Universität Leipzig

Fakultät für Mathematik und Informatik

Institut für Informatik

Thema: **Untersuchung der Online-Videoübertragung unter Windows NT über Dual-Video-Systeme**

Diplomarbeit

Leipzig, Juni, 1999

vorgelegt von

Fiebig, Matthias

Inhaltsverzeichnis

1 Einleitung

Im Laufe der letzten Jahre haben sich die Anwendungsbereiche von Computern immer stärker erweitert. Am Anfang der Entwicklung stand die Verwendung als reine Rechenmaschine im Vordergrund. In der weiteren Entwicklung erfolgte die Verwendung vornehmlich als Eingabe- und Verwaltungsgerät für umfangreiche Datenmengen bzw. zur Prozeßsteuerung. Seit Anfang der 90er Jahre nimmt jedoch die Nutzung als Kommunikations- und Informationsgerät stark zu. In diesem Zusammenhang hat sich auch die Präsentation der vom Computer verarbeiteten Daten stark gewandelt. Statt alphanumerischen Textausgaben werden heute die Daten häufig aufwendig grafisch aufbereitet und dargestellt.

Diese Entwicklung ist in erster Linie durch die gesteigerte Leistungsfähigkeit der Hard- und Software (Prozessoren, Speicher, Netze, Betriebssysteme usw.) möglich geworden. Aufgrund dieser Steigerung sind heute handelsübliche Computer in der Lage, Datenmengen zu verarbeiten, die bisher nur Spezialmaschinen vorbehalten waren. Dazu zählen unter anderem Videodaten in einer Qualität, wie sie von Videorecordern aufgezeichnet werden.

So erlauben z.B. für Personalcomputer[1] angebotene Videoschnittkarten, einen Videodatenstrom zu bearbeiten. Sie stellen damit eine dem Videorecorder ähnliche Funktionalität zur Verfügung. Ausgehend von diesen Produkten wurde am Lehrstuhl „Rechnernetze und Verteilte Systeme“ des Instituts für Informatik der Universität Leipzig begonnen, ein experimentelles Online-Videoübertragungssystem zu entwickeln [NGU98]. Das Ziel der Entwicklung besteht darin, unter Einsatz handelsüblicher PC-Hardware weitestgehend verlustfreie Übertragung von Audio- und Videodaten in Echtzeit zu ermöglichen.

Bei diesen Entwicklungsarbeiten mußten die Probleme von Betriebssystemen, Netzwerken und der eingesetzten spezifischen Hard- und Software untersucht werden. Speziell wurden Unzulänglichkeiten beim Betrieb mehrerer Videoschnittkarten in einem Kommunikationssystem, bei der gleichzeitigen Darstellung von zwei Videos auf dem PC-Bildschirm und der angebotenen Betriebssystemunterstützung festgestellt.

---

[1] Wenn hier von Personalcomputern (PC) die Rede ist, so sind damit IBM-kompatible Personalcomputer gemeint.

Die vorliegende Arbeit soll einen Weg zur Lösung dieser Probleme anbieten; es wird ein Hardwaresystem untersucht, welches es ermöglicht, zwei Videodatenströme gleichzeitig zu verarbeiten. Dabei soll innerhalb der Arbeit geklärt werden, ob die untersuchte Hardware den Anforderungen an die Online-Videoübertragung entspricht, welche Vorteile sie gegenüber der bisher im Projekt eingesetzten Hardware bietet und wie eine Einbindung in das bisher entwikkelte System stattfinden kann.

Im zweiten Kapitel werden die Grundlagen der digitalen Videoübertragung vorgestellt, um eine Überblick über die Bedeutung und die Anforderungen dieser Technik zu vermitteln. Eine Einführung in die zur Verarbeitung von digitalen Videodaten notwendigen Grundlagen, insbesondere der Kompressionsverfahren und Formate, bietet das dritte Kapitel. Dadurch soll dem Leser ein Verständnis für die Notwendigkeit der Datenkompression und die gebräuchlichsten dazu verwendeten Verfahren nahegebracht werden. Innerhalb des vierten Kapitels wird der Stand der Technik bezüglich Videoverarbeitung im PC erläutert. Damit werden die Grundlagen für das Verständnis der in den darauf folgenden Kapiteln vorgestellten Entwicklungen und Untersuchungen gelegt. Im fünften Kapitel erfolgt eine Untersuchung der Möglichkeiten des eingesetzten Systems und es wird ein Überblick über dessen Leistungsmerkmale gegeben. Im sechsten Kapitel wird die Entwicklung eines Testsystems beschrieben, welches es erlaubt, die untersuchte Hardware zur Videoübertragung einzusetzen. Auf der Grundlage dieses Testsystems werden im siebten Kapitel Untersuchungen zu wichtigen Eckwerten der Videokommunikation durchgeführt. Abschließend faßt das achte Kapitel die Ergebnisse der Arbeit zusammen und gibt einen Ausblick zur Weiterentwicklung des Testsystems.

2 Digitale Videoübertragungssysteme

2.1 Bedeutung und Anwendung

Videotelefone (Bildtelefone) sind bereits seit etwa 30 Jahren bekannt, wurden jedoch aus unterschiedlichen Motiven nicht allgemein eingesetzt. Die wohl wichtigsten Gründe sind dabei der einerseits sehr hohe Preis solcher Lösungen und die andererseits geringe Bildqualität. Nur unter Einsatz aufwendiger und somit teurer Technologien war es bisher möglich, Video in vertretbarer Qualität in Echtzeit zu übertragen.

Was unter vertretbarer Qualität zu verstehen ist, hängt von der Anwendung ab. So werden die Maßstäbe des Heimanwenders von den verbreiteten Videoaufzeichnungsgeräten bestimmt (VHS-Videorecorder, Video-8-Kameras), während in der Medizin eine unverfälschte Übertragung der Videodaten unabdingbar ist.

Aufgrund der fortschreitenden technologischen Entwicklung und der daraus resultierenden Verbreitung von leistungsfähigen Arbeitsplatzrechnern, welche in immer stärkerem Maße vernetzt sind, werden die Voraussetzungen für den Einsatz von PC-gestützten leistungsfähigen Echtzeit-Videoübertragungssystemen ständig besser. Es existiert bereits eine Vielzahl von Lösungen zur PC-gestützten Bildtelefonie. Beispiele dafür sind CuSeeMe und Netmeeting (weitere Informationen siehe [CUSEE] und [NETME]). Diese Lösungen verwenden entweder die existierenden Telefonnetze (hauptsächlich ISDN-Verbindungen) oder übertragen die Daten über Computernetzwerke. Bei der Verwendung des Telefonnetzes stellen sie hauptsächlich einen Zusatzdienst zum herkömmlichen Telefongespräch dar, da es sich hier meist um Punktzu-Punkt-Verbindungen handelt. Werden die Daten über Computernetzwerke ausgetauscht, so rückt die Integration in die Gruppenarbeit stärker in den Vordergrund. Es kommen zusätzliche Möglichkeiten der Interaktion hinzu. Besondere Bedeutung hat dabei die Konferenzschaltung mehrerer Teilnehmer. Hierbei treten jedoch auch die höchsten Anforderungen in Bezug auf die eingesetzte Technik auf.

Ein weiterer Vorteil ist die Integration anderer Technologien. So eröffnet im medizinischen Bereich die Online-Videoübertragung medizinischer Aufnahmen unter gleichzeitigem Einsatz datenbankbasierter, mustererkennender Systeme neue diagnostische Möglichkeiten.

Die Verwendung von Videoübertragungssystemen ist in vielen Bereichen möglich, die zukünftige Entwicklung aber derzeit noch nicht abzusehen. Die Gründe dafür sind:

- Nur begrenzte Anwendungsmöglichkeiten: Da es sich bei Video-Konferenzschaltungen um eine wenig verbreitete Technologie handelt, bestehen die hardwareseitigen Grundlagen für eine umfangreiche Anwendung nicht.

- Mangelnder Kenntnisstand: In vielen potentiellen Anwendungsbereichen der Videoübertragung sind die bereits vorhanden Möglichkeiten dieser Technologie noch nicht umfassend bekannt. Nur eine zunehmende Verbreitung wird hier Abhilfe schaffen.

2.2 Aufbau und Anforderungen

Videoübertragungssysteme können anhand der Anforderungen klassifiziert werden, die sie an die Endsysteme und an das Netz stellen. Es ist hierbei sinnvoll, vier Fälle zu unterscheiden:

1. Ein Videoübertragungssystem besteht im einfachsten Fall aus einem Sender und einem Empfänger (Punkt-zu-Punkt-Verbindung). Eine Anwendungsmöglichkeit besteht z.B. in der Überwachungstechnik. Hierbei werden die geringsten Anforderungen an die Endgeräte und an das Übertragungsmedium gestellt, da jedes Endgerät und auch das Netz nur jeweils einen Videostrom verarbeiten muß.

2. Die nächste Stufe stellen Punkt-zu-Mehrpunkt-Systeme dar, hierbei werden die Videodaten von einem Sender an mehrere Empfänger gesendet. Bei Verwendung von Shared-Media-Systemen[2] und Multicast-Diensten zur Übertragung der Daten steigen die Anforderungen an Endsysteme und Übertragungsmedium im Vergleich zum ersten Fall nicht. Eine mögliche Anwendung besteht in der Übertragung von Vorlesungen an mehrere Orte.

3. Für den Fall eines Mehrpunkt-zu-Punkt-Systems treten erhebliche zusätzliche Belastungen auf. Die zum Senden der Videodaten eingesetzten Systeme werden ähnlich belastet wie in den ersten beiden Fällen. Sie er-

[2] Beispiele für LAN-Technologien, die Shared-Media-Systeme verwenden, sind z.B. Ethernet und Token-Ring. Alle Stationen eines Netzsegmentes benutzen dabei dasselbe physikalische Medium.

zeugen ebenfalls einen Videodatenstrom und senden diesen kontinuierlich an den Empfänger. Der Empfänger muß in der Lage sein, alle ankommenden Videoströme zu verarbeiten. Die geforderte Leistungsfähigkeit des Systems hängt dabei direkt mit der Anzahl der Sender zusammen. Dieselben Anforderungen werden auch an das Netzwerksystem gestellt. Es muß in der Lage sein, die geforderte Anzahl von Videoströmen gleichzeitig zu übertragen. Ein sinnvoller Einsatz von Mehrpunkt-zu-Punkt-Systemen ist ebenfalls in der Videoüberwachung und Meßwerterfassung zu sehen. Mit Hilfe eines solchen Systems können die Daten von verschiedenen, räumlich weit voneinander entfernten Videoquellen zentral überwacht und aufgezeichnet werden.

4. Die höchsten Anforderungen an die Endgeräte und die Netz-Infrastruktur stellen Mehrpunkt-zu-Mehrpunkt-Systeme. Hierbei benötigen alle Endgeräte eine Leistungsfähigkeit, die es erlaubt, die ankommenden Datenströme zu verarbeiten. Zusätzlich müssen sie selbst noch einen Datenstrom erzeugen und versenden. Die Anforderungen an das Netzwerksystem sind hier besonders hoch, da jeder Teilnehmer mit der Anzahl von Videoströmen versorgt werden muß, wie Teilnehmer existieren. Der klassische Einsatzfall für ein solches System ist eine Konferenzschaltung mehrerer Teilnehmer. In diesem Fall kann jeder mit jedem gleichzeitig kommunizieren.

3 Digitales Video

3.1 Darstellung von Farbsignalen

Zur Darstellung von Farbbildern auf Computerbildschirmen wird das RGB–System verwendet. Es handelt sich um ein additives Farbsystem, bei dem die drei Komponenten Rot (R), Grün (G) und Blau (B) zu Mischfarben addiert werden. Durch die Variation der Intensität der Farbkomponenten kommt die Vielfalt der Farben zustande. Das RGB–System läßt sich als Würfel darstellen, wobei der x–Koordinate Rot, der y–Koordinate Blau und der z–Koordinate Grün zugeordnet wird (siehe Abbildung 1).

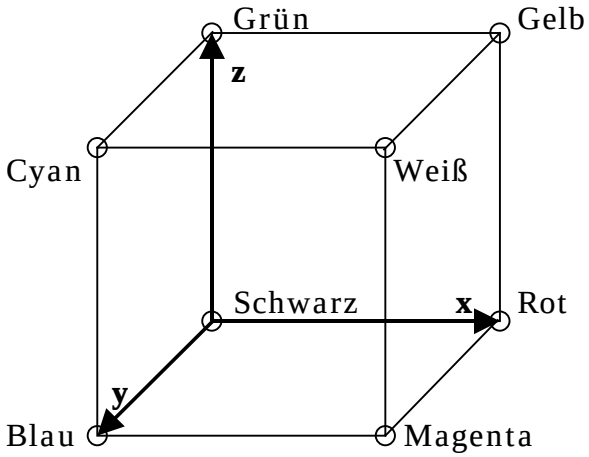


Abbildung 1: RGB–Farbwürfel

Man kann jedem Punkt in diesem Würfel eindeutig einen Farbwert zuordnen. In vielen Computersystemen werden pro Farbkomponente 8 Bit verwendet, damit stehen für jede Komponente 256 Intensitätsstufen zur Verfügung. Somit lassen sich theoretisch 256^3 = 16,7 Millionen Farben mischen.

Da bei der Einführung des Farbfernsehens das Problem bestand, diese Programme auch auf den vorhandenen Schwarzweißgeräten zu empfangen, war das einfachste Verfahren, die getrennte Kodierung der RGB–Signale, nicht akzeptabel. Bei dem in der Videotechnik verwendeten System handelt es sich um die Farbkomponentendarstellung. Bei der Farbkomponentendarstellung werden neben der Helligkeitskomponente Y (Luminanz) zwei Farbdifferenzkomponenten U und V (Chrominanz) gespeichert. Schwarzweißgeräte können diese Signale sehr einfach durch weglassen der Farbdifferenzkomponenten anzeigen. Die Umwandlung der RGB–Signale in die YUV–Signale läßt sich eindeutig durchführen. Die Gleichung (1) repräsentiert die Verhältnisse der

Komponenten bei den in Europa üblichen Farbfernsehsystemen PAL[3] und SECAM[4].

$$\begin{pmatrix} Y \\ U \\ V \end{pmatrix} = \begin{pmatrix} 0{,}299 & 0{,}587 & 0{,}114 \\ -0{,}146 & -0{,}288 & 0{,}434 \\ 0{,}617 & -0{,}517 & -0{,}1 \end{pmatrix} \cdot \begin{pmatrix} R \\ G \\ B \end{pmatrix} \quad \textbf{(1)}$$

Das NTSC–Format[5], welches vor allem in Amerika und Japan üblich ist, verwendet im Gegensatz zum YUV–System das YIQ–System. Es kommen hierbei etwas geänderte Vorfaktoren zum Einsatz (Gleichung (2)).

$$\begin{pmatrix} Y \\ I \\ Q \end{pmatrix} = \begin{pmatrix} 0{,}299 & 0{,}587 & 0{,}114 \\ 0{,}597 & -0{,}277 & -0{,}321 \\ 0{,}213 & -0{,}523 & 0{,}309 \end{pmatrix} \cdot \begin{pmatrix} R \\ G \\ B \end{pmatrix} \quad \textbf{(2)}$$

Mit diesen Gleichungen zur Umwandlung der RGB–Signale in die Farbkomponentendarstellung wird der darstellbare Farbraum an die Empfindlichkeit des menschlichen Sehsinns angepaßt.

3.2 Bildabtastung

Heutzutage wird in der analogen Videotechnik in der Regel das Zeilensprungverfahren (interlace) angewendet. Hierbei wird jedes Bild in 2 Halbbilder (fields) aufgeteilt. Eines der Halbbilder enthält dabei die geradzahligen, das andere die ungeradzahligen Bildzeilen. Das Abtasten und Wiedergeben der Halbbilder erfolgt zeitlich versetzt.

Da der Digitalisierung häufig die Abtastung des analogen Videosignals zugrunde liegt, wird das Zeilensprungverfahren auch in einigen digitalen Bildformaten verwendet. Hieraus resultiert ein Problem bei der Darstellung auf Computerbildschirmen, da diese grundsätzlich mit progressiver Bildwie-

---

[3] PAL (Phase Alternating Line) ist die in weiten Teilen Europas gültige Fernsehnorm (Halbbilder mit 50 Hz Bildwechselfrequenz, 625 Zeilen, davon etwa 576 sichtbar, Farbdarstellung mit YUV–System).

[4] SECAM (Sequentiel Couleur Avec Memoire) heißt das in Frankreich und Osteuropa verwendete Farbsystem. Von der Bildauflösung und der Bildwechselfrequenz entspricht es PAL.

[5] NTSC (National Television Standards Committee) ist die in den USA gültige Fernsehnorm (Halbbilder mit 60 Hz Bildwechselfrequenz, 525 Zeilen, davon 480 sichtbar, Farbdarstellung mit YIQ).

dergabe arbeiten. Der zeitliche Versatz der Halbbilder zueinander ist bei progressiver Bildwiedergabe deutlich zu sehen. Die jeweils aufeinanderfolgenden Zeilen sind je nach Bildveränderung deutlich verschieden. Besonders bei der Darstellung von schnellen Bewegungen ist das zu erkennen.

3.3 Eigenschaften digitaler Bildformate

Es existieren verschiedene normierte digitale Bildformate. Die wichtigsten momentan verwendeten Formate und ihre Einsatzgebiete sind im folgenden aufgeführt:

- QCIF (quarter common intermediate format) wird für Bildtelefon–Dienste bei niedrigen Datenraten verwendet.
- CIF (common intermediate format) und SIF (standard intermediate format) werden für Bildtelefon– und Videokonferenz–Dienste mit höherer Qualität als QCIF angewandt.
- CCIR-472 steht für ein digitales Fernsehformat mit reduzierter Zeilenauflösung, welches bei Bildtelefon– und Videokonferenzanwendungen eingesetzt wird.
- CCIR-601 wird als digitales Fernsehsystem im Studiobereich verwendet. Es entspricht von den Auflösungen und den Bildwechselfrequenzen den jeweiligen analogen Videosignalen (PAL / SECAM sowie NTSC).
- EDTV (enhanced definition TV, auch enhanced quality TV, EQTV) dient der Darstellung von progressiven HDTV–Signalen mit verbessertem Bildseitenverhältnis[6] (16:9) auch bei verringerter Auflösung.
- HD–1440 ist ein HDTV–Format (high definition TV) mit der exakt doppelten Auflösung (siehe Tabelle 1) des konventionellen TV–Signals (CCIR-601).
- HD–I ist ein HDTV–Format, welches mit verbessertem Bildseitenverhältnis (16:9) und Zeilensprungverfahren arbeitet.

---

[6] Das Bildseitenverhältnis gibt das Verhältnis zwischen horizontaler und vertikaler Auflösung an. Die meisten heute in Fernsehgeräten und Computermonitoren verwendeten Bildröhren haben ein Bildseitenverhältnis von 4:3. Mit der Einführung von HDTV soll sich dies auf das beim 35–mm–Film übliche 16:9 ändern.

- HD–P ist ein HDTV–Format mit progressiver Abtastung und erhöhter Bildwechselfrequenz.

Die wichtigsten Eigenschaften wie Auflösung, Bildwechselfrequenz, Bildseitenverhältnis und Datenraten der genannten Bildformate sind in Tabelle 1 zusammengestellt (abgewandelt übernommen aus [OHM95]). Die angegebenen Werte für Zeilenanzahl, Bildpunkte/Zeile, Bildwechselfrequenz und Datenmenge entsprechen den Werten der europäischen Normen (PAL/SECAM), die geklammerten Werte denen des NTSC–Systems.

	QCIF	**CIF/ SIF**	**CCIR-472**	**CCIR-601**	**EQTV**	**HD–1440**	**HD–I**	**HD–P**
Bildpunkte/Zeile (Y)	176	352	256	720	960	1440	1920	1920
Zeilenanzahl (Y)	144 (120)	288 (240)	576 (480)	576 (480)	576 (480)	1152 (960)	1152 (960)	1152 (960)
Bildpunkte/Zeile (U,V)	88	176	128	360	480	720	960	960
Zeilenanzahl (U,V)	72 (60)	144 (120)	576 (480)	576 (480)	288 (240)	1152 (960)	1152 (960)	576 (480)
Bildseitenverhältnis	4:3	4:3	4:3	4:3	16:9	4:3	16:9	16:9
Bildwechselfrequenz [Hz]	5–15	10–30	25 (30)	25 (30)	25 (30)	25 (30)	25 (30)	50 (60)
Datenmenge für Einzelbild [Kbyte][7]	38,02 (31,68)	152,1 (126,7)	294,9	829,4 (691,2)	829,4 (691,2)	3318 (2765)	4424 (3686)	3318 (2765)
Datenrate für Bildsequenz [MBit/s]	0,84–3,8	10,1–30,4	59,0	165,9	165,9	663,5	884,7	1327

Tabelle 1: Eigenschaften digitaler Bildformate [Ohm95]

[7] Die Farbkomponenten werden mit einer Genauigkeit von 8 Bit gespeichert.

3.4 Verfahren zur Datenreduktion

Wie man in der Übersicht (Tabelle 1) erkennen kann, entstehen bei der Digitalisierung von Videosignalen sehr große Datenmengen, für eine Sekunde unkomprimiertes Video nach CCIR-601 fallen z.B. bereits mehr als 20 MB Daten an. Selbst die aktuellen Speicher- und Netzwerksysteme stoßen hierbei schnell an ihre Grenzen. Grundsätzlich ist es deshalb sinnvoll, die Datenmenge zu reduzieren. Die dazu einsetzbaren Kompressionsverfahren lassen sich durch folgende Kriterien charakterisieren:

- verlustfreie und verlustbehaftete Kompression,
- Kompressionsgrad oder Kompressionsrate,
- symmetrisches und asymmetrisches Verhalten.

Bei den verlustfreien Kompressionsverfahren (lossless compression) läßt sich der Originaldatenstrom vollständig aus den komprimierten Daten rekonstruieren. Zu den wichtigsten verlustfreien Kompressionstechniken zählen die statistische Kodierung, die arithmetische Kodierung, die Lemple–Ziv–Technik und die Lauflängenkodierung. Weitere Einzelheiten zur statistischen Kodierung und zur Lauflängenkodierung folgen in Abschnitt 3.4.1 und Abschnitt 3.4.2.

Verlustfreie Kompressionstechniken werden natürlich in allererster Linie dort angewendet, wo die vollständige Wiederherstellung der Originaldaten unverzichtbar ist, wie zum Beispiel bei Programmcode, Texten oder Zahlenmaterial.

Im Gegensatz dazu zielen die verlustbehafteten Kompressionsverfahren (lossy compression) darauf ab, unter Tolerierung eines Verlustes einen höheren Kompressionsgrad zu erreichen. Hierbei wird im Bereich der Bildverarbeitung darauf geachtet, daß das visuelle Wahrnehmungssystem des Menschen möglichst wenig davon bemerkt. Das Ziel besteht darin, einen gleichen visuellen Eindruck zu erreichen, obwohl sich die dekomprimierten Daten von den Originaldaten unterscheiden. Im allgemeinen steigt mit dem Kompressionsgrad dieser Verfahren auch der Informationsverlust, somit sinkt die Qualität. Die Bewertung der Qualität der dekomprimierten Daten gestaltet sich aufgrund des stark subjektiven Eindrucks schwierig. Um die Qualität der Bildkodierung

mathematisch zu bewerten, wird häufig das Verzerrungsmaß PSNR[8] verwendet. Aufgrund psychovisueller Faktoren stimmen diese Aussagen aber nicht immer mit dem subjektiven Eindruck der Qualität überein. So wird zum Beispiel ein relativ detailarmes Bild (viel Himmel oder einfarbiger Hintergrund) sehr gute PSNR–Werte erzielen, wohingegen ein stark detailliertes Bild (feine Strukturen wie Wald oder Wiese) relativ schlecht bewertet wird. Der Betrachter bemerkt die schlechtere Qualität jedoch häufig nicht, da die Fehler in der Informationsfülle des detaillierten Bildes untergehen.

Der Kompressionsgrad oder die Kompressionsrate (compression ratio) gibt das Verhältnis der originalen Datenmenge zur komprimierten Datenmenge an. Bei einer Kompressionsrate von 10:1 würde sich zum Beispiel der Datenstrom für das CCIR-601-Format (siehe 3.3) von 165 MBit/s auf 16,5 MBit/s reduzieren lassen. Problematisch ist jedoch, daß die erzielbaren Kompressionsfaktoren stark von der Qualität des verwendeten Bildmaterials sowie den darin enthaltenen Details abhängen. Ein Vergleich von unterschiedlichen Kompressionsverfahren anhand des Kompressionsfaktors ist also nur mit Hilfe identischen Bildmaterials möglich.

Verfahren, die bei hohen Kompressionsraten wenig Qualitätsverlust erzeugen, erreichen dies durch eine sehr genaue Analyse der Originaldaten. Dieser Vorgang wirkt sich auf die Kompressionsgeschwindigkeit und die Dekompressionsgeschwindigkeit aus. Man unterscheidet hierbei zwischen symmetrischen und asymmetrischen Kompressionsverfahren. Symmetrische Kompressionsverfahren sind durch einen ähnlichen Aufwand für die Kompression und die Dekompression gekennzeichnet. Verfahren mit einem solchen Verhalten werden in erster Linie dort eingesetzt, wo ein Realzeitverhalten nötig ist. Bei asymmetrischen Kompressionsverfahren wird wesentlich mehr Aufwand in die Kompression als in die Dekompression investiert. Die Folge ist eine verbesserte Bildqualität bei gleicher Kompressionsrate oder eine höhere Kompressionsrate bei vergleichbarer Bildqualität gegenüber den symmetrischen Kompressionsverfahren. Aufgrund der Tatsache, daß die Kompression aber nicht immer in Realzeit erfolgen kann (siehe Abschnitt 3.6), muß der Datenstrom zwischengespeichert werden. Durch die dabei auftretenden Verzögerungen sind diese Verfahren unter Umständen ungeeignet für interaktive Anwendungen. Die hohe Bildqualität bei gleichzeitig hohem Kompres-

[8] Das PSNR (peak signal to noise ratio) läßt eine direkte Aussage über den mittleren quadratischen Fehler des rekonstruierten Bildes zu [Ohm95, S. 186].

sionsgrad und relativ schneller Dekomprimierung erlauben es jedoch, diese Verfahren in dokumentbasierenden Systemen oder in der Videoarchivierung einzusetzen.

3.4.1 Lauflängenkodierung

Die einfachste und am leichtesten zu verstehende Form der Redundanz in Daten ist eine lange Folge von gleichen Zeichen. Es ist normal, daß man die Zeichenfolge „00000000555555111111“ mit acht mal 0, sechs mal 5 und fünf mal 1 beschreibt, anstatt die Ziffern einzeln aufzuzählen. Die Lauflängenkodierung (run–length encoding) verwendet genau dieses Verfahren. Die redundanten Daten werden in Form von Paaren kodiert. Die Paare bestehen jeweils aus einem Zähler und einem Zeichen. Der Zähler gibt an, wie oft das Zeichen in den Ausgangsdaten an dieser Stelle steht. Um beim Dekodieren zu erkennen, ob nun ein Paar oder nur ein einfaches Zeichen kommt, wird ein sogenanntes Escape–Zeichen verwendet. Dieses sollte in den Daten ansonsten nur sehr selten vorkommen. Die Kombination aus Escape–Zeichen und einem Paar wird als Escape–Sequenz bezeichnet. Da diese bereits aus drei Zeichen besteht, lohnt es sich nicht, Folgen von weniger als vier gleichen Zeichen zu verschlüsseln. Das Escape–Zeichen selbst wird durch sich selbst mit einer anschließenden Null für den Zähler dargestellt. Zur Verdeutlichung des eben beschrieben Verfahrens hier noch ein Beispiel. Es soll die folgende Folge von Zeichen kodiert werden, wobei @ als Escape–Zeichen verwendet werden soll.

Zeichenfolge :

hhhhhhh00ppppp@3333333666666uuuuuuuuulllmm@nnnnnnnn

kodierte Zeichenfolge:

@**7**h00@**5**p@**0**@**7**3@**6**6@**9**ulllmm@**0**@**8**n

Damit die kodierte Zeichenfolge besser gelesen werden kann, wurden für die Zähler (fett gedruckt) die wirklichen Zahlen verwendet, normalerweise steht dort aber bei **1** natürlich ein Byte mit dem Wert eins und nicht der Zeichenkode von **1**. Die Lauflängenkodierung ist nur in sehr speziellen Fällen effektiv einsetzbar. Einer dieser Fälle wird im Abschnitt 3.5 vorgestellt.

3.4.2 Kodierung mit variabler Länge

Wenn man sich Daten (z.B. Texte) genauer anschaut, sind in vielen Fällen gewisse Häufungen bestimmter Zeichen zu erkennen. So kommen zum Beispiel in der deutschen Sprache die Buchstaben ‚e' und ‚n' relativ häufig vor. Bei der Speicherung belegt aber ein ‚e' genau soviel Platz wie das viel seltenere ‚z'. Die Idee besteht nun darin, für Zeichen mit häufigem Auftreten weniger Speicher zu verwenden als für Zeichen, die eher selten auftreten. Dies bedeutet, daß zum Beispiel beim Speichern eines Textes nicht für jedes Zeichen 8 Bit verwendet werden, sondern das häufigste Zeichen nur mit 2 Bit kodiert wird. Es entsteht hierbei jedoch ein Problem: Bei der Dekodierung ist nicht bekannt, wie lang das aktuelle Zeichen ist. Dieses Problem könnte, wie bei der Lauflängenkodierung, mit Hilfe eines Escape–Zeichens gelöst werden, aber es gibt eine einfachere und effektive Lösung: Wenn keiner der neuen Zeichenkodes mit dem Anfang eines anderen übereinstimmt, läßt sich die kodierte Zeichenfolge eindeutig dekodieren. Es ist aber auf alle Fälle notwendig, das neue Alphabet mit den kodierten Daten zu übertragen, da dies für die Dekodierung zwingend erforderlich ist. Es folgt nun ein Beispiel, welches kurz darstellt, wie das Verfahren prinzipiell funktioniert. Es soll das Wort „genannt" kodiert werden. Dazu wird als erstes ein neues Alphabet definiert:

a=00; e=10; g=010; t=011; n=11.

Mit Hilfe diese Alphabetes läßt sich die kodierte Zeichenfolge erzeugen:

0101011001111011.

Schaut man sich das Ergebnis der Kodierung an, wird man sehen, daß es anhand des Alphabetes nur eine Möglichkeit zur Dekodierung gibt. Die einfachste Möglichkeit zu garantieren, daß kein Zeichenkode mit dem Anfang eines anderen übereinstimmt, ist die Darstellung des Alphabetes als Trie[9]. Der Zeichenkode wird durch den Pfad von der Wurzel zu diesem Zeichen bestimmt, wobei ‚0' für „gehe nach links" und ‚1' für „gehe nach rechts" steht. Die Dekodierung läuft damit sehr einfach ab: An der Wurzel des Baumes beginnend wird dieser nach den Vorgaben der kodierten Zeichenfolge durchlaufen. Je-

[9] Der Begriff Trie leitet sich von „re trieval datastructure" ab und stellt einen Baum dar, in welchem nur in den Blättern (Endknoten) Informationen gespeichert sind. Der Pfad durch diesen Baum wird zur Kodierung der Informationen verwendet. Weitere Informationen zu Tries sind z.B. in [SED91] zu finden.

desmal, wenn ein äußerer Knoten erreicht ist, wird das diesem Knoten zugeordnete Zeichen ausgeben und an der Wurzel des Baumes fortgefahren. In Abbildung 2 wird der Trie dargestellt, welcher das neu definierte Alphabet darstellt.

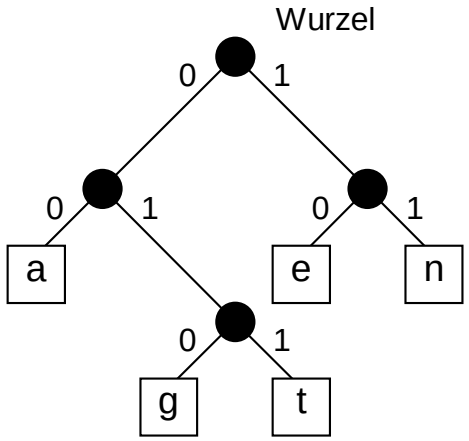


Abbildung 2: Darstellung eines Trie, erzeugt aus dem Wort „genannt“

Es stellt sich nun die Frage: Wie läßt sich für eine beliebige gegebene Zeichenfolge ein Trie berechnen, welcher zu einer Bit-Folge minimaler Länge führt? Ein allgemeines Verfahren dafür wurde 1952 von D. Huffman entwickelt und später nach ihm benannt. Die Huffman–Kodierung wird im folgenden kurz beschrieben, für eine genauere Erläuterung und Details zur Implementierung sei auf weiterführende Literatur verwiesen, z.B. [SED91].

Als erster Schritt wird die Häufigkeit der einzelnen Zeichen in der zu kodierenden Zeichenfolge festgestellt. Dies erfolgt durch einfaches Durchlaufen der Zeichenfolge und die Erhöhung des jeweiligen Eintrages in einer Tabelle, welche alle Zeichen des Alphabetes enthält. Für die Wortgruppe „Beschreibung der Huffman–Kodierung“ sieht diese Häufigkeitstabelle wie in Tabelle 2 dargestellt aus.

Zeichen	a	b	c	d	e	f	g	h	i	m	n	o	r	s	u	B	H	K	–
Häufigkeit	1	1	1	2	4	2	2	1	2	1	3	1	3	1	3	1	1	1	1

Tabelle 2: Häufigkeiten für "Beschreibung der Huffman–Kodierung"

Aus dieser Tabelle wird nun der Kodierungs–Trie aufgebaut. Während der Erzeugung wird ein binärer Baum verwendet, wobei die Häufigkeiten in den

Knoten gespeichert werden. Als erstes wird für jedes Zeichen, dessen Häufigkeit ungleich Null ist, ein Baum erzeugt, welcher nur aus dem Wurzelknoten besteht. Die jeweilige Häufigkeit wird in diesem gespeichert (siehe Abbildung 3 „Schritt 1“). Nun werden zwei Wurzelknoten mit den niedrigsten Häufigkeiten ausgewählt. Diese bekommen einen neuen sogenannten Vater. In diesem wird die Summe der Häufigkeiten der beiden ausgewählten Knoten gespeichert (siehe Abbildung 3 „Schritt 2“). Sollten mehr als zwei Wurzelknoten in Frage kommen, ist es egal, welche gewählt werden. Dieses Verfahren wird jetzt mit den verbleibenden Wurzelknoten fortgesetzt. In jedem Schritt werden zwei Wurzeln entfernt und eine hinzugefügt. Die Zahl der verbleibenden Wurzeln verringert sich somit in jedem Schritt um eins, dadurch bleibt nach endlich vielen Schritten nur noch ein Baum übrig (siehe Abbildung 3 „Schritt 19“). Es brauchen nun nur noch die in den Blättern gespeicherten Häufigkeiten durch die passenden Zeichen aus der Tabelle ersetzt werden. Weiterhin werden die Häufigkeiten in den inneren Knoten nicht mehr benötigt (siehe Abbildung 3 „Letzter Schritt“). Deutlich ist zu erkennen, daß Buchstaben mit hoher Häufigkeit (‚e‘), weiter oben im Trie angeordnet sind als Buchstaben mit geringer Häufigkeit. Demzufolge ist der Pfad zu diesen Buchstaben kürzer und deshalb auch die Länge ihres Kodes.

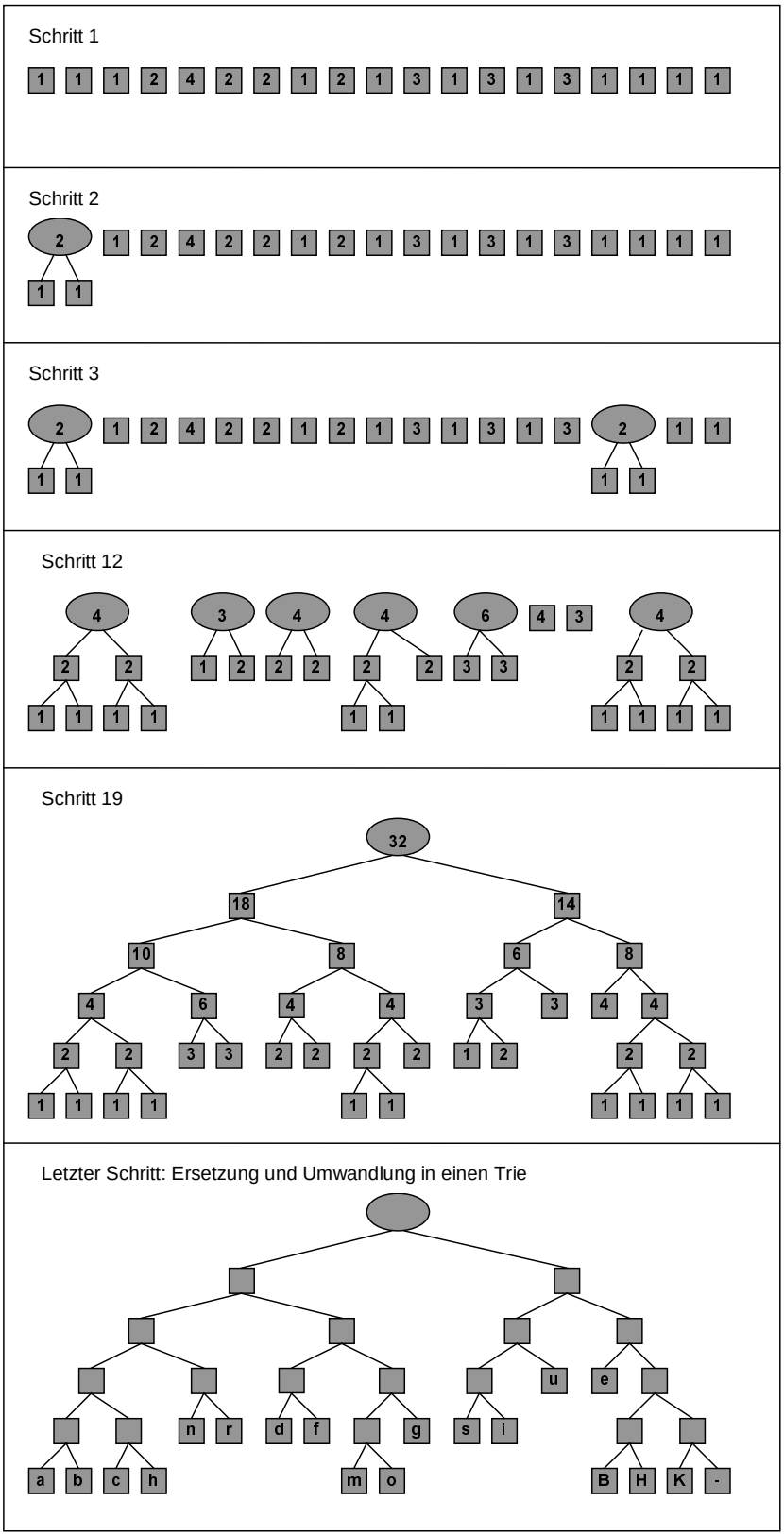


Abbildung 3: Aufbau des Huffman–Baumes

3.4.3 Diskrete Cosinus–Transformation

Das Ziel bei verlustbehafteten Kompressionsverfahren besteht darin, wichtige Informationen auf wenige und bekannte Elemente zu konzentrieren. Nachdem die Trennung zwischen wichtigen und unwichtigen Elementen erfolgt ist, können mit Hilfe der Quantisierung die unwichtigen Elemente entfernt werden. Der Informationsverlust hält sich somit in Grenzen. Die Trennung läßt sich zum Beispiel durch Transformationen durchführen. Es ergeben sich aber zwei besonders wichtige Kriterien, die solche Transformationen erfüllen müssen. Da man die wichtigen Informationen wieder benötigt, müssen die Transformationen invertierbar sein. Weiterhin sollte eine schnelle Berechenbarkeit gewährleistet sein, um sowohl Softwarelösungen als auch preiswerte Hardwareimplementierungen zu ermöglichen. Diese beiden Kriterien werden im Bereich der Bildverarbeitung besonders gut von der zweidimensionalen Diskreten–Cosinus–Transformation (DCT) erfüllt.

Damit man die DCT auf ein Bild anwenden kann, wird dieses zuerst in Blöcke von 8×8 Pixeln $\{F(x,y)\}_{x,y_\in\{0,...,7\}}$ zerlegt. Diese Blöcke werden nun mit Hilfe der Gleichung (3) so transformiert, daß sich die wichtigen Informationen links oben in dem neuen 8×8-Block $\{f(k,n)\}_{k,n_\in\{0,...,7\}}$ befinden (siehe Abbildung 8).

$$f(k,n) := \frac{C(k)}{2}\frac{C(n)}{2}\sum_{x=0}^{7}\sum_{y=0}^{7}F(x,y)\cos\left(\frac{\pi(2x+1)k}{16}\right)\cos\left(\frac{\pi(2y+1)n}{16}\right) \qquad \textbf{(3)}$$

wobei $k,n \in \{0,\ldots,7\}$ und

$$C(z) := \begin{cases} 1/\sqrt{2}, & \text{für } z = 0 \\ 1, & \text{für } z > 0 \end{cases}$$

Die Idee besteht darin, daß das menschliche Auge feine Details, sogenannte hohe Ortsfrequenzen, nicht so gut erkennen kann. Gleichung (3) kann folgendermaßen beschrieben werden: Das aus 8×8 Pixeln bestehende Ausgangsbild $\{F(x,y)\}_{x,y_\in\{0,...,7\}}$ wird als Matrix in $M_8(R)$ aufgefaßt und bezüglich einer anderen Basis $\{B_{k,n}\}_{k,n_\in\{0,...,7\}}$ von $M_8(R)$ dargestellt (siehe Abbildung 4).

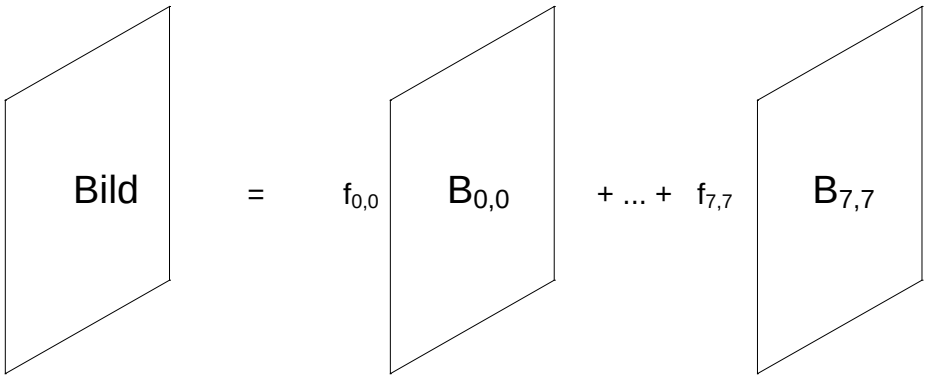
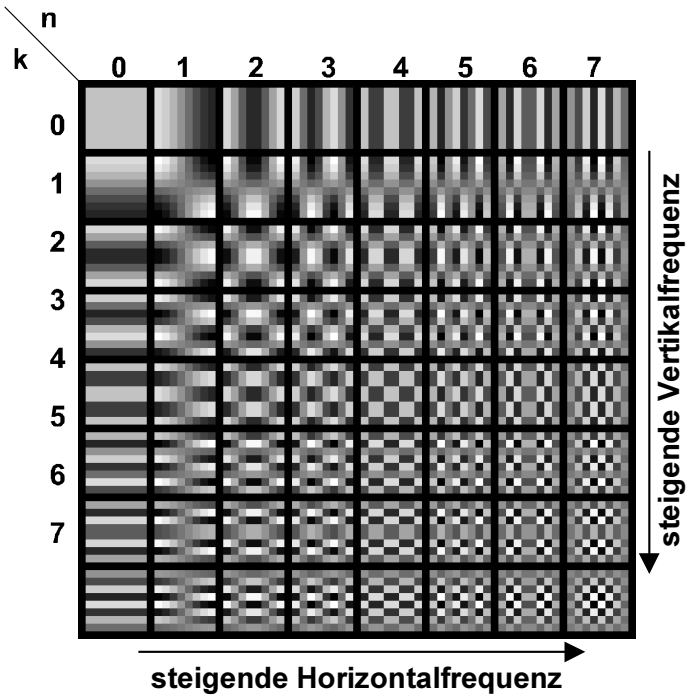


Abbildung 4: Darstellung eines Bildes mit Basisbildern nach [MIL95]

Die in der Matrix $\{f(k,n)\}_{k,n\in\{0,\ldots,7\}}$ abgespeicherten Werte stellen also die Koeffizienten dar, welche die Wertigkeit der einzelnen Basisbilder $\{B_{k,n}\}_{k,n\in\{0,\ldots,7\}}$ (siehe Abbildung 5) in Bezug auf das Ausgangsbild $\{F(x,y)\}_{x,y\in\{0,\ldots,7\}}$ angeben. Abbildung 5 gibt einen optischen Eindruck der 64 Basisbilder, welche von Gleichung (4) und Gleichung (5) beschrieben werden. Der Koeffizient f(0,0) wird als DC–Koeffizient bezeichnet und definiert die Grundfarbe des Ausgangsbildes. Die restlichen 63 Koeffizienten werden als AC–Koeffizienten bezeichnet. Gleichung (4) stellt in den Zeilen der Matrix $A:=\{A(k,n)\}_{k,n\in\{0,\ldots,7\}}$ die Basisfunktionen der eindimensionalen DCT dar. Durch Matrizenmultiplikation der transponierten k–ten und der n–ten Zeile von A werden dann die Basisbilder $\{B_{k,n}\}_{k,n\in\{0,\ldots,7\}}$ definiert (siehe Gleichung (5)).

$$A(k,n) := \begin{cases} \frac{1}{2\sqrt{2}} \text{ für } k = 0 \text{ und } n \in \{0,\ldots,7\} \\ \frac{1}{2}\cos\frac{\pi(2n+1)k}{16} \text{ für } k \in \{1,\ldots,7\} \text{und } n \in \{0,\ldots,7\} \end{cases} \quad \textbf{(4)}$$

$$B_{k,n} := \begin{pmatrix} A(k,0) \\ \vdots \\ A(k,7) \end{pmatrix} \cdot (A(n,0),\ldots,A(n,7)) \quad \textbf{(5)}$$



Die DCT-Basisbilder $\{B_{k,n}\}_{k,n\in\{0,\ldots,7\}}$ bestehen aus jeweils 8 x 8 Pixeln

Abbildung 5: DCT–Basisbilder nach [MIL95]

Bei der bis jetzt durchgeführten Transformation (siehe Gleichung (3)) treten mathematisch gesehen keinerlei Informationsverluste auf. Bei der praktischen Umsetzung kommt es jedoch aufgrund von Rechenungenauigkeiten bereits zu ersten Verlusten. Aus Optimierungsgründen werden die Berechnungen häufig nur ganzzahlig durchgeführt, dadurch läßt sich bei günstiger Implementierung ein großer Teil der Rechenzeit einsparen. Von W. B. Pennebaker und J. C. Mitchell werden in [PEMI92] Verfahren beschrieben, die anstelle der 64 Multiplikationen und 63 Additionen zur Berechnung eines DCT–Koeffizienten durchschnittlich weniger als eine Multiplikation und neun Additionen benötigen.

Zur Umwandlung der Koeffizientenmatrix $\{f(k,n)\}_{k,n\in\{0,\ldots,7\}}$ in das Ausgangsbild $\{F(x,y)\}_{x,y\in\{0,\ldots,7\}}$ wird die zweidimensionale inverse DCT (IDCT) verwendet (siehe Gleichung (6)). Mit Hilfe der Gleichung (5) zur Definition der Basisbilder läßt sich Gleichung (6) anschaulicher mit Gleichung (7) beschreiben, welche der Abbildung 4 entspricht.

$$F(x,y) := \sum_{k=0}^{7} \sum_{n=0}^{7} \frac{C(k)}{2} \frac{C(n)}{2} f(k,n) \cos\left(\frac{\pi(2x+1)k}{16}\right) \cos\left(\frac{\pi(2y+1)n}{16}\right) \quad \textbf{(6)}$$

wobei $x, y \in \{0,\ldots,7\}$ und

$$C(z) := \begin{cases} 1/\sqrt{2}, & \text{für z = 0} \\ 1, & \text{für z > 0} \end{cases}$$

$$F(x,y) = \sum_{k=0}^{7} \sum_{n=0}^{7} f(k,n) B_{k,n}(x,y) \quad x,y \in \{0,\ldots,7\} \quad \textbf{(7)}$$

3.5 Standbildkompression

Unter der Standbildkompression (Intraframekompression) versteht man Kompressionsverfahren, welche zu einer Reduzierung der räumlichen Redundanz in einem Bild führen.

Im folgenden soll der JPEG–Standard erläutert werden. JPEG (Joint Photographic Experts Group) steht für den Namen der Expertengruppe, die in Zusammenarbeit der ISO[10] und CCITT[11] zur Ausarbeitung eines Standards für „Digital Compression and Coding of Continuoustone Still–Images“[12] eingesetzt wurde. Der Standard erhielt den Namen der Expertengruppe und ist heute weit verbreitet. Es existieren sowohl ausgereifte Softwareimplementierungen als auch eine Vielzahl leistungsfähiger Hardwareimplementierungen.

Der JPEG–Standard definiert vier verschiedene Grundverfahren, einen Überblick über deren wichtigste Eigenschaften gibt Tabelle 3.

---

[10] ISO – International Standardization Organization

[11] CCITT – Consultative Committee for International Telephone and Telegraph

[12] Der Standard besteht aus zwei Teilen:

- Part 1 – Requirements and Guidelines: ISO/IEC 10918–1 bzw. ITU/CCITT T.81;
- Part 2 – Compliance Testing: ISO/IEC 10918–2 bzw. ITU/CCITT T.83;

Sequentielles DCT–basiertes Verfahren (Basisverfahren)
DCT–basiertes Verfahren Quellbild: 8–Bit–Farbtiefe pro Sample Sequentiell Quantisierung: 4 Tabellen Huffman–Kodierung: 2 AC– und 2 DC–Tabellen Interleaved und non–interleaved Scans Nicht skalierbar
Erweitertes DCT–basiertes Verfahren (Basisverfahren)
DCT–basiertes Verfahren Quellbild: 8– oder 12–Bit–Farbtiefe pro Sample Sequentiell oder progressiv Quantisierung: 4 Tabellen Huffman– oder Arithmetische Kodierung 4 AC– und 4 DC–Tabellen Interleaved und non–interleaved Scans Nicht skalierbar
Verlustfreies Verfahren
Auf Vorhersage beruhendes Verfahren Quellbild: N–Bit–Samples ($2 \leq N \leq 16$) Sequentiell Huffman– oder Arithmetische Kodierung: 4 DC–Tabellen Interleaved und non–interleaved Scans Nicht skalierbar
Hierarchisches Verfahren
Verwendet erweitertes DCT–basiertes und / oder verlustfreies Verfahren Skalierbar

Tabelle 3: JPEG–Verfahren und ihre wichtigsten Eigenschaften (übernommen aus [MIL95])

Im folgenden wird der sequentielle DCT–basierende Modus vorgestellt, da nur er im Moment eine Bedeutung bei der Verarbeitung von digitalem Video besitzt. Die meisten JPEG–Implementierungen beschränken sich auf den sequentiellen DCT–basierten Modus. Eine der MJPEG–Karten, welche Gegenstand der Vergleichsuntersuchungen ist, stellt jedoch auch den verlustfreien Modus zur Verfügung. Für weitergehende Informationen sei auf die einschlägige Fachliteratur verwiesen, so zum Beispiel [MIL95] und [OHM95].

Vom Prinzip her ist JPEG von Farbmodell, Farbtiefe und Bildauflösung unabhängig. Ein Quellbild darf jedoch nicht aus mehr als 255 verschiedenen Farbkomponenten bestehen und muß ein rechteckiges Format haben. Bei der Beschreibung wird davon ausgegangen, daß das Quellbild im RGB–Format vorliegt, wobei pro Farbkomponente 8–Bit verwendet werden. Für die Ausgabe werden dieselben Vorgaben angenommen. Unter diesen Voraussetzungen sieht der Ablauf von Kodierung und Dekodierung beim sequentiellen DCT–basierten Modus wie in Abbildung 6 dargestellt aus.

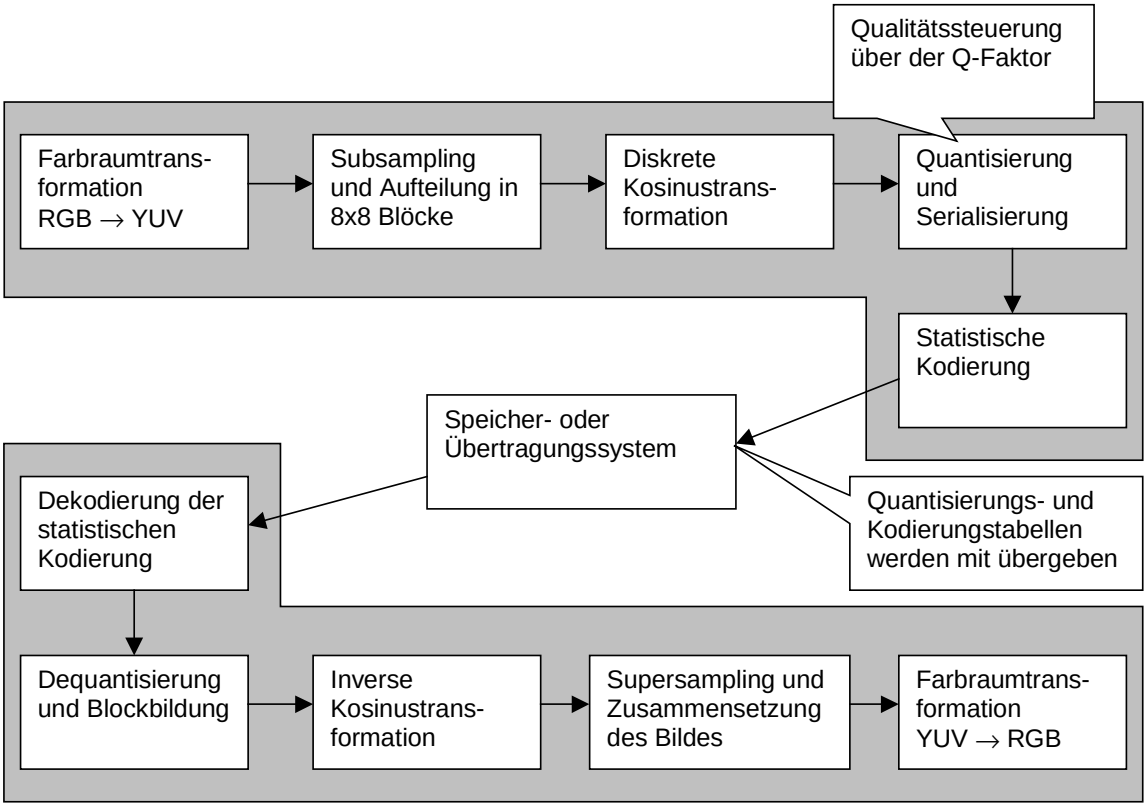


Abbildung 6: Ablauf der Kodierung und Dekodierung bei JPEG (modifiziert nach [MEI94])

Die Farbraumtransformation wird anhand der in Abschnitt 3.1 vorgestellten Gleichung (1) vorgenommen.

Das Subsampling erfolgt für den hier angenommenen Fall[13] mit dem Abtastverhältnis 4:2:2 für die YUV–Werte. Die Halbierung der Auflösung für die U– und V–Werte erfolgt in horizontaler Richtung, es wird also zu je zwei Pixeln der Y–Komponente nur je ein Pixel der U– und V–Komponente

[13] Es wird der CCIR-601-Standard als Beispiel angenommen (siehe Abschnitt 3.3, Tabelle 1).

gespeichert. Diese wird durch Mittelwertbildung aus den zwei Original-Pixeln gebildet. Abbildung 7 illustriert diese Verfahren.

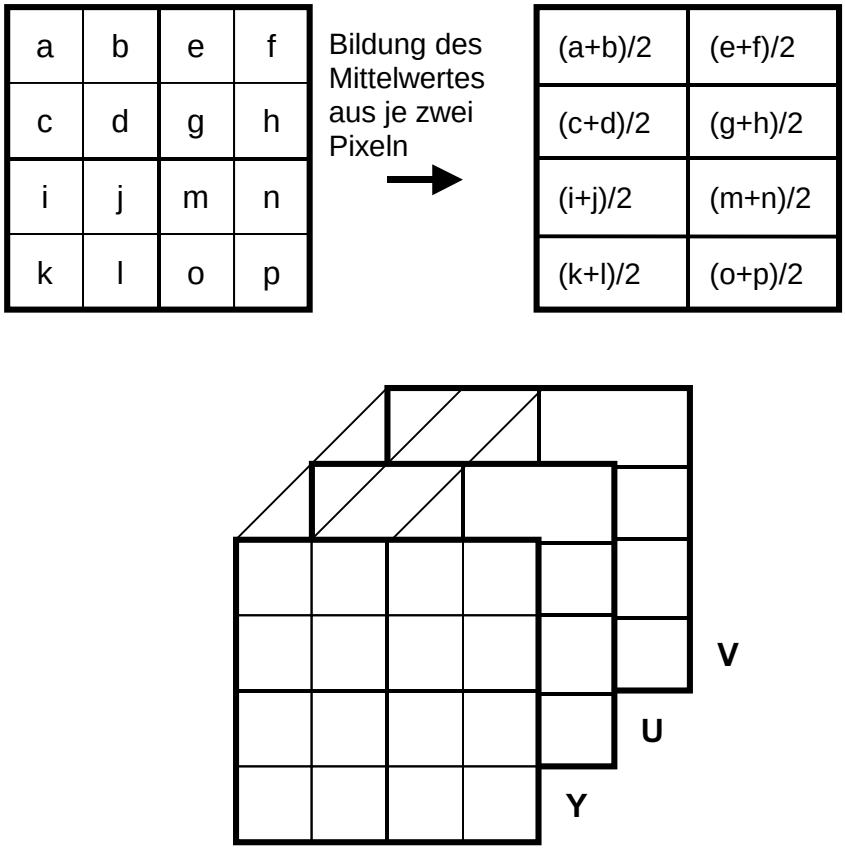


Abbildung 7: Subsampling 4:2:2 YUV

Als nächster Schritt werden die einzelnen Komponenten in 8x8 Blöcke aufgeteilt und diese der in Abschnitt 3.4.3 vorgestellten DCT unterzogen. Diese Aufteilung in 8x8 Blöcke ist der Grund dafür, daß es sinnvoll ist, für die Auflösung der Quellbilder gewisse Vorgaben zu machen. Da für U– und V–Komponenten beim Subsampling eine Halbierung der Auflösung stattfindet, sollte die horizontale Auflösung der Quellbilder ein Vielfaches von 16 sein, in vertikaler Richtung ein Vielfaches von acht. Die nach der DCT erhaltenen 8x8-Blöcke haben den Vorteil, daß die für das menschliche Auge wichtigen Informationen sich hauptsächlich innerhalb des in Abbildung 8 links oben markierten Bereiches befinden.

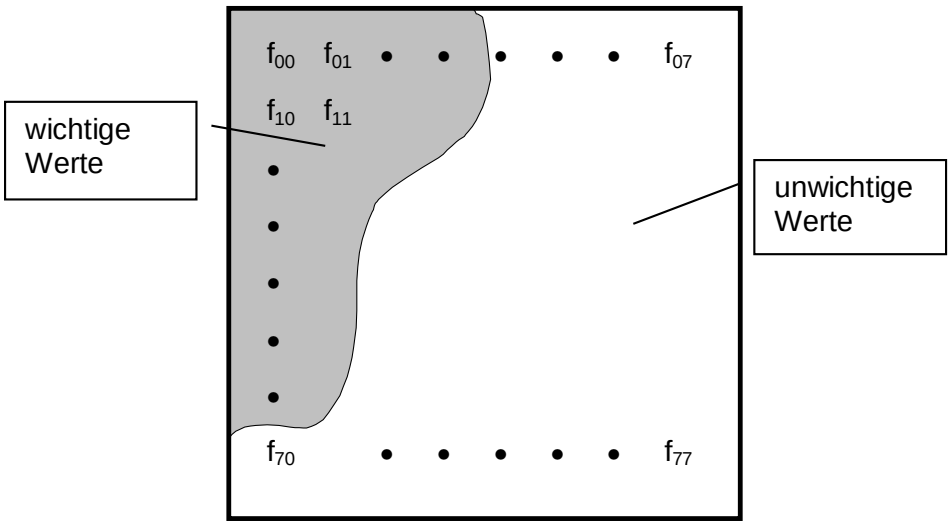


Abbildung 8: Bedeutung der DCT–Koeffizienten (nach [MEI94])

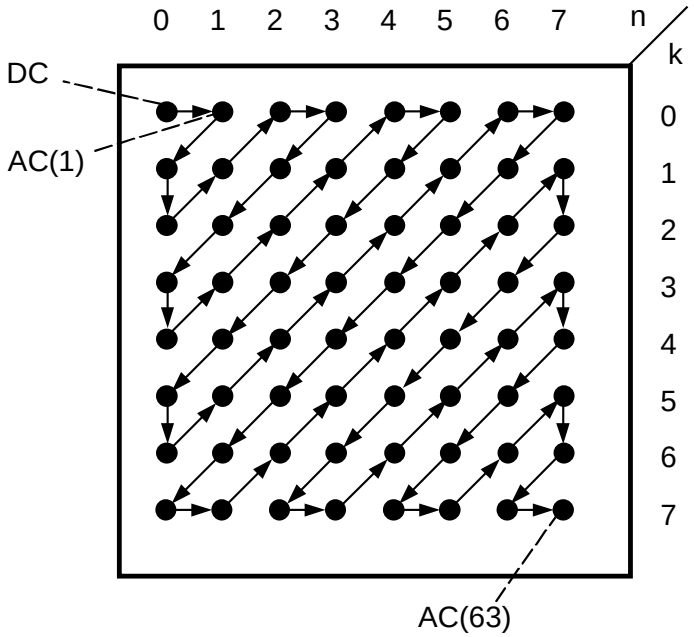


Abbildung 9: Zickzack–Abtastung der Koeffizienten (nach [MIL95])

Damit man bei den nachfolgenden Kodierungsschritten eine möglichst hohe Kompression erreicht, erfolgt nun die Quantisierung der Daten. Dieser Schritt verursacht den eigentlichen Informationsverlust und ermöglicht es dem Benutzer, Einfluß auf die Kompressionsrate zu nehmen. Dafür werden vom Kodierer unter Berücksichtigung des Q–Faktors[14] Quantisierungstabellen er-

[14] Der Q–Faktor (0-255) ermöglicht es, den Kompressionsfaktor zu beeinflussen. Je größer der Q–Faktor gewählt wird, desto stärker wird das Bild komprimiert. Jedoch sinkt damit auch die Qualität.

stellt. Hierbei werden häufig Quantisierungstabellen verwendet, welche im JPEG–Standard (siehe [JPEG93]) vorgestellt werden. Durch das Multiplizieren dieser Tabellen mit dem Q-Faktor werden die eigentlichen Quantisierungstabellen erstellt. Diese Tabellen enthalten für jeden der 63 AC–Koeffizienten einen eigenen Wert. Die Werte in den vorgeschlagenen Tabellen sind darauf ausgelegt, hochfrequente Bereiche besonders stark zu quantisieren. Dadurch werden hauptsächlich die Informationen vernichtet, welche für das menschliche Auge nicht so gut unterscheidbar sind. Bei der Quantisierung werden die Koeffizienten durch den ihnen in den Tabellen zugeordneten Wert dividiert und auf ganze Zahlen gerundet. Es entstehen dabei besonders in der rechten unteren Hälfte der Matrix fast ausschließlich Nullwerte. Die bei der Kodierung verwendeten Tabellen sind bei der Dekodierung unbedingt erforderlich, müssen also zusammen mit den Bilddaten gespeichert werden. Wie man Tabelle 3 entnehmen kann, sind bis zu vier Quantisierungstabellen erlaubt, somit kann für jede Farbkomponente eine auf das Sehvermögen des menschlichen Auges angepaßte Tabelle verwendet werden.

Die Koeffizienten–Matrix wird nun sequentiell abgelegt, hierbei werden die DC– und die AC–Koeffizienten verschieden behandelt. Der DC–Koeffizient eines Blocks wird jeweils in Bezug auf den DC–Koeffizienten des vorhergehenden Blocks DPCM–kodiert[15]. Dies betrifft jeweils die DC–Koeffizienten einer Komponente und basiert auf der Annahme, daß sich die Grundfarben zweier aufeinanderfolgender Blöcke nicht wesentlich unterscheiden. Die AC–Koeffizienten werden durch Zickzack–Abtastung (siehe Abbildung 9) der Koeffizienten–Matrix in eine Folge gebracht. In dieser Folge besteht besonders die zweite Hälfte fast ausschließlich aus Nullen, deshalb läßt sie sich sehr gut durch Lauflängenkodierung (siehe Abschnitt 3.4.1) komprimieren. Die DPCM–kodierten DC–Koeffizienten und die lauflängenkodierten AC–Koeffizienten werden nun Huffman-kodiert (siehe Abschnitt 3.4.2). Es werden hier meist getrennte Kodierungs–Tries für die verschiedenen Koeffizienten verwendet. Diese Tries müssen, ebenso wie die Quantisierungstabellen, an den Dekodierer übermittelt werden. Aus diesem Grund werden sie im Header des JPEG–Datenstroms gespeichert.

[15] Die Differential Pulse Code Modulation (DPCM) soll den Wertebereich von numerischen Eingabezeichen verringern und damit eine bessere Entropie–Kodierung ermöglichen. In dem hier eingesetzten Fall wird der Fehler zum vorhergehenden Wert (die Differenz) gespeichert.

Mit dem sequentiellen DCT–basierenden Modus lassen sich sehr hohe Kompressionsraten erreichen. Je nach Anwendung werden die bei der Kompression auftretenden Qualitätsverluste bis zu Kompressionsraten von 60:1 als tolerierbar angesehen. Wenn man Kompressionsraten von etwa 5:1 verwendet, sind die Qualitätsverluste kaum noch wahrnehmbar.

Die im Rahmen dieser Arbeit untersuchten Hardwarekomponenten benutzen zur Aufzeichnung von Video das JPEG-Verfahren. Da es sich dabei um die Speicherung von bewegten Bildern handelt, wird es in diesem Fall als *Motion*-JPEG bezeichnet. Als Abkürzung dafür steht häufig entweder M-JPEG oder MJPEG.

3.6 Bewegtbildkompression

Das Prinzip der Bewegtbildkompression soll hier am Beispiel von MPEG (Motion Picture Expert Group) erläutert werden. MPEG steht für den Namen einer Arbeitsgruppe der ISO, welche seit 1988 an der Standardisierung von Verfahren der Bewegtbildkompression arbeitet. Der von dieser Gruppe ausgearbeitete Standard heißt *„Coding of Moving pictures and associated audio“*. Im Allgemeinen spricht man jedoch meist vom MPEG–Standard. Obwohl im Standard neben der Videokompression auch die Audiokompression enthalten ist, wird hier nur der Videoteil behandelt.

Im Jahr 1991 wurde der MPEG–1 Standard veröffentlicht (ISO 11172). MPEG–1 ist in erster Linie darauf ausgerichtet, bei relativ geringer Bandbreite (1 bis 1,5 MBit/s) digitales Video mit akzeptabler Bildfrequenz (25 bzw. 30 Hz) und das dazugehörige digitale Audiosignal zu speichern. Die Zielplattform war vor allem die Video–CD, welche auf Single–Speed–CD–ROM's (Datenrate ≈ 1,2 MBit/s) abgespielt werden kann. Aufgrund dieser Zielsetzung wird bei MPEG–1 im allgemeinen das CIF/SIF-Bildformat (siehe Tabelle 1) verwendet. Es sind zwar prinzipiell höhere Auflösungen möglich, ein beliebiger MPEG–1–Dekodierer muß diese jedoch nicht dekodieren können. Schon vor der Verabschiedung des MPEG–1–Standards begannen die Arbeiten an MPEG–2. MPEG–2 sollte ursprünglich vor allem der Aufnahme und Übertragung von digitalem Video in Studioqualität nach CCIR-601 (siehe Tabelle 1) dienen. Die Bandbreite sollte dabei etwa zwischen 4–6 MBit/s liegen. Im letztendlich verabschiedeten Standard (ISO / IEC 13818–(1–10)) werden jedoch wesentlich flexiblere Vorgaben gemacht.

Damit die Implementierung von Dekodierern nicht zu komplex wird, sind die grundlegenden Eigenschaften in Profilen (Kodierungsmethoden) und Levels (maximale Datenrate, zugehörige Bildgröße und Bildfrequenz) eingeteilt (siehe Tabelle 4). Somit läßt sich relativ genau angeben, welche Funktionen ein Dekodierer unterstützt.

(Parametergrenzen)			„Levels“					
Maximale Anzahl von Pixel/s		Max. Bitrate						
1920 x 1080 x 30	1920 x 1152 x 25	80 MBit/s (1)	High	*	MP@HL	*	*	HP@HL
1440 x 1080 x 30	1400 x 1152 x 25	60 MBit/s (2)	High–1440	*	MP @H14L	*	SSP @H14L	HP @14L
720 x 480 x 30	720 x 576 x 25	15 MBit/s (3)	Main	SP@ML	MP@ML	SNRP @ML	*	HP@ML
352 x 240 x 30	352 x 288 x 25	4 MBit/s	Low	*	MP@LL	SNRP @LL	*	*
				Simple	Main	SNR Scalable	Spatial Scalable	High
				4:2:0, keine bidirektionale Prädiktion	4:2:0, keine Skalierbarkeit	Main + SNR–Skalierbarkeit	Main + Auflösungsskalierbarkeit	Gesamte Funktionalität (einschließlich 4:2:2)

„Profiles“
(Kodierungswerkzeuge, Funktionalität)

Tabelle 4: „Profile“- und „Level“ Organisation in MPEG–2–Video [MIL95]

Mit den zur Zeit in Entwicklung begriffenen Standards MPEG–4 und MPEG–7 werden neue Ziele angestrebt. Wo bisher nur eine Unterteilung in einzelne Videobilder möglich war, soll es nun möglich sein, mit einzelnen Objekten zu arbeiten. MPEG–4 stellt damit vor allem einen Zugang zu multimedialen

Informationen zur Verfügung. Im Datenstrom ist dabei zusätzlich zu den Objekten ihre zeitliche und räumliche Position zueinander definiert (siehe [MPEG4]). MPEG–7 erweitert dieses Konzept, indem es eine Beschreibung der Objekte und Daten ermöglicht. Es eignet sich damit voraussichtlich besonders zur Archivierung von multimedialen Daten, da anhand der Beschreibungen eine effektive Suche nach Informationen möglich wird (siehe [MPEG7]).

Die nun folgenden Erläuterungen zum prinzipiellen Ablauf der MPEG–Kodierung gelten sowohl für MPEG–1 als auch für MPEG–2.

MPEG zählt zu den sogenannten Bewegtbildkompressionsverfahren. Im Gegensatz zu den Standbildkompressionsverfahren wird dabei nicht nur die räumliche Redundanz der Daten genutzt, sondern auch die zeitliche Redundanz. Wenn ein Sprecher vor gleichbleibendem Hintergrund aufgezeichnet wird, so ist dieser Hintergrund nahezu unverändert auf allen Bildern der Aufzeichnung vorhanden. Ein weiteres Beispiel stellen Objekte dar, welche zwar im Bild ihre Position verändern, aber selbst keine Veränderung erfahren. Standbildbasierende Verfahren können aus diesen Wiederholungen keinen Vorteil (höherer Kompressionsgrad) ziehen. Bei der MPEG–Kompression wird diese Redundanz genutzt, um höhere Kompressionsgrade bei vergleichbarer Bildqualität zu erzielen.

MPEG unterscheidet zwischen vier verschiedenen Bildtypen, um einerseits gute Kompressionswerte zu erreichen und andererseits einen möglichst wahlfreien Zugriff auf Einzelbilder zu gewährleisten.

- I–Bilder (intraframe–kodierte Bilder) lassen die zeitliche Redundanz unberücksichtigt und werden nach den Prinzipien des JPEG–Basisverfahrens (siehe Tabelle 3) kodiert. Diese Bilder ermöglichen den wahlfreien Zugriff im Datenstrom, da sie vollkommen unabhängig von vorhergehenden bzw. folgenden Bildern dekodierbar sind. Wenn bei der Übertragung von Videodaten Fehler auftreten, so sind die I–Bilder also die Möglichkeit, trotz vorher fehlender Daten die Übertragung fortzusetzen. Die I–Bilder dienen als Referenz für die nächsten zwei Bildtypen. Da die I–Bilder die zeitliche Redundanz unberücksichtigt lassen, erzielen sie von den im MPEG–Standard definierten Bildtypen den geringsten Kompressionsfaktor.

- P–Bilder (prädikativ–kodierte Bilder) basieren auf vorherigem I– oder P–Bild. Bei der Kompression wird sowohl die räumliche als auch die zeitliche Redundanz genutzt. Für die Kodierung und die Dekodierung wird das Re-

ferenzbild benötigt, dadurch erhöht sich das Datenvolumen je nach Auflösung der Bilder. Es treten aber keine zusätzlichen Verzögerungen auf, da das Referenzbild bereits bearbeitet wurde. Bei der Dekodierung kann nicht wahlfrei auf die P–Bilder zugegriffen werden, sondern es muß das letzte I– oder P–Bild bekannt sein. Zum Zugriff auf ein beliebiges P–Bild, müssen sowohl das letzte I–Bild als auch alle dazwischen liegenden P–Bilder dekodiert werden. Der Nachteil des nicht wahlfreien Zugriffs wird durch die wesentlich höheren Kompressionsfaktoren im Vergleich zu den I–Bildern jedoch ausgeglichen.

- B–Bilder (bidirektional–kodierte Bilder) basieren auf dem vorhergehenden und/oder folgenden I– und/oder P–Bild. Sie werden grundsätzlich nicht selbst als Referenz für andere Bilder verwendet. Da B–Bilder auch auf nachfolgenden Bildern basieren, müssen diese zum Zeitpunkt der Kodierung bzw. Dekodierung bereits verarbeitet sein. Es besteht also bei der Verwendung von B–Bildern ein Unterschied in der Reihenfolge der Aufnahme und der Kodierung (bzw. der Anzeige und der Dekodierung). Bekommt der Kodierer die Aufgabe, die Bildsequenz »IBP« zu kodieren, so muß er sie in der Reihenfolge »IPB« kodieren, da er zur Bearbeitung des B–Bildes das darauffolgende Referenzbild benötigt. Dies hat neben einem erhöhten Datenvolumen (es müssen zwei Referenzbilder zwischengespeichert werden) erhebliche Verzögerungen sowohl bei der Kodierung als auch bei der Dekodierung zur Folge. Bei Anwendungen, welche nur mit geringen Verzögerungen zufriedenstellende Ergebnisse liefern (z.B. Videokonferenzsysteme), stellt dies ein Hindernis dar. Die Rechtfertigung für den Sinn der B–Bilder liegt in den hohen erreichbaren Kompressionsfaktoren, welche die der P–Bilder noch überschreiten. Ein Beispiel dafür ist in Abbildung 10 zu sehen. Große Teile des zweiten Bildes lassen sich nicht im ersten Bild wiederfinden, damit könnten sie in einem P–Bild nicht effizient kodiert werden. Im dritten Bild sind nun aber einige dieser Teile aus dem zweiten Bild immer noch sichtbar, mit Hilfe des Vorwärtsbezuges der B–Bilder läßt sich diese Redundanz nutzen.

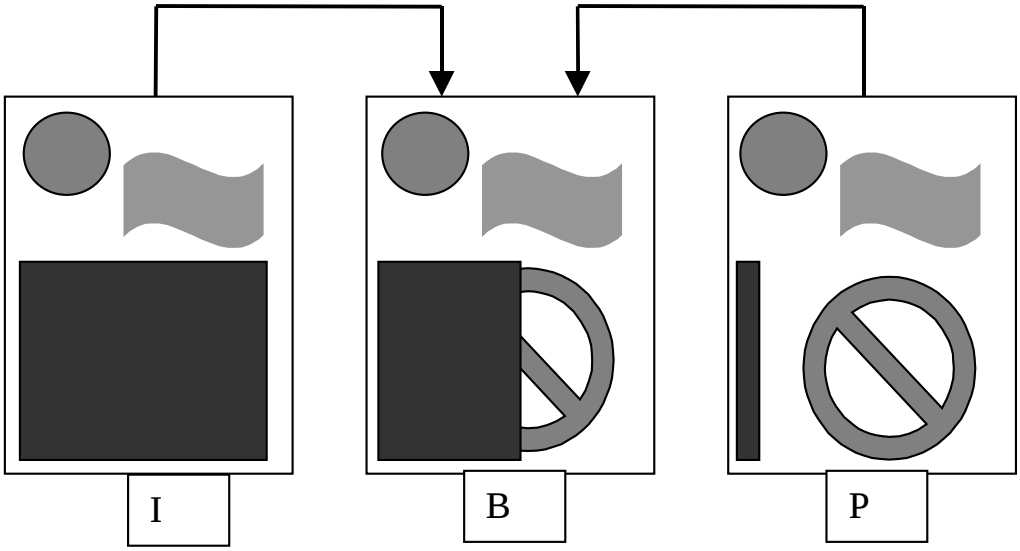


Abbildung 10: Einsatz von B–Bildern zur Nutzung der zeitlichen Redundanz

- D–Bilder (DC–kodierte Bilder) dürfen nicht im normalen Datenstrom vorkommen, sondern können vielmehr in einem eigenen Datenstrom die Möglichkeit einer schnellen Vorschau zur Verfügung stellen. Sie werden nach dem gleichen Verfahren wie die I–Bilder behandelt, jedoch werden nur die DC–Koeffizienten gespeichert. Dadurch ergibt sich nur eine sehr grobe Abstraktion des Originals, die aber wenig Speicher verbraucht.

Die Reihenfolge, in der die Bildtypen in einem Videodatenstrom vorkommen, ist im MPEG–Standard nicht vorgeschrieben. Für einen Datenstrom wird die Folge der Bildtypen in der Datenstruktur GOP (Group of Pictures) vorgegeben. Sowohl der Kodierer als auch der Dekodierer erkennen anhand dieser Struktur, von welchem Typ das aktuelle Bild ist und welche Bilder eventuell als Referenz verwendet werden müssen. Der Videodatenstrom setzt sich nun aus Bildfolgen zusammen, die der GOP entsprechen (siehe Abbildung 11 und Abbildung 12). Mit Hilfe der GOP kann man maßgeblichen Einfluß auf die Eigenschaften des Datenstromes nehmen. Da das erste Bild einer GOP immer ein I–Bild ist und normalerweise nur ein I–Bild pro GOP verwendet wird, beeinträchtigt eine lange GOP den wahlfreien Zugriff. Wird die GOP zu kurz gewählt, kommen verhältnismäßig viele I–Bilder vor, dadurch kann die zeitliche Abhängigkeit des Bildmaterials nicht richtig ausgenutzt werden und die Kompressionsfaktoren erreichen keine optimalen Werte. Der Extremfall tritt ein, wenn die GOP nur aus einem I–Bild besteht, damit wird jedes Bild einer Standbildkompression unterzogen und es werden auch nur die dort möglichen Kompressionsfaktoren erzielt. Da die Entscheidung, wie die GOP aufgebaut

sein soll, nicht offensichtlich ist, wurden für verschieden Anwendungsgebiete Vorgaben geschaffen. Für Bildmaterial mit 25 Hz Bildfrequenz bildet zum Beispiel eine GOP aus neun Bildern, bei der das erste Bild ein I–Bild, jedes weitere dritte ein P–Bild und alle restlichen B–Bilder sind (IBBPBBPBB) einen guten Kompromiß zwischen Kompressionsrate und wahlfreier Zugriffsmöglichkeit (siehe Abbildung 11) [MIL95]. Bei Anwendungen, in denen eine geringe Verzögerung notwendig ist, sollte auf die Verwendung von B–Bildern verzichtet werden.

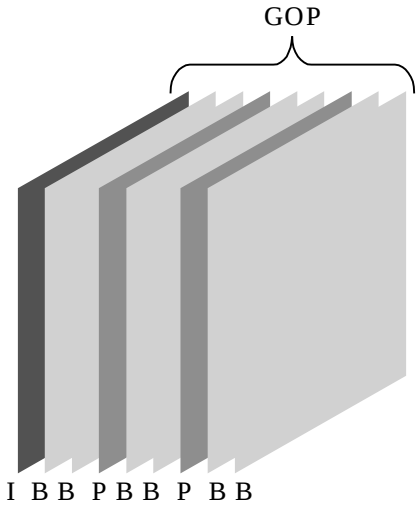


Abbildung 11: Aufbau einer GOP

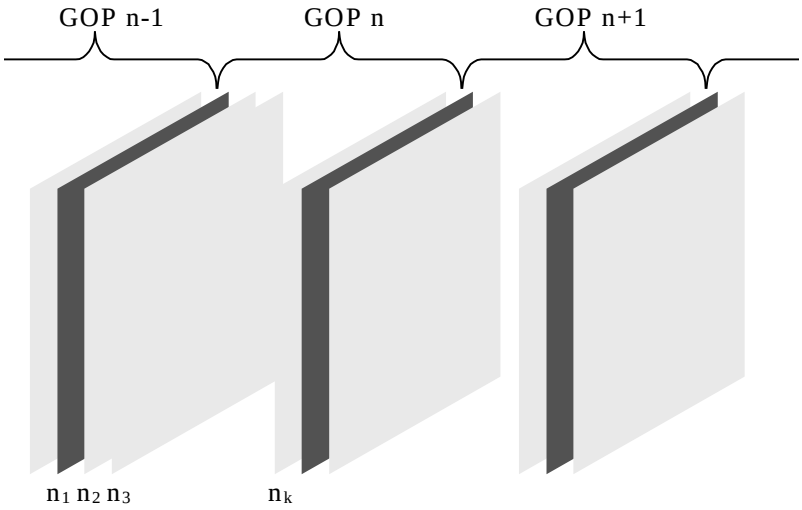


Abbildung 12: Unterteilung einer Videosequenz in GOP‘s von k Bildern

4 Videoverarbeitung im PC

Im folgenden Abschnitt werden die derzeitigen Möglichkeiten der Videoverarbeitung mittels Personalcomputer diskutiert. Es wird dabei ausschließlich auf die Möglichkeiten der Microsoft-Windows-Betriebssysteme (Windows 95, Windows NT 4.0) eingegangen, da zum jetzigen Zeitpunkt nur dort eine Treiberunterstützung der getesteten Hardwareprodukte zur Verfügung steht.

4.1 Anforderungen

Im Rahmen dieser Arbeit werden bestimmte Qualitätsmerkmale von den zu verarbeitenden Videoströmen gefordert:

- Das Videomaterial soll dem CCIR-601-Format (siehe Tabelle 1) entsprechen.
- Es soll möglich sein, mindestens zwei Videoströme gleichzeitig zu verarbeiten und auf dem Bildschirm darzustellen.
- Bei der Komprimierung sollen möglichst keine sichtbaren Verluste entstehen. Damit diese Forderung erfüllt werden kann, muß bei Standbildkompressionsverfahren wie JPEG eine Kompressionsrate von etwa 5:1 verwendet werden (siehe Abschnitt 3.5).

Da bei unkomprimiertem Videomaterial im CCIR-601-Format 30 MByte/s (8-Bit-RGB-Darstellung) bzw. 20 MByte/s (YUV 4:2:2) an Daten pro Datenstrom verarbeitet werden müssen, ergibt sich zusammen mit den komprimierten Datenströmen eine Gesamtdatenmenge von 76 MByte/s bei RGB-Darstellung bzw. 56 MByte/s bei YUV-Darstellung. In Abbildung 13 werden die Datenströme für eine Beispielkonfiguration dargestellt.

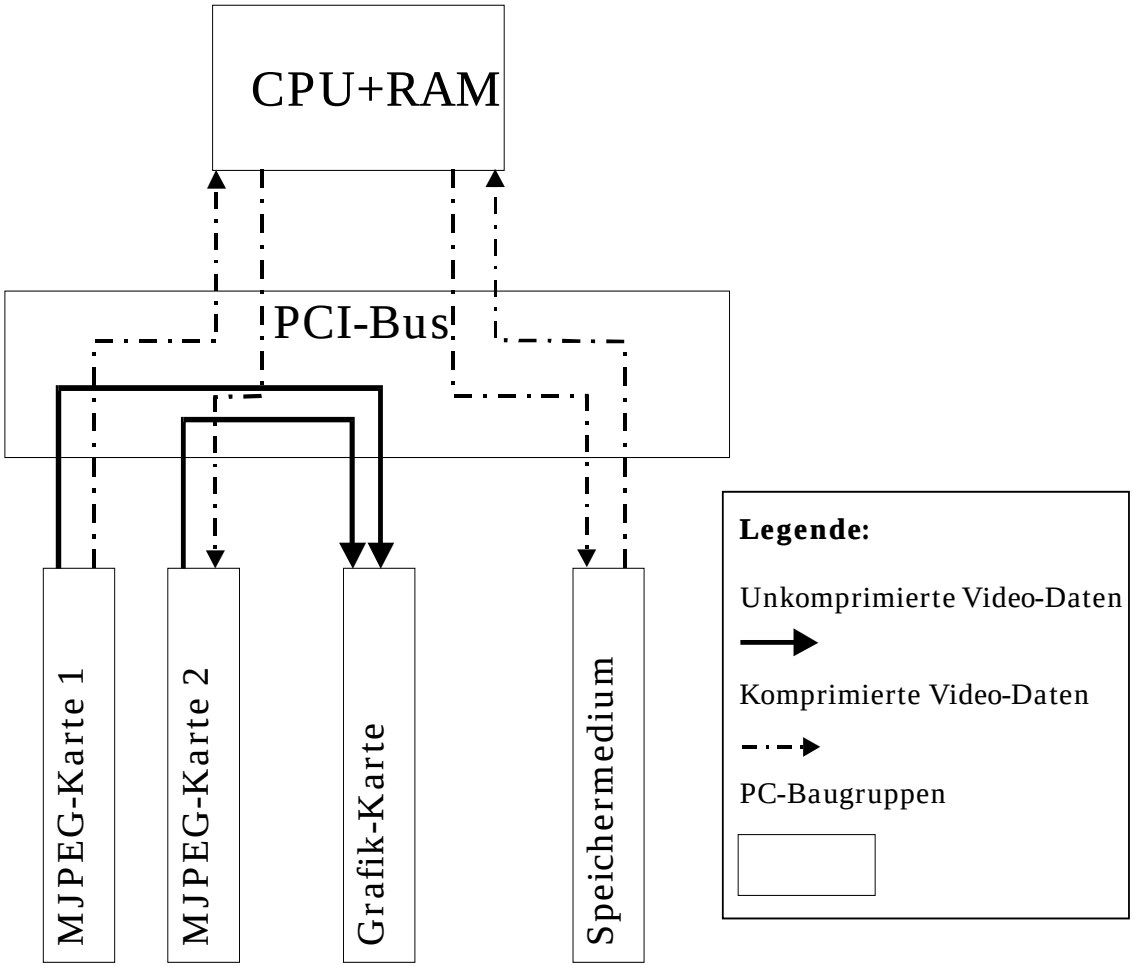


Abbildung 13: Darstellung der Datenströme beim Einsatz zweier unabhängiger MJPEG-Karten

4.2 Hardwarebetrachtungen

Die heute eingesetzten Personalcomputer verwenden zur internen Datenübertragung fast ausschließlich den PCI-Bus. Eine Haupteigenschaft des PCI-Konzeptes ist die Entkopplung des Prozessor-Hauptspeicher-Subsystems vom Standarderweiterungsbus. Die Verbindung zwischen Prozessor-Hauptspeicher-Subsystem und PCI-Bus stellt die PCI-Bridge her. Mit deren Hilfe ist es möglich, daß einzelne PCI-Einheiten untereinander Daten austauschen, ohne dabei den Prozessor zu belasten (Busmaster-Transfer). Der Prozessor initiiert lediglich die Übertragung und kann sofort wieder andere Aufgaben übernehmen. Die Standardbusbreite für den PCI-Bus beträgt 32 Bit[16], damit

[16] Bereits im Entwurf wurde eine Erweiterung auf 64-Bit vorgesehen, inzwischen wird diese im Serverbereich auch verwendet.

ergibt sich bei einer Betriebsfrequenz von 33 MHz eine maximale Transferrate von 133 MByte/s. Da der PCI-Bus ein Multiplexing-Schema[17] verwendet, stellt diese Zahl nur einen Wert dar, der in speziellen Ausnahmefällen erreicht wird. Bei sich nicht sequentiell ändernden Adressen beträgt die maximale Datenübertragungsrate 66 MByte/s (Schreiben) bzw. 44 MByte/s (Lesen). In der Praxis werden jedoch häufig größere Blöcke zusammenhängender Daten hintereinander übertragen, diese Fälle werden vom sogenannten Burst-Modus[18] unterstützt. Die PCI-Bridge hat dabei die Aufgabe, sequentielle Einzeltransfers zu erkennen und sie selbständig zu Bursttransfers zusammenzusetzen. Nur dadurch ist es möglich, die maximale Transferrate von 133 MByte/s zu erreichen.

Momentan werden Transferraten in dieser Größenordnung hauptsächlich von Grafikkarten unterstützt und benötigt. Bei Bildschirmauflösungen von 1024 x 768 Bildpunkten und 32 Bit Farbtiefe (3 MByte/Bild) werden erhebliche Datenraten produziert, sobald häufig wechselnde Bilder (z.B. Videos) dargestellt werden. Die meisten anderen PCI-Adapter (SCSI-Adapter, Netzwerk-Adapter) benötigen im gegenwärtigen Moment noch keine vergleichbaren Transferraten, so daß der PCI-Bus, für diese Adapter jeweils separat betrachtet, keine Behinderung darstellt.

Einen weiteren wichtigen Faktor stellt die Speicherperformance dar. Die derzeit aktuelle Rechnergeneration (Pentium II mit 400 MHz internem und 100 MHz externem Takt) erreicht unter Verwendung von SDRAMs Speichertransferraten von etwa 100 MByte/s. Wie beim PCI-Bus ist auch das ein eher theoretischer Wert, da er ausschließlich bei linearen Speicherzugriffen erreicht werden kann.

Die Voraussetzung zur Videoverarbeitung (in der im Abschnitt 4.1 geforderten Qualität) im PC sind spezielle Erweiterungskarten, welche die Aufgabe der Komprimierung und Dekomprimierung übernehmen. Abbildung 13 stellt eine Beispielkonfiguration dar, in der zwei MJPEG-Adapter diese Aufgaben übernehmen. Die Notwendigkeit ergibt sich aus der Rechenleistung, die von diesen

---

[17] Die Leitungen des Bussystems werden abwechselnd als Adreß- und Datenleitungen verwendet, wodurch Leitungen eingespart werden, aber für Einzeltransfers nun zwei bis drei Taktzyklen nötig sind.

[18] Nach dem einmaligen Übergeben der Adresse erhöhen Sender und Empfänger die Adresse mit jedem Taktzyklus, somit ist die Adresse stets bekannt und braucht nicht neu gesetzt werden.

beiden Vorgängen benötigt wird. Die aktuell in PCs eingesetzten Prozessoren (z.B. Pentium-II mit 400 MHz) erlauben zwar die Dekomprimierung eines Datenstroms in Echtzeit, bei der Komprimierung und Verarbeitung mehrerer Datenströme sind sie jedoch überfordert.

4.3 Die Multimedia-Architektur unter Windows

Seit der Einführung von Windows 3.0 unterstützt Microsoft das Abspielen von Audio- und Videodaten mit verschiedenen Programmierschnittstellen. Zur Wahrung der Kompatibilität werden auch in aktuellen Windows-Versionen ältere Schnittstellen unterstützt. In den folgenden Abschnitten werden die derzeit aktuellen Schnittstellen beschrieben und es wird erörtert, inwieweit sie sich zum Zwecke der Online-Videoübertragung einsetzen lassen. Abbildung 14 gibt einen Überblick über die im folgenden erläuterten Beziehungen der verschiedenen in Windows integrierten Grafik- und Videoschnittstellen.

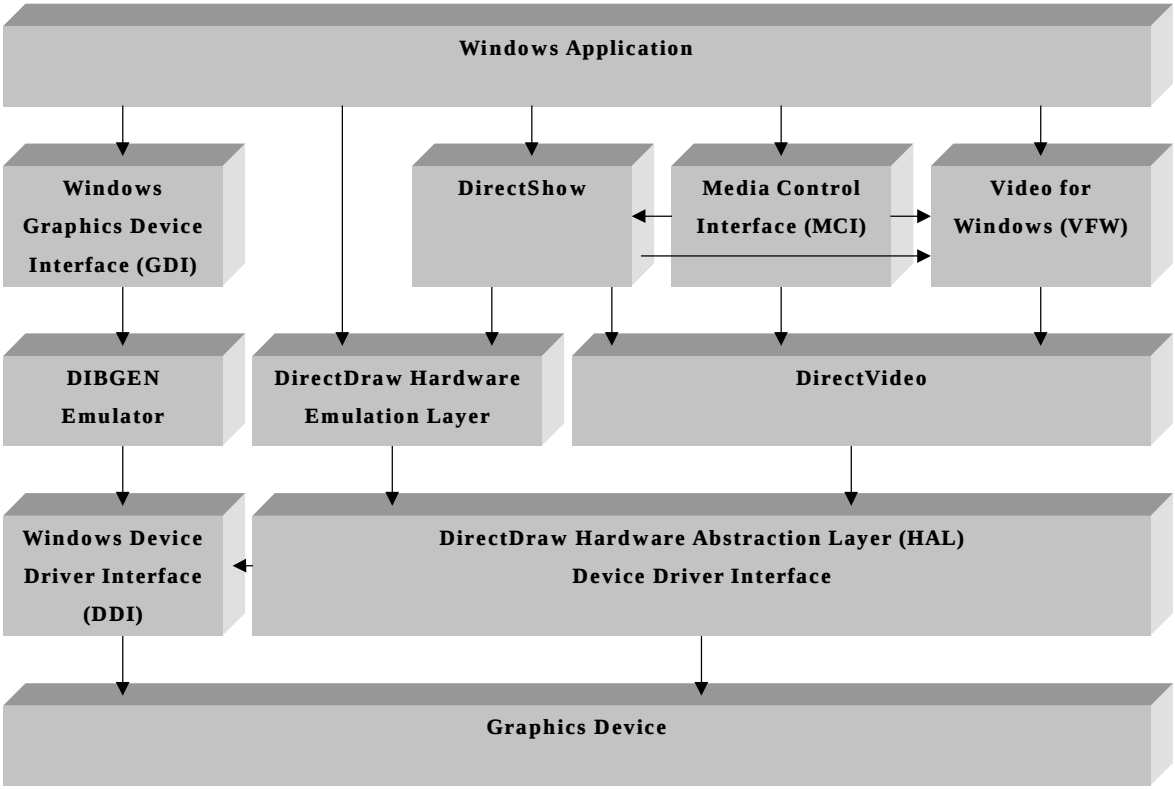


Abbildung 14: Grafik- und Videoarchitektur unter Windows

4.3.1 Das Media-Control-Interface

Die ersten Ansätze zur Unterstützung multimedialer Daten unter Windows wurde durch das *Media Control Interface* (MCI) zur Verfügung gestellt. Das

MCI erlaubt mit Hilfe von Kommandos, die an die Bedienung von Videorecordern angelehnt sind (Starten, Pause, Stop, Zurückspulen usw.), die Daten zu präsentieren. Es fehlen jedoch wichtige Funktionen zum Aufnehmen, Komprimieren und Bearbeiten der Daten.

Die vom MCI zur Verfügung gestellten Funktionen eignen sich nicht zum Implementieren eines Online-Videoübertragungssystems, da keinerlei Möglichkeiten des direkten Datenzugriffs vorhanden sind.

4.3.2 Video-for-Windows

Im Jahr 1993 führte Microsoft ergänzend zum MCI mit *Video for Windows* (VFW) ein neue Schnittstelle ein. Mit Hilfe der Funktionen von VFW ist es möglich, Video- und Audiodaten aufzunehmen und zu editieren. Es wurde eine Schnittstelle geschaffen, die es erlaubt, die Daten zu komprimieren und zu dekomprimieren, wobei sich neue Algorithmen mit Hilfe des Kompressionsmanagers in das System integrieren lassen. Gleichzeitig erfolgte mit dem *Audio Video Interleaved* (AVI) Format die Einführung eines Standards zur Speicherung der Daten. Zur Wahrung der Kompatibilität dient eine Schnittstelle zwischen dem MCI und VFW. Somit können MCI-Applikationen ebenfalls auf Daten zugreifen, die mit den Funktionen von VFW erstellt oder bearbeitet worden sind.

VFW hat jedoch drei entscheidende Schwachstellen, welche eine komplette Neuentwicklung (*DirectShow*) als sinnvoll erscheinen ließen:

1. Es ist nicht möglich, Audio- und Videodatenströme zu synchronisieren.

2. Große Teile von VFW sind weiterhin als 16-Bit-Code implementiert, woraus im Zusammenspiel mit 32-Bit-Betriebssystemen teilweise erhebliche Performanceeinbußen resultieren.

3. Aufgrund von Limitierungen im Header des AVI-Formates ist es nicht möglich, Dateien zu verarbeiten, die größer als 2 GByte sind. Bei Datenraten von 4 MByte/s lassen sich somit nur Videos von etwa 8 Minuten aufnehmen.

Da mit Hilfe der VFW-Funktionen ein direkter Zugriff auf die aufgenommenen Videodaten möglich ist, läßt sich VFW gut zur Implementierung von Online-Videoübertragungssystemen einsetzen.

Ein Beispiel für die Implementierung eines solchen Systems stellt das experimentelle Online-Videokommunikationssystem „Visitphone“ (siehe [NGU98]) dar.

4.3.3 DirectShow

DirectShow wurde 1996 noch unter dem Namen *ActiveMovie* eingeführt. In der ersten Version wird ausschließlich das Abspielen von Daten unterstützt, zum Aufnehmen und Editieren muß weiter auf VFW zurückgegriffen werden. Dafür lassen sich Audio- und Videodatenströme nun synchronisieren und die Performance wurde wesentlich verbessert. Da keinerlei Möglichkeiten zur Aufnahme von Daten vorhanden sind, läßt sich mit Hilfe der ersten Version maximal ein Empfänger (einer Online-Videoübertragung) implementieren. Aufgrund dessen haben viele Hersteller von Videokompressionskarten kein Interesse an der Unterstützung von *ActiveMovie.*

Diese Einstellung hat sich erst mit der Einführung von *ActiveMovie 2.0* geändert. Die Version wurde Anfang 1998 unter dem Namen *DirectShow* vorgestellt. Im folgenden werden ihre wichtigsten Eigenschaften zusammengefaßt, da die im Rahmen der Arbeit zu untersuchende Hardware nur mit *DirectShow* vollständig nutzbar ist.

DirectShow stellt eine komponentenbasierte-Architektur[19] dar, welche die Wiedergabe von Multimediadatenströmen aus lokalen Dateien oder dem Internet erlaubt und mit Hilfe geeigneter Geräte die Aufnahme solcher Datenströme unterstützt. Die einzelnen Funktionen (wie zum Beispiel: Datei lesen, Datei schreiben, Komprimieren, Dekomprimieren, Anzeigen) sind dabei in einzelne Module, die sogenannten Filter, verpackt. Filter sind COM-Objekte und besitzen mindestens einen Ein- oder Ausgang (Pin). Durch das Verbinden der Pins verschiedener Filter entsteht ein sogenannter Filter-Graph. In Abbildung 15 ist ein Filter-Graph dargestellt, welcher eine AVI-Datei liest, diese dekomprimiert und auf dem Bildschirm anzeigt. In diesem einfachen Beispiel kommen die drei verschiedenen Filterarten zum Einsatz, zwischen denen bei *DirectShow* unterschieden wird.

---

[19] Es wird das *Component Object Model* (COM) verwendet, eine ausführliche Darstellung dieses Modells ist zum Beispiel in [BOX98] zu finden.

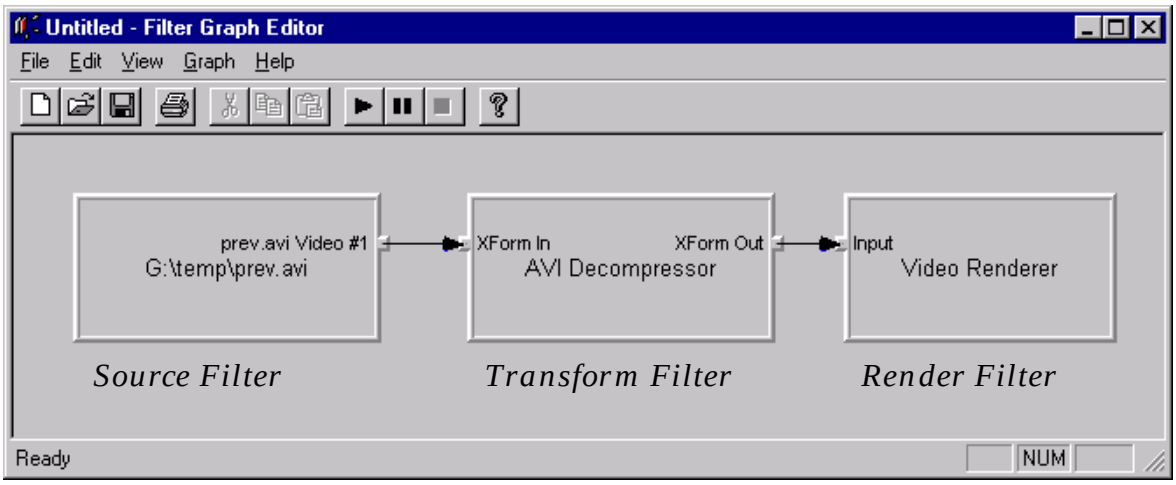


Abbildung 15: Einfacher Filter-Graph zur Darstellung einer AVI-Datei, erzeugt mit Hilfe des Filter-Graph-Editors

- *Source Filter:* Sie stellen die Datenquellen dar und besitzen keine Eingänge, sondern lesen Daten aus einer lokalen Datei, aus dem Internet, von einem Aufnahmegerät bzw. einer anderen vom Programmierer vorgesehen Stelle. Diese Daten stellen sie an mindestens einem Ausgang anderen Filtern zur Verfügung. Zur Datenübergabe werden zwei verschiedene Konzepte eingesetzt. *Source Filter,* die nach dem *Push-Model* arbeiten, übermitteln die Daten selbständig an den Graphen. Sie rufen dazu jedesmal, wenn neue Daten vorhanden sind, die Empfangsroutine des angeschlossenen Eingangs auf und übergeben die Daten. Die zweite Möglichkeit besteht darin, daß der *Source Filter* passiv darauf wartet, daß die Daten abgeholt werden. Der Eingangs-Pin des nachfolgenden Filters ruft dabei die Daten mittels einer Callback-Funktion vom Ausgang des *Source Filters* ab.

- *Transform Filter:* Diese Filter besitzen mindestens einen Eingang und einen Ausgang. Sie erhalten an den Eingängen Daten von einem *Source Filter* oder einem anderen *Transform Filter* und verarbeiten diese. Häufige Aufgaben sind die Komprimierung und Dekomprimierung, Farbraumtransformationen oder das Zusammenfügen und Trennen von verschachtelten Datenströmen. Die bearbeiteten Daten werden an den Ausgängen anderen *Transform* oder *Render Filtern* zur Verfügung gestellt.

- *Render Filter:* Diese Filter sind für die Ausgabe der Daten verantwortlich und besitzen mindestens einen Eingang, aber keine Ausgänge. Üblicherweise erhalten sie die Daten von einem *Transfom Filter*, es ist jedoch auch möglich, einen *Source Filter* direkt an einen *Render Filter* zu koppeln. Die

entgegengenommenen Daten werden zum Beispiel auf dem Bildschirm dargestellt, in eine Datei geschrieben oder an ein Gerät ausgegeben. Hierbei versucht *DirectShow* automatisch, die zur Verfügung stehende Hardware optimal auszunutzen, wobei es die Dienste anderer Betriebssystemmodule (z.B. *DirectVideo*), so wie in Abbildung 14 dargestellt, benutzt.

Zum Erzeugen und Verwalten der Filter-Graphen wird der Filter-Graph-Manager (FGM) eingesetzt. Dieser ist als COM-Objekt implementiert und seine *Interfaces* stellen die Schnittstelle zu den Funktionen von *DirectShow* dar.

Eine Windows-Anwendung kann auf drei verschiedene Arten mit *DirectShow* kommunizieren (siehe Abbildung 16):

1. Durch direkten Zugriff auf den FGM mittels seiner COM-Schnittstellen (siehe Abbildung 16 „1."): Diese Variante ermöglicht es, den gesamten Funktionsumfang zu nutzen und nahezu beliebige Filter-Graphen zu erstellen. Als Testtool bei der Erstellung von Filter-Graphen dient der Filter-Graph-Editor (siehe Abbildung 15).

2. Mit Hilfe des MCI, welches von Microsoft intern so erweitert wurde, daß es *DirectShow* als Werkzeug für die von ihm angebotenen Dienste verwenden kann (siehe Abbildung 16 „2."): Hierdurch lassen sich Geräte verwenden, die nur von *DirectShow* direkt unterstützt werden. Da die MCI-Schnittstelle selbst nicht erweitert wurde, lassen sich aber keine neu angebotenen Dienste nutzen.

3. Die dritte Möglichkeit besteht in der Nutzung eines mit *DirectShow* gelieferten *ActiveX Controls* (siehe Abbildung 16 „3."): Dieses stellt einerseits ein Plugin für den Internet Explorer dar und kann andererseits von Anwendungsentwicklern relativ einfach dazu verwendet werden, *DirectShow*-Medien abzuspielen. Ein direkter Zugriff auf die Daten ist dabei nicht möglich.

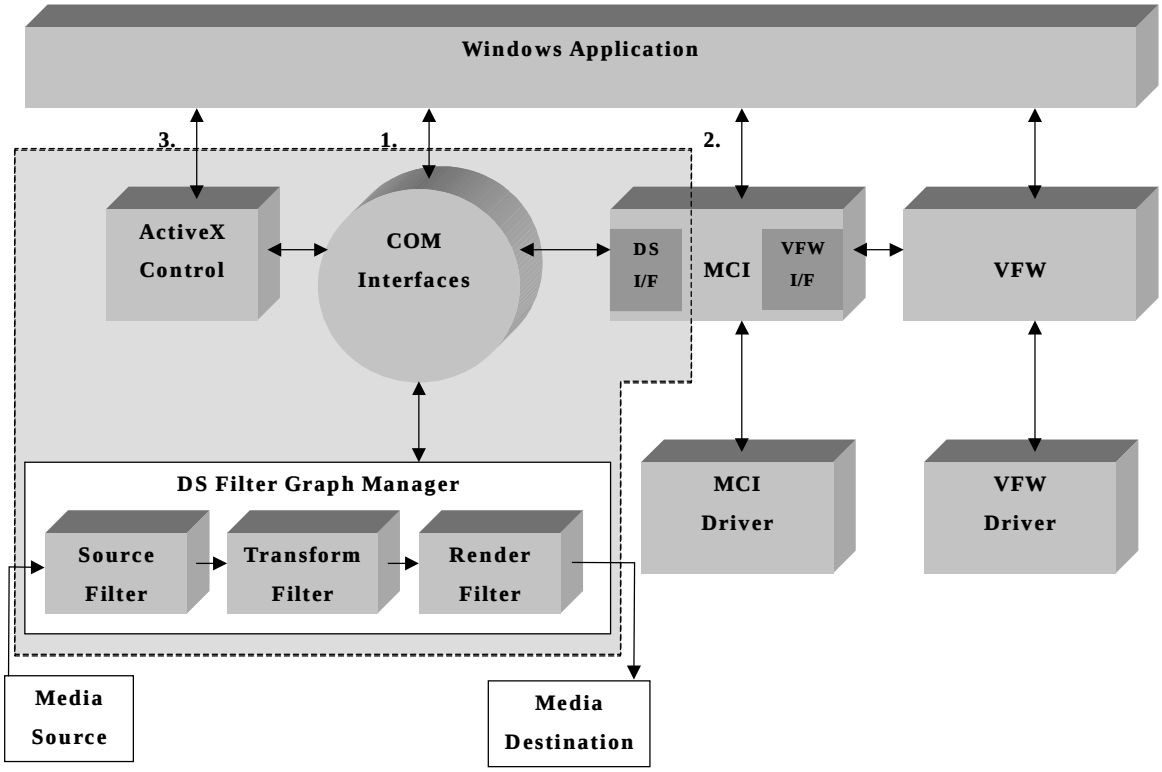


Abbildung 16: *DirectShow* Architektur im Detail nach [COE98]

5 Systemuntersuchungen

In diesem Abschnitt wird die Arbeitsumgebung vorgestellt, in der das zu untersuchende Video-System getestet wurde. Anschließend werden die wesentlichen Merkmale dieses Systems näher erläutert und ein Vergleich mit bisher eingesetzter Technik vorgenommen.

5.1 Arbeitsumgebung

Die Untersuchungen fanden auf einem Dual-Pentium-II-System statt, welches mit zwei Prozessoren ausgestattet ist. Die Prozessoren werden mit 300 MHz getaktet, der Zugriff auf das Hauptspeicher-Subsystem erfolgt mit 66 MHz und der PCI-Bus wird mit 33 MHz betrieben. Das System verfügt über einen Hauptspeicher von 128 MByte RAM und als Festplatte wird eine 9 GByte SCSI-II-Platte verwendet. Mit dieser sind Dauertransferraten von 10 MByte/s möglich. Die Netzwerkanbindung des Systems erfolgt über einen 10 MBit Ethernet-Adapter und einen 155 MBit ATM-Adapter (ForeRunner-200LE)[20].

Zur Videoverarbeitung kam ein Video-Adapter-System der Firma Matrox zum Einsatz, welches im Abschnitt 5.2 näher beschrieben wird.

5.2 Video-Adapter-System

Das eingesetzte System wird unter dem Namen „DigiSuite“ kommerziell vertrieben. Dieser Name steht als Oberbegriff für verschiedene Adapter-Kombinationen. Zur Zeit werden von Matrox fünf Adapter angeboten, welche für die unterschiedlichsten Aufgabengebiete gedacht sind. Bei der Zusammenstellung des Testsystems wurden die folgenden für den geplanten Einsatzzweck nötigen Adapter berücksichtigt:

- DigiMix: Dieser dient als analoger Video-Ein- und Ausgang.
- DigiMotion: Dieser stellt zwei unabhängige MJPEG-Kanäle und einen SCSI-II-Kontroller zur Verfügung.

[20] Der Einsatz dieser Adapter ist aufgrund der angebotenen API Unterstützung notwendig (siehe [FRAN99]).

- DigiDesktop: Er wird als Grafikadapter verwendet. Dabei ist die Benutzung von zwei Bildschirmen und die Darstellung von bis zu vier Video-Overlay-Fenstern möglich.

Alle „DigiSuite“ Baugruppen verfügen über einen Movie-2-Bus-Anschluß. Der Movie-2-Bus dient als Datenbus, über den die einzelnen Adapter zu einem System zusammengefügt werden. Er wird über eine Steckverbindung auf der Oberseite der Steckkarten realisiert. Für die verschiedenen Möglichkeiten der Kombination der Baugruppen gibt es jeweils verschiedene Steckverbinder.

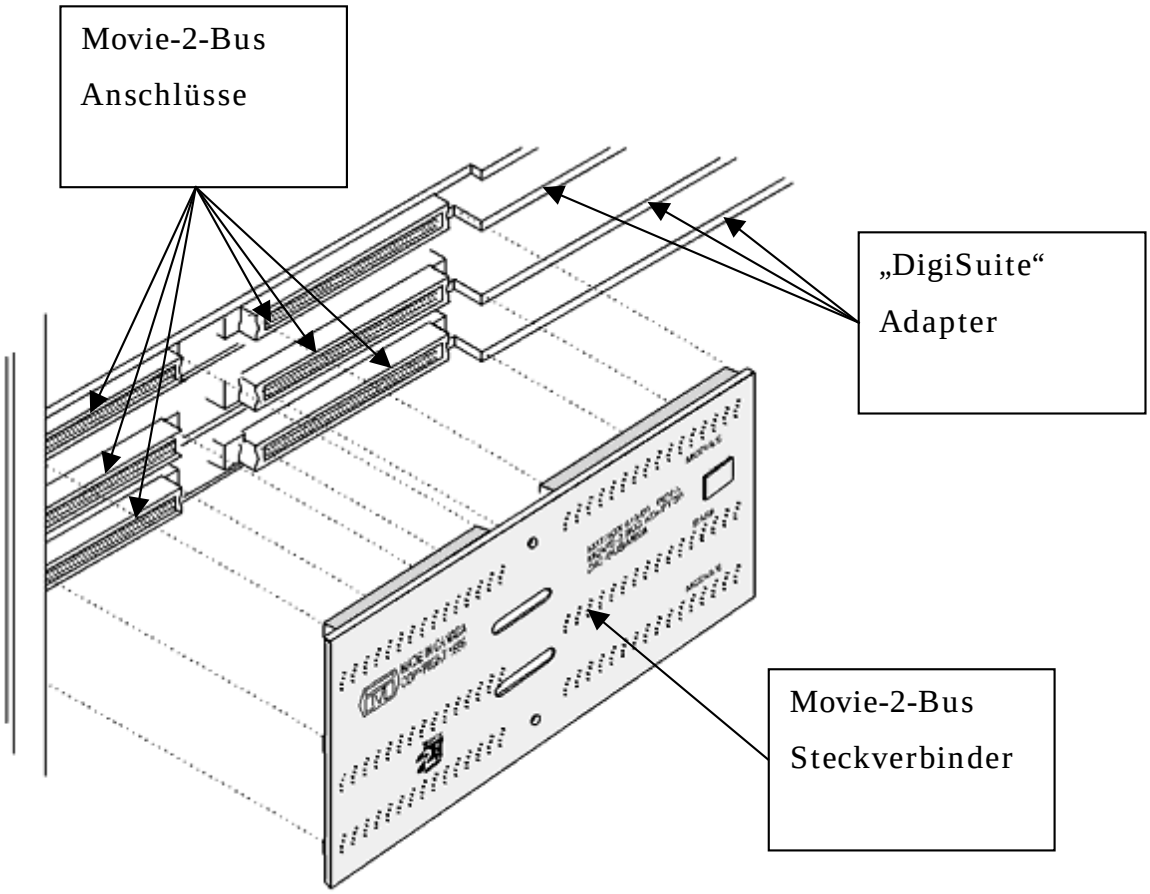


Abbildung 17: Schematische Ansicht eines Movie-2-Bus Systems (abgewandelt übernommen aus [M2BUS]).

5.2.1 Movie-2-Bus

Die technischen Daten der nun folgenden Ausführungen basieren auf dem Dokument [M2BUS].

Der Grund für die Entwicklung des Movie-2-Busses ist der gesteigerte Bedarf an Bandbreite bei der Bearbeitung von Videodaten. Bei der Entwicklung wurden die besonderen Anforderungen des digitalen Bildformats CCIR-601 (siehe Abschnitt 3.3) berücksichtigt. Im einzelnen sind folgende Informationen gleichzeitig übertragbar:

- bis zu acht unabhängige Videodatenströme nach CCIR-601 (unkomprimiert, mit 8 oder 10 Bit pro Komponente, YUV 4:2:2),
- bis zu sechs unabhängige 8-Bit-Transparenzkanäle nach CCIR-601,
- bis zu acht digitale Audio-Kanäle mit 48-kHz-Samplingfrequenz und 16-Bit-Genauigkeit,
- verschiedene Steuersignale (z.B.: *I2C Philips device control bus*) sowie Referenz-Signale für die Synchronisation von Audio- und Videodatenströmen.

Die Vorteile dieses Bussystems liegen in der sehr hohen Datentransferrate von bis zu 242 MByte/s (siehe Abbildung 18) und der speziellen Ausrichtung auf synchrone Übertragung von Audio- und Videodaten. Gleichzeitig entfällt in vielen Fällen die Notwendigkeit, unkomprimierte Videodaten über den PCI-Bus zu übertragen.

Als Nachteil ist die geringe Verbreitung des Systems zu sehen. Weiterhin ist aufgrund der speziellen Steckverbinder die Flexibilität eingeschränkt. Das Konzept der einheitlichen Steckplätze, welches beim PCI-Bus verwendet wird, ist wesentlich universeller verwendbar.

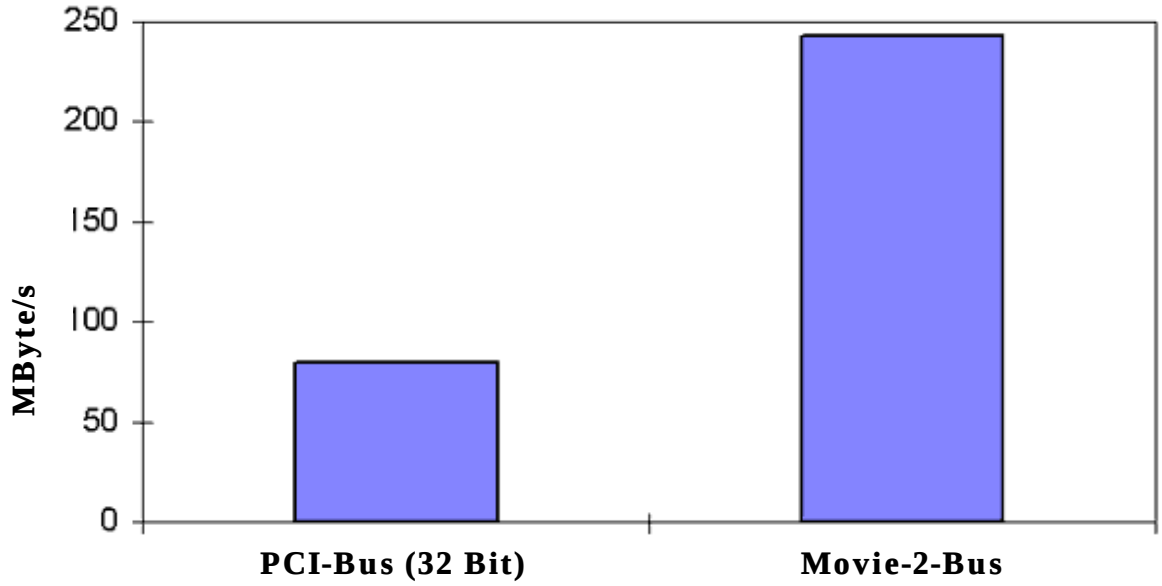


Abbildung 18: Vergleich der effektiv nutzbaren Bandbreite zwischen PCI-Bus (32 Bit) und Movie-2-Bus (übernommen aus [M2BUS])

Die Performanceuntersuchungen haben ergeben, daß im eingesetzten System die gleichzeitige Darstellung zweier Videoquellen (z.B. analoges Videosignal einer Kamera vom Eingang der DigiMix und ein von der DigiMotion kommendes Videosignal) in zwei Overlay-Fenstern keine Beeinträchtigung der Systemperformance mit sich bringt. Während der Darstellung der Videosignale ist es für den Benutzer ohne weiteres möglich, normal weiterzuarbeiten, da der Prozessor und der PCI-Bus kaum belastet werden.

Damit schafft der Movie-2-Bus eine wesentliche Voraussetzung für die Online-Videoübertragung.

5.2.2 DigiMix

Die DigiMix wird im eingesetzten System als analoger Videoeingang benutzt. Die analogen Signale werden nach der CCIR-601 digitalisiert. Sie können danach entweder direkt auf dem Adapter mit Videoeffekten bearbeitet und wieder analog ausgegeben oder über den Movie-2-Bus an einen anderen Adapter übermittelt werden.

Nach Aussagen von Matrox ist die DigiMix momentan in einem DigiSuite-System zwingend erforderlich. Die Untersuchungen haben ergeben, daß digitale Videosignale, die von der DigiMotion kommen, vor der Darstellung auf der

DigiDesktop unbedingt einen Video-Filter auf der DigiMix durchlaufen müssen, da ansonsten Farbverfälschungen auftreten.

Durch den Einsatz von DigiMix-Filtern sind eine Vielzahl von digitalen Effekten möglich, welche aber für den angestrebten Einsatz der Online-Videoübertragung nicht unmittelbar verwendbar sind.

5.2.3 DigiMotion

Die DigiMotion kombiniert drei unterschiedliche Funktionen auf einem Adapter. In Abbildung 19 ist der schematische Aufbau des Adapters dargestellt.

- Die DigiMotion stellt zwei unabhängig voneinander arbeitende MJPEG-Kodiereinheiten zur Verfügung (siehe Abbildung 19, Motion-JPEG CODEC 0 und 1). Diese können Video nach CCIR-601 in Echtzeit komprimieren und dekomprimieren. Dabei werden sowohl das DCT–basierte Verfahren als auch das verlustfreie Verfahren (siehe Abschnitt 3.5, Tabelle 3) unterstützt. Von jeder der beiden Einheiten können Datenraten bis zu 15 MByte/s (beim verlustfreien Verfahren) gleichzeitig bewältigt werden. Es ist also möglich, einen mit einer Datenrate von 15 MByte/s aufgezeichneten Datenstrom mittels einer der Einheiten zu dekomprimieren, während die zweite Einheit einen anderen Datenstrom auf 15 MByte/s komprimiert. Die unkomprimierten Daten werden über den Movie-2-Bus an- und abtransportiert. Die komprimierten Daten werden über den adapterinternen PCI-Bus an das System übergeben. Beim Kopieren der Daten traten während der Entwicklung des Testsystems jedoch Probleme auf. Diese werden im Abschnitt 7.1 näher erläutert.

- Die zweite auf dem Adapter zur Verfügung stehende Baugruppe ist für die Audioverarbeitung zuständig. Sie dient sowohl als AD- als auch als DA-Wandler. Dabei arbeitet sie mit einer 16-Bit-Auflösung und ermöglicht folgende Samplingraten: 32000 Hz, 44056 Hz, 44100 Hz, 47952 Hz und 48000 Hz. Es können bis zu vier Audiosignale gleichzeitig aufgezeichnet werden, was bei 48000 Hz ungefähr einer Datenrate von 0,37 MByte/s entspricht.

- Die dritte Baugruppe ist ein SCSI-II-Kontroller, der Festplatten mit einer Geschwindigkeit von bis zu 40 MByte/s ansprechen kann. Er ist an den internen PCI-Bus des Adapters angekoppelt und steht auch dem restlichen System transparent zur Verfügung.

Zusätzlich zu diesen drei Baugruppen verfügt der Adapter noch über eine Speicherbaugruppe, die mit bis zu 64 MByte DRAM bestückt werden kann und hauptsächlich zur Zwischenspeicherung der komprimierten Videodaten dient. So kann bei einer kurzzeitigen Überlastung des PC-Systems in vielen Fällen ein Datenverlust verhindert werden.

Während der Untersuchungen konnten die maximalen Datentransferraten (die MJPEG-Kodierer können zusammen eine Datenmenge von etwa 30 MByte/s erzeugen) nicht überprüft werden, da das zur Verfügung stehende Festplattensubsystem nur etwa 10 MByte/s verarbeiten kann. In diesem Bereich arbeitete das System jedoch zuverlässig und ohne Datenverluste. Da in diesem Fall das Testsystem am Rande seiner Leistungsfähigkeit (bezüglich der Datenaufzeichnung) betrieben wurde, war es nicht mehr möglich, zusätzlich Applikationen zu betreiben, welche Festplattenzugriffe durchführen. Wurde dies dennoch getan, so kam es zu Datenverlusten in Form von nicht aufgezeichneten Bildern. Im praktischen Einsatz ist es daher unabdingbar, zur Aufzeichnung der Videodaten eine separate Platte zu verwenden.

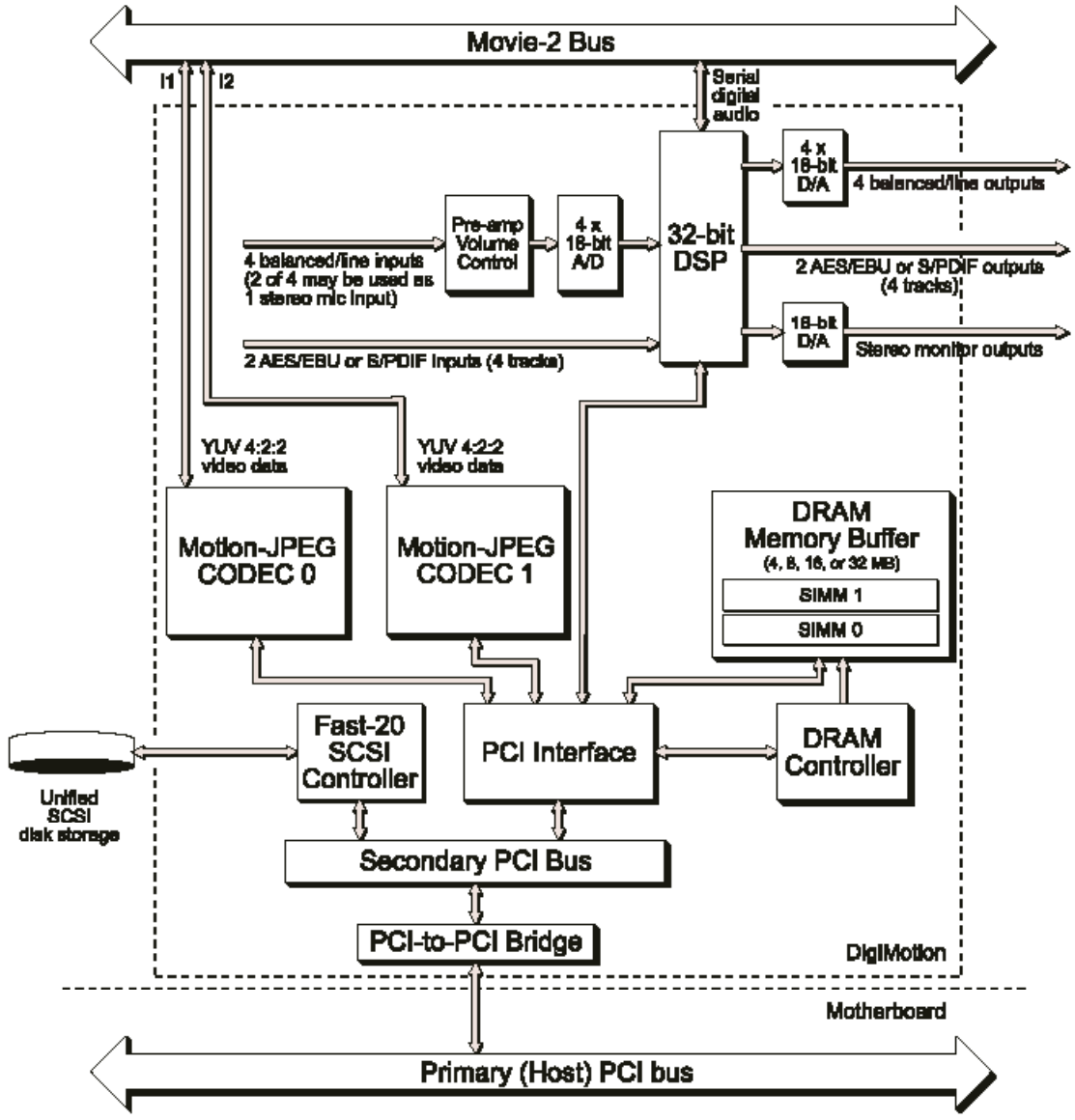


Abbildung 19: Blockschaltbild der DigiMotion (übernommen aus [DigiSDK])

5.2.4 DigiDesktop

Die DigiDesktop ist ein Grafikkarte, die über zwei unabhängige Grafikchips vom Typ MGA-2064W verfügt. Damit ist sie in der Lage, zwei Bildschirme anzusteuern und dem Benutzer somit die doppelte Darstellungsfläche zur Verfügung zu stellen. Es sind Bildschirmauflösungen von bis zu 1600x1200 Pixel bei einer Farbtiefe von 24 Bit möglich. Somit beträgt die maximale Desktopauflösung bei Verwendung von 2 Bildschirmen 3200x1200 Pixel.

Jeder der beiden Grafikchips wird von einem Videomodul unterstützt, welches es erlaubt, bis zu zwei Hardware-Overlay-Fenster mit einem Videosignal darzustellen. Um die Overlay-Fenster zu benutzen, müssen Bildschirmmodi mit einer Farbtiefe von 24 Bit verwendet werden. Dabei sind entweder je zwei Fenster mit 25 Hz (30 Hz beim NTSC-Format) oder je ein Fenster mit 50 Hz (60 Hz beim NTSC-Format) möglich. Zur synchronen Darstellung ist es notwendig, die Bildwiederholfrequenzen der Grafikmodi an die Frequenz der Videosignale anzupassen. Bei der Verwendung des PAL-Formates werden 75 Hz vom Hersteller empfohlen, beim NTSC-Format 60 Hz.

Die Videomodule erhalten die Daten entweder vom Movie-2-Bus oder von den Videodekodern, die bis zu vier analoge Videosignale digitalisieren können. Die zu verarbeitenden Videodaten müssen im CCIR-601-Format vorliegen. Es ist leider nicht möglich, die von den Videodekodern erzeugten digitalen Videodaten ins System zu übernehmen oder über den Movie-2-Bus zu übertragen. Die vier analogen Eingänge sind somit lediglich zum Überprüfen analoger Videosignale direkt auf dem Bildschirm geeignet.

Das im Test eingesetzte DigiDesktop-Modell stammt nicht aus der Serienproduktion. Während der Untersuchungen traten einige Probleme bezüglich der Videodarstellung auf. Diese wurden aber größtenteils durch neue Treiber behoben. Das verbleibende Problem besteht in einer geringfügigen Verschiebung (etwa zwei bis fünf Bildpunkte) des Videobildes nach links und einem damit auftretenden schwarzen Rand auf der rechten Seite der Videofensters. In Diskussionen mit anderen Anwendern des Systems stellte sich jedoch heraus, daß dieses Problem bei den Produkten aus der Serienherstellung nicht mehr auftritt.

Beim Verwenden der Hardware-Overlay-Fenster kam es zu keinen spürbaren Beeinträchtigungen der Systemperformance. Das einzige Problem trat auf, wenn gleichzeitig Konsolenfenster geöffnet wurden. In einem solchen Fall werden die Overlay-Fenster häufig nicht mehr korrekt dargestellt. Der Hersteller weist allerdings in seiner Dokumentation ausdrücklich auf dieses Problem hin (siehe [DigiSDK]).

6 Entwicklung des Testsystems

Zur Untersuchung der Anwendbarkeit des Video-Adapter-Systems für die Online-Videoübertragung wurden im Rahmen dieser Arbeit *DirectShow*-Filter entwickelt. In diesem Kapitel wird dargelegt, wie der Entwurf dieser Komponenten erfolgte und wie sie implementiert wurden. Weiterhin werden einige Werkzeuge beschrieben, die bei der Überprüfung der Komponenten sowie der Leistungsfähigkeit des untersuchten Systems zum Einsatz gekommen sind.

6.1 Entwicklungsumgebung

Bei der Entwicklung der Filter und des darauf aufbauenden Testtools kam Microsoft-Visual C++ 5.0 zum Einsatz. Es gab mindestens zwei zwingende Gründe dafür:

- Zu Beginn dieser Arbeit war es die einzige Entwicklungsumgebung, mit der das *DirectShow-SDK* verwendet werden konnte. Das lag vor allem daran, daß sowohl die *Header-Files* als auch die mitgelieferten Bibliotheken nicht für andere Compiler verfügbar waren.

- Der zweite Grund war das von Matrox gelieferte SDK zur Unterstützung der Videoadapter. Es war nur für Visual C++ 5.0 verfügbar und eine Entwicklerunterstützung erfolgte nur bei Einsatz dieses Systems.

Die Entwicklung der *DirectShow*-Filter erfolgte unter Verwendung des „Microsoft *DirectShow-SDK*“. Es stellt Basisklassen für die verschiedenen Filterarten zur Verfügung und ermöglicht damit die Entwicklung von Anwendungen, welche die Funktionalität von *DirectShow* verwenden.

Zum Testen der Filter im Zusammenhang mit dem untersuchten Video-Adaptersystem wurde das „*DigiSuite-SDK*“ verwendet. Hierbei handelt es sich zum einen um eine Sammlung von *DirectShow*-Filtern, welche die Softwareschnittstellen zu den Adaptern darstellen. Zum anderen wird eine Klassenbibliothek („ActiveMovie FilterGraph Class Library“) zur Verfügung gestellt, die verschiedene Funktionen der Videoadapter in vereinfachter Weise anbietet. Es war nicht möglich, eine Anwendung zu entwickeln, die ohne diese Klassenbibliothek auskommt und trotzdem eine effektive Nutzung der Videoadapter erlaubt.

Zur Übertragung der Videodaten an einen anderen Rechner wurde das *Socket*-Modul[21] verwendet, welches detailliert in [FRAN99] beschrieben ist. Da dieses Modul auch im System „Visitphone“ verwendet wird, ist hiermit die Grundlage für eine spätere Kompatibilität zu diesem System gelegt.

6.2 Entwurf der Komponenten

Beim Entwurf der Komponenten gab es mehrere wichtige Forderungen zu beachten:

- Erzeugung einer möglichst geringen Systembelastung,
- Gewährleistung einer geringen Verzögerung bei der Übertragung,
- gute Ausnutzung der Hardwareressourcen,
- angestrebte Kompatibilität mit „Visitphone“.

Zur Nutzung der in Abschnitt 5.2 vorgestellten Videoadapter muß *DirectShow* verwendet werden. Deshalb wurden *DirectShow*-Filter entwickelt, die eine Online-Videoübertragung ermöglichen.

Videodaten, die über das Netz empfangen und anschließend dargestellt oder gespeichert werden, sind in einem Filter-Graphen als *Source-Filter* dargestellt. Aus diesem Grund wurde für die Implementierung des Empfängers ein entsprechender *Source-Filter* als Datenquelle für den Filter-Graphen entwickelt. Da man das Senden der Daten in ein Netzwerk auch mit dem Speichern der Daten in einer Datei vergleichen kann, wurde zum Senden ein *Render-Filter* entwickelt.

Es wurden zwei verschiedene Ansätze zur Architektur der Videoübertragung untersucht, die nachfolgend beschrieben werden.

6.2.1 Zentrale Verwaltung der Videodaten

Die erste Variante orientiert sich an dem bereits existierenden System „Visitphone“ (siehe [NGU98]). Hierbei werden die Daten zentral vom

[21] Das *Socket-Modul* stellt die Funktionalität des WinSock-Programmierinterfaces in vereinfachter Form zur Verfügung und erlaubt es, die Transportprotokolle ATM-AAL-5 sowie auch UDP und TCP zu nutzen.

Anwendungsprogramm verwaltet und an die einzelnen Module übergeben (siehe Abbildung 20). In Abbildung 21 ist die während der Arbeit zu Testzwecken implementierte und untersuchte Architektur dargestellt. Die Videodaten werden dabei vom Anwendungsprogramm verwaltet.

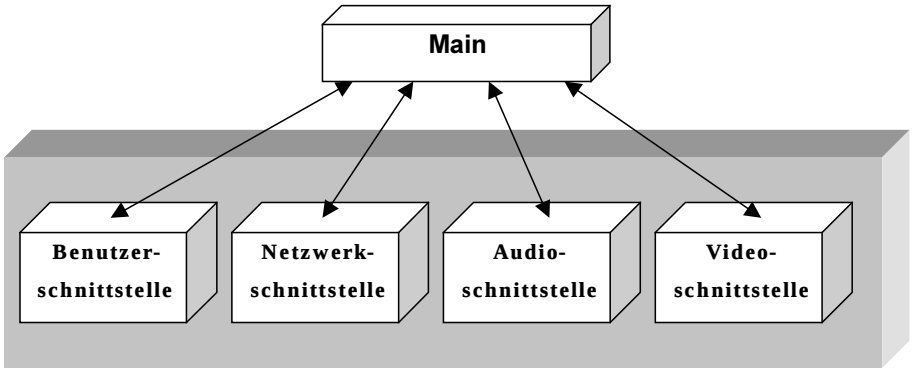


Abbildung 20: Übersicht des Programmaufbaus von „Visitphone“ (übernommen aus [NGU98])

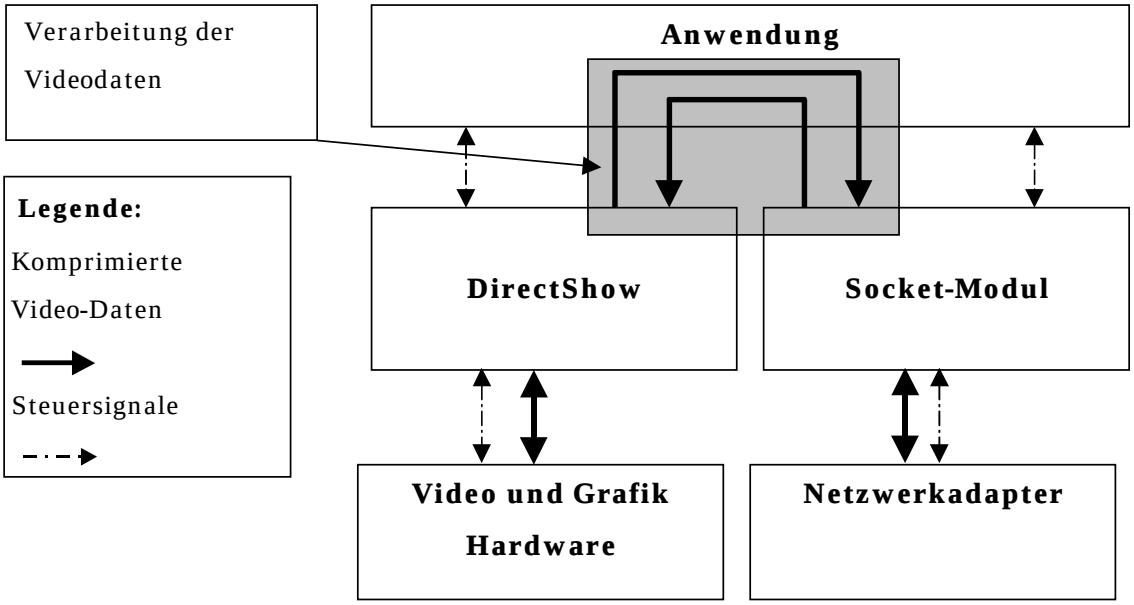


Abbildung 21: Architektur 1; Zentrale Verarbeitung der Videodaten

Die von den Kompressionskarten erzeugten Videodaten werden vom *DirectShow*-Modul an die Anwendung übergeben und von dieser an das *Socket*-Modul weitergereicht. Videodaten, die das *Socket*-Modul von anderen Sendern empfängt, werden erst an die Anwendung übermittelt und dann von dieser mittels *DirectShow* weiterverarbeitet. Die Videoübertragungsapplikation steuert somit den gesamten Datenfluß.

Der Vorteil dieser Lösung besteht in der Flexibilität, da die einzelnen Module ausgetauscht werden können. So kann das *Socket*-Modul gegen ein anderes

Netzwerkmodul ausgewechselt werden, ohne an der Anwendung etwas zu verändern. Voraussetzung dafür sind klar definierte Schnittstellen für die einzelnen Module.

Der Nachteil besteht in dem erhöhten Verwaltungsaufwand, da die Daten zwischen den einzelnen Modulen übergeben werden. Es ist ein gesteigerter Aufwand an Kommunikation innerhalb der Anwendung nötig, und die Daten müssen aufgrund der Übergabe an die Anwendung mindestens einmal mehr kopiert werden. Dies führt zu größeren Verzögerungen und zu erhöhter Systembelastung.

Während der Untersuchungen mit dieser Architektur traten vor allem Probleme mit der Nachrichtenbehandlung auf. Die Kommunikation der einzelnen Module erfolgte durch den Austausch von Nachrichten. Beim Verschieben des Anwendungsfensters wurden jedoch keine Nachrichten innerhalb der Anwendung verarbeitet. Dadurch konnten in dieser Zeit weder Daten gesendet noch empfangen werden. Es ist aber inakzeptabel, daß während der Verschiebung des Fensters durch den Anwender der Empfänger nicht mehr mit Daten versorgt werden kann. Aus den soeben geschilderten Gründen wurde die Implementierung dieser Variante nicht fortgesetzt. Es wurde statt dessen eine zweite Möglichkeit untersucht und realisiert, die im nächsten Kapitel beschrieben wird.

6.2.2 Dezentrale Verarbeitung der Daten

Zwei wesentliche Anforderungen an ein Online-Videoübertragungssystem sind geringe Verzögerungszeiten und eine geringe Systembelastung. Es wurde daher versucht, den Verwaltungsaufwand bei der Verarbeitung der Videodaten zu verringern. Der Anwendung kommt hierbei eine steuernde Funktion zu. Mit ihrer Hilfe kann der Anwender festlegen,

- welche Daten gesendet und empfangen werden,
- wer die Empfänger und die Sender sind,
- wie die Daten dargestellt werden.

Zur Realisierung dieser Variante wurden zwei *DirectShow*-Filter entwickelt:

1. Ein *Source-Filter*, welcher Videodaten selbständig mittels des *Socket*-Moduls empfangen kann und diese anschließend zur Weiterverarbeitung im Filter-Graphen zur Verfügung stellt.

2. Ein *Render-Filter*, der die komprimierten Videodaten mit Hilfe des *Socket*-Moduls versendet.

Abbildung 22 stellt die Beziehungen der Komponenten zueinander dar. Die Filter wurden mit Schnittstellen versehen, die es der Anwendung ermöglichen, Sender und Empfänger festzulegen.

Der Vorteil dieser Variante besteht im geringen Kommunikationsbedarf der Anwendung mit den Komponenten. Es ist nicht mehr nötig, daß jedes Datenpaket von der Anwendung verarbeitet wird, dies erledigen die Komponenten intern. Die Anwendung stellt lediglich eine Benutzerschnittstelle dar, welche die Komponenten mit den nötigen Informationen versorgt. Es ist möglich, mit Hilfe des im *DirectShow-SDK* enthaltenen Filter-Graph-Editors (siehe Abschnitt 4.3.3) eine Videoübertragung zu realisieren.

Ein Nachteil ist die höhere Integration von Video- und Netzwerkverarbeitung im Hinblick auf die Austauschbarkeit der einzelnen Teile. Es nicht mehr möglich, eine andere Netzwerkschnittstelle zu verwenden, ohne die Komponenten zu verändern. Dies ist jedoch bei der zentralen Verarbeitung der Daten (siehe Abschnitt 6.2.1) möglich.

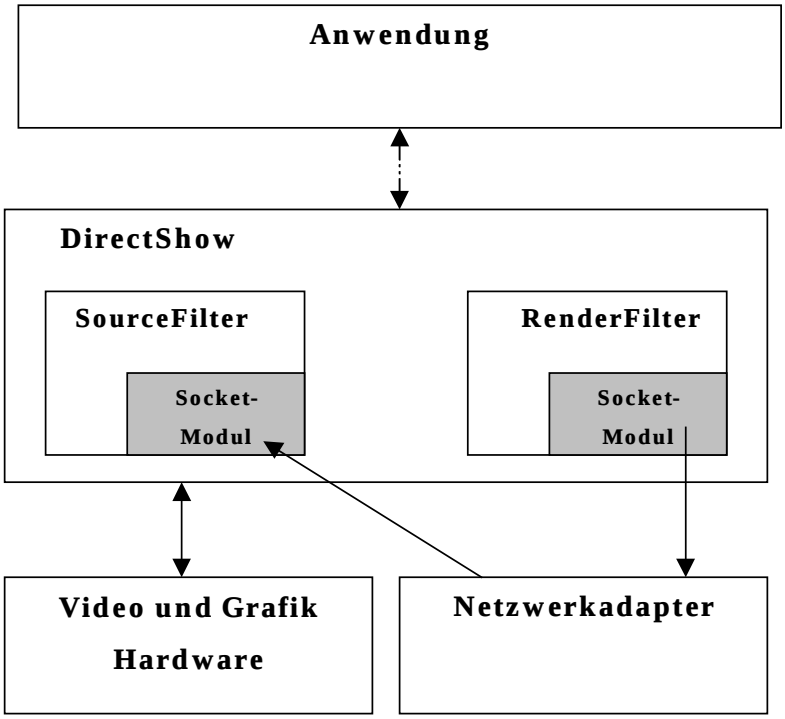


Abbildung 22: Architektur 2; Dezentrale Verarbeitung der Videodaten

6.3 Implementierung der Komponenten

Im folgenden werden die grundlegenden Methoden beschrieben, die zum Zugriff auf die Videodaten notwendig sind. Weiterhin werden die Funktionen der zwei Komponenten beschrieben. Es wird hierbei auf die in Abschnitt 6.2.2 vorgestellte Variante der dezentralen Verarbeitung der Daten eingegangen. Die verwendeten Basisklassen sind im *DirectShow-SDK* enthalten.

6.3.1 Der Sender

Die Aufgabe des Senders ist es, die Videodaten, welche innerhalb des Filter-Graphen an ihn übergeben werden, an die oder den Empfänger zu übermitteln.

Der Sender wurde als *Render-Filter* (siehe Abschnitt 4.3.3) entwickelt. Er besitzt einen Eingang. An diesem erhält er die Daten vom Filter-Graphen und versendet diese an eine vom Anwender vorgegebene Netzwerkadresse.

Die Implementierung basiert auf der Klasse ***CBaseFilter***. Um mit Hilfe dieser abstrakten Basisklasse einen funktionsfähigen *Render-Filter* zu erstellen, muß ihr mindestens ein Eingang hinzugefügt werden. Dieser wird von der Klasse ***CRenderedInputPin*** abgeleitet. In dieser Klasse sind alle Funktionen imple-

mentiert, die zur Entgegennahme von Daten nötig sind. Bevor die Daten jedoch versendet werden, ist es notwendig zu spezifizieren, wie und wohin sie geschickt werden sollen. Das Festlegen der Empfängeradresse und weiterer Übertragungsparameter ist auf zwei Wegen möglich:

1. Es kann die Schnittstelle ***IProperty*** eingesetzt werden. Diese wurde implementiert, um Anwendungen den Zugriff auf bestimmte Parameter des Filters zu ermöglichen. Mit den in dieser Schnittstelle definierten Funktionen können sowohl die Parameter verändert als auch ihre aktuellen Werte ermittelt werden. Die wichtigste Funktion stellt

   ```
   HRESULT IProperty::put_Receiver(
       char  *Protocol,
       char  *Receiver,
       int   Param1,
       int   Param2
       );
   ```

   dar. Mit dieser Funktion wird das Netzwerkprotokoll gewählt und die Netzwerkadresse des Empfängers festgelegt. Je nach Protokoll müssen noch verschiedene Parameter spezifiziert werden. Momentan werden drei Protokolle unterstützt. Es handelt sich dabei um:

   - SVC (*Switched Virtual Connection*),
   - PVC (*Permanent Virtual Connection*) und
   - UDP (*User Data Protocol*).

   Die Unterstützung basiert auf dem *Socket*-Modul, welches im Rahmen der Arbeit [FRAN99] entwickelt wurde. Tabelle 5 gibt einen Überblick über die Bedeutung der jeweiligen Parameter. Das Protokoll wird dabei durch einen String (SVC, PVC oder UDP) spezifiziert. Der Zeiger auf diesen String sowie auf die Adresse des Empfängers wird zusammen mit den zwei Parametern an den Filter übergeben. Als Rückgabewert wird ein Parameterwert, der das Ergebnis des Verbindungsaufbaus charakterisiert, geliefert. Der Wert ist vom Typ ***HRESULT***.

2. Die zweite Möglichkeit besteht in einem Dialog (siehe Abbildung 23). Dieser kann entweder vom Entwickler innerhalb einer Anwendung aktiviert werden oder ist bei Verwendung des Filter-Graph-Editors direkt aufrufbar.

Damit ist eine interaktive Einstellung der Verbindungsparameter möglich. Diese werden mittels ***IProperty::put_Receiver(...)*** an den Filter übergeben. Anschließend wird das Ergebnis des Verbindungsaufbaus dargestellt. Der Dialog stellt also einen visuellen Zugang zur Schnittstelle ***IProperty*** dar.

Protokoll	Parameter 1	Parameter 2	Beschreibung
SVC	Selector	Bitrate	Selector = Selektorbyte der lokalen ATM-Adresse zum Binden an den Socket; Bitrate = max. Datentransferrate in Kbit
PVC	vpi	vci	Pfadnummer (vpi) und Kanalnummer (vci) des zu öffnenden PVC
UDP	Port	Adapter	Port = Portnummer zum Binden an den Sokket; Adapter = Adapter, welcher verwendet werden soll

Tabelle 5: Übersicht über Protokollparameter (übernommen aus [FRAN99])

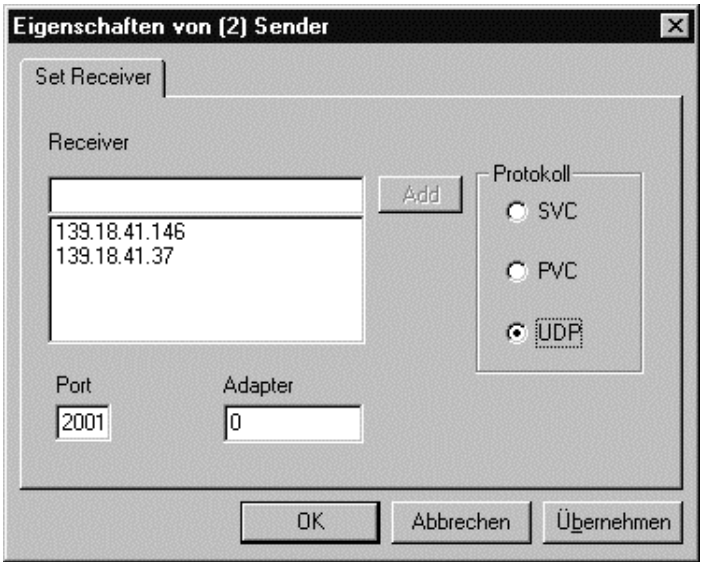


Abbildung 23: Dialog zum Einstellen der Übertragungsparameter am Sender

Nachdem eine korrekte Verbindung aufgebaut wurde, ist der Sender bereit, die Videodaten an den Empfänger zu übertragen.

Die Datenstrukturen, die zum Übertragen verwendet werden, orientieren sich dabei an den in „Visitphone“ verwendeten Spezifikationen. Dies soll vor allem ermöglichen, daß „Visitphone“ als Empfänger verwendet werden kann. Es ist dazu notwendig, zusätzlich zu den eigentlichen Bilddaten noch Informationen über den Bildaufbau zu übertragen. Die nötigen Informationen sind in der Struktur ***BITMAPINFOHEADER*** (siehe Anhang) zusammengefaßt und werden den Bildern jeweils als *Header* vorangestellt.

Da *DirectShow* auf die Verarbeitung von Datenströmen ausgelegt ist, werden neben den Informationen über die einzelnen Bilder auch Eckdaten des Datenstroms benötigt. Dabei handelt es sich um die folgenden Werte:

- Anzahl der Bilder pro Sekunde,
- Kompressionsformat,
- Angaben zum Bildformat.

Es ist nicht möglich, diese Daten zusammen mit den Einzelbildern zu übertragen, ohne die vorhandenen Datenstrukturen zu verändern. Es wurde daher ein zusätzlicher Pakettyp definiert. Dieser sollte von Anwendungen, die ihn nicht kennen, standardmäßig ignoriert werden. Es wird darin die Struktur ***AM_MEDIA_TYPE*** (siehe [MDS97]) übertragen, welche die gewünschten Informationen enthält. Diese sind notwendig, bevor auf der Empfängerseite der Filter-Graph komplett erstellt werden kann. Es ist nicht nötig, diese Struktur nach jedem Bild neu zu übertragen, da sich die Werte innerhalb eines Datenstroms nicht ändern. Damit jedoch auch beim Einsatz verbindungsloser Protokolle (z.B. UDP) jederzeit eine korrekte Darstellung der Daten auf der Empfängerseite möglich ist, werden diese Daten jeweils nach 25 Bildern erneut versendet. Somit kann ein Empfänger spätestens nach 25 empfangenen Bildern mit dem Darstellen der Daten beginnen. Das Senden der Daten findet in der Memberfunktion

```
HRESULT CRenderedInputPin::Receive(
    IMediaSample *pSample
    );
```

statt. Diese ist bereits in der Basisklasse definiert und muß je nach Anwendung entsprechend überladen werden. Sie wird immer dann aufgerufen, wenn ein neues Datensegment an den Eingang übergeben wird. Wenn es sich dabei um Videodaten handelt, wird üblicherweise ein komplettes Bild geliefert. Die für den Sender implementierte Variante von ***Receive(...)*** hat die Aufgabe, jedes ankommende Bild und aller 25 Bilder auch die Struktur ***AM_MEDIA_TYPE*** an das *Socket*-Modul zu übergeben. Die komprimierten Bilddaten sind über einen Zeiger zugänglich, der mit dem Aufruf

pSample->GetPointer(BYTE* buffer);

ermittelt wird. Die Anzahl der dort gespeicherten Bytes wird über den Aufruf von

pSample->GetActualDataLength();

geliefert. Diese Bilddaten werden zusammen mit dem ***BITMAPINFOHEADER*** und einem Byte, das die Bildwechselfrequenz des Videos definiert, in einen Puffer (***LPSTR buf***) kopiert. Die Datenstruktur zum Übertragen der Bilddaten ist in Abbildung 24 dargestellt.

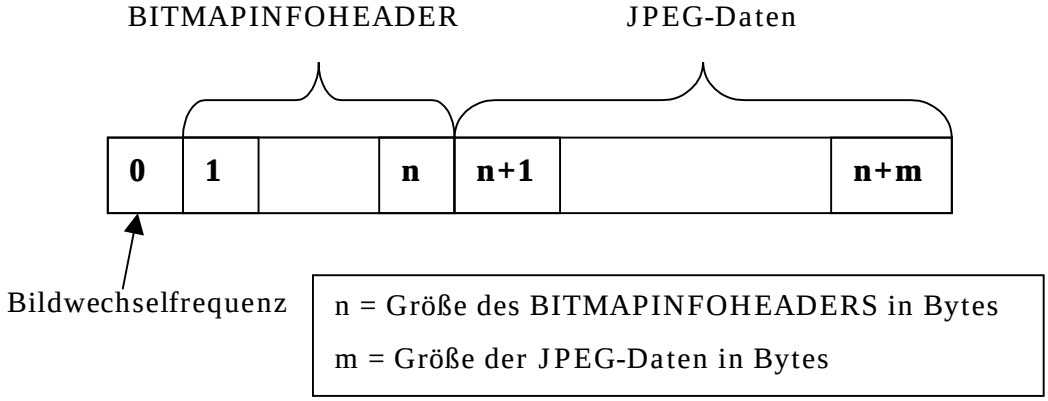


Abbildung 24: Datenstruktur zum Versenden der Bilddaten

Das Senden der Daten erfolgt mittels der Funktion

SendSeqData(LPSTR buf, DWORD len, WORD type).

Dabei stellt

- ***buf*** den Zeiger auf die Daten dar,
- ***len*** definiert das Datenvolumen (n+m+1 siehe Abbildung 24) und

- ***type*** gibt an, was für Daten versendet werden.

Diese Funktion ist im *Socket*-Modul definiert und überträgt die Daten entsprechend den Voreinstellungen.

Zur Analyse der Datenübertragung werden auf Anforderung zusätzlich Meßwerte ermittelt und ausgegeben. Die Ausgabe erfolgt entweder in eine Datei oder in eine Konsole. Zu jedem Bild werden das Datenvolumen sowie der Start und das Ende des Sendevorgangs protokolliert. Die Zeitmessung erfolgt dabei über *High-Resolution-Timer*. Die entsprechenden Funktionen greifen hierfür auf Zähler zu, die im Prozessor ständig mitlaufen. Nur so ist es möglich, exakte Zeiten, die nicht von der Belastung des Systems beeinflußt werden, zu ermitteln.

Zusätzlich zur Funktion ***Receive(...)*** werden noch einige andere Funktionen der Basisklassen überladen:

- Konstruktoren, um Initialisierungen durchzuführen;
- ***CBaseFilter::NonDelegatingQueryInterface*** gibt Informationen über die im Objekt (im Filter) enthaltenen Schnittstellen zurück;
- ***CBaseFilter::Run(),CBaseFilter::Pause()*** und ***CBaseFilter::Stop()*** wurden an die Erfordernisse der Online-Videoübertragung angepaßt;
- ***CBaseFilter::GetPin(int)*** übergibt den von der Klasse ***CRenderedInputPin*** abgeleiteten Eingang;
- ***CRenderedInputPin::CheckMediaType*** prüft die Kompatibilität der Datentypen beim Verbinden und erlaubt im Fall des Sende-Filters jeden Datentyp.

6.3.2 Der Empfänger

Die Aufgabe des Empfängers besteht in der kontinuierlichen Bereitstellung von Daten (im speziellen Fall Videodaten). Er dient in einem Filter-Graphen als Datenquelle und stellt somit einen *Source-Filter* dar.

Die Implementierung basiert auf der Klasse ***CSource***. Diese Klasse ist eine Erweiterung der Klasse ***CBaseFilter***. Mit Hilfe der Klasse ***CSourceStream*** lassen sich Ausgänge zu einem von ***CSource*** abgeleiteten Filter hinzufügen.

Ausgänge, die durch Ableitung aus ***CSourceStream*** hervorgehen, arbeiten nach dem *Push-Model*. Sie erzeugen Daten, die sie an einen angeschlossenen Filter übergeben.

Der Empfänger unterstützt die gleichen Protokolle wie der Sender (SVC, PVC, UDP). Deshalb ist es notwendig festzulegen, welches Protokoll verwendet werden soll. Dies ist wieder auf zwei Arten möglich:

1. Mit Hilfe der Schnittstelle ***INetCom***: Die Implementierung dieser Schnittstelle dient der Kommunikation einer Anwendung mit dem Filter. Neben dem Setzen wichtiger Parameter lassen sich auch Informationen über den momentanen Status einer Übertragung ermitteln. Die wesentlichste Funktion ist jedoch das Einstellen der Empfangsparameter. Dazu stellt sie die Funktion

   ```
   HRESULT set_Protokoll(
     char  *Protocol,
     int   Param1,
     int   Param2);
   ```

   zur Verfügung. Das Protokoll wird durch einen String definiert („SVC“, „PVC“ oder „UDP“), der Zeiger auf diesen wird an den Filter übergeben. Die beiden Parameter entsprechen den in Tabelle 5 gemachten Angaben. Der Rückgabewert vom Typ ***HRESULT*** gibt Aufschluß darüber, ob das Starten der Empfangsroutine gelungen ist.

2. Das Festlegen des Protokolls ist ebenfalls interaktiv über einen Dialog möglich. Dieser erlaubt die Auswahl des Protokolls und das Einstellen der eventuell zusätzlich nötigen Parameter. Der Dialog ist vom Entwickler innerhalb einer Anwendung aktivierbar, es besteht jedoch auch die Möglichkeit, ihn bei Benutzung des Filter-Graph-Editors zu verwenden. Zum Übergeben der Einstellungen an den Filter und zum Ermitteln der Parameter bereits bestehender Verbindungen dient die Schnittstelle ***INetCom***.

Nachdem die Empfangsparameter eingestellt sind, wartet der Empfänger auf das Eintreffen von Daten. Anhand des Pakettyps wird entschieden, ob es sich um Videodaten oder um die Struktur ***AM_MEDIA_TYPE*** handelt. Sobald diese Struktur erstmalig empfangen worden ist, kann die Konfiguration des Ausgangs erfolgen. Alle weiteren Pakete dieses Typs werden nicht verwendet.

Mit den nun am Ausgang vorhandenen Informationen über die Videodaten läßt sich ein Filter-Graph erstellen. Dieser könnte die Daten z.B. darstellen und/oder speichern.

Pakete vom Typ „Videodaten“ werden zwischengespeichert, damit beim Starten des Filter-Graphen sofort Daten zur Verfügung stehen. Die Speicherung erfolgt dabei in einer bestimmten Anzahl von Puffern, standardmäßig beträgt deren Anzahl drei. Die Verwaltung wird durch die dafür entwickelte Klasse ***RingBuffer*** ermöglicht. In dieser Klasse sind unter anderem zwei Funktionen implementiert, die einerseits den Zeiger auf einen Schreib- und andererseits auf einen Lesepuffer zurückgeben:

RingBuffer::BufferStruct *GetWriteBuffer();
RingBuffer::BufferStruct *GetReadBuffer();.

Wenn Daten in einem Puffer gespeichert werden sollen, muß ein Schreibpuffer angefordert werden. Sollten bereits alle Puffer gefüllt sein, so wird der älteste überschrieben. In der Klasse werden Zeiger verwaltet, die dafür sorgen, daß beim Anfordern eines Lesepuffers immer der älteste noch verfügbare Puffer geliefert wird. Diese Art der Datenverwaltung hat zur Folge, daß eine zusätzliche Verzögerung während der Übertragung entsteht. Gleichzeitig können so aber in gewissem Maße Schwankungen bei der Übertragung ausgeglichen werden. Es hat sich in Versuchen gezeigt, daß das Akzeptieren einer etwas höheren Verzögerung besser ist als der visuelle Eindruck, der beim etwaigem Stocken der Übertragung auftritt. Die Größe der maximalen zusätzlichen Verzögerung hängt von der Anzahl der Puffer ab und berechnet sich nach Gleichung (8).

$$t_v = (P-1) * \frac{1}{f} \qquad \textbf{(8)}$$

wobei : P = Anzahl der Puffer
f = Bildwechselfrequenz

Sobald der Filter-Graph gestartet wird, werden die empfangenen Videodaten unter Verwendung der Memberfunktion

```
HRESULT CSourceStream::FillBuffer(
    IMediaSample *pms
    );
```

an diesen übermittelt. Diese Funktion wird in regelmäßigen Abständen automatisch aufgerufen. Die dazu notwendigen Steuerungsmechanismen sind bereits in der Basisklasse ***CSourceStream*** vorhanden. Die Frequenz der Aufrufe hängt von den Videodaten ab und ist in der Struktur ***AM_MEDIA_TYPE*** definiert. Bei jedem Aufruf von ***FillBuffer(...)*** wird ein Bild an den Filter-Graphen übergeben. Dazu wird mittels ***GetReadBuffer*** ein Zeiger auf die nächsten auszuliefernden Videodaten ermittelt. Nach erfolgreicher Beendigung dieses Vorgangs werden die Daten in den über ***IMediaSample*** ermittelten Speicherbereich kopiert. Zusätzlich wird die Anzahl der benötigten Bytes übergeben. Sollten keine aktuellen Videodaten verfügbar sein, so wird das letzte Bild wiederholt.

Um bei den Untersuchungen genaue Aussagen zu Verzögerungszeiten treffen zu können, wird nach dem Kopieren der Daten ein Zeitstempel ausgegeben. Der Zeitstempel entspricht dem im Sender verwendeten Format.

Die soeben beschriebene Memberfunktion ***FillBuffer(...)*** erfüllt die eigentliche Arbeit. Damit sie jedoch ausgeführt werden kann, sind im Vorfeld und parallel zu ihrer Ausführung noch andere Aufgaben zu erledigen. Dazu wurden die folgenden Funktionen der Basisklasse überladen bzw. zum Filter hinzugefügt:

- ***CSourceStream::DecideBufferSize*** initialisiert einen internen Speicherbereich des Filters, der für die Datenübergabe an einen angeschlossenen Filter nötig ist. Der Aufruf erfolgt beim Initialisieren des Filters. Zu diesem Zeitpunkt ist noch nicht bekannt, welches Datenvolumen die komprimierten Daten haben werden. Deshalb wird hier ein relativ großer Wert (420000 Bytes, das entspricht bei 25 Bildern in der Sekunde einer Datenrate von etwa 10 MByte/s) verwendet. Zusätzlich muß festgelegt werden, wieviel solche Speicherbereiche nötig sind. In den zum *DirectShow-SDK* mitgelieferten Beispielen wird jeweils nur ein Puffer verwendet. Es hat sich jedoch gezeigt, daß mindestens zwei Puffer nötig sind, wenn der Filter Daten an die DigiMotion übergeben soll.

- ***CSourceStream::GetMediaType*** informiert den Filter-Graphen über die von diesem Filter unterstützten Datentypen. Der Aufruf erfolgt, wenn ein Filter mit dem Ausgang verbunden werden soll. Die hierbei übergebenen Werte entsprechen denen, die in der Struktur ***AM_MEDIA_TYPE*** übertragen werden. Solange keine Daten empfangen wurden, liefert die Funktion einen leeren Typ zurück.

- ***RecvData*** stellt eine Ausnahme dar. Es handelt sich hierbei nicht um eine Memberfunktion. Ihre Aufgabe ist es, dem *Socket*-Modul einen Speicherbereich zur Verfügung zu stellen, in den das *Socket*-Modul die empfangenen Daten ablegen kann. Der Zeiger auf ***RecvData*** wird bei der Initialisierung des Empfängers an das *Socket*-Modul übergeben. Wenn ein neues Paket eintrifft, wird die Funktion aufgerufen. Sie führt eine Prüfung hinsichtlich des Datentypes durch und reagiert entsprechend. Sollte es sich um Videodaten handeln, wird ein neuer Schreibpuffer angefordert und an das *Socket*-Modul übergeben. Handelt es sich um Formatinformationen (Struktur ***AM_MEDIA_TYPE)***, ist eine Speicherung dieser nur einmal notwendig. In allen späteren Fällen kann der soeben verwendete Schreibpuffer wieder an das *Socket*-Modul übergeben werden.

6.4 Beschreibung des Testaufbaus

Zum Testen der Filter kamen zwei Tools zum Einsatz:

1. Der Filter-Graph-Editor aus dem Microsoft *DirectShow-SDK* wurde auf den Testrechnern eingesetzt, auf denen „DigiSuite“-Adapter nicht zur Verfügung standen. Mit Hilfe dieses Editors können Filter-Graphen grafisch aufgebaut und getestet werden. Eine ausführliche Dokumentation ist Bestandteil des SDK (siehe [MDS97]).

2. Beim Einsatz der „DigiSuite“-Adapter ist es notwendig, den von Matrox im Rahmen des SDK mitgelieferten Filter-Graph-Editor zu verwenden. Dieser basiert auf der *„ActiveMovie FilterGraph Class Library*“, welche Initialisierungen an der Videohardware durchführt. Die genauen Spezifikationen bzw. die Vorgehensweise für diese Initialisierungen wurden vom Hersteller nicht offengelegt. Damit ist die Verwendung dieser Klassenbibliothek zwingend notwendig. In der prinzipiellen Funktionalität unterscheidet sich dieser Editor nicht vom Filter-Graph-Editor aus dem Microsoft *DirectShow-SDK*. Es sind jedoch einige Details praktischer umgesetzt worden, so lassen sich z.B. wesentlich mehr Informationen über die Anschlüsse der verwendeten Filter ermitteln. Weiterführende Erläuterungen zur Benutzung sind der dem SDK beiliegenden Dokumentation (siehe [DigiSDK]) zu entnehmen.

Mit Hilfe der genannten Tools wurden mehrere Szenarien geschaffen. Diese dienten dem Zweck der Funktionsüberprüfung der Hardware und der entwik-

kelten Softwarekomponenten. Im folgenden werden die wichtigsten von ihnen vorgestellt und erläutert. Bei der Beschreibung der Filter-Graphen erfolgt eine Numerierung der Filter, die in der dazugehörigen Abbildung korrespondierenden Nummern werden dabei jeweils in Klammern hinter dem Filternamen angegeben.

6.4.1 Funktionsüberprüfung der Hardware

Vor dem Beginn der Entwicklungsarbeiten mußte die verwendete Hardware auf die notwendige Funktionalität hin überprüft werden. Zwei wesentliche Anforderungen sollen durch die eingesetzten Adapter erfüllt werden:

1. Es sollen zwei Videodatenströme gleichzeitig verarbeitet werden (Komprimieren und Dekomprimieren).

2. Es sollen zwei Videodatenströme gleichzeitig dargestellt werden können.

Zur Überprüfung des ersten Punktes ist es notwendig, zwei Fälle zu untersuchen. Es soll zum einen möglich sein, zwei Videodatenströme zu dekomprimieren, andererseits muß auch die Möglichkeit bestehen, einen Datenstrom zu komprimieren und gleichzeitig einen zweiten zu dekomprimieren. Unter Verwendung dieser Funktionalität ließe sich eine Videotelefonie-Anwendung realisieren.

Für den ersten Teil der Untersuchung werden zwei zuvor auf Festplatte abgespeicherte Videos gleichzeitig in zwei Fenstern dargestellt. Der dazu notwendige Filter-Graph ist in Abbildung 25 abgebildet. Die Videos werden durch die zwei *Source-Filter* (1 und 4) dargestellt. Die von den *Source-Filtern* kommenden Daten werden an die *„MJPEG Video Renderer"* (2 und 5) weitergeleitet. Diese Filter repräsentieren die zwei MJPEG-Kodiereinheiten der DigiMotion im Dekompressionsmodus. Die von den Kodierern erzeugten unkomprimierten Videodaten werden auf den Movie-2-Bus transferiert. Dieser Bus wird im Filter-Graphen durch zwei Filter repräsentiert: Das sind zum einen der *„Matrox Movie-2 Bus Control Input Filter"* (3), welcher es ermöglicht, Daten auf den Bus zu transferieren, und zum anderen der *„Matrox Movie-2 Bus Control Output Filter"* (6), der als *Source-Filter* dient und die Daten direkt vom Movie-2-Bus erhält. Über den Bus werden die Daten zur DigiMix transportiert. Im Filter-Graphen ist dies durch den *„DigiMix Movie-2 Bus Input Filter"* (7) dargestellt. Auf der DigiMix müssen die Videodaten verschiedene Verarbeitungsschritte (Filter 8-10 sowie 11 und 12) durchlaufen, bevor sie in

einem Format vorliegen, welches von der DigiDesktop korrekt dargestellt werden kann.[22] Die transformierten Daten gelangen über den Movie-2-Bus auf die DigiDesktop. Der *„DigiDesktop Movie-2 Bus Input Filter“* (13) stellt den Eingang der DigiDesktop dar. Anschließend erfolgt die Anzeige der Videodaten in den Overlay-Fenstern der Grafikkarte. Dazu wird festgelegt, welches Overlay-Fenster jeweils verwendet wird *(„DigiDesktop Video Window A“* oder *„DigiDesktop Video Window B“* (14 und 16))[23]. Den Abschluß des Filter-Graphen bilden die „Video Renderer“ (15 und 17) des *DirectShow-SDK*. Sie sorgen für die Darstellung der Fenster auf dem Bildschirm.

Mit dem beschriebenen Filter-Graphen lassen sich zwei Videodatenströme gleichzeitig dekodieren und auf dem Bildschirm in voller Auflösung (CCIR-601-Format, siehe Tabelle 1) darstellen. Damit ist ein Teil der Anforderungen des ersten Punktes erfüllt und Punkt zwei ist erfolgreich überprüft worden.

Im zweiten Schritt war der Nachweis zu erbringen, daß gleichzeitig ein Datenstrom komprimiert und ein Datenstrom dekomprimiert werden kann. Es ist dabei auch eine Darstellung der Videosignale auf dem Monitor nötig.

Der zu diesem Zweck verwendete Filter-Graph ist in Abbildung 26 dargestellt. Er unterscheidet sich nur in einigen Teilen von dem vorhergehenden Graphen. Der wesentliche Unterschied besteht darin, daß nur ein Videodatenstrom aus einer Datei gelesen wird (diese stellt die komprimierten Daten zur Verfügung). Das Lesen der Daten aus einer Datei, das Dekomprimieren und das Darstellen erfolgen analog dem vorhergehenden Graphen durch die Filter 12, 13, 8, 9, 2, 10, 11, 8, 16, 19 und 20. Der zweite Datenstrom wird von einer analogen Videoquelle geliefert. Im hier beschriebenen Versuch handelte es sich um eine an den Videoeingang der DigiMix angeschlossene Videokamera. Mit Hilfe des *„DigiMix Input Select Filter“* (1) wird der Videoeingang selektiert. Der Filter verfügt zu diesem Zweck über eine Schnittstelle, die sich innerhalb des Editors mittels der rechten Maustaste aktivieren läßt. Am Ausgang des Filters stehen die digitalisierten Videodaten zur Verfügung. Diese müssen noch einige Verarbeitungsschritte (Filter 2 bis 4) durchlaufen, bevor sie im richtigen Format vorliegen. Da das Komprimieren und das Darstellen der Daten von zwei

---

[22] Die genaue Reihenfolge der Verarbeitungsschritte ist Beispielen aus dem *„DigiSuite-SDK“* entnommen. Die meisten der verwendeten Filter dienen eigentlich dem Zweck des Videomischens. Die entsprechende Funktionalität wird aber in diesem Fall nicht benötigt.

[23] Mit A und B wird der Grafikchip festgelegt, auf der das Overlay-Fenster erzeugt wird. Die dabei zu beachtenden Einschränkungen sind in Abschnitt 5.2.4 beschrieben.

unterschiedlichen Adaptern vorgenommen wird, ist es notwendig, den Datenstrom an beide zu übermitteln. Zu diesem Zweck wird der *„Matrox Hardware Connection Tee Filter“* (5) verwendet. Er stellt die an seinem Eingang anliegenden Daten mehrfach zur Verfügung. Bei jeder Belegung eines freien Ausgangs wird ein neuer erzeugt. Die Videodaten werden anschließend über den Movie-2-Bus (Filter 8 und 9) an die DigiMotion und an die DigiDesktop übermittelt.

Der Filter *„DigiMotion M-JPEG Video Capture 0“* (14) stellt eine der beiden MJPEG-Kodiereinheiten der DigiMotion im Kompressionsmodus dar. Am Ausgang liegen die Daten komprimiert vor. Das Einstellen der Kompressionsrate erfolgt über eine Schnittstelle des Filters. Der dafür zuständige Dialog ist mittels der rechten Maustaste erreichbar.[24] Die komprimierten Videodaten werden durch den *„Matrox RIFF Sink Filter“* (15) in einer Datei gespeichert. Die Angabe des Dateinamens erfolgt ebenfalls über eine Schnittstelle, die im Editor mittels der Maus zu aktivieren ist.

Auf der DigiDesktop erfolgt die Darstellung mit Hilfe der gleichen Schritte wie im zuvor beschriebenen Graphen.

Der Test des beschriebenen Graphen verlief erfolgreich. Beide Videodatenströme wurden korrekt auf dem Bildschirm dargestellt. Bei den durch die Kamera gelieferten Bildern waren im Rahmen der menschlichen Wahrnehmungsfähigkeit keine Verzögerungen gegenüber den aufgenommen Bewegungen festzustellen. Der in einer Datei gespeicherte Datenstrom konnte in weiteren Versuchen als Datenquelle verwendet werden. Durch diesen Test wurde nachgewiesen, daß die getestete Hardware in der Lage ist, sowohl zwei Videodatenströme gleichzeitig zu komprimieren und zu dekomprimieren als auch zwei Datenströme gleichzeitig auf dem PC-Monitor darzustellen. Sie erfüllt damit die an sie gestellten Anforderungen.

[24] Dabei ist zu beachten, daß diese Option nur dann besteht, wenn der Filter weder am Eingang noch am Ausgang mit dem Graphen verbunden ist.

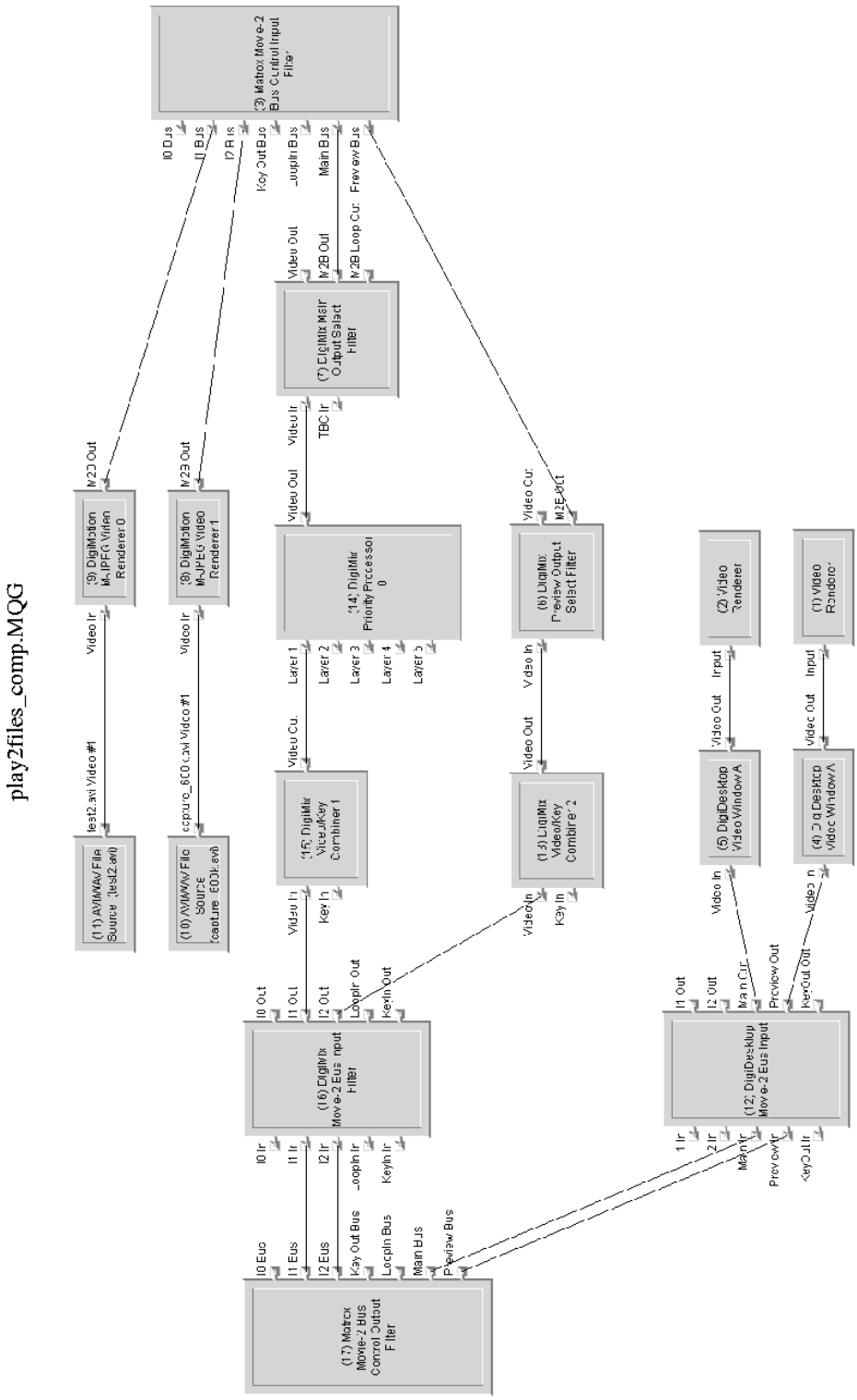


Abbildung 25: Filter-Graph zum gleichzeitigen Abspielen von zwei Videodateien

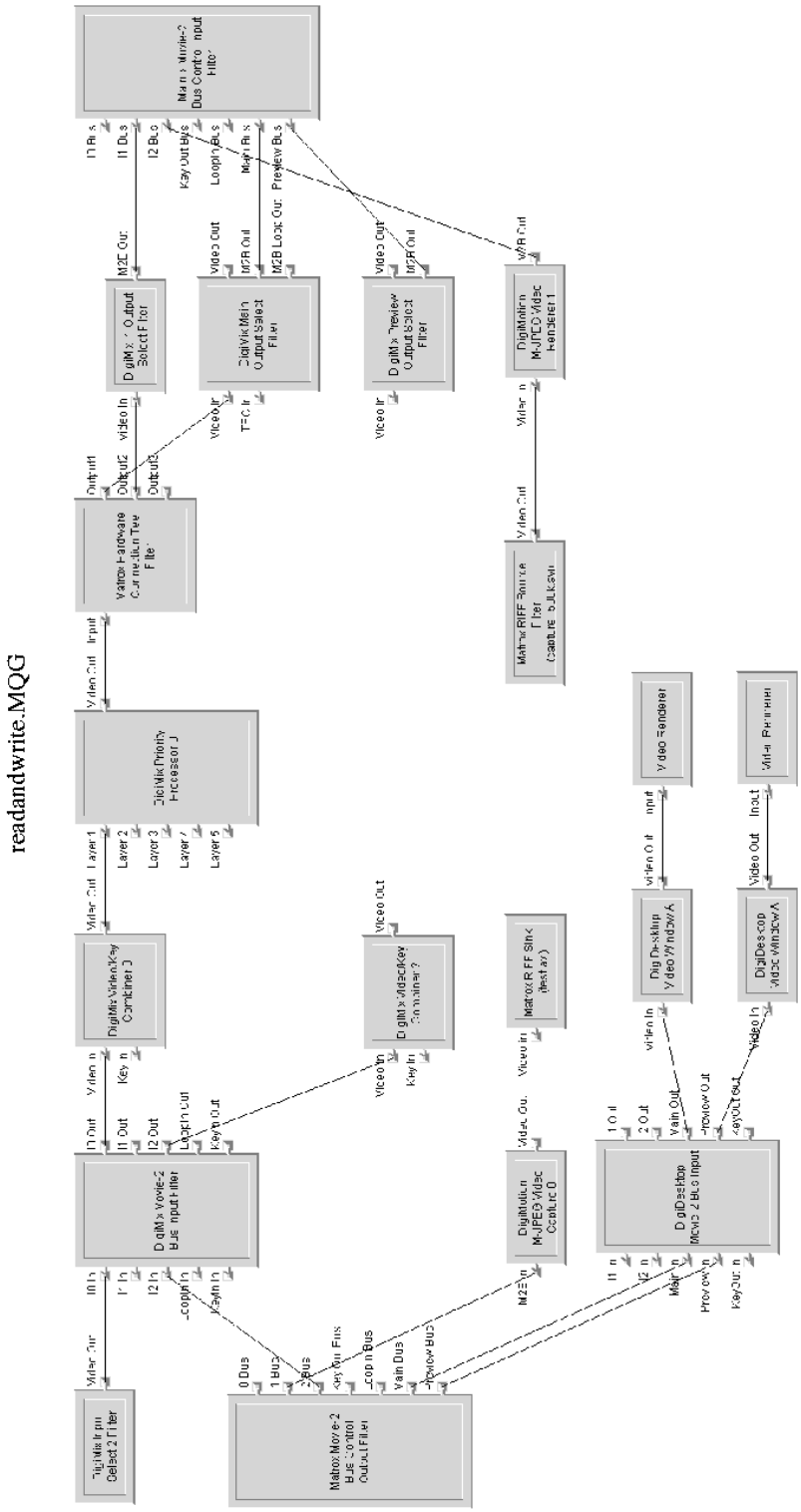


Abbildung 26: Filter-Graph zum gleichzeitigen Aufnehmen und Abspielen

6.4.2 Funktionsüberprüfung der entwickelten Komponenten

Nach dem Nachweis der Verwendbarkeit der Hardware wurden die Komponenten zur Videoübertragung entwickelt (siehe Abschnitt 6.2 und 6.3). Während der Implementierung war es notwendig, Versuche durchzuführen, um die Verwendbarkeit der Konzepte zu überprüfen. Im folgenden werden die dazu verwendeten Graphen erläutert.

Die Komplexität der in Abschnitt 6.4.1 beschriebenen Graphen ist relativ hoch, dementsprechend fehleranfällig gestaltet sich die Arbeit mit ihnen. Aus diesem Grund wurde am Anfang der Entwicklung auf den Einsatz der Videohardware verzichtet. Die Komponenten wurden dabei folgendermaßen untersucht:

1. Die Funktionsüberprüfung des Senders erfolgte mit Hilfe des in Abbildung 27 dargestellten Graphen. Zum Erzielen reproduzierbarer Ergebnisse wurden Videodaten aus einer Datei versendet. Es konnten damit immer dieselben Daten verwendet werden. Parallel zum Senden wurden die Videodaten dargestellt. Bei der Darstellung der Daten wird von *DirectShow* auf die korrekte Geschwindigkeit (Bildfrequenz) geachtet, die im Videodatenstrom vermerkt ist. Dadurch erfolgt automatisch auch das Absenden der Daten in der richtigen Geschwindigkeit.

   Der *„AVI/WAV File Source Filter“* (1) liest die Daten aus einer Videodatei. Anschließend erfolgt das Duplizieren des Datenstroms mit Hilfe des *„Infinite-Pin Tee Filter“*[25] (2). Eine der beiden Kopien wird nun mit Hilfe des Filters *„AVI Decompressor“* (3) dekomprimiert und anschließend durch den *„Video Renderer“* (4) auf dem Bildschirm dargestellt, die andere Kopie wird direkt an den Sender (5) übergeben. Dieser verschickt die Daten an den Empfänger, sobald ein solcher festgelegt wurde (siehe Abschnitt 6.3.1, Abbildung 23).

[25] Dieser Filter ist nicht in der Laufzeitversion von *DirectShow* enthalten. Er wird jedoch mit dem *DirectShow-SDK* mitgeliefert.

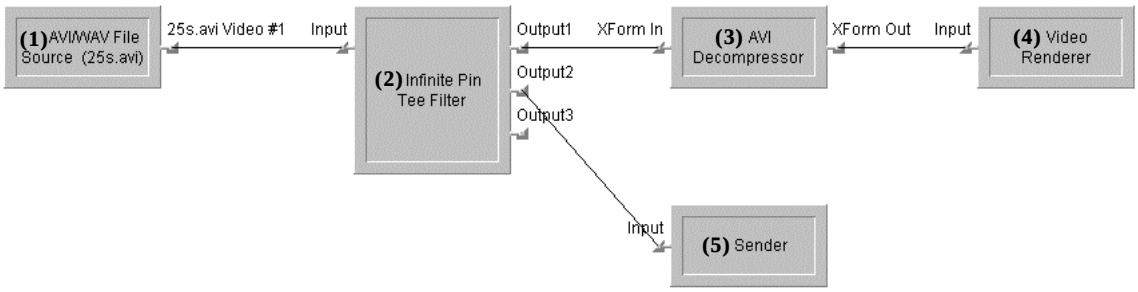


Abbildung 27: Vereinfachter Filter-Graph zum Testen des Senders

2. Auf der Empfängerseite werden die Videodaten sowohl dargestellt als auch gespeichert. Dies ermöglicht einen Vergleich der versandten mit den empfangenen Daten. In Abbildung 28 ist der Filter-Graph dargestellt, der diese Funktionalität zur Verfügung stellt.

   Die Daten kommen direkt vom Empfänger (1). Der *„Infinite-Pin-Tee-Filter“* (2) dupliziert sie. Anschließend erfolgt die Darstellung (Filter (3) und (4)) der ersten Kopie analog der Vorgehensweise beim Sender. Die zweite Kopie wird mittels der zwei Filter „AVI MUX“[26] (5) und „File Writer“[27] (6) in einer Datei zu Testzwecken gespeichert.

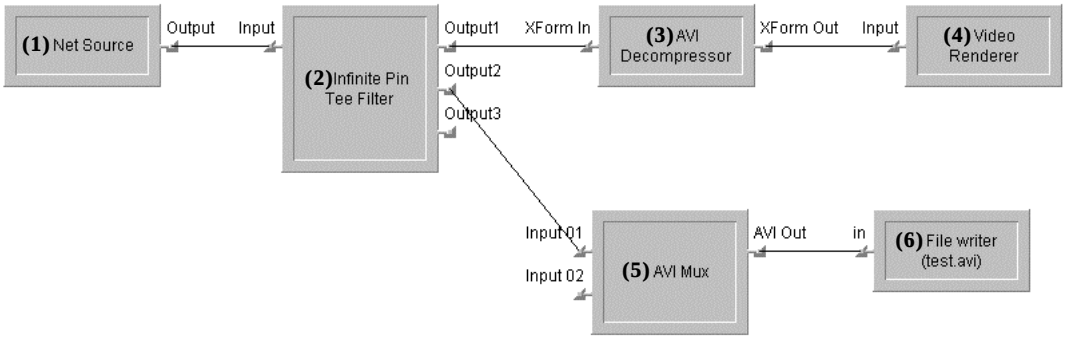


Abbildung 28: Vereinfachter Filter-Graph zum Testen des Empfängers

Unter Verwendung der beiden beschriebenen Filter-Graphen (Abbildung 27 und Abbildung 28) wurde die Funktionalität der Komponenten erfolgreich überprüft. Am Ende der Entwicklung konnten die auf der Empfängerseite gespeicherten Daten bei einem Vergleich nicht mehr von den Originaldaten unterschieden werden.

---

[26] Dieser Filter fügt die einzelnen Bilddaten so zusammen, daß sie dem AVI-Format entsprechen.

[27] Dieser Filter schreibt die ankommenden Daten in eine zuvor festgelegte Datei.

Als abschließender Schritt der Überprüfung mußte der Nachweis erbracht werden, daß die Komponenten in Zusammenarbeit mit den untersuchten Videoadaptern korrekt funktionieren. Der dabei entwickelte Graph wurde später auch bei den in Abschnitt 7 durchgeführten Leistungstests verwendet.

Die Basis für den hierzu benutzten Filter-Graphen ist der in Abbildung 26 dargestellte und im Abschnitt 6.4.1 erläuterte Graph. Die Änderungen bestehen darin, daß der *„Matrox RIFF Sink Filter“* entfernt und durch den in Abschnitt 6.3.1 vorgestellten Sender (15) ersetzt wird. Abbildung 29 zeigt den derart geänderten Graphen. Der in Abschnitt 6.3.2 beschriebene Empfänger (12) ersetzt den *Source-Filter*, welcher die Daten aus einer Datei zur Verfügung stellte. Zu beachten ist dabei, daß die Verbindung zwischen dem Empfänger und dem *„MJPEG Video Renderer”* (13) erst dann hergestellt werden kann, wenn die ersten Daten empfangen worden sind. Es ist also notwendig, zuerst die Adresse des Empfängers am Sender anzugeben und anschließend den Graphen zu starten. Danach kann der Graph wieder gestoppt und der Empfänger mit dem *„MJPEG Video Renderer”* (13) verbunden werden. Nach dem erneuten Start des Graphen erfolgt nun eine Übertragung des am Videoeingang der DigiMix anliegenden Videosignals. Beide Videosignale, sowohl das lokale als auch das über das Netzwerk empfangene, werden auf dem Bildschirm dargestellt. Die Übertragung erfolgt dabei über das Netzwerk, wobei sich der Empfänger auf demselben Rechner wie der Sender befindet. Grund hierfür ist die Tatsache, daß lediglich ein Testsystem mit den zu untersuchenden Adaptern zur Verfügung stand. Gleichzeitig konnte auch die volle Funktionalität bei der Verarbeitung von zwei Datenströmen überprüft werden.

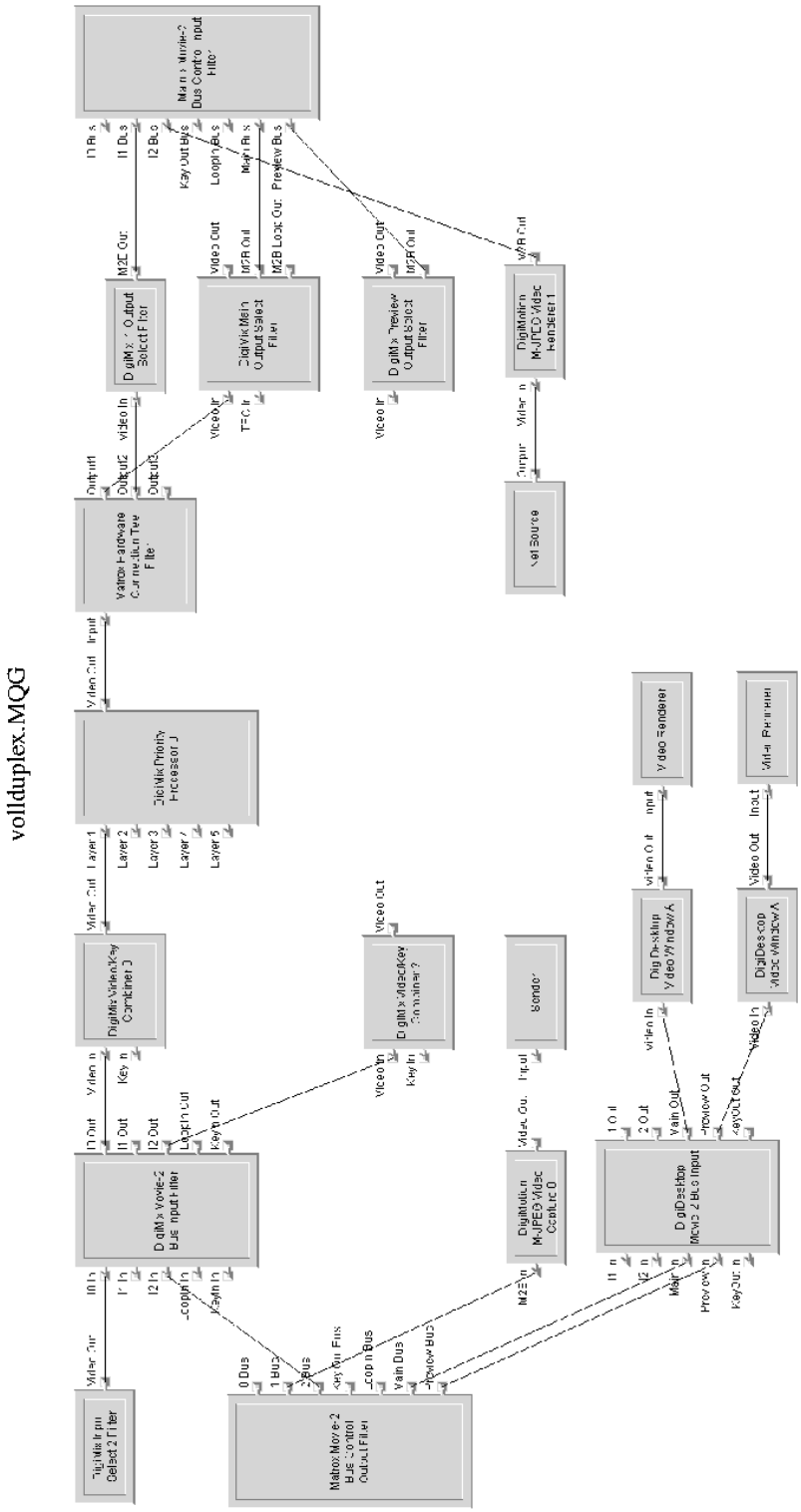


Abbildung 29: Filter-Graph zum Testen im Vollduplexbetrieb

7 Leistungstests

Zur Überprüfung der im Laufe der Arbeit gewonnenen Erkenntnisse und zum Analysieren wesentlicher Leistungsmerkmale der zu untersuchenden Videoadapter wurden mehrere Experimente durchgeführt. Dabei kam vor allem das in Abschnitt 6 vorgestellte Testsystem zum Einsatz. Folgende Gesichtspunkte waren zu untersuchen und werden in den folgenden Abschnitten ausführlich beschrieben:

- Systembelastung während einer Videoübertragung unter Verwendung der „DigiSuite“;
- Untersuchungen zur Übertragungsverzögerung;
- Vergleich der Kompressions- und Dekompressionsqualität der DigiMotion mit anderen MJPEG-Systemen.

7.1 Untersuchung der Systembelastung

Im Abschnitt 4.2 wurden die Voraussetzungen zur Verarbeitung von Videodaten im PC dargelegt. Beim experimentellen Videokonferenzsystem „Visitphone“ kommen zwei MJPEG-Adapter zum Einsatz. Genauere Informationen zur Auswahl dieser Adapter sind der Arbeit [NGU98] zu entnehmen. Die dort verwendeten Adapter übertragen sowohl die komprimierten als auch die unkomprimierten Videodaten über den PCI-Bus. Die dabei entstehenden Datenströme sind in Abbildung 13 dargestellt. Wie im Abschnitt 4.1 erläutert, werden dabei, im Vergleich zur effektiven Bandbreite des PCI-Bus, relativ große Datenmengen transportiert.

Im untersuchten Testsystem werden nur die komprimierten Videodaten mittels PCI-Bus ausgetauscht. Die zur Darstellung auf dem Bildschirm notwendigen unkomprimierten Bildinformationen gelangen über den Movie-2-Bus auf den Grafikadapter. Dadurch erfolgt eine erhebliche Entlastung des PCI-Bus. Im hier verwendeten Beispiel werden zwei Videodatenströme verarbeitet, d.h. gesendet bzw. empfangen und dargestellt. Mit der Maßgabe, daß die unkomprimierten Videodaten an den Grafikadapter im YUV 4:2:2 Format übergeben werden, tritt eine Entlastung des PCI-Bus von etwa 40 MByte/s ein. Dies entspricht rund 50% der real nutzbaren Bandbreite. Abbildung 30 stellt die Datenströme beim Einsatz des Movie-2-Bus dar. Es ist zu erkennen, daß nur die komprimierten Videodaten über den PCI-Bus transferiert werden.

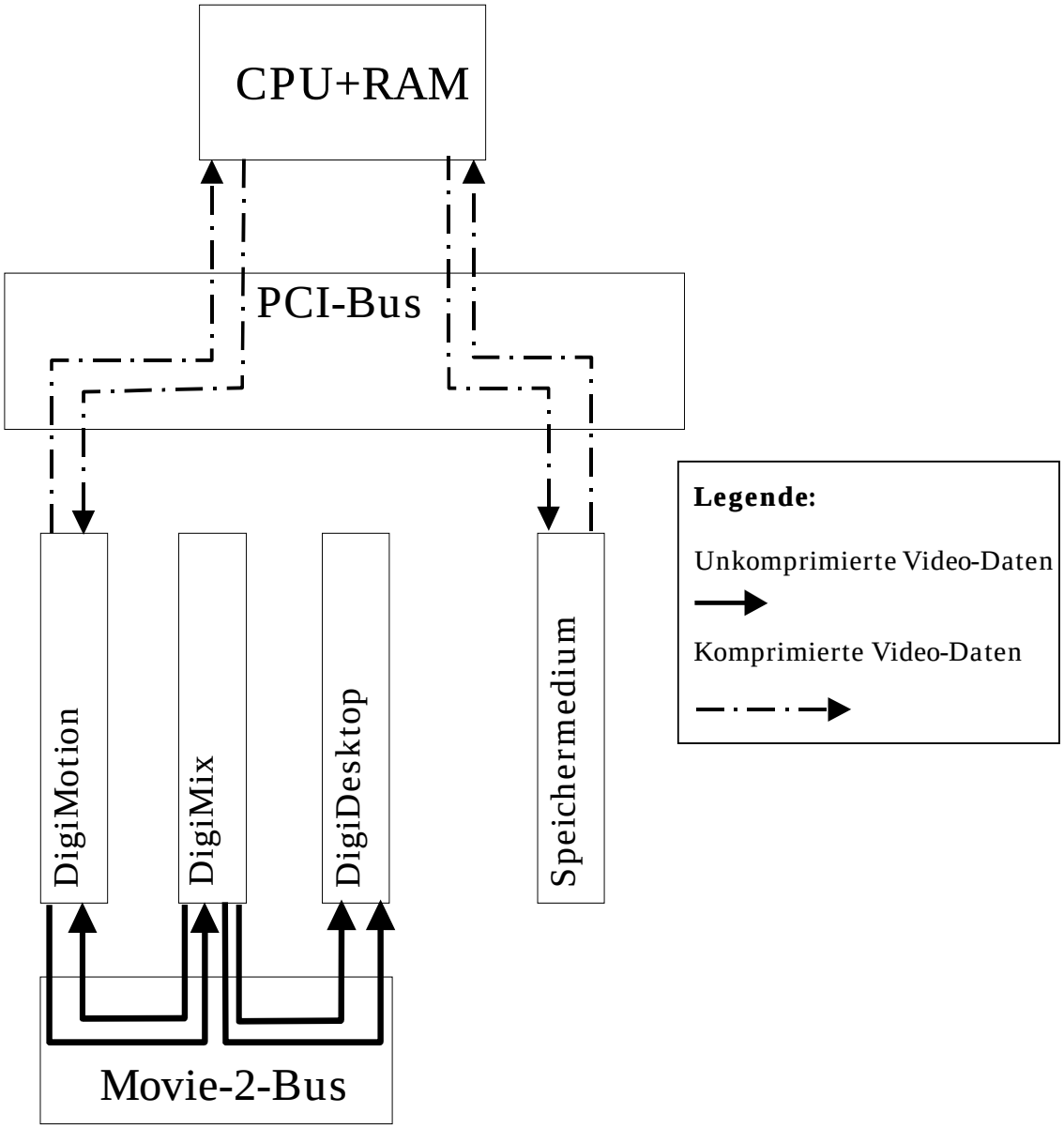


Abbildung 30: Darstellung der Datenströme beim Einsatz der "DigiSuite" zum Verarbeiten zweier Videodatenströme

Innerhalb der vorliegenden Arbeit konnten diese Werte nicht experimentell überprüft werden, da es keine Möglichkeit gibt, mittels Software Informationen über den Auslastungszustand des PCI-Bus abzufragen. Eine theoretische Möglichkeit bestände in Vergleichsmessungen, bei denen versucht wird, die gesamte verfügbare Bandbreite zu belegen. So könnte indirekt die während einer Videoübertragung auftretende Last ermittelt werden. Aufgrund der ständig stattfindenden Prozeßwechsel ist es jedoch nicht möglich den PCI-Bus kontinuierlich zu belasten.

Die einzige praktikable Möglichkeit zur Ermittlung der PCI-Bus-Belastung besteht im Einsatz zusätzlicher Hardware, dabei handelt es sich um PCI-Adapter, die Statistiken über die Aktivitäten auf dem Bus erstellen. Solche Geräte werden von verschiedenen Firmen auf dem Markt angeboten. Derartige Systeme sind im benutzten Rechnersystem nicht verwendbar, da in diesem bereits alle PCI-Steckplätze belegt sind.

Im Gegensatz zur Busbelastung läßt sich die Prozessorbelastung während einer Videoübertragung exakt ermitteln. Mit Hilfe des „Windows NT Task-Manger“ ist es möglich, jederzeit die CPU-Auslastung zu überwachen. Während der im Abschnitt 7.2 durchgeführten Messungen zu Übertragungsverzögerungen wurde die CPU-Belastung gemessen. In Abbildung 31 ist die CPU-Auslastung in Abhängigkeit von der Datenrate dargestellt. Die statistischen Fehlergrenzen liegen bei zirka 10% des jeweiligen Meßwertes und sind deshalb nicht extra angegeben. Es ist zu beachten, daß jeweils ein Datenstrom mit dieser Datenrate gesendet und einer empfangen wird und gleichzeitig die Darstellung beider Datenströme auf dem PC-Bildschirm erfolgt.

Der Vergleich der Meßwerte und der im Testsystem zur Verfügung stehenden Leistung führte zu einem Widerspruch: Aufgrund der theoretisch möglichen Datentransferraten des Hauptspeicher-Subsystems und des PCI-Bus dürfte bei den verwendeten Datenraten keine so starke CPU-Nutzung vorliegen. Deshalb wurde eine detaillierte Untersuchung der Abläufe vorgenommen.

Laufzeitmessungen einzelner Teile der entwickelten Softwaremodule haben folgendes ergeben: Der Kopiervorgang der komprimierten Videodaten von der DigiMotion in den Hauptspeicher wird im Vergleich zum theoretisch möglichen Wert verhältnismäßig langsam durchgeführt. Dies wirkt sich blockierend auf das System aus. Die exakten Ursachen für diese Verzögerungen konnten nicht geklärt werden. Sie sind in der Arbeitsweise der Treiber-Programme bzw. im Hardware-Aufbau begründet. Bis zum Abschluß der Arbeit wurde vom Hersteller der DigiSuite keine Lösung für das Problem zur Verfügung gestellt.

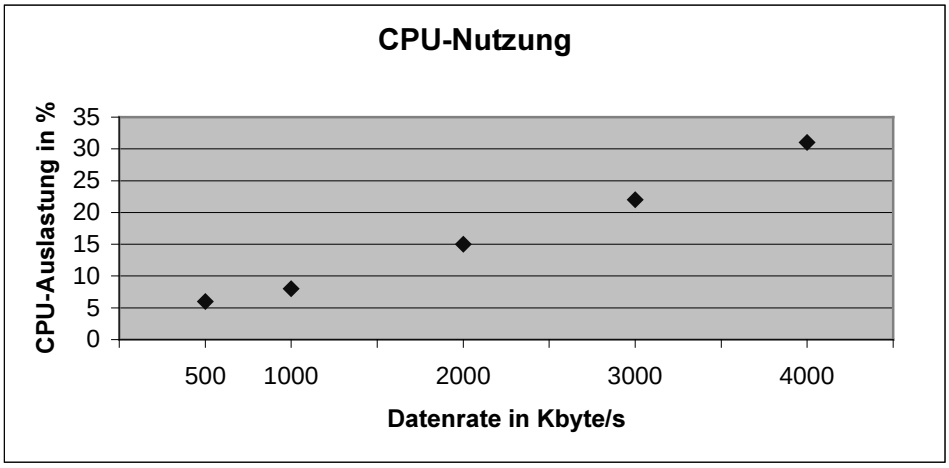


Abbildung 31: Abhängigkeit der CPU-Nutzung von der Datenrate

7.2 Untersuchung der Übertragungsverzögerung

Zur Gewährleistung einer ungestörten Kommunikation zwischen Gesprächspartnern darf die Übertragungsverzögerung gewisse Werte nicht überschreiten. Menschen reagieren besonders empfindlich auf Verzögerungen bei der Tonübertragung (siehe [STEIN94]). Da jedoch die Übertragung von Ton und Bild synchron erfolgen muß, gelten diese Werte auch für die Bildübertragung. Damit die Übertragung als synchron angesehen werden kann, sollten die Datenströme um nicht mehr als +/- 80 ms gegeneinander verschoben sein. Im Rahmen dieser Arbeit wurde die Verarbeitung von Audiodaten nicht untersucht, die Bedingungen dafür fließen jedoch in die Bewertung der Videoübertragung mit ein.

Ende-zu-Ende-Verzögerung	Auswirkung
50 ms	Verzögerung ist nicht wahrnehmbar
150 ms	Bei direktem Vergleich ist eine Verzögerung wahrnehmbar, sie wirkt sich auf die Kommunikation nicht negativ aus.
250 ms	Die Verzögerung macht sich störend bemerkbar und erfordert eine Anpassung des Kommunikationsverhaltens.
600 ms	Ab diesem Wert ist es kaum noch möglich normal zu kommunizieren.

Tabelle 6: Auswirkung von Verzögerungen auf die Kommunikation [NGU98]

Zum Feststellen der Ende-zu-Ende-Verzögerung ist es notwendig, die exakten Zeitpunkte der Bildaufnahme und der Bildwiedergabe zu ermitteln. Im Rahmen der verwendeten Software ist es aber nur möglich, Informationen darüber zu erhalten, wann die komprimierten Bilddaten vorliegen und wann sie an den Dekoder übergeben werden. Der Zeitbedarf, der für Kompression, Dekompression und Darstellung benötigt wird, ist nicht direkt meßbar. Mit Hilfe der durch Sender und Empfänger ermittelten Daten für die Zeitpunkte des Sendens und Empfangens (siehe Abschnitt 6.3) läßt sich jedoch sehr genau die Zeit feststellen, die zur Übertragung der Daten benötigt wird. Diese ist abhängig von dem verwendeten Netzwerk-Typ, dem eingesetzten Übertragungsprotokoll und der zur Verfügung stehenden Bandbreite. In Abbildung 32 sind die Verzögerungswerte für eine Übertragung mittels ATM dargestellt. Die dort aufgezeigten genauen Messungen waren möglich, da der Rechner die Daten an sich selbst gesendet hat, die Zeitbasis war demzufolge gleich. Die Daten wurden über einen ATM-Switch (IBM8265) transferiert. Als Transportprotokoll kam ATM-AAL-5 zum Einsatz, wobei eine SVC-Verbindung hergestellt wurde.

In dem Diagramm (siehe Abbildung 32) ist zu erkennen, daß die Übertragungsdauer für ein komplettes Bild mit zunehmender Datenmenge linear ansteigt. Bei einer Bildgröße von 150 Kbyte und einer Bandbreite von 20000 Kbit/s erreicht die Übertragungszeit 40 ms. 150 Kbyte pro Bild stellen bei dieser Bandbreite damit das absolute Maximum für eine Übertragung von 25

Bildern pro Sekunde dar. Dies entspricht einer Datenrate von 3750 Kbyte/s und einem Kompressionsfaktor von 5,3:1.

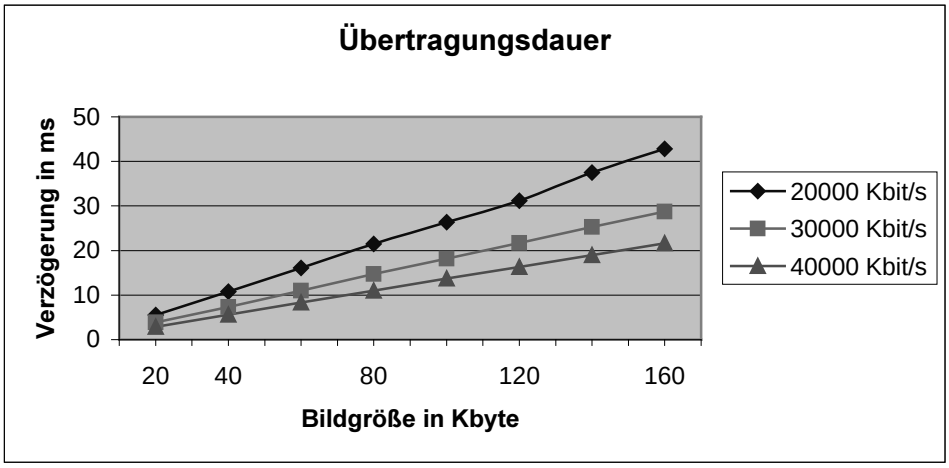


Abbildung 32: Übertragungsdauer bei Übertragung mittels SVC

Zum Ermitteln der Ende-zu-Ende-Verzögerung wurde folgendes Experiment durchgeführt:

Eine Stoppuhr mit einer Genauigkeit von 10 ms wurde von der Kamera aufgenommen und direkt auf dem Bildschirm dargestellt. Es wird dabei davon ausgegangen, daß dies nahezu ohne Verzögerung geschieht. Das Videosignal wurde komprimiert, über das Netz an denselben Rechner gesendet, dekomprimiert und ebenfalls dargestellt. Zur Realisierung dieses Verfahrens kam der im Abschnitt 6.4.2 (siehe Abbildung 29) beschriebene Filter-Graph zum Einsatz.

Während der Übertragung konnten so in regelmäßigen Abständen Bildschirmfotos aufgenommen werden. In Abbildung 33 ist eine solche Aufnahme zu sehen. Die linke der beiden Uhren ist die lokale Aufnahme, wie an dem fortgeschrittenen Zählerstand zu erkennen ist. Das Problem bei dieser Art der Messung besteht darin, daß die Zahlen nicht immer genau zu erkennen waren. Es wurden deshalb für verschiedene Datenraten jeweils 50 Messungen durchgeführt, um genügend „saubere“ Bilder zu erhalten.

Aus der Differenz der beiden Zeiten läßt sich die Ende-zu-Ende-Verzögerung errechnen. Die gemessenen Werte sind in Abbildung 34 als Diagramm dargestellt. Es sind jeweils der arithmetische Mittelwert und die Standardabweichung angegeben. Die relative große Standardabweichung erklärt sich durch die geringe Genauigkeit des Meßinstrumentes und durch die Art des

Meßvorgangs. Da die beiden Videos nicht synchron laufen und jedes Bild genau 40 ms lang angezeigt wird, ist es möglich, daß genau dann ein Bildschirmfoto aufgenommen wird, wenn ein Bild gerade eine Millisekunde und das andere bereits 39 ms lang angezeigt wird. Damit läßt sich auch der starke Anstieg der Ende-zu-Ende-Verzögerung beim Übergang der Datenrate von 2000 Kbyte/s auf 3000 Kbyte/s erklären. Die Übertragungsverzögerung nimmt zwar nur um etwa 10 ms zu, aber wenn genau in dieser Zeit ein Bildwechsel erfolgt, so wird eine höhere Verzögerung gemessen.

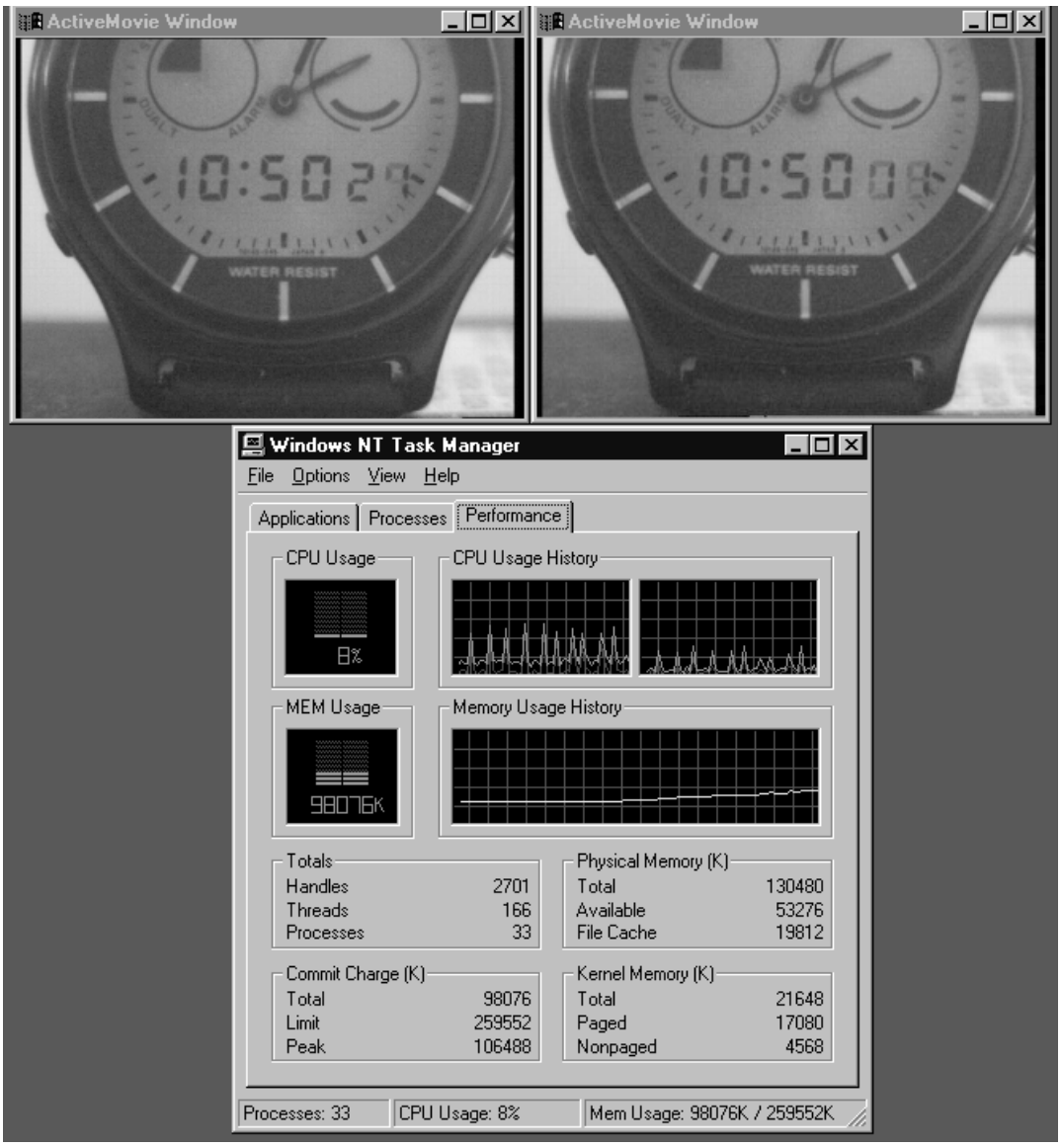


Abbildung 33: Bildschirmfoto zu den Verzögerungs- und Belastungsmessungen. Das linke der beiden oberen Fenster stellt das lokale Videobild, das rechte das über das Netzwerk übertragene Video dar. Das untere Fenster zeigt die während der Übertragung genutzte CPU-Leistung an.

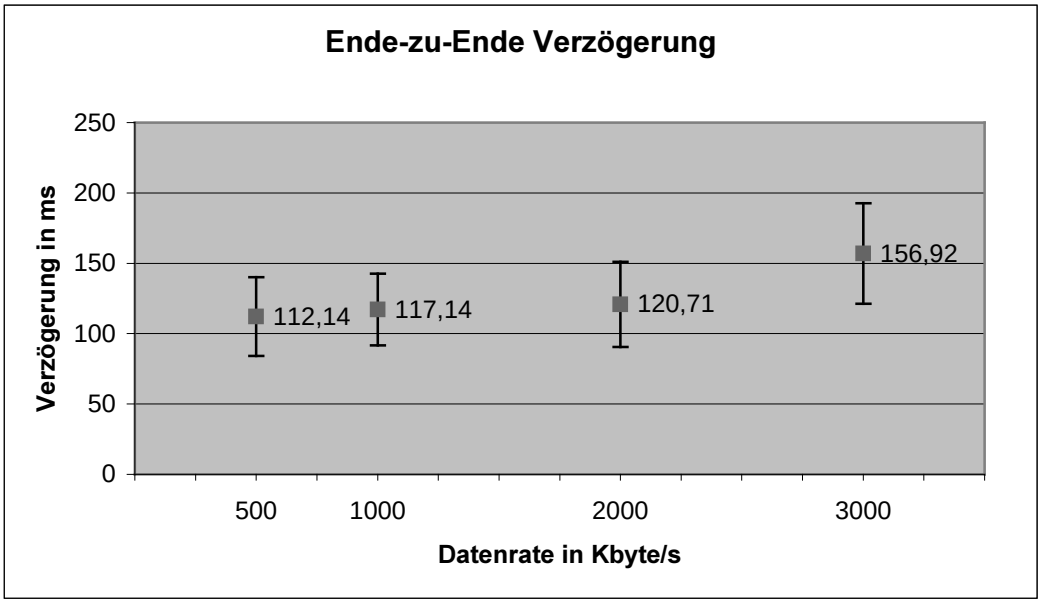


Abbildung 34: Darstellung der Ende-zu-Ende Verzögerung

7.3 Vergleich der Bildqualität

Der Vergleich der Bildqualität dient der Bewertung der MJPEG-Kodiereinheiten der DigiMotion. Dabei wurden drei Systeme miteinander verglichen (siehe Tabelle 7).

DigiMotion MJPEG-Kodiereinheit	Sie stellt das zu untersuchende System dar.
MIRO DC30	Als Vertreter der bei „Visitphone“ verwendeten Videoadapter.
Software MJPEG-Kodierer	Ein zu Testzwecken frei verfügbarer MJPEG-Kodierer, der sowohl unter Windows 95 als auch unter Windows NT einsetzbar ist.

Tabelle 7: Übersicht über die Testsysteme

Drei verschiedene Bilder wurden mit den Testsystemen komprimiert und dekomprimiert. Dieser Vorgang erfolgte für jedes der Bilder in fünf Qualitätseinstellungen (siehe Tabelle 8). Zusätzlich erfolgte die Untersuchung der

DigiMotion im verlustfreien Modus. Zur Auswahl der Bilder trug entscheidend das Ziel bei, verschiedene Einsatzbedingungen zu untersuchen. Das erste Testbild „Bugs“ (siehe Abbildung 39) ist ein von einem Programm synthetisch erzeugtes Bild, in dem keine Störungen enthalten sind. Das zweite Bild „Seal“ (siehe Abbildung 43) stellt ein Landschaftsbild dar. In diesem sind einerseits sehr viele Details (z.B. Wald) und zusätzlich aufgrund der Datenquelle (Video) bereits Störungen enthalten. Diese Gegebenheiten erschweren die Datenreduktion erheblich. Das dritte Bild „Zelda“ (siehe Abbildung 47) stellt ein Szenario dar, welches für Videotelefonie üblich ist, nämlich einen Kopf vor neutralem Hintergrund.

Die Angaben beziehen sich auf Bilder mit einer Auflösung von 720 x 576 Bildpunkten. Dies entspricht einer Datenmenge von 1215 Kbyte pro Bild bei RGB-Darstellung, bei Darstellung im YUV-System (Subsampling 4:2:2) beträgt die Datenmenge 810 Kbyte pro Bild. Die Kompressionsrate bezieht sich auf das Verhältnis zur Darstellung im YUV-System.

Datenmenge für Einzelbild komprimiert [Kbyte]	**Datenrate für Bildsequenz (25 Hz) komprimiert [Kbyte/s]**	**Kompressionsfaktor**
20	500	40:1
40	1000	20:1
80	2000	10:1
160	4000	5:1
320	8000	2,5:1

Tabelle 8: Übersicht der untersuchten Qualitätsstufen

Nach dem Erzeugen der Testbilder folgte ein Vergleich dieser mit den Originalen. Als Vergleichsmaßstab kam das Verzerrungsmaß PSNR zum Einsatz (siehe Abschnitt 3.4).

Die Ermittlung der Werte erfolgte mit dem im Rahmen der Arbeit dafür entwickelten Tool. Abbildung 35 gibt einen Eindruck von dessen Benutzeroberfläche. Die links oben angeordneten Bilder (*Image1* und *Image2*) stellen die Vergleichsobjekte dar. Nach dem Laden des ersten Bildes (*Image1*) können

verschiedene Vergleichsbilder (*Image2*) nacheinander geladen werden. Der jeweilige PSNR-Wert wird aktualisiert. Mit Hilfe der beiden Felder (X und Y) ist es möglich, Randbereiche der Bilder nicht mit in den Vergleich einzubeziehen[28]. Das auf der rechten Seite zu sehende Bild zeigt die Unterschiede zwischen dem ersten und dem zweiten Bild in Form von Graustufen[29]. Da diese Unterschiede bei geringem Kompressionsgrad sehr klein sind, kann der Kontrast mittels eines Faktors erhöht werden (*Multiplikator*).

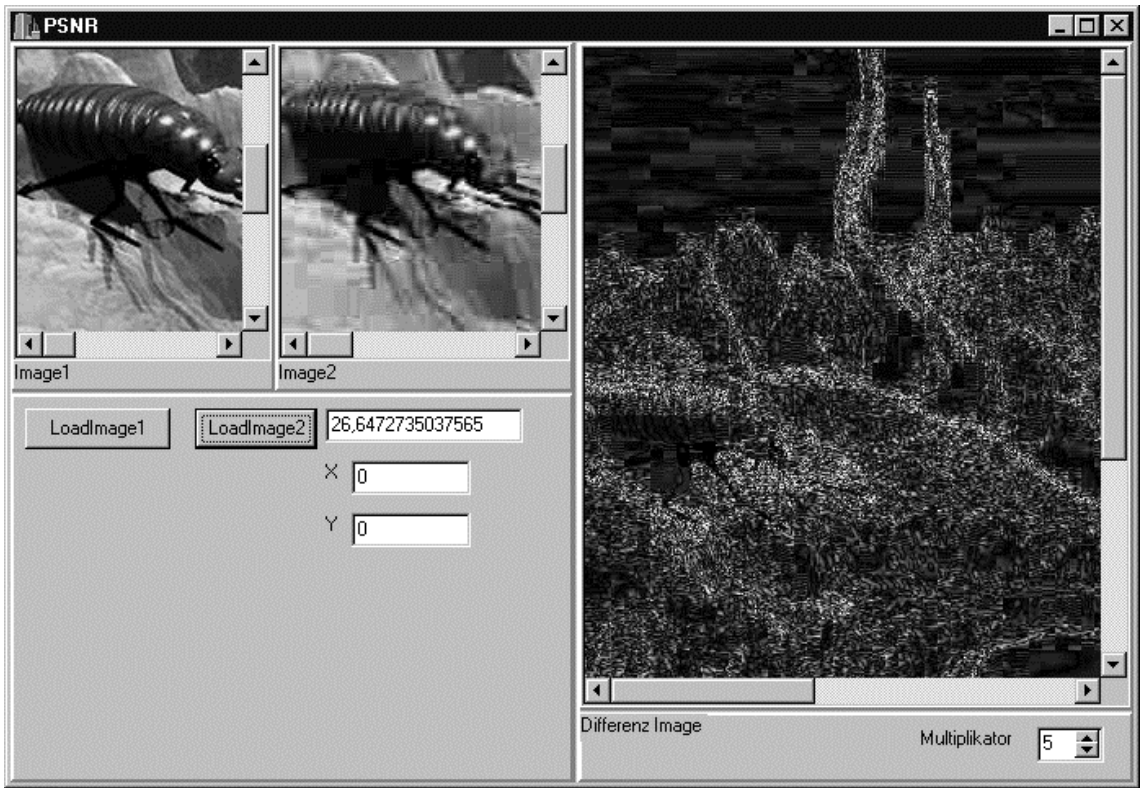


Abbildung 35: Programm zur Ermittlung der PSNR-Werte

Den in den Diagrammen dargestellten Meßwerten (siehe Abbildung 36 bis Abbildung 38) sind drei Aussagen zu entnehmen:

1. Alle Testkandidaten liegen relativ nah beieinander, das heißt, es bestehen keine wesentlichen Unterschiede in der Qualität der einzelnen Kodierer. Nur im Bereich sehr starker und sehr geringer Kompression treten bemerkbare Unterschiede auf. Dies liegt zum einen daran, daß einer der Testkandidaten (MIRO DC30) gar nicht für so geringe Kompressionsraten

[28] Die Werte werden dabei jeweils oben und unten (Y) bzw. links und rechts (X) abgezogen.

[29] Je heller ein Punkt dargestellt ist, desto stärker weicht der jeweilige Punkt im zweiten Bild von dem im ersten ab.

(hohe Datenraten) ausgelegt, zum anderen die DigiMotion speziell für geringe Kompressionsraten gedacht ist. Die Entwickler haben also die Produkte mit hoher Wahrscheinlichkeit den an sie gestellten Anforderungen angepaßt, und dies spiegelt sich in den Meßwerten wider.

2. Es zeigt sich, daß Qualität und Inhalt der Ausgangsbilder einen bedeutenden Einfluß auf das Ergebnis der Kompression und Dekompression haben. Die durchweg besten PSNR-Werte erzielt das Bild „Zelda“, da dort sehr wenig Details vorkommen. Zumindest bei hohen Datenraten macht sich die gute Ausgangsqualität des Bildes „Bugs“ bemerkbar. Das Bild „Sail“ bereitet den Kodierern die meisten Probleme, die Verfremdungen gegenüber dem Original sind hier am stärksten.

3. Die Qualitätsbewertung anhand der PSNR-Werte ist nicht unproblematisch. Beim Betrachten der Bilder wird man nicht sofort feststellen, daß sich die Qualität der verschiedenen Bilder so stark unterscheidet. Denn besonders im Bild „Sail“ sind die Unterschiede für das menschliche Auge schwer zu erkennen, da sie z.B. in den Details der Bäume untergehen. Zur Veranschaulichung der Qualitätsunterschiede sind noch einige Ausschnitte der Testbilder dargestellt. Dabei handelt es sich jeweils um das Originalbild und drei Qualitätsstufen, die Bilder wurden mit der DigiMotion verarbeitet (Abbildung 39 bis Abbildung 50).

Bei der Analyse der Testbilder zeigte sich, daß der sogenannte „verlustfreie“ Modus der DigiMotion nicht zu 100% verlustfrei arbeitet. Die Unterschiede sind für das menschliche Auge jedoch nicht sichtbar, erst im Differenzbild sind kleine Abweichungen zu erkennen. Die Ursache dafür liegt in Rundungsfehlern während der Kompression und Dekompression.

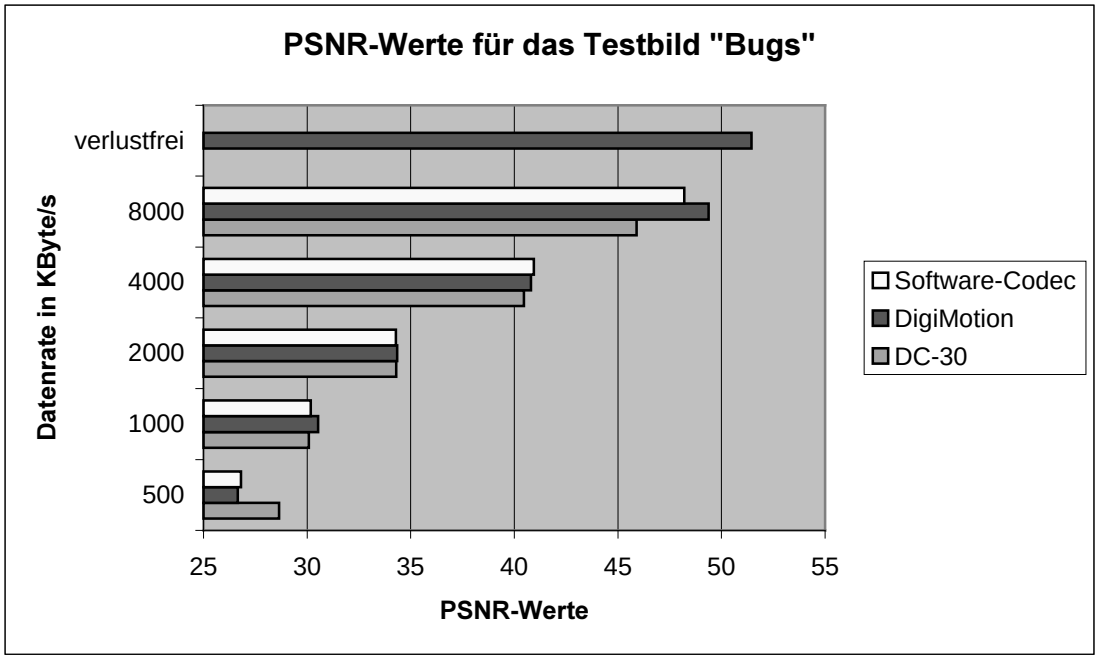


Abbildung 36: Darstellung der PSNR-Werte für das Bild „Bugs“

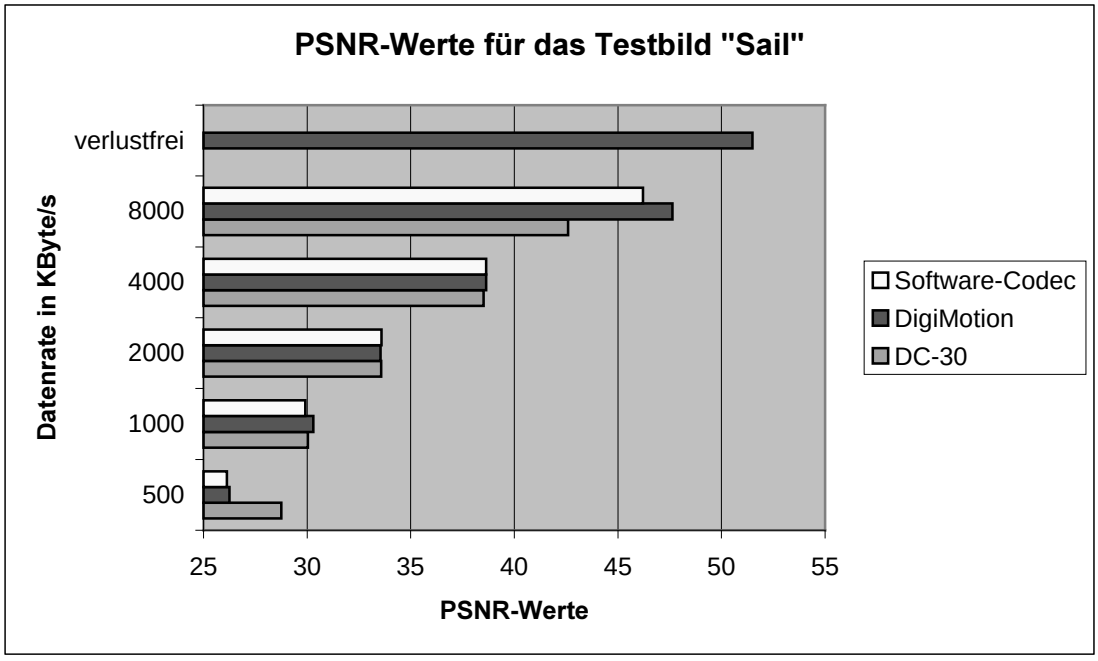


Abbildung 37: Darstellung der PSNR-Werte für das Bild „Sail“

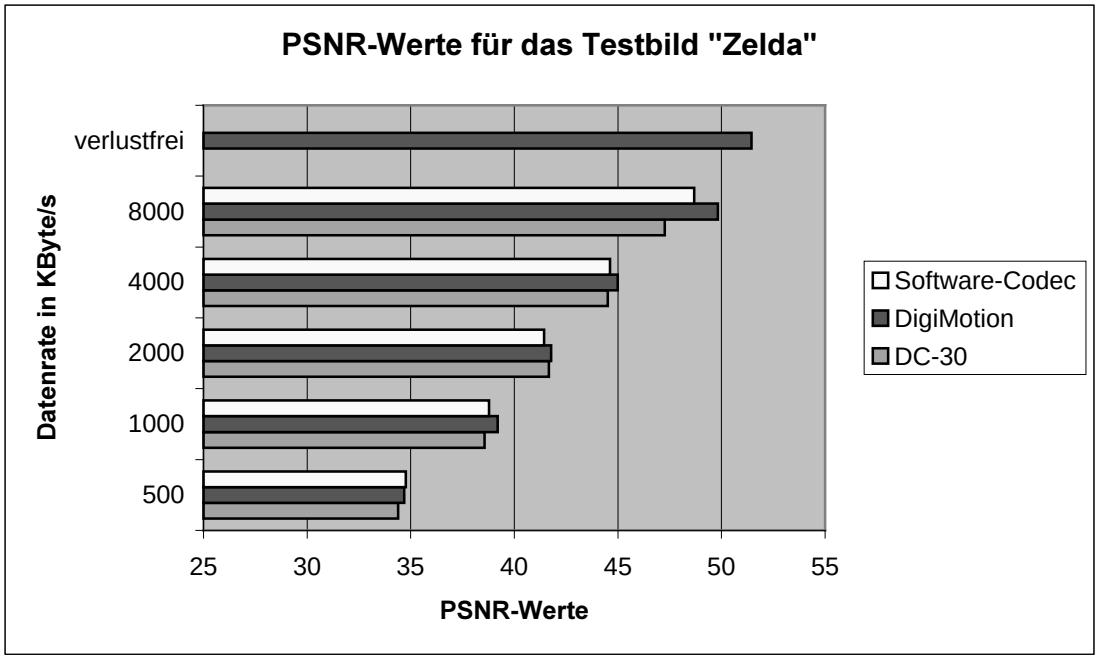


Abbildung 38: Darstellung der PSNR-Werte für das Bild „Zelda“

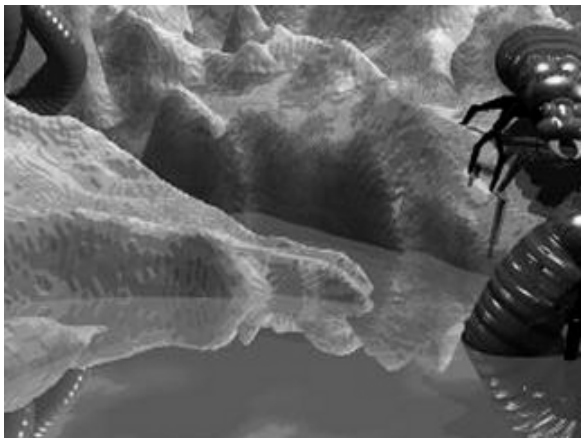


Abbildung 39: „Bugs“ Original

Abbildung 40: „Bugs“ (40:1, PSNR=26,65)

Abbildung 41: „Bugs“ (10:1, PSNR=34,34)

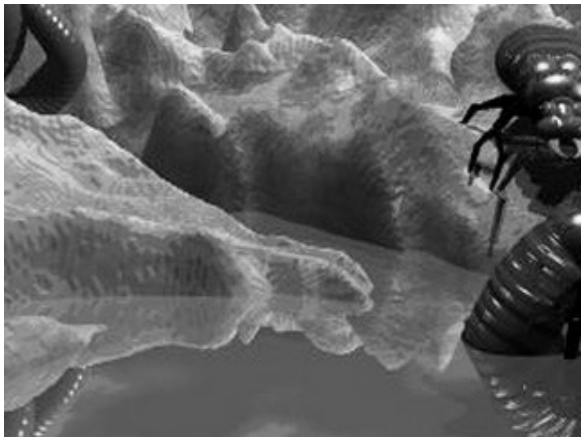


Abbildung 42: „Bugs“ (5:1, PSNR=40,80)

Abbildung 43: „Sail“ Original

Abbildung 44: „Sail“ (40:1, PSNR=26,26)

Abbildung 45: „Sail“ (10:1, PSNR=33,54)

Abbildung 46: „Sail“ (5:1, PSNR=38,64)

Abbildung 47: „Zelda“ Original

Abbildung 48: „Zelda“ (40:1, PSNR=34,68)

Abbildung 49: „Zelda“ (10:1, PSNR=41,77)

Abbildung 50: „Zelda“ (5:1, PSNR=44,99)

8 Zusammenfassung

Im Rahmen der vorliegenden Arbeit wurden Konzepte zur Online-Videoübertragung mittels *DirectShow* entworfen, die dazu notwendigen Softwaremodule entwickelt und Untersuchungen zur Leistungsfähigkeit der eingesetzten Videoadapter durchgeführt. Aufbauend auf den Entwicklungen konnte die Funktionalität der Adapter für derartige Anwendungen nachgewiesen werden.

Das System „DigiSuite“ stellt in der untersuchten Konfiguration eine gute Basis zur Übertragung von Videodaten im CCIR-601-Format dar. Unter Verwendung von Windows NT wird eine stabile Arbeitsumgebung zur Verfügung gestellt. Bei den abschließenden Leistungsuntersuchungen haben sich folgende Fakten ergeben:

- Die Prozessorbelastung durch die Übertragung und Darstellung der Videodaten im CCIR-601-Format ist bei einem auf 2000 Kbyte/s komprimierten Datenstrom gering. Bei Datenraten oberhalb von 5000 Kbyte/s stellt der Kopiervorgang der komprimierten Videodaten von der DigiMotion in den Hauptspeicher ein Hemmnis dar. Dieses Problem ist durch den Hersteller der Hardware zu beheben und konnte deshalb bis zum Abschluß dieser Arbeit nicht gelöst werden.

- Durch die Ausnutzung des „DigiSuite“-eigenen Movie-2-Bus kann der Systembus auch während der Videoverarbeitung weitestgehend voll für andere Aufgaben genutzt werden. Die unkomprimierten Videodaten werden dabei ausschließlich über der Movie-2-Bus transferiert. Dies ermöglicht insbesondere auch während einer Videokommunikation ein störungsfreies Arbeiten mit Anwendungen, die eine hohe Buslast erzeugen.

- Die Qualität der von der Kompressionshardware verarbeiteten Bilder unterscheidet sich nicht wesentlich von anderen MJPEG-Lösungen. Es besteht aber die Möglichkeit, sehr geringe Kompressionsfaktoren zu verwenden. Dadurch lassen sich Videodaten mit minimalem Qualitätsverlust verarbeiten.

Die entwickelten Softwarekomponenten erlauben einen zukünftigen Einsatz der getesteten Hardware im System „Visitphone“.

Aufbauend auf den Erkenntnissen dieser Arbeit sind weitere Entwicklungen notwendig bzw. denkbar:

- Zur Bereitstellung eines Videokommunikationssystems muß die Audioübertragung in das vorgestellte System integriert werden.

- Eine Integration in das bereits realisierte „Visitphone“-Projekt ist notwendig. Es besteht hierbei das Problem, daß der für die Entwicklung von „Visitphone“ eingesetzte Compiler keine *DirectShow*-Unterstützung anbietet.

- Weitere Video-Hardware ist auf die Verwendbarkeit mit dem vorgestellten Konzept zu überprüfen, um eine breitere und flexiblere Basis für den Einsatz zu schaffen.

- Unter Ausnutzung der durch die vorliegenden Untersuchungen festgestellten geringen Systembelastung ist eine Erweiterung des Systems in drei Richtungen vorstellbar:

    1. die gleichzeitige Dekodierung und Darstellung weiterer Videodatenströme mittels Software;

    2. die Anbindung von schmalbandigen Videokonferenzsystemen (ISDN) mit Hilfe entsprechender Software-Kodierer;

    3. eine Verschlüsselung der Videodaten zur Realisierung sicherer Verbindungen.

Abkürzungsverzeichnis

AAL	ATM Adaption Layer
AD	Analog-Digital
ATM	Asynchronous Transfer Mode
AVI	Audio/Video Interleaved
CCIR	Comité Consultatif International de Radiocommunications
CCITT	Comité Consultatif International de Téléphonie et Télégraphie
CD	Compact Disc
CIF	Common Intermediate Format
COM	Component Object Model
CPU	Central Processing Unit
DA	Digital-Analog
DCT	Diskreten–Cosinus–Transformation
DDI	Device Driver Interface
DPCM	Differential Pulse Code Modulation
DRAM	Dynamic Random Access Memory
DS	DirectShow
EDTV	Enhanced Definition Television
EQTV	Enhanced Quality Television
FGM	Filter-Graph-Manager
GDI	Graphics Device Interface
GOP	Group of Pictures
HAL	Hardware Abstraction Layer
HDTV	High Definition Television
IDCT	Inverse-DCT
IEC	International Electrotechnical Commission
ISDN	Integrated Service Digital Network
ISO	International Electrotechnical Commission
ITU	International Telecommunication Union

JPEG	Joint Photographic Expert Group
LAN	Local Area Network
MCI	Media Control Interface
MJPEG	Motion Joint Photographic Expert Group
MPEG	Motion Picture Expert Group
NT	New Technology
NTSC	National Television Standards Committee
PAL	Phase Alternating Line
PC	Personalcomputer
PCI	Peripheral Component Interconnect
PSNR	Peak Signal to Noise Ratio
PVC	Permanent Virtual Connection
QCIF	Quarter Common Intermediate Format
RGB	Rot-Grün-Blau
RIFF	Resource Interchange File Format
ROM	Read Only Memory
SCSI	Small Computer System Interface
SDK	Software Development Kit
SDRAM	Synchronous DRAM
SECAM	Sequentiel Couleur Avec Memoire
SVC	Switched Virtual Connection
TCP	Transport Control Protocol
TV	Television
UDP	User Data Protocol
VFW	Video for Windows
VHS	Video Home System
WAV	Waveform
YIQ	Y – Luminanz; I – (Cyan-Orange-Balance); Q – (Magenta-Grün-Balance)
YUV	Y – Luminanz; U – (Rot-Cyan-Balance); V – (Gelb-Blau-Balance)

Literatur– und Quellenverzeichnis

(Alle aufgeführten Internet-Quellen basieren auf dem Stand von Juni 1999.)

[BOX98] Box, Don: „Essential COM“. Addison Wesley Longman, Inc., 1998

[COE98] Coelho, Rohan: „DirectX®, RDX, RSX, and MMX™ technology: a jumpstart guide to high performance APIs“. Addison–Wesley, 1998

[CUSEE] White Pine–Produktspezifikation / CU-SeeMe Homepage, (http://www.wpine.com/Products/CU-SeeMe/)

[DigiSDK] DigiSuite SDK Reference Manual. Matrox Electronic Systems, 1998

[FRAN99] M. Franke, „Entwicklung einer Native-ATM Schnittstelle für Visitphone“, Diplomarbeit, Universität Leipzig, 1999

[JAIN89] Jain, A. K.: „Fundamentals of Digital Image Processing“. Prentice Hall, 1989

[JPEG93] Information Technology: „Digital Compression and Coding of Continous-tone Still Images“. International Standard ISO/IEC IS 10918, 1993

[M2BUS] Over-The Top: The Movie-2 Digital Audio/Video Expansion Bus. Matrox Electronic Systems, 1998 (http://www.matrox.com/videoweb/movie2bus.html)

[MDS97] Microsoft DirectShow SDK, Microsoft, 1997

[MEI94] Meissner, Hansgeorg: „Digitale Multimediasysteme“. Verlag Technik, 1994

[MIL95] Milde, Torsten: „Videokompressionsverfahren im Vergleich: JPEG, MPEG, H.261, XCCC, Wavelets, Fraktale“. Dpunkt, Verl. Für digitale Technologie, 1995

[MPEG4] MPEG–4 FAQ, (http://www.cselt.it/mpeg/faq/faq_mpeg-4.htm)

[MPEG7] MPEG–7 FAQ (http://www.cselt.it/mpeg/faq/faq_mpeg-7.htm)

[NETME] Netmeeting, Funktionen und Features (Microsoft), (http://www.microsoft.com/products/prodref/113_ov.htm)

[NGU98] Nguyen, G.: „Entwicklung eines Videotransportprotokolls über UDP“. Diplomarbeit, Universität Leipzig, 1998

[OHM95] Ohm, Jens–Rainer: „Digitale Bildcodierung: Repräsentation, Kompression und Übertragung von Bildsignalen“. Springer, 1995

[PEMI92] Pennebaker, W. B. und Mitchell, J. C.: „JPEG Still Image Data Compression Standard“. Van Nostrand Rheinhold, 1992

[SED91] Sedgewick, Robert: „Algorithmen“. Addison–Wesley, 1991

[STEIN94] Steinmetz, R.: „Multimedia Technologie, Einführung und Grundlagen“. Springer International, 1994.

[TAN97] Tanenbaum, Andrew S.: „Computernetzwerke“. Prentice Hall, 1997

Abbildungsverzeichnis

Tabellenverzeichnis

Anhang

IMediaSample: Diese Interface enthält Informationen über Daten die zwischen einzelnen Filtern ausgetauscht werden. Die folgenden Funktionen erlauben den Zugriff auf Informationen über die Daten. Es sind nur die für die Arbeit wesentlichen Funktionen aufgeführt, eine komplette Beschreibung ist in [MDS97] zu finden.

GetPointer: Liefert einen Zeiger auf die Daten, über den sowohl Schreib- wie auch Lesezugriffe möglich sind.

- ***GetSize*** und ***GetActualDataLength***: Hierüber wird das Datenvolumen in Byte ermittelt.

- ***GetTime*** und ***GetMediaTime***: Beide Funktionen liefern Informationen über Start und Endzeit der Daten in Bezug auf den Datenstrom dem sie entstammen.

Es fällt auf, daß es teilweise mehrere Funktionen gibt, die ähnliche Informationen liefern, eine Erklärung dafür konnte nicht gefunden werden. Es wurde jedoch festgestellt, daß in Einzelfällen nicht alle Funktionen korrekte Werte liefern. Bei der Verwendung dieses *Interfaces* sollte daher immer eine Überprüfung stattfinden, welche Funktionen verwendet werden können.

BITMAPINFOHEADER: In dieser Struktur sind Daten über die Bildgeometrie und die Auflösung eines Bildes vermerkt. Die folgenden Angaben stammen aus [MDS97].

```
typedef struct tagBITMAPINFOHEADER {
      DWORD biSize;
      LONG biWidth;
      LONG biHeight;
      WORD biPlanes;
      WORD biBitCount;
      DWORD biCompression;
      DWORD biSizeImage;
      LONG biXPelsPerMeter;
      LONG biYPelsPerMeter;
      DWORD biClrUsed;
      DWORD biClrImportant;
} BITMAPINFOHEADER;
```

□□□□□□	Specifies the number of bytes required by the structure.
□□□□□□□	Specifies the width of the bitmap, in pixels.
□□□□□□□□	Specifies the height of the bitmap, in pixels. If biHeight is positive, the bitmap is a bottom-up DIB (device-independent bitmap) and its origin is the lower left corner. If biHeight is negative, the bitmap is a top-down DIB and its origin is the upper left corner.

▫▫▫▫▫▫▫▫	Specifies the number of planes for the target device. This value must be set to 1.
▫▫▫▫▫▫▫▫▫▫	Specifies the number of bits per pixel. Some compression formats need this information to properly decode the colors in the pixel.
▫▫▫▫▫▫▫▫▫▫▫▫▫	Specifies the type of compression used or requested. Both existing and new compression formats use this member.
▫▫▫▫▫▫▫▫▫▫▫	Specifies the size, in bytes, of the image. This can be set to 0 for uncompressed RGB bitmaps.
▫▫▫▫▫▫▫▫▫▫▫▫▫▫▫	Specifies the horizontal resolution, in pixels per meter, of the target device for the bitmap. An application can use this value to select a bitmap from a resource group that best matches the characteristics of the current device.
▫▫▫▫▫▫▫▫▫▫▫▫▫▫▫	Specifies the vertical resolution, in pixels per meter, of the target device for the bitmap.
▫▫▫▫▫▫▫▫▫	Specifies the number of color indices in the color table that are actually used by the bitmap. If this value is zero, the bitmap uses the maximum number of colors corresponding to the value of the biBitCount member for the compression mode specified by biCompression.
▫▫▫▫▫▫▫▫▫▫▫▫▫▫▫	Specifies the number of color indices that are considered important for displaying the bitmap. If this value is zero, all colors are important.

AM_MEDIA_TYPE: Beschreibt die Eckwerte eines Videodatenstroms. Die genauen Angaben wurden [MDS97] entnommen.

```
typedef struct _MediaType{
      GUID majortype;
      GUID subtype;
      BOOL bFixedSizeSamples;
      BOOL bTemporalCompression;
      ULONG lSampleSize;
      GUID formattype;
      IUnknown *pUnk;
      ULONG cbFormat;
      /* [size_is] */ BYTE __RPC_FAR *pbFormat;
} AM_MEDIA_TYPE;
```

▫▫▫▫▫▫▫▫▫	Major type of the media sample.
▫▫▫▫▫▫▫	Subtype of the media sample.
▫▫▫▫▫▫▫▫▫▫▫▫▫▫▫▫▫	If TRUE, samples are of a fixed size.
▫▫▫▫▫▫▫▫▫▫▫▫▫▫▫▫▫▫▫▫▫	If TRUE, samples are compressed.
▫▫▫▫▫▫▫▫▫▫▫	Size of the sample in bytes.
▫▫▫▫▫▫▫▫▫▫	Registered (GUID) format type.
▫▫▫▫	Pointer to the IUnknown interface.
▫▫▫▫▫▫▫▫	Size of the format section of the media type.
▫▫▫▫▫▫▫▫	Pointer to the format section of the media type. The layout of this is determined by the format type GUID

Erklärung

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

Leipzig, _____

Matthias Fiebig