D Wertgrenzen
SFInt32	position3DNbBits	Anzahl der Position3D Quantisierungsbits
SFBool	position2DQuant	Aktiviert das Position2D Verfahren.
SFVec2f	position2DMin	minimale Position2D Wertgrenzen
SFVec2f	position2DMax	maximale Position2D Wertgrenzen
SFInt32	position2DNbBits	Anzahl der Position2D Quantisierungsbits
SFBool	drawOrderQuant	Aktiviert das DrawingOrder Verfahren.
SFFloat	drawOrderMin	minimale DrawingOrder Wertgrenze
SFFloat	drawOrderMax	maximale DrawingOrder Wertgrenze
SFInt32	drawOrderNbBits	Anzahl der DrawingOrder Quantisierungsbits
SFBool	colorQuant	Aktiviert das Color Verfahren.
SFFloat	colorMin	minimale Color Wertgrenze
SFFloat	colorMax	maximale Color Wertgrenze
SFInt32	colorNbBits	Anzahl der Color Quantisierungsbits
SFBool	textureCoordinateQuant	Aktiviert das TextureCoordinate Verfahren.
SFFloat	textureCoordinateMin	minimale TextureCoordinate Wertgrenze
SFFloat	textureCoordinateMax	maximale TextureCoordinate Wertgrenze
SFInt32	textureCoordinateNbBits	Anz. TextureCoordinate Quantisierungsbits
SFBool	angleQuant	Aktiviert das Angle Verfahren.
SFFloat	angleMin	minimale Angle Wertgrenze
SFFloat	angleMax	maximale Angle Wertgrenze
SFInt32	angleNbBits	Anzahl der Angle Quantisierungsbits
SFBool	scaleQuant	Aktiviert das Scale Verfahren.
SFFloat	scaleMin	minimale Scale Wertgrenze
SFFloat	scaleMax	maximale Scale Wertgrenze
SFInt32	scaleNbBits	Anzahl der Scale Quantisierungsbits
SFBool	keyQuant	Aktiviert das Key Verfahren.
SFFloat	keyMin	minimale Key Wertgrenze
SFFloat	keyMax	maximale Key Wertgrenze
SFInt32	keyNbBits	Anzahl der Key Quantisierungsbits
SFBool	normalQuant	Aktiviert Normals und Rotations Verfahren.
SFInt32	normalNbBits	Anz. Normals/Rotations Quantisierungsbits
SFBool	sizeQuant	Aktiviert das ObjectSize3D/2D Verfahren.
SFFloat	sizeMin	minimale ObjectSize3D/2D Wertgrenze
SFFloat	sizeMax	maximale ObjectSize3D/2D Wertgrenze
SFInt32	sizeNbBits	Anz. ObjectSize3D/2D Quantisierungsbits
SFFloat	useEfficientCoding	Aktiviert EfficientFloat (siehe Kapitel 3.3.1)

Tabelle C.6: Datenfelder eines QP Knotens

Feldtyp	Feldname	Kurzerklärung
`SFBool`	`enable`	Bestimmt, ob das Kommando verschickt werden soll.
`MFString`	`url`	Angabe des Objektdeskriptors bzw. Elementary Streams, über den das Kommando geschickt werden soll.
`SFString`	`command`	Kommando mit beliebiger Syntax, das zum Sendeterminal geschickt werden soll.

Tabelle C.7: Datenfelder eines `ServerCommand` Knotens

Feldtyp	Feldname	Kurzerklärung
`MFString`	`url`	Angabe des Objektdeskriptors, der die ES-Deskriptoren für den eingebetteten Szenengraphen enthält.

Tabelle C.8: Datenfelder eines `Inline` Knotens

C.2 Bitbedarf zur Kodierung von Feldern und Knoten

SF-Felder werden nach ihrem Typ unterschiedlich kodiert. Der Bitbedarf für die unkomprimierte Ablage im Bitstrom wurde bereits in Tabelle 3.1 angegeben. Die einzelnen Elemente der MF-Felder werden wie ihre entsprechenden SF-Typen kodiert. Um die Anzahl der Elemente im MF-Feld zu kodieren, werden zwei Möglichkeiten angeboten: `MFListDescription` und `MFVectorDescription`. Bei der ersten Möglichkeit entscheidet ein Bit nach jedem Element, ob noch ein weiteres Element folgt. Bei der zweiten Möglichkeit wird zuerst die Anzahl n der Elemente kodiert und anschließend n einzelne Elemente hintereinander ohne weitere Lücke. Wie in [95] angegeben wird, ist die `MFListDescription` effizienter solange $n < 9$ gilt. Da jedoch im Umfeld dieser Arbeit meist mit sehr langen Listen gearbeitet wird, entfallen hier weitere Angaben über die Kodierung mit `MFListDescription`. Die Kodierung eines MF-Felds mit n Elementen benötigt die folgende Anzahl von Bits:

Bitanzahl	Verwendung
(1)[a)]	Dieses Bit bestimmt, ob Predictive MFField verwendet wird.
1	Reserviert; sollte immer auf den Wert `falsch` gesetzt werden.
1	Entscheidung zwischen `MFList-` oder `MFVectorDescription`.
5	Bitanzahl für die Kodierung von n.
0-31	Anzahl der Listenelemente n
n · N	Anzahl der Listenelemente mal Kodierungslänge eines Elements

[a)] Nur vorhanden, wenn ein BIFS Elementary Stream der Version 2 verwendet wird.

Wird ein `MFVec3f` Feld mit 5 000 3D-Punkten unkomprimiert kodiert, werden demnach

$$1 + 1 + 1 + 5 + \lceil log_2(5\,000) \rceil + 5\,000 \cdot 3 \cdot 32 = 21 + 480\,000$$

Bits benötigt. Für jedes kodierte MF-Feld kann angenommen werden, dass neben den Nutzdaten maximal 39 weitere Bits benötigt werden.

Auch für die Kodierung aller Felder eines Knotens werden zwei Möglichkeiten angegeben: `MaskNodeDescription` und `ListNodeDescription`. Bei der ersten Möglichkeit wird für jedes Feld des Knotens mit einem Bit angegeben, ob es kodiert wird, oder nicht. Ein Feld sollte dann nicht kodiert werden, wenn es seinen Standardwert enthält. Ein `QP` Knoten hat 40 Felder, also werden hier immer 40 Bit für die Kodierung der Maske benötigt. Bei der zweiten Möglichkeit wird vor dem kodierten Feld zuerst der Index des Feldes mit einer festgelegten Anzahl von Bits kodiert und abschließend mit einem Bit entschieden, ob noch ein weiteres Feld folgt. Sollen bei einem `QP` Knoten nur drei Felder kodiert werden, benötigt die `ListNodeDescription` deutlich weniger als 40 Bits. Im Weiteren wird von einer `MaskNodeDescription` ausgegangen, da hier für jeden Knoten eine feste Anzahl von Bits zur Kodierung verwendet werden kann.

Neben der Kodierung der Felder fallen folgende Bitkosten für einen normalen Knoten an:

Bitanzahl	Verwendung
1	Wert = `falsch`: ein normaler Knoten ist kein Verweisknoten
nIdx	Index des Knotentyps in seinem Kontext (siehe unten)
1	Wert = `wahr` wenn der Knoten eine Knoten-ID hat
nodeIdBits	Wenn der Knoten eine Knoten-ID hat, dann ist sie hier kodiert. Der Wert nodeIdBits entspricht dem gleichnamigen BIFSConfig Parameter.
1	Wert = `wahr` für eine Maskendarstellung der Felder (siehe oben)
n_F	Pro Feld des Knoten ein Bit, das angibt, ob das Feld kodiert wird

Die folgende Tabelle gibt für einige Knoten die Werte nIdx und n_F an, sowie die minimal notwendige Anzahl von Bits zur Kodierung eines solchen Knotens:

Knoten	Kontext	nIdx	n_F	Minimum
`Shape`	SF3DNode	6	2	11
`IFS`	SFGeometryNode	5	14	22
`Coordinate`	SFCoordinateNode	1	1	5
`QuantizationParameter`	SF3DNode	6	40	49

Für einen Verweisknoten werden hauptsächlich Bits zur Kodierung der Knoten-ID verwendet:

Da nodeIdBits Werte im Bereich $[0 \ldots 31]$ annehmen kann, benötigt die Kodierung eines Verweisknotens maximal 32 Bit.

Bitanzahl	Verwendung
1	Wert = `wahr`: hier wird ein Verweisknoten kodiert
nodeIdBits	die Knoten-ID des Knotens, auf den verwiesen wird

C.3 SL-Paket Header

Bei der Aufteilung einer Access Unit in mehrere kleinere SL-Pakete wird jedem SL-Paket ein Header hinzugefügt. Für das erste SL-Paket einer Access Unit fällt dieser Header etwas länger aus, da dort die auf die Access Unit bezogenen Informationen abgelegt werden. Die folgenden SL-Header enthalten nur noch auf das jeweilige SL-Paket bezogene Daten.

In der Konfiguration jedes Elementary Streams ist bereits bekannt, welche SL-Informationen in die Header eingebunden werden soll. Für jedes SL-Paket können das sein:

- **Start-Flag**
  Ein Bit signalisiert den Beginn einer neuen Access Unit.
- **End-Flag**
  Ein Bit signalisiert das Ende einer Access Unit.
- **SL-Paket Zähler**
  Ein inkrementierender Zähler der aktuellen SL-Paket Nummer. Dieser kann verwendet werden, um verloren gegangene SL-Pakete zu detektieren.
- **Object Clock Reference (OCR)**
  Ein Zeitstempel der Uhr des Sendeterminals. Mit Hilfe der Folge von empfangenen OCR Zeitstempeln kann der Empfänger den Verlauf der Uhr des Sendeterminals schätzen.
- **Padding**
  Damit der Header eine durch acht teilbare Anzahl von Bits belegt, können mit diesem Wert zwischen 0 und 7 beliebige Bits eingefügt werden.

Für das erste SL-Paket einer AU können folgende Werte hinzukommen:

- **Random-Flag**
  Dieses Bit teilt dem Empfänger mit, das mit der Dekodierung dieser Access Unit sämtliche zu diesem Zeitpunkt bekannten Daten des entsprechenden Medienobjekts übertragen werden. Für einen BIFS Elementary Stream entspricht dies einem `SceneReplace` BIFS-Update Kommando. Nachdem der Empfänger einen Übertragungsfehler detektiert hat, muss er so lange warten, bis er wieder eine Random Access Unit empfängt.
- **Access Unit Zähler**
  Ein inkrementierender Zähler der aktuellen Access Unit Nummer. Dieser kann verwendet werden, um verloren gegangene Access Units zu detektieren.
- **Länge der Access Unit**

Die Angabe der Bytelänge einer Access Unit hilft dem Empfänger insoweit, als dass er bereits beim ersten SL-Paket weiß, wie viel Speicher er für den weiteren Empfang reservieren muss.

- **Composition Time Stamp (CTS)**
  Dieser Zeitstempel gibt an, wann der Inhalt der Access Unit dargestellt werden soll.
- **Decoding Time Stamp (DTS)**
  Dieser Zeitstempel gibt an, wann der Inhalt der Access Unit dekodiert werden soll.

Im *SLConfigDescriptor* kann das Sendeterminal für jeden einzelnen der oben erwähnten Wert einstellen, ob er im SL-Header kodiert wird. Je nach Bedarf kann somit gezielt gesteuert werden, wie viele zusätzliche Informationen mitgeschickt werden. Sind wenig Übertragungsfehler zu erwarten, müssen weder der SL-Paket oder Access Unit Zähler verwendet werden. Wird die Präsentation aus einer MP4 Datei gelesen, schreibt der Standard sogar einen SL-Header ohne Start-, End- und Random-Flag vor.

Der Empfänger kann auf mehrere Arten entscheiden, wann in einer Folge von SL-Paketen eine Access Unit beginnt, und wann sie endet:

- **Weder Start- noch End-Flag**
  In diesem Fall schreibt der Standard vor, dass jedes SL-Paket eine vollständige Access Unit enthält.
- **Nur Start-Flag**
  Steht dem Empfänger nur das Start-Flag zur Verfügung, kann er das Ende einer Access Unit dadurch detektieren, wenn das nächste SL-Paket mit gesetztem Start-Flag empfangen wird. Der Empfänger kann also nur mit einem kontinuierlichen Datenstrom arbeiten. Die letzte gesendete Access Unit kann so lange nicht bearbeitet werden, bis das nächste SL-Paket mit gesetztem Start-Flag ankommt.
- **Nur End-Flag**
  Ähnlich wie beim Start-Flag kann der Empfänger das erste empfangene SL-Paket nach einem mit gesetztem End-Flag als Beginn einer neuen Access Unit interpretieren. Im Gegensatz zur Lösung „nur Start-Flag“ kann eine Access Unit sofort bearbeitet werden, sobald das letzte SL-Paket empfangen wurde.
- **Start- und End-Flag**
  Ohne eventuell falsche Annahmen machen zu müssen, kann der Empfänger Anfang und Ende einer Access Unit erkennen.
- **Start-Flag und Länge der Access Unit**
  Auch in dieser Konstellation kann sowohl Anfang als auch Ende erkannt werden. Zusätzlich ist hier der Vorteil, dass der Empfänger bereits mit dem ersten SL-Paket weiß, wie viel Speicher er für die folgenden SL-Pakete reservieren muss.

Die minimale Größe für einen SL-Header sind 0 Byte. Die für die Versuche verwendeten SL-Header haben neben Start-, End- und Random-Flag zusätzlich die beiden Zähler und CTS enthalten. Außerdem wurde Padding verwendet, da dadurch das Zusammensetzen

der SL-Pakete zu Access Units wesentlich effizienter implementiert werden konnte. Der erste SL-Header belegte 13 Byte, während die folgenden SL-Header nur 3 Byte benötigten.

Literaturverzeichnis

[1] Behrendt, Stephan: *Rendering Dynamic Real-World Scenes Using Image Spheres*. In: Bebis, George, Richard Boyle, Bahram Parvin, Darko Koracin, Paolo Remagnino, Ara V. Nefian, Meenakshisundaram Gopi, Valerio Pascucci, Jiri Zara, Jose Molineros, Holger Theisel und Thomas Malzbender (Herausgeber): *ISVC (2)*, Band 4292 der Reihe *Lecture Notes in Computer Science*, Seiten 467–479. Springer, 2006. 2

[2] Boughoufalah, Souhila, Jean-Claude Dufourd und Frédéric Bouilhaguet: *MPEG-Pro: An Authoring System for MPEG-4 with Temporal Constraints and Template Guided Editing*. In: *IEEE Int. Conf. on Multimedia and Expo*, Band 1, Seiten 175–178, New York, NY, USA, 2000. 12

[3] Bray, David A. und Benn Konsynski: *Virtual Worlds, Virtual Economies, Virtual Institutions*. Emory University - Department of Decision & Information Analysis and Emory University - Goizueta Business School, Februar 2007. 11

[4] Burkert, Tim, Jan Leupold und Georg Passig: *A Photo-Realistic Predictive Display*. PRESENCE: Teleoperators and Virtual Environments, 13(1):22–4, 2004. 2

[5] Calvin, J., J. Seeger, G. Troxel und D. Van Hook: *STOW real-time information transfer and networking system architecture*. In: *Proceedings of the 12th DIS Workshop*, März 1995. 9

[6] Capps, Michael, Don McGregor, Don Brutzman und Michael Zyda: *NPSNET-V: A New Beginning for Dynamically Extensible Virtual Environments*. IEEE Computer Graphics and Applications, 20(5):12–15, September/October 2000. 9

[7] Celakovski, Sasko, Marius Preda, Slobodan Klajdziski, Danco Davcev und Françoise Preteux: *MPEG-4 3D Graphics: from specifications to the screen*. In: *Proc. WSEAS*, Juli 2006. 12

[8] Concolato, Cyril, Jean-Claude Dufourd und Jean-Claude Moissinac: *Creating and Encoding of Cartoons Using MPEG-4 BIFS: Methods and Results*. IEEE Transactions on Circuits and Systems for Video Technology, 13(11), November 2003. 12

[9] Coors, Volker und Matthias Finke: *Compressed mixed reality maps for LBS using MPEG4-BIFS*. In: *Eurescom Summit 2003: Evolution of Broadband Services*, Seiten 161–166, September 2003. 13

[10] http://de.wikipedia.org/wiki/Constructive_Solid_Geometry. Übersicht zu Constructive Solid Geometry. 5

[11] Daras, Petros, Ioannis Kompatsiaris, Theodoros Rapis und Michael G. Strintzis: *MPEG-4 Authoring Tool for the Composition of 3D Audiovisual Scenes*. In: *IEEE Int. Symposium on Circuits and Systems*, Band 2, Seiten 201–204, Sydney, Australia, May 2001. 11

[12] Daras, Petros, Ioannis Kompatsiaris, Theodoros Rapis und Michael G. Strintzis: *An MPEG-4 tool for composing 3D scenes*. IEEE Multimedia Magazin, 11(2):58–71, April 2004. 11

[13] *IEEE standard for information technology – protocols for distributed simulation applications: Entity information and interaction. IEEE Standard 1278-1993*. IEEE Computer Society, New York, 1993. 9

[14] http://www.divx.com. Internetpräsenz des kommerziellen DivX MPEG-4 Video-Codecs. 11

[15] Dobermann, F., D. Jackèl und U. von Lukas: *Synchrone Telekooperation im CAD-Umfeld auf der Basis von MPEG-4*. Diplomarbeit, Universität Rostock, Fachbereich Informatik, 2001. 12

[16] Dobermann, Falk und Uwe von Lukas: *Using MPEG-4 in the engineering domain*. In: *Proc. European Concurrent Engineering Conference 2002*, Seiten 183–187, 2002. 12

[17] http://www.idsoftware.com/games/doom/doom-final, Januar 2008. Internetpräsenz von idSoftware's Doom. 10

[18] Durbridge, Neil: *Is VRML A Forgotten Language?* Doktorarbeit, School of Humanities, Oxford Brookes University, July 2004. Beschreibt die Geschichte und Werdegang der verschiedenen VRML Standards. 6

[19] Eisert, Peter, Yong Guo, Anke Riechers und Juergen Rurainsky: *High-Resolution Interactive Panoramas with MPEG-4*. In: *Proc. Vision Modeling and Visualization*, 2004. 12

[20] Feuvre, Jean Le, Cyril Concolato und Jean-Claude Moissinac: *GPAC, Open Source Multimedia Framework*. In: *Proc. 15th ACM Int. Conf. on Multimedia*, Seiten 1009–1012, 2007. 11

[21] http://ffmpeg.mplayerhq.hu. Internetpräsenz des Open Source FFMpeg Video-Codec Projekts. 11

[22] Folea, Octavian, Marius Preda und Francoise Prêteux: *MPEG-4 SDK:*

From Specification to Real Applications. In: *9th WSEAS International Conference on Communications*, Athen, Griechenland, 14.-16. Juli 2005. 11

[23] Frécon, Emmanuel und Mårten Stenius: *DIVE: A scaleable network architecture for distributed virtual environments*. Distributed Systems Engineering Journal (special issue on Distributed Virtual Environments), 5(3):91–100, September 1998. 9, 76

[24] http://www.sics.se/dce/dive/dive.html. Internetpräsenz des DIVE Forschungsprojekts. 9

[25] Funk, Wolfgang: *Impact of MPEG-4 3D Mesh Coding on Watermarking Algorithms for Polygonal 3D meshes*. In: Edward J. Delp III, Ping W. Wong (Herausgeber): *Proc. of SPIE: Security, Steganography, and Watermarking of Multimedia Contents VI*, Band 5306, Seiten 336–344, June 2004. 13

[26] http://graphics.stanford.edu/~sliang/CS448B_win00/sliang-cs448b-contrib.html, Januar 2008. Linksammlung zum Thema Geometriekompression. 14

[27] http://www.angelfire.com/space2/dineshshikhare/compression/geomComp.html, Februar 2008. Linksammlung zum Thema Geometriekompression. 14

[28] http://earth.google.de, Januar 2008. Internetpräsenz von Google Earth. 11

[29] Greenhalgh, Chris, Adrian Bullock, Emmanuel Frècon, David Lloyd und Anthony Steed: *Making Networked Virtual Environments Work*. PRESENCE: Teleoperators and Virtual Environments, 10(2):142–159, April 2001. 9

[30] Gross, M., S. Würmlin, M. Naef, E. Lamboray, C. Spagno, A. Kunz, E. Koller-Meier, T. Svoboda, L. Van Gool, S. Lang, K. Strehlke, A. Vande Moere und O. Staadt: *blue-c: A Spatially Immersive Display and 3D Video Portal for Telepresence*. In: *Proc. ACM SIGGRAPH*, Seiten 819–827, San Diego, USA, July 2003. 10

[31] http://blue-c.ethz.ch/, Januar 2008. Internetpräsenz des blue-c Forschungsprojekts. 10

[32] http://edc.usgs.gov/products/elevation/gtopo30/gtopo30.html, Januar 2008. Frei verfügbare Höhenkarte der Erde. 11

[33] Gumhold, Stefan und Wolfgang Strasser: *Real Time Compression of Triangle Mesh Connectivity*. In: *Proc. Int. Conf. Computer Graphics and Interactive Techniques*, Seiten 133–140, 1998. 13

[34] Hoffmann, Christoph Martin: *Geometric & Solid Modeling*. Morgan Kaufmann Publishers, California, 1989. ISBN 1-55860-067-1, verfügbar unter http://www.cs.purdue.edu/homes/cmh/distribution/books/geo.html. 5

[35] Holbrook, H. W., S. K. Singhal und D. R. Cheriton: *Log-Based Receiver-*

Reliable Multicast for Distributed Interactive Simulation. Computer Communications Review, 25(4), October 1995. 9

[36] Hosseini, Mojtaba und Nicolas D. Georganas: *Suitability of MPEG4's BIFS for Development of Collaborative Virtual Environments.* In: *Proc. 10th IEEE Int. Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, 2001. WET ICE 2001.*, Seiten 299–304, 2001. 12

[37] Hosseini, Mojtaba und Nicolas D. Georganas: *MPEG-4 Based Recording and Replay of Collaborative Virtual Reality Sessions.* In: *IEEE Proc. Virtual Reality*, Seiten 271–272, 2002. 12

[38] Hosseini, Mojtaba und Nicolas D. Georganas: *MPEG-4 BIFS Streaming of Large Virtual Environments and their Animation on the Web.* In: *Proc. 7th Int. Conf. on 3D Web technology*, Seiten 19–25, 2002. 12

[39] *IEEE Standard for Binary Ffloating-Point Arithmetic for microprocessor systems (ANSI/IEEE Std 754-1985).* Verfügbar unter http://754r.ucbtest.org/standards/754.pdf. 26

[40] ISO/IEC: *Information Technology – Coding of audio-visual objects – Part 6: Delivery Multimedia Integration Framework (DMIF)*, 2000. 14496-6, w3713. 37

[41] ISO/IEC: *Information Technology – Coding of audio-visual objects – Part 5: Reference Software*, 2001. 14496-5, verfügbar unter http://www.m4if.org/resources.php. 28, 111, 112

[42] ISO/IEC: *Information Technology – Coding of audio-visual objects – Part 1: Systems*, 2002. 14496-1, w5277. 15

[43] ISO/IEC: *Information Technology – Coding of audio-visual objects – Part 11: Scene Description and Application Engine*, 2005. 14496-11, w6960. 7, 12, 26, 30, 77, 112

[44] ISO/IEC: *Information Technology – Coding of audio-visual objects – Part 16: Animation Framework eXtension (AFX)*, 2005. 14496-16, w7881. 109

[45] ISO/IEC: *Information Technology – Coding of audio-visual objects – Part 20: Lightweight Application Scene Representation (LASeR) and Simple Aggregation Format (SAF)*, 2006. 14496-20, w9385. 7

[46] http://www.mpeg-laser.org/html/techSection_laserSpec.htm. Verweis auf die frei erhältliche LASeR Spezifikation. 7

[47] http://www.m4if.org/resources.php, Januar 2008. Internetpräsenz des MPEG-4 Industy Forum. Die MPEG-4 Referenzimplementierung ist hier erhältlich. 11

[48] Jang, Euee S.: *3D Animation Coding: its History and Framework.* IEEE Int. Conf. on Multimedia and Expo, 2000. 12

[49] Jang, Euee S., James D. K. Kim, Seok Yoon Jung, Mahn-Jin Han, Sang Oak Woo und Shin-Jun Lee: *Interpolator Data Compression for MPEG-*

4 Animation. IEEE Transactions on Circuits and Systems for Video Technology, 14(7):989–1008, Juli 2004. 12

[50] Kim, J. D. K., Seok Yoon Jung, Mahnjin Han, E. S. Jang, Sang Oak Woo, Shin Jun Lee und Gyeong Ja Jang: *Animation data compression in MPEG-4: interpolators*. In: *IEEE Int. Conf. on Image Processing (ICIP)*, Band 3, 2002. 12

[51] http://gpac.sourceforge.net. Internetpräsenz des Open Source GPAC Multimedia Projekts. 11, 112

[52] Lehane, B., N. O'Conner und N. Murphy: *MPEG-4 Tools and Applications: An Overview*. In: *Irish Machine Vision and Image Processing Conference*, 2003. 11

[53] Lindenschmidt, K.-E., F. B. Hesser und M. Rode: *Integrating water quality models in the High Level Architecture (HLA) environment*. Advances in Geosciences, 4:51–56, 2005. 9

[54] Macedonia, Michael R., Michael J. Zyda, David R. Pratt, Paul T. Barham und Steven Zeswitz: *NPSNET: A Network Software Architecture for Large Scale Virtual Environments*. PRESENCE: Teleoperators and Virtual Environments, 3(4), 1994. 9

[55] Meehan, Michael: *Survey of Multi-User Distributed Virtual Environments*. In: *Course Notes: Developing Shared Virtual Environments, Siggraph '99*, 1999. 9

[56] Miller, D. und J. A. Thorpe: *SIMNET: The advent of simulator networking*. Proceedings of the IEEE, 83(8):1114–1123, August 1995. 9

[57] Morán, Francisco, Marius Preda, Gauthier Lafruit, Paulo Villegas und Robert-Paul Berretty: *3D Game Content Distributed Adaption in Heterogeneous Environments*. EURASIP Journal on Advances in Signal Processing, 2007, September 2007. 12

[58] Naef, Martin, Edouard Lamboray, Oliver Staadt und Markus Gross: *The blue-c Distributed Scene Graph*. In: *Proc. IPT/EGVE Workshop*, 2003. 10

[59] http://worldwind.arc.nasa.gov, Januar 2008. Internetpräsenz von NASA World Wind. 11

[60] http://www.tec.army.mil/TD/tvd/survey/NPSNET_Visual_Simulation_System.html. NPSNET Internetpräsenz. 9

[61] http://www.dsg.stanford.edu/paradise.html. PARADISE Internetpräsenz. 9

[62] Pope, A.: *The SIMNET network and protocols*. Technischer Bericht 7102, MA: BBN Systems and Technologies, Cambridge, MA, 1989. 9

[63] Requicha, Aristides A. G.: *Representations for Rigid Solids: Theory, Methods, and Systems*. ACM Computing Surveys (CSUR), 12(4):437–464, Dezember 1980. Überblick über bekannte Beschreibungsformen solider Körper. 5

[64] Ringbeck, Thorsten und Bianca Hagebeuker: *A 3D Time of Flight Camera for Object Detection*. In: *Optical 3-D Measurement Techniques*, ETH Zürich, July 2007. 39

[65] Schaeffer, Benjamin, Peter Brinkmann, George Francis, Camille Goudeseune, Jim Crowell und Hank Kaczmarski: *Myriad: Scalable VR via peer-to-peer connectivity, PC clustering, and transient inconsistency*. Computer Animation and Virtual Worlds, 18(1):1–17, February 2007. 10

[66] http://secondlife.com, Januar 2008. Internetpräsenz von Linden Lab's Second Life. 11

[67] http://www.libsecondlife.org, Januar 2008. Dokumentation des Second Life Protokolls. 11

[68] http://www.sfb453.de, Januar 2008. Internetpräsenz des SFB 453. 1, 3

[69] Shannon, Claude Elwood und Warren Weaver: *The Mathematical Theory of Communication*. University of Illinois Press, 1963. 30

[70] Shaw, C. und M. Green: *The MR toolkit peers package and experiment*. In: *Proc. 1993 IEEE Virtual Reality Annual International Symposium*, Seiten 463–469, Seattle, September 1993. 9

[71] Shaw, Chris, Mark Green, Jiandong Liang und Yunqi Sun: *Decoupled Simulation in Virtual Reality with the MR Toolkit*. ACM Transactions on Information Systems, 11(3):287–317, July 1993. 9

[72] Signès, Julien, Yuval Fischer und Alexandros Eleftheriadis: *MPEG-4's binary format for scene description*. Elsevier Signal Processing: Image Communication, 15:321–345, 2000. 11

[73] Singh, G., L. Serra und W. Png: *BrickNet: A software toolkit for network-based virtual environments*. PRESENCE: Teleoperators and Virtual Environments, 3(1):19–34, 1994. 9

[74] Singh, Gurminder, Luis Serra, Willie Png, Audrey Wong und Hern Ng: *BrickNet: Sharing Object Behaviors on the Net*. In: *Proc. IEEE Virtual Reality Annual International Symposium*, Seiten 19–25, 1995. 9

[75] Singhal, Sandeep und Michael Zyda: *Networked Virtual Environments*. Siggraph Series, ACM Press Books, 1999. 8

[76] Singhal, Sandeep. K. und David R. Cheriton: *Using a Position History-Based Protocol for Distributed Object Visualization*, Kapitel 10. Designing Real-Time Graphics for Entertainment (Course Notes for SIGGRAPH '94), 1994. 9

[77] Smolic, Aljoscha, Carsten Grünheit und Thomas Wiegand: *Interaktives Streaming von hochaufgelösten 360°-Panoramen*. Fachzeitschrift für Fernsehen, Film und Elektronische Medien (FKT), Januar 2003. 12

[78] Strassburger, Steffen: *Distributed simultation based on the high level architecture in civilian application domains*. Doktorarbeit, Otto-von-Guericke-Universität Magdeburg, 2001. 9

[79] Strauss, Paul S. und Rikk Carey: *An object-oriented 3D graphics toolkit*. ACM SIGGRAPH Computer Graphics, 26(2):341–349, July 1992. 10

[80] http://www.w3.org/TR/SVG/. aktuelle Version der SVG Spezifikation. 7

[81] Taubin, Gabriel, André Guéziec, William Horn und Francis Lazarus: *Progressive Forest Split Compression*. In: *Proc. Int. Conf. Computer Graphics and Interactive Techniques*, Seiten 123–132, 1998. 13

[82] Taubin, Gabriel, William P. Horn, Francis Lazarus und Jarek Rossignac: *Geometry Coding and VRML*. Proceedings of the IEEE, 86(6):1228–1243, June 1998. 7

[83] Taubin, Gabriel und Jarek Rossignac: *Geometric Compression Through Topological Surgery*. ACM Transactions on Graphics, 17(2):84–115, April 1998. 13

[84] Tran, S. M., K. Lajos, E. Balazs, K. Fazekas und z. Csaba: *A survey on the interactivity feature of MPEG-4*. In: *Proc. 46th Int. Symposium Electronics in Marine (Elmar)*, Seiten 30–38, June 2004. 12, 22

[85] Tran, S. M., M. Preda, K. Fazekas und F. Prêteux: *Improvement of the temporal constraint in MPEG-4 BIFS*. In: *Proc. 6th COST 276 Workshop*, Seiten 83–87, Thessaloniki, Greece, May 2004. 12

[86] Tran, S. M., M. Preda, F. Prêteux und A. Gschwindt: *An MPEG-4 BIFS based authoring tool for multimedia content*. In: *Proceedings IAPR Int. Conf. on Image and Signal Processing*, Seiten 38–46, Agadir, Morocco, 2003. 12

[87] Tran, S. M., M. Preda, F. J. Preteux und K. Fazekas: *Case study: A basic composing tool for editing MPEG-4 System Information*. In: *Proc. 4th European Workshop on Image Analysis for Multimedia Interactive Services*, Seiten 449–455, London, UK, 2003. 12

[88] Tran, S. M., M. Preda, F. J. Preteux und K. Fazekas: *Exploring MPEG-4 BIFS features for creating multimedia games*. In: *IEEE Int. Conf. on Multimedia and Expo*, Band 1, Seiten I–429 – I–432, 2003. 12, 22

[89] http://www.ecma-international.org/publications/standards/Ecma-363.htm. aktuelle Version des Universal 3D File Format. 7

[90] http://tools.ietf.org/html/rfc3629. Beschreibung des UTF-8 Encoding. 19

[91] http://www.virtualworldsreview.com, Januar 2008. Übersicht über bekannte virtuelle Welten. 11

[92] http://www.web3d.org/x3d/specifications/vrml/VRML1.0/index.html. VRML 1.0 Spezifikation. 6

[93] http://www.mitra.biz/vrml/vrml2/19960202/spec_full.html. VRML 2.0 Spezifikation. 6

[94] http://www.web3d.org/x3d/specifications/vrml. VRML97 Spezifikation. 6, 17

[95] Walsh, Aaron E. und Mikaël Bourges-Sévenier: *MPEG-4 Jump Start*. Prentice Hall, 2002. 120

[96] http://maps.live.de, Januar 2008. Internetpräsenz von Microsoft's Windows Live Maps. 11

[97] Witten, Ian H., Radford M. Neal und John G. Cleary: *Arithmetic coding for data compression*. Communications of the ACM, 30(6):520–540, 1987. Arithmetische Kodierung mitsamt C-Quelltext. 30, 31

[98] Woodward, Peter, Yakup Paker und Alan Pearmain: *A software system for MPEG-4 encoding of multi-media studio content for 3D television*. In: *Int. Conf. on Visual Information Engineering*, Seiten 77–80, July 2003. 12

[99] http://www.wow-europe.com, Januar 2008. Internetpräsenz von Blizzard's World-Of-Warcraft. 10

[100] http://www.web3d.org/x3d/specifications. X3D Spezifikationen. 6

[101] http://www.xvid.org. Internetpräsenz des Xvid Open Source MPEG-4 Video-Codec Forschungsprojekts. 11

[102] Zarki, Magda El, Liang Cheng, Haining Lui und Xiaoping Wei: *An Interactive Object Based Multimedia System for IP Networks*. In: *Proc. of the 8th Int. Workshop on Object-Oriented Real-Time Dependable Systems*, Seiten 312–318, January 2003. 12