Technische Informatik I
Prof. Dr.-Ing. Rolf-Rainer Grigat

Farbmanagement mit ICC-Profilen

Studienarbeit

Marco Budde

September 1999

Inhaltsverzeichnis

Kapitel 1

Einleitung

1.1 Problemstellung

Bei der digitalen Bildverarbeitung sieht man sich mit dem Problem konfrontiert, daß Farbbilder auf verschiedenen Ausgabegeräten teilweise doch sehr unterschiedlich wiedergegeben werden. So wird im Extremfall z.B. eine Farbe, die auf dem Monitor als Gelb wahrgenommen wird, auf einem Ausdruck eher Orange aussehen. Ja sogar die Ausdrucke eines Bildes auf zwei Druckern können sich sehr stark unterscheiden.

Verursacht wird dieses Problem durch die Art und Weise, wie das Auge des Menschen Farben wahrnimmt und wie Farben durch die jeweiligen Ausgabegeräte dargestellt werden.

1.1.1 Das Auge

Das Auge des Menschen besteht aus Zapfen und und Stäbchen. Mittels der Stäbchen nimmt der Mensch Helligkeiten wahr, wobei die Stäbchen vor allem für Lichtreize niedriger Intensität empfindlich sind. Die Zapfen reagieren hingegen vor allem auf hohe Intensitäten und sind für den Farbeindruck zuständig.

Es existieren drei verschiedene Arten von Zapfen, die jeweils für einen anderen spektralen Bereich des Lichtes empfindlich sind [1]. Die Empfindlichkeitskurve der »roten«, »grünen« und »blauen« Zapfen ist in Abbildung 1.1 zu sehen.

Wenn die Strahlung eines Selbstleuchters, z.B. eines Fernsehers, auf die Netzhaut des Auges trifft, entsteht folgender Farbreiz:

$$\varphi = S(\lambda)$$

Hierbei ist $S(\lambda)$ die spektrale Strahlungsverteilung des Lichtes.

Bei einem Nichtselbstleuchter wird das Umgebungslicht oder das Licht einer künstlichen Beleuchtung teilweise von dem bestrahlten Körper reflektiert. Dieses reflektierte Licht trifft dann aufs Auge und löst dort diesen Farbreiz aus:

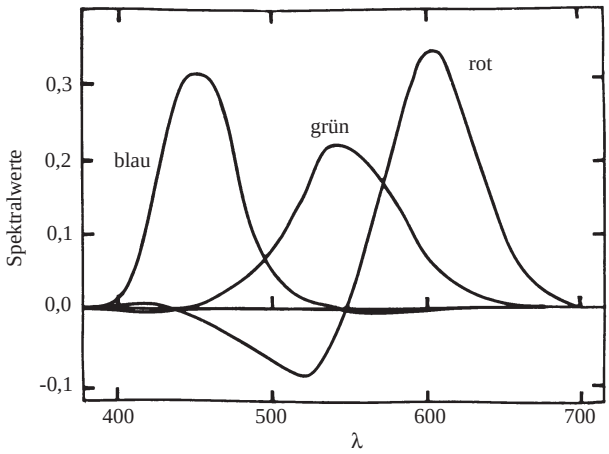

Abbildung 1.1: Empfindlichkeitskurve der Zapfenempfänger

$$\varphi = S(\lambda)\,\beta(\lambda)$$

Der spektrale Remissionsgrad wird hierbei durch $\beta(\lambda)$ beschrieben. Der Körper, z.B. bedrucktes Papier, wirkt also wie ein Filter, der bestimmte Wellenlängen des Lichtes mehr oder weniger stark durchläßt; siehe auch Bild 1.2. Der Farbreiz hängt also vom Licht, vom Papier und von der Druckfarbe ab.

Der Farbeindruck entsteht durch Multiplikation mit der Empfindlichkeitskurve einer Zapfenart und Integration über die Wellenlänge des sichtbaren Lichtes. Für die »roten«-Zapfen ergibt sich dann z.B.:

$$\alpha_{rot} = \int_{\lambda_{min}}^{\lambda_{max}} \varphi(\lambda)\, r(\lambda)\, d\lambda$$

Wie man leicht an der Formel erkennen kann, können völlig unterschiedliche Spektren den gleichen Farbeindruck beim Menschen erzeugen. Aufgrund der verschiedenen Zapfenarten des Auges läßt sich jeder Farbreiz als Linearkombination dreier Primärvalenzen erzeugen, wobei sich nicht jede Farbe durch die drei gleichen Primärvalenzen darstellen läßt [1]. Das führt dazu, daß sich mit den drei Grundfarben eines Monitors oder eines Druckers nicht alle Farbe erzeugen lassen. Der darstellbare Farbraum ist also begrenzt.

1.1.2 Ein-/Ausgabegeräte

Die Vorlagen werden in der Regel mit Hilfe eines Scanners in eine digitale Darstellung überführt. Scanner arbeiten meistens mit dem RGB-Farbraum. Jeder Pixel wird also durch drei Primärvalenzen dargestellt, deren Wellenlängen im Bereich der Farben Rot, Grün und Blau liegt.

Allerdings unterscheiden sich die verwendeten Primärvalenzen von Scanner zu Scanner mehr oder weniger stark. Verursacht wird dieses unter anderem durch die unterschiedlichen Übertragungsfunktionen der verwendeten Filter vor den CCD-Elementen. Dieses führt dazu, daß die digitalen RGB-Werte einer Vorlage von dem verwendeten Scanner abhängig sind. Jeder Scanner hat also einen gerätespezifischen Farbraum.

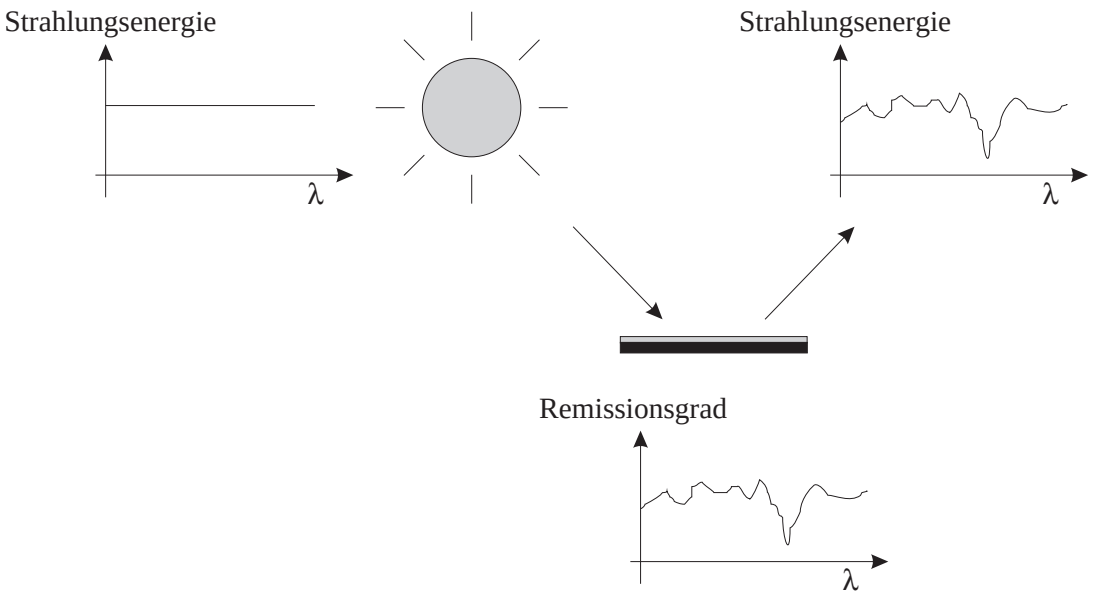

Abbildung 1.2: Nichtselbstleuchter

Das gleiche Problem hat man bei der Ausgabe von digitalen Daten auf einem Monitor. Monitore arbeiten ebenfalls mit dem RGB-Farbraum, wobei jeder Pixel aus rotem, grünem und blauem Phosphor besteht. Allerdings verwenden die Hersteller der Geräte nicht unbedingt Phosphor, das mit der gleichen Wellenlänge strahlt wie das des anderen Herstellers. Durch unterschiedliche Phosphore und Prozeßschwankungen bei der Herstellung hat also auch jeder Monitor einen eigenen Farbraum.

Drucker arbeiten heute meistens mit vier Primärvalenzen: Cyan, Magenta, Yellow und Black. Bei einigen Consumergeräten wird auf das Schwarz verzichtet. Im Gegensatz zu Scannern und Monitoren arbeiten Drucker allerdings nicht mit additiver sondern mit substraktiver Farbmischung [2]. Die Kombination aus Druckfarbe und Papierfarbe wirkt als Filter für das Umgebungslicht.

Wie man sich leicht vorstellen kann, ist gerade die Wiedergabe von Farben auf Druckern relativ problematisch, da der Farbreiz von vielen Faktoren abhängig ist. So wird es immer gewisse Schwankungen bei den eingesetzten Farbstoffen geben. Auch der Wechsel der Papiersorte bzw. -art ist nicht unproblematisch, da ja das Spektrum des Papiers seinen Teil zum Farbeindruck beiträgt.

Überhaupt keinen Einfluß hat man auf die Betrachtungsbedingungen. Zwei Farben, die bei Kunstlicht vielleicht den gleichen Farbreiz beim Menschen erzeugen, können bei Tageslicht einen völlig unterschiedlichen Farbreiz erzeugen: metamere Farben.

1.2 Lösungsansatz mittels CMYK

Um die Probleme mit der Farbwiedergabe in den Griff zu bekommen, erfolgt der gesamte Arbeitsablauf in der professionellen Druckindustrie im CMYK-Farbraum. Schon vor dem Scannen der Vorlagen wird festgelegt, wie und auf welchem Papier das fertige Werk ausgedruckt werden soll.

Die Vorlagen werden dann mit einem Scanner eingelesen und vom RGB-Farbraum des jeweiligen Scanners in den CMYK-Farbraum des zu verwendenen Druckprozesses konvertiert. Zur Darstellung der Daten auf einem Monitor müssen diese wieder nach RGB konvertiert werden.

Problematisch an diesem Ansatz ist vor allem die Tatsache, daß die Daten selbst auf genau einen Druckprozeß optimiert werden. Ein Wechsel des zu verwendenden Papieres oder der Druckmaschine fällt bei diesem Ansatz schwer, da dafür die Daten von einem CMYK-Farbraum auf einen anderen umgerechnet werden müssen.

Auch muß zwischen Ersteller der Druckdaten und der Druckerei vorab eine intensive Kommunikation stattfinden, um die genauen Parameter des Druckprozesses zu definieren, auf die die Druckdaten dann optimiert werden.

Aus diesen Gründen ist dieser Ansatz für den Heimbereich und den semiprofessionellen Bereich eher ungeeignet. Kaum ein Privatverbraucher wird die Geduld aufbringen, vor der eigentlichen Arbeit den eigenen Drucker und das verwendete Papier genau auszumessen. Auch wird ihm oftmals das notwendige Fachwissen zur Optimierung der Scans fehlen.

1.3 Lösungansatz mit ICC-Profilen

Einen völligen anderen Ansatz verfolgt das *International Color Consortium* mit seinen ICC-Profilen [4, 5].

Statt die Rohdaten des Scanners in den Farbraum eines bestimmten Ausgabegerätes zu konvertieren, wird bei diesem Ansatz direkt mit den Rohdaten gearbeitet. Erst für die Ausgabe werden die Daten in den jeweiligen Ausgabefarbraum konvertiert, wobei jedoch auch dabei die Rohdaten erhalten bleiben.

Man legt den Rohdaten im RGB-Format eine ICC-Profil-Datei bei, die die Konvertierung der Rohdaten in einen geräteunabhängigen Farbraum wie CIE Lab oder CIE XYZ beschreibt. Ein solches Profil wird dann auch für die Ausgabegeräte (Monitor, Drucker) benötigt. Hier beschreibt das Profil jedoch den umgekehrten Weg: Konvertierung von CIE Lab bzw. CIE XYZ in z.B. den Farbraum des Druckers (CMYK).

Durch die Kopplung eines Eingabe- und eines Ausgabeprofiles können dann die Rohdaten des Scanners in den Farbraum des Ausgabegerätes konvertiert werden, ohne dabei die Rohdaten zu zerstören. Auf dem Monitor werden also z.B. die für den Monitor konvertierten Daten angezeigt, intern arbeitet das Programm aber weiterhin mit den Rohdaten, so daß diese nach der Bearbeitung problemlos in den Druckerfarbraum konvertiert werden können.

Wie man leicht erkennen kann, finden dank der ICC-Profile weit weniger zerstörerische Konvertierungen statt, bei denen durch Überläufe und Rundungsfehler Daten verloren gehen.

Außerdem ist es theoretisch nicht mehr nötig, vor der eigentlichen Arbeit Absprachen mit der Druckerei zu treffen, da erst in der Druckerei die Rohdaten mittels der Profile des Scanners und des Druckprozesses für den Druck optimiert werden.

Das Konzept der ICC-Profile ist auch für den Privatanwender mit seinen nicht so hohen Ansprüche gut geeignet, da der Hersteller eines Gerätes - eine geringe Serienstreuung vorausgesetzt - jedem Gerät gleich die passende Profil-Datei beilegen kann, so daß der Anwender das Gerät nicht selbst kalibrieren muß, was doch recht aufwendig sein kann.

1.4 Ziel dieser Studienarbeit

Um das ICC-Konzept in der Praxis nutzen zu können, muß dieses von allen am Bearbeitungsprozeß beteiligten Programmen unterstützt werden. Bisher stellen nur die Betriebssysteme MS Windows 98 und Apple MacOS eine Infrastruktur für ein Farbmanagement auf Basis von ICC-Profilen bereit, die auch nur von wenigen Anwendungen genutzt wird. Dem immer mehr an Bedeutung gewinnenden Betriebssystem Linux fehlt wie die anderen Unix-Systemen auch jegliche Unterstützung für ein solches System.

Das Ziel dieser Arbeit war deshalb die Entwicklung eines solchen Farbmanagementsystemes für Linux und Unix-Betriebssysteme in Form einer Laufzeitbibliothek, die dann von Anwendungen für ein farbsicheres Arbeiten genutzt werden kann.

Zu Demonstrationszwecken wurde eine Anwendung entwickelt, die TIFF-Grafikdateien [11] mittels zweier ICC-Profile von einem geräteabhängigen Farbraum in einen anderen konvertiert. Diese Anwendung bedient sich dabei dem entwickeltem Farbmanagementsystem.

Um die Qualität des entwickelten Systems zu testen, wurden die 17 CIE-Testfarben nach DIN 6169 ausgedruckt, einscannt und das Distanzmaß zum Original ermittelt. Zur visuellen Kontrolle wurden Photos gescannt, gedruckt und mit den Originalen verglichen.

Kapitel 2

ICC-Profile

Dieses Kapitel beschreibt die Funktionsweise des ICC-Ansatzes und den Aufbau der ICC-Profile.

2.1 Allgemeine Struktur

Wie bereits in Abschnitt 1.3 beschrieben wurde, wird für jedes Gerät eine ICC-Profil-Datei benötigt, die die Konvertierung vom geräteabhängigen in einen geräteunabhängigen Farbraum und umgekehrt beschreibt.

Die Daten der Quelle werden mit dem Profil der Quelle in den wahrnehmungsbasierten CIE Lab bzw. CIE XYZ Farbraum konvertiert. Die so transformierten Grafikdaten können dann mit dem ICC-Profil des Zielgerätes in dessen geräteabhängigen Farbraum konvertiert werden.

Diese geräteunabhängigen Farbräume stellen die Farben nicht mehr als Linearkombination dreier Primärvalenzen dar. Vielmehr wird eine Farbe durch ihre Helligkeit, Farbart und Sättigung beschrieben [2]. Der Vorteil dieser Darstellung besteht darin, daß diese Farbräume groß genug sind, um die Farben verschiedener Farbräume darzustellen.

Der CIE RGB-Farbraum (mit den Primärvalenzen 700 nm, 546,1 nm und 438,8 nm) läßt sich z.B. mit folgender Formel in XYZ-Koordinaten transformieren:

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} 0,490 & 0,310 & 0,200 \\ 0,177 & 0,813 & 0,011 \\ 0,000 & 0,010 & 0,990 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

Die durch die ICC-Profile beschriebenen Transformationen können auch invers ausgeführt werden. Dieses erlaubt z.B. die Simulation der Farben eines Druckers auf dem Bildschirm. Hierzu werden die Daten des Scanners zuerst nach Lab / XYZ transformiert. Diese werden dann mit dem Profil des Druckers in den Druckerfarbraum und von dort wieder in den Lab- / XYZ-Farbraum transformiert. Die so entstandenen Daten enthalten jetzt nur noch Farben, die vom Drucker wiedergegeben werden können. Mittels des Profiles eines Monitors können die simulierten Druckdaten auf dem Bildschirm dargestellt werden. So kann man bereits bei der Bearbeitung dafür sorgen, daß die verwendeten Farben überhaupt auf dem Drucker darstellbar sind.

Ein ICC-Profil besteht aus einem Header, einer Tabelle der in dem Profil enthaltenen Tags und den eigentlich Daten dieser Tags. Im Header wird unter anderem festgelegt, welche geräteabhängigen und geräteunabhängigen Farbkoordinaten das Profil verwendet, für welches Betriebssystem das Profil gedacht ist und ob es sich um ein Profil für ein Eingabegerät, einen Monitor oder ein Ausgabegerät handelt.

Durch das Konzept der Tags, das auch bei anderen Formaten wie z.B. den TIFF-Dateien [11] Verwendung findet, kann man den Standard für ICC-Profile jederzeit erweitern und neue Tags hinzufügen, ohne daß dadurch Kompatibilitätsprobleme entstehen würden. Dieses ist sehr wichtig, da Grafikdaten und Eingabeprofil ja eine Einheit bilden und auch nach Jahren noch lesbar sein müssen.

Die Transformationen zwischen den verschiedenen Farbkoordinaten werden durch die numerischen Werte beschrieben, die in den Tags enthalten sind. Hierbei werden vor allem Übersetzungstabellen und Mischmatrizen verwendet. Jedes Profil muß abhängig von dem Gerätetyp, für das es gedacht ist, bestimmte Tags enthalten. Zusätzlich zu den vorgeschriebenen Elemente sind auch noch optionale möglich. Optional sind z.B. die Elemente, die die genauen Betrachtungsbedingungen und Geräteeinstellungen festlegen, für die das Profil gedacht ist: z.B. Papierart, -farbe und die Beleuchtung.

Denn wie bei der Optimierung mittels des CMYK-Farbraumes, siehe Abschnitt 1.2, ist jedes Profil im Prinzip nur für genau einen Druckprozeß gültig. Für jede Papiersorte muß die Druckerei also ein Profil erzeugen. Im Privatbereich wird man auf diesen Aufwand in der Regel verzichten und dabei die mehr oder weniger großen Abweichungen durch die Umgebungsbedingungen in Kauf nehmen. Denn während sich Profile für Scanner noch mit relativ einfach Mitteln erzeugen lassen, werden für Monitore und Drucker relativ teure Meßgeräte benötigt, über die Privatanwender nicht verfügen.

Der ICC-Profil-Standard sieht zwei verschiedene Methoden vor, die Transformation von einem geräteabhängigen in einen geräteunabhängigen Farbraum numerisch zu beschreiben: durch eine Matrix oder durch eine mehrdimensionale Tabelle. Im nachfolgenden werden beide Methode vorgestellt.

2.2 TRC

Die erste Methode, die im weiteren Text die *TRC-Methode* genannt wird, ist sehr einfach gehalten. Sie ist ausschließlich für die Transformation von RGB-Daten in den XYZ-Farbraum und umgekehrt definiert. Gemäß ICC-Profil-Standard findet dieses Modell nur bei Eingabe- und Monitorprofilen Anwendungen. Dieses ist auch sinnvoll, da Drucker in der Regel mit dem CMYK-Farbmodell arbeiten.

Die RGB-Rohdaten z.B. eines Scanners werden zuerst mit drei eindimensionalen Tabellen linearisiert:

$$
\begin{aligned}
R_{lin} &= redTRC\,[R_{dev}] \\
G_{lin} &= greenTRC\,[G_{dev}] \\
B_{lin} &= blueTRC\,[B_{dev}]
\end{aligned}
$$

Die so entstandenen, linearisierten RGB-Daten werden dann mit folgender 3x3 Mischmatrix in den XYZ-Farbraum transformiert:

$$
\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} =
\begin{bmatrix}
redColorant_X & greenColorant_X & blueColorant_X \\
redColorant_Y & greenColorant_Y & blueColorant_Y \\
redColorant_Z & greenColorant_Z & blueColorant_Z
\end{bmatrix}
\begin{bmatrix} R_{lin} \\ G_{lin} \\ B_{lin} \end{bmatrix}
$$

Soll die TRC-Methode zur Ausgabe von Daten auf einem Monitor verwendet werden, hilft die obige Rechnung nicht direkt, da man ja nicht RGB nach XYZ transformieren möchte; vielmehr liegen die Daten im XYZ-Farbraum vor und müssen in den RGB-Farbraum des Monitores konvertiert werden. Dazu werden die Tabellen und die Matrix invertiert. Außerdem wird die Reihenfolge der Rechenschritte vertauscht. Die XYZ-Daten werden zuerst mit der Mischmatrix in RGB-Daten transformiert und dann mit den Tabellen an die nicht lineare Ansteuerung des Monitores angepaßt:

$$\begin{bmatrix} R_{lin} \\ G_{lin} \\ B_{lin} \end{bmatrix} = \begin{bmatrix} redColorant_X & greenColorant_X & blueColorant_X \\ redColorant_Y & greenColorant_Y & blueColorant_Y \\ redColorant_Z & greenColorant_Z & blueColorant_Z \end{bmatrix}^{-1} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

$$\begin{aligned} R_{dev} &= redTRC^{-1}\left[R_{lin}\right] \\ G_{dev} &= greenTRC^{-1}\left[G_{lin}\right] \\ B_{dev} &= blueTRC^{-1}\left[B_{lin}\right] \end{aligned}$$

Die Tabellen und die Matrix sind in den ICC-Profilen unabhängig davon, ob es sich um Eingabe- oder Monitorprofile handelt, immer in der nicht invertierter Form abgelegt. Ein Farbmanagementsystem muß diese Invertierung bei Bedarf also selbst durchführen, was bei einigen Profilen zu Problemen führt.

In einem ICC-Profil sind die Tabellen unter folgenden Tagnamen abgelegt: *redTRC*, *greenTRC* und *blueTRC*. Jeder TRC besteht aus einem Zähler, der angibt, aus wieviel Stützstellen sich der TRC zusammensetzt, und den eigentlichen Werten der Stützstellen. Bei einem Zähler von »0« wird von einer linearen Kurve ausgegangen. Ein Zähler von »1« signalisiert, daß die erste Stützstelle den Wert einer Gamma-Kurve definiert. Jede Stützstelle ist ein 16 Bit-Wert.

Die Mischmatrix ist durch folgende Tags definiert: *redColorant*, *greeenColorant* und *blueColorant*. Jeder Colorant besteht aus einem XYZ-Wert, wobei ein Wertebereich von 0,0 bis 1,0 benutzt wird.

In abgeänderter Form wird das TRC-Modell auch für Geräte benutzt, die nur Graustufen wiedergeben können. Hierzu gehören z.B. Graustufen-Monitore und -Scanner. Da die Pixel bei solchen Geräten nicht aus drei sondern nur aus einem Kanal bestehen, enthält das Profil auch nur einen TRC: *grayTRC*. Im XYZ-Farbraum wird die Helligkeit durch den Y-Kanal definiert, so daß die Formel für die Transformation so aussieht:

$$Y = grayTRC\left[dev\right]$$

Auf eine Mischmatrix kann für Graustufen-Daten verzichtet werden, da ja nur ein Eingabe- und ein Ausgabekanal vorhanden ist. Für die Transformation von Y in den Graustufenwert eines Gerätes muß die TRC-Kurve wie bei Farbdaten invertiert werden.

Beachtet werden muß bei dem TRC-Modell, daß der geräteunabhängige Farbraum, im ICC-Profil-Standard [4] auch PCS genannt, immer in XYZ-Farbkoordinanten vorliegt, selbst wenn das Profil laut Header Lab-Koordinaten für den PCS verwendet.

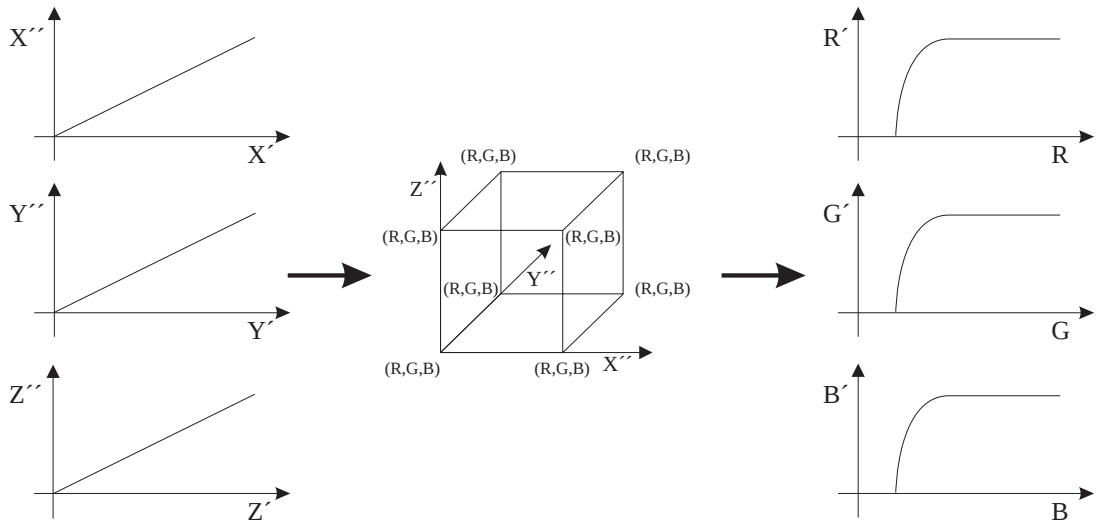

Abbildung 2.1: Flußdiagramm der LUT-Methode

2.3 LUT

Im Vergleich zur TRC-Methode ist die LUT-Methode erheblich komplexer, was sich auch im nötigen Rechenaufwand niederschlägt. LUT steht für »lookup table«. Ein Flußdiagramm der LUT-Methode ist in Abbildung 2.1 zu sehen. Die in dem Flußdiagramm verwendeten Kennlinien entsprechen denen, die das ICC-Profil für HP-Deskjet-Drucker verwendet.

Die LUT-Methode ist im Gegensatz zur TRC-Methode nicht nur auf eine Transformation zwischen RGB- und XYZ-Koordinaten beschränkt. Neben XYZ-Koordinaten können für den geräteunabhängigen Farbraum auch Lab-Koordinaten verwendet werden. Für den geräteabhängigen Farbraum stehen zusätzlich Systeme wie CMY oder CMYK bereit. Außerdem sind allgemeine geräteabhängige Farbräume mit zwei bis fünfzehn Grundfarben benutzbar.

Falls es sich um ein Profil für die Ausgabe auf einem Drucker handelt und für die geräteunabhängigen Koordinaten XYZ verwendet wird, so muß vor der eigentlichen Transformation folgende Matrix angewendet werden, deren Werte im LUT-Tag des Profiles enthalten sind:

$$\begin{bmatrix} X' \\ Y' \\ Z' \end{bmatrix} = \begin{bmatrix} e_{00} & e_{01} & e_{02} \\ e_{10} & e_{11} & e_{12} \\ e_{20} & e_{21} & e_{22} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

Sowohl bei XYZ- als auch Lab-Koordinaten läuft die weitere Rechnung dann wie in Abbildung 2.1 ab. Für jeden Eingangskanal gibt es eine eindimensionale Ersetzungstabelle. Jede Tabelle besteht aus zwei bis 4096 Stützstellen. Die genaue Aufgabe dieser Ersetzungstabelle wird vom ICC-Profil-Standard nicht festgelegt. Häufig wird sie dazu verwendet, die Eingangsdaten so zu verzerren, daß man beim nachfolgenden CLUT mit möglichst wenig Stützstellen auskommt.

Nach den Tabellen folgt dann der CLUT. Die Dimension dieser Tabelle entspricht der Anzahl der Eingangskanäle. Jeder Stützstelle dieser mehrdimensionalen Tabelle ist ein Ausgabewert in den Zielkoordinaten zugeordnet. So wird in dem Beispiel jedem XYZ-Wert ein RGB-Wert zugeordnet. Wie man sich leicht vorstellen kann, sind die Möglichkeiten dieses Ansatz deutlich größer wie die der 3x3-Matrix, die bei der TRC-Methode verwendet wurde.

Allerdings ist auch der Speicherbedarf für eine solche Tabelle deutlich größer. Der Speicherbedarf in Byte läßt sich folgendermaßen berechnen [4]:

$$\text{Größe} = \text{Stützstellanzahl}^{\text{Eingangskanäle}} * \text{Ausgabekanäle} * 2$$

Je nach Auflösung der Kanäle und Anzahl der Stützstellen kann eine solche Tabelle also schnell über 100 MByte belegen, was auch heutige Rechner schnell überlastet und vor allem den Austausch der Profile erschwert. Aus diesem Grund ist die Anzahl der Stützstellen in der Regel relativ klein. Selbst hochwertige Profile verwenden je Koordinate deutlich weniger wie 50 Stützstellen. Die nicht enthaltenen Werte werden aus den Stützstellen dann per Software interpoliert.

Der CLUT führt also die eigentliche Transformation zwischen den verschiedenen Farbkoordinaten durch. Die so gewonnenen Werte werden schließlich noch mit einer weiteren eindimensionalen Tabelle für jeden Farbkanal bearbeitet. Auch die Aufgabe dieser Ausgangstabelle ist nicht im Standard definiert. Sie kann z.B. zum Ausgleich der Nichtlinearitäten des Druckers verwendet werden.

Der ICC-Profil-Standard sieht zwei verschiedene LUT-Tags vor: einmal mit 8 Bit- und einmal mit 16 Bit-Auflösung. So kann der Speicherbedarf recht einfach an die mit einem Gerät erreichbare Genauigkeit angepaßt werden.

Im Gegensatz zur TRC-Methode müssen bei der LUT-Methode die Tabellen und Matrizen nicht invertiert werden, wenn man die Daten in die entgegengesetzte Richtung transformieren möchte. Vielmehr kann ein Profil für jede Richtung ein eigenes LUT-System definieren.

Tatsächlich kann nicht nur für jede Richtung ein eigenes LUT-System im Profil enthalten sein, sondern für jede Richtung können drei LUT-Systeme Bestandteil eines Profiles sein. Jedes dieser drei LUT-Systeme definiert dabei eine für eine bestimmte Aufgabe optimierte Transformation. Folgende Optimierungen - auch *Rendering Intent* genannt - sieht der Standard vor:

- fotografische Optimierung
- relative farbmetrische Optimierung
- sättigungbasierte Optimierung
- absolute farbmetrische Optimierung

Die absolute farbmetrische Optimierung ist nicht im Profil gespeichert, sondern läßt sich aus der relativen farbmetrischen Optimierung gemäß folgender Formeln berechnen:

$$\begin{aligned} X_a &= \left(\frac{X_{mw}}{X_i}\right) X_r \\ Y_a &= \left(\frac{Y_{mw}}{Y_i}\right) Y_r \\ Z_a &= \left(\frac{Z_{mw}}{Z_i}\right) Z_r \end{aligned}$$

Hierbei ist X_{mw}, Y_{mw} und Z_{mw} der Weißpunkt des Papiers und X_i, Y_i und Z_i das Weiß der Beleuchtung.

Der Standard selbst beschreibt erstaunlicherweise nicht, welches Ziel die Optimierungen genau haben sollen. Informationen darüber finden sich ausschließlich in der Sekundärliteratur wie [2, 12]. Grundsätzlich ist eine Optimierung deshalb notwendig, weil sich die Größe der zwei zu verknüpfenden Farbräume unterscheiden kann, so daß z.B. der Druckerfarbraum nicht alle Farben des Monitorfarbraumes wiedergeben kann. Es muß also entschieden werden, was mit den nicht darstellbaren Farben passieren soll.

Die *fotografische Optimierung*, im Standard auch *perceptual rendering intent* genannt, ist vor allem für die Wiedergabe von Fotos geeignet. Der gesamte Quellfarbraum wird gleichmäßig so komprimiert, daß er komplett im Zielfarbraum wiedergegeben wird. Ein Farbe mit maximaler Sättigung im Quellfarbraum wird also auch im Druckerfarbraum die maximal mögliche Sättigung haben. Die Idee ist hierbei, daß der Abstand zwischen allen Farben gleichmäßig verkleinert wird. Diese Optimierung verändert jede Farbe, selbst wenn sie im Zielfarbraum direkt darstellbar wäre. Dieses führt dazu, daß diese Optimierung völlig ungeeignet ist, wenn es darum geht, eine genau definierte Farbe auszugeben, wie dieses z.B. bei Logos der Fall sein könnte.

Wenn die genaue Wiedergabe definierter Farbwerte im Vordergrund steht, sollte die *farbmetrische Optimierung*, auch *colorimetric rendering intent* genannt, Verwendung finden. Diese Optimierung gibt die Farbwerte, soweit dieses möglich ist, exakt wieder. Farbwerte, die im Zielfarbraum nicht darstellbar sind, werden auf den nächsten gerade noch darstellbaren Wert gerundet. Ist also die z.B. die Sättigung für den Zielraum zu groß, so wird die Sättigung auf die maximale Sättigung des Zielfarbraumes verringert. Dieses Verfahren eignet sich für Fotos eher schlecht, da bestimmte Farben zulaufen; verschiedene Farben werden also gleich wiedergegeben, so daß der Betrachter im Foto weniger Details wahrnehmen kann.

Schließlich gibt es noch die *Sättigungsoptimierung*, bei der versucht wird, die Sättigung nicht zu verändern. Statt dessen variert man den Farbton, um im Zielfarbraum nicht darstellbare Farben wiederzugegeben. Im Standard wird diese Optimierung *saturation rendering intent* genannt.

Die LUT-Systeme verwenden im ICC-Profil die Tags mit dem Namen *AToB* für die Transformation von geräteabhängigen in geräteunabhängige Farbkoordinaten und die Namen *BToA* für die entgegengesetzte Richtung. Die Optimierung wird durch eine zusätzlich angehängte Ziffer gekennzeichnet: fotometrisch (0), relativ farbmetrisch (1), Sättigung (2) und absolut farbmetrisch (3).

2.4 Andere Tags

Neben den Tags, die für die TRC- und LUT-Methode verwendet werden, enthalten viele Profile weitere standardisierte und proprietäre Tags. So definieren z.B. die Tags *mediaBlackPoint* und *mediaWhitePoint* die Schwarz- und Weißpunkte des Mediums in XYZ-Koordinaten.

Sehr wichtig ist natürlich auch, im Profil zu speichern, für welches Gerät, bei welchen Einstellungen, für welches Papier, für welche Betrachtungsbedingungen usw. das Profil erzeugt worden ist. Denn jedes Profil ist nur für die Randbedingung gültig, für die es erzeugt worden ist. Weicht die Realität von diesen Bedingung ab, wird es zu mehr oder weniger großen Abweichungen in der Qualität des Resultates kommen. Für die Angabe dieser Randbedingungen gibt es eine große Anzahl von Tags. Näheres dazu findet sich im ICC-Profil-Standard [4, 5].

Schließlich enthalten manche Profile noch Tags, die in Verbindung mit dem Farbmanagement der Seitenbeschreibungsprache PostScript verwendet werden können. Diese werden im Rahmen dieser Studienarbeit allerdings nicht weiter behandelt, da die Umrechnung zwischen den Farbräumen von der entwickelten Laufzeitbibliothek übernommen wird.

Kapitel 3

Implementierung

Wie die Laufzeitbibliothek für ein Farbmanagementsystem für Linux-/Unixrechner implementiert wurde, welche Algorithmen benötigt wurden und welche Probleme aufgetreten sind, wird in dem folgenden Kapitel behandelt.

3.1 Nutzungskonzept der Bibliothek

Das Farbmanagementsystem, das im Rahmen dieser Studienarbeit entwickelt wurde, ist in Form einer Laufzeitbibliothek realisiert worden. Diese Bibliothek kann von Programmen oder Treibern verwendet werden, um den Benutzern ein Arbeiten mit kalibrierten Farben zu erlauben.

Ein Diagramm, wie die Integration der Bibliothek in ein bestehendes Programm aussehen kann, ist in Abbildung 3.1 zu sehen. Das Programm liest wie bisher die Quelldateien, die z.B. von einem Scanner stammen, im RGB-Farbraum ein. Zur Darstellung der Daten auf dem Monitor übergibt das Programm dann die RGB-Daten, das Scanner-Profil und das Monitor-Profil an die Bibliothek. Die von der Bibliothek in der Farbraum des Monitors transformierten Daten stellt die Anwendung dann auf dem Monitor dar. Hierbei gilt zu beachten, daß die Anwendung intern weiterhin im RGB-Farbraum des Scanners arbeitet.

Verändert der Benutzer also die Grafik, so werden diese Änderungen einmal an den Originaldaten durchgeführt und zum zweiten werden diese Daten dann für die Monitordarstellung wiederum mit der Bibliothek in den Monitorfarbraum transformiert.

Für die Ausgabe der Grafik auf einem Drucker transformiert die Anwendung diese mit den Profilen des Scanners und des Druckers in den Farbraum des Druckers.

Dieses Nutzungskonzept der Bibliothek hat zwar den Vorteil, daß man bestehende Anwendungen kaum ändern muß, um das Farbmanagement zu nutzen; es kann aber teilweise zu Problemen führen, wenn man z.B. zwei Bilder, die mit verschiedenen Geräten erzeugt wurden, zusammenfügen will. Aus diesem Grund wäre es sicherlich sinnvoller, wenn die Programme intern mit einem geräteunabhängigen Farbraum wie CIE Lab arbeiten würden; siehe Abbildung 3.2.

Der Scannertreiber könnte dann z.B. die RGB-Rohdaten mit der Bibliothek nach Lab transformieren und die Daten dann auch in diesem Farbraum in einer Grafikdatei ablegen. Das Bearbeitungsprogramm würde intern mit Lab-Koordinaten rechnen und die Daten nur zur Darstellung mit der Bibliothek vom

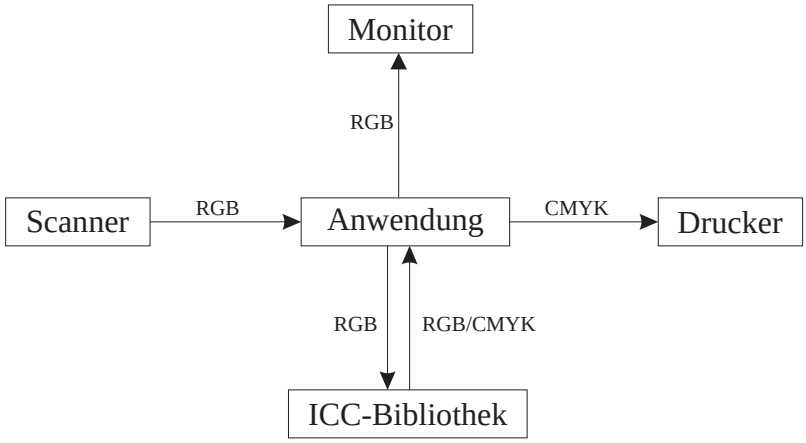

Abbildung 3.1: Verwendung der Bibliothek in bestehenden Anwendungen

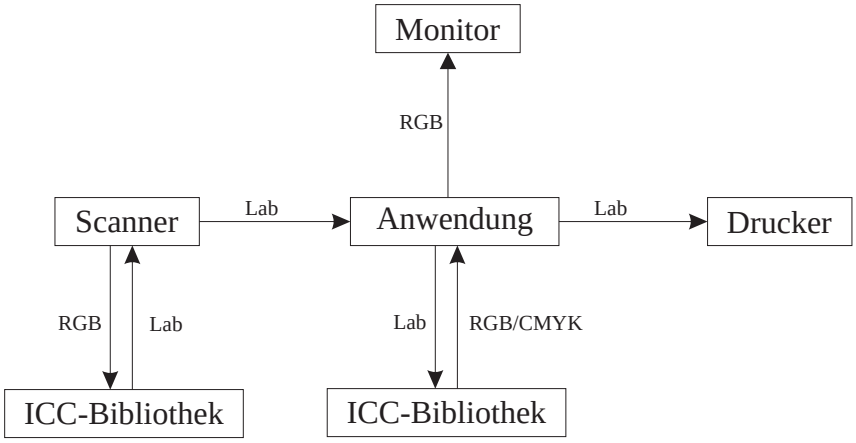

Abbildung 3.2: Verwendung der Bibliothek in neuen Anwendungen

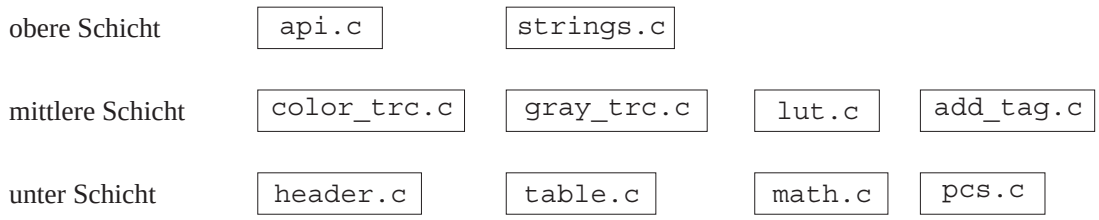

Abbildung 3.3: Struktur der Source der Bibliothek

Lab-Farbraum in den Farbraum des Monitors transformieren. Die Ausgabe auf dem Drucker würde den Lab-Farbraum von PostScript verwenden. Die eigentliche Transformation in den Druckerfarbraum würde entweder im Drucker selbst oder im RIP (Raster Image Processor) auf dem Computer stattfinden. Bei letzterem könnte sich der RIP dann wiederum der Bibliothek bedienen, um die Transformation durchzuführen.

3.2 Struktur der Sourcen

In Abbildung 3.3 ist die Struktur der Sourcen der entwickelten Bibliothek zu sehen. Die Sourcen bestehen aus einer Anzahl von Dateien, die sich grob in drei Schichten einteilen lassen.

Die Low-Level-Schicht besteht aus vier Quelldateien: die Datei `header.c` enthält Funktionen für das Auslesen des ICC-Profil-Headers. Funktionen zum Zugriff auf das Inhaltsverzeichnis eines ICC-Profiles sind in der Quelldatei `table.c` zu finden. Mathematische Funktionen wie die Interpolation oder Inversion einer Funktion sind in der Datei `math.c` implementiert. Die Datei `pcs.c` stellt schließlich mathematische Funktionen für den geräteunabhängigen Farbraum bereit.

In der mittleren Schicht sind dann die eigentlichen Routinen für die Transformation zwischen geräteabhängigen und -unabhängigen Farbräumen mittels der TRC- und LUT-Methode zu finden. Einige zusätzliche Tags wie der Weißpunkt können mit den Routinen aus `add_tag.c` ausgelesen werden.

Die meisten Anwendungen werden die Bibliothek mittels der High-Level-API benutzen, die in der Datei `api.c` realisiert ist. Mit den in `strings.c` enthaltenen Funktionen, können einige Konstanten und Zahlwerte aus ICC-Profilen in Textmeldungen übersetzt werden.

3.3 Untere Schicht

3.3.1 header.c

Den Header einer ICC-Profil-Datei kann man mit der in dieser Datei enthaltenen Funktion `cm_Get Header()` in eine Struktur im Hauptspeicher einlesen.

Da gemäß Standard Integer-Zahlen in einer ICC-Profil-Datei in Big-Endian Reihenfolge abgelegt sind und die entwickelte Bibliothek sowohl auf Big-Endian wie auch Little-Endian Systemen funktionieren soll, muß hier eine Endian-Umwandlung stattfinden. Dieses geschieht mit einem C-Makro, wobei dieses für 16 Bit Zahlen z.B. so aussieht:

```
#define ENDIAN16(a) (((a)[0] << 8) | (a)[1])
```

Das erste Byte im Speicher wird also um 8 Bits nach links verschoben. Die so entstandene 16 Bit Zahl wird mittels ODER mit dem zweiten Byte der Zahl verknüpft. Das Makro für 32 Bit Zahlen sieht entsprechend aus.

Neben Integer-Zahlen enthält der Header auch Festkommazahlen. Diese bestehen aus vier Bytes, wobei die ersten beiden Bytes die Vorkommastellen und das Vorzeichen und die letzten beiden Bytes die Nachkommastellen der Zahl repräsentieren. Mit folgendem Makro werden die vier Bytes in eine Fließkommazahl konvertiert, wobei gleich das Endian-Problem gelöst wird:

```
#define FIXED2FLOAT(a) ((((signed char)(a)[0]) << 8) + (a)[1] + \
                       ((double)((a)[2] << 8) + (a)[3]) / 65536.0)
```

Durch den Cast »(signed char)« wird dafür gesorgt, daß das Vorzeichen nicht verloren geht, während der Cast »(double)« Sorge dafür trägt, daß der C-Compiler die Nachkommazahlen in eine Fließkommazahl konvertiert, da es ansonsten bei der Division zu Rechenfehlern kommt.

Diese Makros werden nicht nur für den Header benötigt, sondern auch in den anderen Funktionen, die direkt das ICC-Profil auf Dateiebene lesen.

3.3.2 table.c

Die Funktion `cm_GetTagTable()` liest das Verzeichnis der in einem Profil enthaltenen Tags aus einer Profildatei ein und legt sie als Liste im Hauptspeicher ab. Gemäß dem objektorientierten Ansatz, den die Bibliothek verfolgt, greift der Programmierer auf die eingelesene Liste nicht direkt zu, sondern bedient sich dazu den Funktionen dieser Quelldatei.

Wenn die Liste nicht mehr benötigt wird, kann sie mittels `cm_FreeTagTable()` aus dem Hauptspeicher wieder entfernt werden. Einen bestimmten Tag kann mit der Funktion `cm_GetTag()` gesucht werden. Als Ergebnis liefert die Funktion die Position der Daten des Tags in der ICC-Profil-Datei und wieviel Bytes die Daten belegen.

Allerdings werden diese Daten wiederum nicht direkt benutzt, sondern der Programmierer liest die eigentlichen Datei des Tags mit der Funktion `cm_GetTagData()` in den Hauptspeicher ein.

3.3.3 math.c

Die bei der TRC- und LUT-Methode verwendeten eindimensionalen Tabellen enthalten in der Regel nicht alle Punkte des verwendeten Wertebereiches. Um Platz zu sparen, sind stattdessen nur einige Stützstellen im ICC-Profil enthalten. Der Funktion `cm_Interpolate()` können eine Anzahl solcher Stützstellen übergeben werden. Die Funktion erzeugt dann mittels Interpolation eine Kennlinie mit der gewünschte Bitauflösung in x- und y-Richtung.

Eine Veränderung der Auflösung in y-Richtung kann notwendig sein, um die Auflösung der Grafikdaten an die interne Auflösung der ICC-Bibliothek anzupassen.

Die Funktion verwendet eine linaere Interpolation. Dabei gilt es zu beachten, daß die Anzahl der Stützstellen nicht unbedingt eine Zweierpotenz sein muß; die Anzahl kann also auch ungerade sein. Außer-

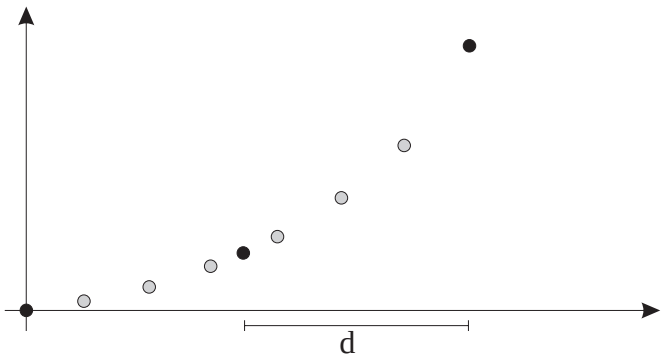

Abbildung 3.4: Interpolation

dem dürfen die Werte der ersten und der letzten Stützstelle nicht verändert werden, da es ansonsten bei der Anwendung der erzeugenden Kennlinie zu Rundungsverlusten kommen kann. Die Interpolation ist in Abbildung 3.4 dargestellt.

Die lineare Interpolation berechnet einen Wert unter der Berücksichtigung den zwei am nächsten liegenden Stützstellen. Die beiden jeweils benötigen Stützstellen müssen also ermittelt werden. Die Funktion berechnet dazu zuerst den Abstand d zweier Stützstellen in der Zielauflösung:

$$d = \frac{\mathrm{nr}_{t_x} - 1}{\mathrm{nr}_{s_x} - 1}$$

Hierbei sind nr_{s_x} bzw. nr_{t_x} die Anzahl der x-Werte im gegebenen bzw. im zu berechnenden System. Wenn eine Tabelle aus drei Stützstellen besteht und eine Auflösung von 3 Bit haben soll, würde man folgendes für den Abstand erhalten:

$$d = \frac{2^3 - 1}{3 - 1} = 3,5$$

Die eigentliche Interpolation erfolgt gemäß der Geradengleichung:

$$y = ax + b$$

Für a und b werden immer dann neue Werte berechnet, wenn der x-Wert des zu berechnenden Wertes größer ist als das Produkt $n\,d$, wobei n die zur Interpolation herangezogenen untere Stützstelle definiert. Die zweite Stützstelle ist einfach $n + 1$:

$$\begin{aligned} a &= \frac{s\,(\mathrm{data}[n+1] - \mathrm{data}[n])}{d} \\ b &= \mathrm{data}[n]\,s - n\,a\,d \\ n &= n + 1 \end{aligned}$$

Hierbei sorgt der Wert s für die Anpassung der Auflösung in y-Richtung. Berechnet wird s so:

$$s = \frac{\mathrm{nr}_{t_y} - 1}{\mathrm{nr}_{s_y} - 1}$$

Um Rundungsfehler zu vermeiden, testet die Funktion `cm_Interpolate()` vor der Interpolation, ob eine solche überhaupt notwendig ist. Sollte das nicht der Fall sein, werden die Daten unverändert zurückgegeben. Außerdem wird der Fall behandelt, daß die Quelltabelle mehr Stützstellen enthält wie überhaupt benötigt werden. Ist die Auflösung der Quelltabelle zu groß, wird diese vor der Interpolation zuerst reduziert.

Die zweite wichtige Funktion dieser Quelldatei ist `cm_CurveInverse()`. Auch sie führt eine Interpolation durch, allerdings wird die Kurve gleichzeitig invertiert. Benötigt wird diese Inversion der Tabellen für die TRC-Methode.

Die Funktion sucht bei der Inversion für jeden Zielwert die Stützstelle n, deren Funktionswert gerade größer ist als die x-Koordinate des Zielwertes. Die zweite Stützstelle ist $n - 1$. Zwischen den beiden Stützstellen wird dann wie bei der ersten Funktion linear interpoliert:

$$\begin{aligned} d &= \frac{\mathrm{nr}_{t_y} - 1}{\mathrm{nr}_{s_x} - 1} \\ a &= \frac{d}{\mathrm{data}[n] - \mathrm{data}[n-1]} \\ b &= d\,(n-1) - a\,\mathrm{data}[n-1] \end{aligned}$$

Problematisch ist bei der numerischen Interpolation theoretisch allerdings, daß sich einige der in den Profilen enthaltenen Tabellen im Prinzip nicht eindeutig invertieren lassen. Der entwickelte Algorithmus ignoriert dieses Problem einfach dadurch, daß er davon ausgeht, daß die Werte monoton wachsen. Dieses Verfahren hat sich im Rahmen dieser Studienarbeit auch bewährt.

Mit den Funktionen `cm_Gamma()` und `cm_GammaInverse()` wird eine Gamma-Kurve bzw. deren Inverse mit dem angegebenen Gamma-Wert und der angegebenen Auflösung in x- und y-Richtung erzeugt. Da die `pow()`-Funktion von C mit Fließkommazahlen arbeitet, was man oftmals garnicht benötigt, ist sie sehr langsam. Deswegen wurde die Funktion `cm_PowInt()` implementiert, die x^y für Ganzzahlen mittels einer Schleife berechnet.

3.3.4 pcs.c

Der ICC-Profil-Standard sieht als geräteunabhängige Farbräume CIE Lab und CIE XYZ vor. Es kann also bei der Konvertierung zwischen zwei geräteabhängigen Farbräumen vorkommen, daß die ICC-Profile nicht den gleichen geräteunabhängigen Farbraum benutzen. Für einen solchen Fall wird also ein Algorithmus benötigt, der die Daten zwischen CIE Lab und CIE XYZ konvertieren kann. Für die Transformation von XYZ nach Lab gilt [1]:

$$L = 116 \sqrt[3]{\frac{Y}{Y_n}} - 16$$

$$a = 500 \left[\sqrt[3]{\frac{X}{X_n}} - \sqrt[3]{\frac{Y}{Y_n}} \right]$$

$$b = 200 \left[\sqrt[3]{\frac{Y}{Y_n}} - \sqrt[3]{\frac{Z}{Z_n}} \right]$$

Diese Formel sind jedoch nur gültig, wenn $\frac{X}{X_n}$, $\frac{Y}{Y_n}$ und $\frac{Z}{Z_n} > 0,008856$ sind; sonst gilt:

$$L = 116 \left(7,787 \frac{Y}{Y_n} + 0,138 \right) - 16$$

$$a = 500 \left[\left(7,787 \frac{X}{X_n} + 0,138 \right) - \left(7,787 \frac{Y}{Y_n} + 0,138 \right) \right]$$

$$b = 200 \left[\left(7,787 \frac{Y}{Y_n} + 0,138 \right) - \left(7,787 \frac{Z}{Z_n} + 0,138 \right) \right]$$

Die Transformation von Lab nach XYZ ist dann so definiert:

$$X = \left(\frac{a}{500} + \sqrt[3]{\frac{Y}{Y_n}} \right)^3 X_n$$

$$Y = \left(\frac{L+16}{116} \right)^3 Y_n$$

$$Z = \left(\sqrt[3]{\frac{Y}{Y_n}} - \frac{b}{200} \right)^3 Z_n$$

X_n, Y_n und Z_n sind der XYZ-Wert des Weißpunktes.

Neben den beiden Transformationen enthält `pcs.c` die Implementation der in Abschnitt 2.3 behandelten Umrechnung von relativer nach absoluter farbmetrischer Optimierung. Mit der Funktion `cm_ColorError()` können schließlich der Farbabstände zwischen den Elementen zweier Arrays berechnet werden, wobei der erste Werte des zurückgelieferten Arrays den Mittelwert der Farbabstände enthält. Der Farbabstand wird gemäß DIN 6169 so berechnet [1]:

$$\Delta E = \sqrt{(\Delta L)^2 + (\Delta a)^2 + (\Delta b)^2}$$

3.4 Mittlere Schicht

3.4.1 color_trc.c

Funktionen zur Transformation zwischen geräteabhängigen und geräteunabhängigen Farbräumen nach der in Abschnitt 2.2 beschriebenen TRC-Methode wurden in dieser Quelldatei implementiert. Die Schnittstelle zur Außenwelt bilden die Funktionen `cm_ColorTRCInit()`, `cm_ColorTRCConvert()` und `cm_ColorTRCFree()`.

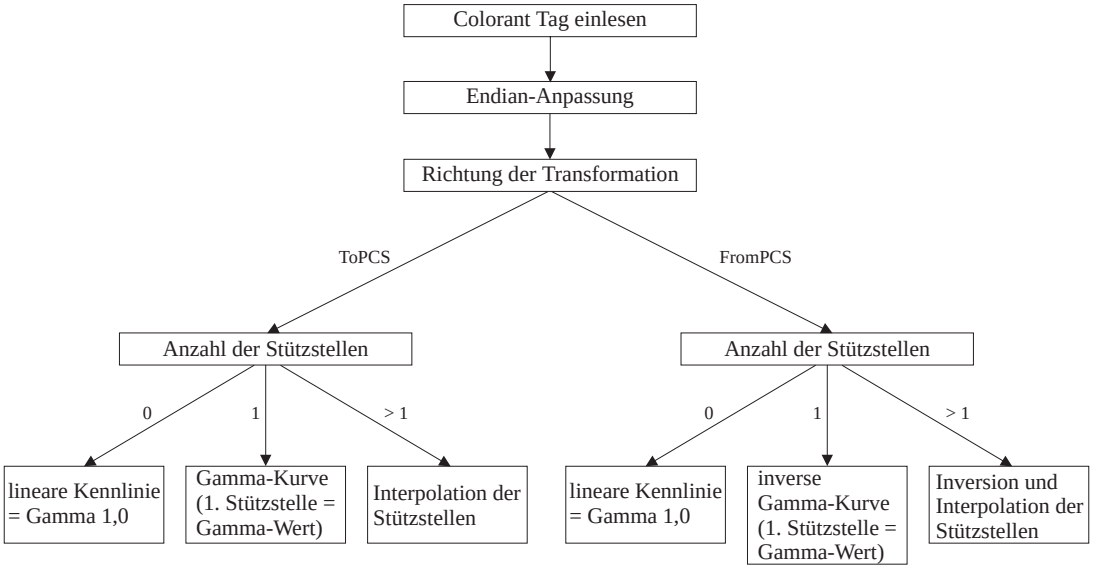

Abbildung 3.5: Flußdiagramm der Funktion `cm_GetTRCCurve()`

Die erste dieser Funktionen liest ein ICC-Profil ein und baut daraus die notwendigen Strukturen für die Transformation im Speicher auf. Als Ergebnis liefert die Funktion ein Handle zurück, daß den beiden anderen Funktionen übergeben werden muß.

Die eindimensionalen Tabellen für jeden Eingangsfarbkanal sind im ICC-Profil durch eine variable Anzahl von Stützstellen mit einer y-Auflösung von 16 Bit definiert. Im Hauptspeicher wird die Auflösung der x-Achse an die Bittiefe der zu verarbeitenden Grafikdaten angepaßt. Die Auflösung in y-Richtung bleibt gleich.

Soll die TRC-Methode verwendetet werden, um Grafikdaten von einem geräteunabhängigen in einen geräteabhängigen Farbraum zu transformieren, so müssen die Tabellen außerdem invertiert werden. Die Funktion `cm_ColorTRCInit()` führt diese Berechnungen nicht selbst durch, sondern bedient sich hierfür der Funktion `cm_GetTRCCurve()`. Diese wählt dann die jeweils passende Funktion aus, um aus einer Tabelle aus dem ICC-Profil eine Tabelle im Hauptspeicher zu erzeugen. Benutzt werden dazu die Funktion `cm_Interpolate()`, `cm_CurveInverse()`, `cm_Gamma()` und `cm_GammaInverse()`, die alle in der unteren Schicht implementiert sind; siehe Abschnitt 3.3.3. In Abbildung 3.5 ist das Flußdiagramm der Funktion `cm_GetTRCCurve()` zu sehen.

Auf jede so erzeugte Tabelle im Hauptspeicher zeigt ein 8 und ein 16 Bit Zeiger in dem Handle, das die Initialisierungsfunktion am Ende zurückliefert. Zwei Zeiger sind aus folgendem Grund notwendig: Bei der Konvertierungsrichtung unabhängige nach abhängige Koordinaten wird zuerst die 3x3 Matrix durchlaufen und dann die Linearsierungstabellen. Die Tabellen liefern also das Endergebnis. Der Programmierer, der die TRC-Funktionen benutzt, kann festlegen, welche Auflösung die geräteabhängigen Koordinaten haben sollen, wobei Werte zwischen einem und 16 Bit zulässig sind. Das führt folglich dazu, daß die Tabellen je nach gewünschter Ausgabeauflösung 8 oder 16 Bit Werte benutzen müssen.

Nachdem die Linearisierungstabellen nun im Speicher vorliegen, werden noch die 9 Werte der 3x3 Matrix eingelesen und im Handle gespeichert. Je nach Transformationsrichtung wird die Matrix außerdem invertiert.

Die eigentliche Konvertierung zwischen den Farbräumen wird dann mittels der Funktion `cm_Color TRCConvert()` durchgeführt, der dazu das erzeugte Handle und die Grafikdaten übergeben werden.

Da bereits während der Erzeugung des Handles alle notwendigen Interpolationen und Inversionen durchgeführt worden sind, verläuft die Konvertierung sehr schnell.

Bei der Konvertierung von geräteabhängigen in geräteunabhängigen Farbkoordinaten wird jeder Kanal einfach so berechnet:

```
xyz_pnt->x = (hd->redTRC16[data8[0]] * hd->XColorant[0] +
              hd->greenTRC16[data8[1]] * hd->XColorant[1] +
              hd->blueTRC16[data8[2]] * hd->XColorant[2]) /
              65535.0;
```

Die Eingangskanäle werden also linearisiert, dann mit einem Vektor aus der 3x3 Matrix multipliziert und schließlich auf den Wertebereich 0,0 bis 1,0 skaliert. Die Gegenrichtung sieht ungefähr so aus:

```
zw32 = (hd->redColorant[0] * xyz_pnt->x +
        hd->redColorant[1] * xyz_pnt->y +
        hd->redColorant[2] * xyz_pnt->z) * 65535.0
        + 0.5;
if (zw32 > 65535)
  data8[0] = hd->redTRC8[65535];
else if (zw32 < 0)
  data8[0] = hd->redTRC8[0];
else
  data8[0] = hd->redTRC8[zw32];
```

Der jeweilige XYZ-Wert wird zuerst mit einem Vektor aus der invertierten 3x3 Matrix multipliziert und dann wieder auf den Wertebereich einer 16 Bit Ganzzahl skaliert. Die Addition von 0,5 ist notwendig, da ANSI C Fließkommazahlen bei der Umwandlung in Ganzzahlen nicht rundet, sondern einfach die Nachkommastellen abschneidet. Der so entstandene Wert wird dann mit der für den Farbkanal passenden Tabelle an die Nichtlinearitäten des Ausgabefarbraums angepaßt.

Vor der Verwendung der Tabelle muß allerdings noch die Einhaltung des Wertebereiches einer 16 Bit Ganzzahl sichergestellt werden. Überläufe nach oben sind möglich, da XYZ-Werte manchmal den definierten Bereich von 0,0 bis 1,0 verlassen, was gemäß ICC-Profil-Standard auch beabsichtigt ist. Überläufe nach unten entstehen dadurch, daß die 3x3 Matrizen auch negative Koeffizienten enthalten können, so daß das Endergebnis negativ wird.

Wird ein Handle nicht länger benötigt, kann der durch das Handle belegt Hauptspeicher mit der Funktion `cm_ColorTRCFree()` wieder freigegeben werden.

3.4.2 gray_trc.c

Die TRC-Methode ist für RGB- und Graustufenfarbräume definiert. Die Routinen zur Behandlung von TRCs für Graustufen ist in der Quelldatei `gray_trc.c` implementiert. Das Programmierinterface besteht wie bei den farbigen TRCs aus drei Funktionen: `cm_GrayTRCInit()`, `cm_GrayTRCConvert()` und `cm_GrayTRCFree()`.

Die Initialisierung berechnet mit `GetTRCCurve()` eine Linearisierungskurve für den Helligkeitskanal. Eine 3x3 Mischmatrix gibt es bei dieser Methode nicht.

Mit `cm_GrayTRCFree()` kann der durch ein Handle belegte Speicherplatz wieder freigegeben werden.

3.4.3 lut.c

Eine Implementation der sehr komplexen LUT-Methode, die in Abschnitt 2.3 beschrieben wurde, findet sich in der Quelldatei `lut.c`. Auch bei dieser Implementation besteht die Programmierschnittstelle aus drei Funktionen: `cm_LUTInit()`, `cm_LUTConvert()` und `cm_LUTFree()`.

Die Funktion zur Initialisierung ermittelt als erstes den zur Optimierung der Farben und der Transformationsrichtung passenden Tag im ICC-Profil. Die Eingabe- und Ausgabetabellen, die e-Matrix und die Werte der mehrdimensionalen Tabelle, auch CLUT genannt, des so bestimmten Tags werden eingelesen. Hierbei gilt zu beachten, daß die Daten der LUT-Methode im ICC-Profil entweder mit 8 oder 16 Bit Auflösung abgelegt sein können. Die Tabellen werden interpoliert und mit folgenden Auflösungen im Hauptspeicher abgelegt:

LUT-Typ	Richtung	Eingabetabelle(Bit)		Ausgabetabelle (Bit)	
		x	y	x	y
8 Bit	nach CIE	Quelle	8	8	16
	von CIE	8	8	16	Ziel
16 Bit	nach CIE	Quelle	16	16	16
	von CIE	16	16	16	Ziel

Im Gegensatz zur TRC-Methode wird bei der LUT-Methode viel mit Fließkommazahlen gearbeitet, so daß trotz der bei der Initialisierung bereits interpolierten Tabellen auch bei der eigentlichen Transformation zwischen den Werten der Tabellen interpoliert werden muß, um den Fehler nicht zu groß werden zu lassen.

Die Funktion `cm_LUTConvert()` ruft je nach Transformationsrichtung die Funktion `cm_LUTConvert_ToPCS()` oder `cm_LUTConvert_FromPCS()` auf. Falls eine absolute farbmetrische Optimierung verlangt wird, führt die Funktion diese mittels `cm_XYZRel2Abs()` durch.

Wird ein erzeugtes LUT-Handle irgendwann nicht mehr benötigt, so kann der dadurch belegte Speicherplatz mit `cm_LUTFree()` wieder freigegeben werden.

Sehr problematisch bei der Implementation der Transformationsroutinen der LUT-Methode ist die Tatsache, daß der Standard für ICC-Profile [4] nicht genau definiert, welchen Wertebereich die geräteunabhängigen Farbräume im Profil verwenden. Es wurden deshalb die in Anhang A.1 beschriebenen Wertebereiche verwendetet. Der XYZ-Wertebereich sieht demnach so aus:

$$0,0 < x < 1 + \frac{32767}{32768}$$
$$0,0 < y < 1 + \frac{32767}{32768}$$
$$0,0 < z < 1 + \frac{32767}{32768}$$

Die Lab-Wertebereiche sind für 8 und 16 Bit LUT-Systeme unterschiedlich. Dieses sind die Wertebereich für ein 8 Bit System:

$$0,0 < L < 100,0$$

$$-128,0 < a < 127,0$$
$$-128,0 < b < 127,0$$

Ein 16 Bit System ist hingegen so definiert:

$$0,0 < L < 100,0 + \frac{25500}{65280}$$
$$-128,0 < a < 127,0 + \frac{255}{256}$$
$$-128,0 < b < 127,0 + \frac{255}{256}$$

Wie man sehr deutlich sehen kann, läßt der ICC-Profil-Standard auch Werte zu, die eigentlich keinen Sinn machen und nicht definiert sind.

FromPCS

Bei der Transformation geräteunabhängiger in geräteabhängige Farbkoordinaten wird als erstes überprüft, ob es sich um XYZ- oder Lab-Koordinaten handelt. Falls es sich um XYZ-Koordinaten handelt, wird der XYZ-Vektor mit der e-Matrix multipliziert. Der so entstandene XYZ-Vektor wird mit 32768,0 multipliziert, um den Wertebereich an die LUT-Methode anzupassen.

Bei einem Lab-Farbraum entfällt die e-Matrix. Die Anpassung an den Wertebereich der 8 Bit LUT-Methode sieht so aus:

$$L' = 2,55\,L$$
$$a' = a + 128,0$$
$$b' = b + 128,0$$

Für die 16 Bit LUT-Methode wird folgende Skalierung verwendet:

$$L' = 652,8\,L$$
$$a' = 257,0\,(a + 128,0)$$
$$b' = 257,0\,(b + 128,0)$$

Als nächster Schritt wird dafür gesorgt, daß die Werte von einer 8 bzw. 16 Bit Ganzzahl aufgenommen werden können. Dann wird jeder Wert mit den eindimensionalen Eingangstabellen verzerrt. Da die XYZ-Werte noch immer Fließkommazahlen sind, können die Werte jedoch nicht einfach in den Tabellen

»nachgeschlagen« werden. Vielmehr muß aus zwei Werten einer Tabelle der passende Wert interpoliert werden. Für die lineare Interpolation wird diese Formel verwendet [3]:

$$v_x = v_0 + f_x(v_1 - v_0)$$

Hierbei sind v_0 und v_1 die zwei Funktionswerte aus einer Tabelle, zwischen denen interpoliert werden soll. f_x beschreibt, wo der zu berechnende Wert zwischen den beiden x-Werten der Funktionswerte liegt, wobei $0 < f_x < 1$ gilt.

Nach der Verzerrung folgt dann die mehrdimensionale Tabelle, auch CLUT genannt, deren Stützstellen jedem XYZ / Lab-Wert einen Ausgangswert in dem Zielfarbraum zuordnen. Hier findet also die eigentliche Transformation zwischen den Farbräumen statt. Auch hier muß wieder zwischen den Stützstellen zur Laufzeit interpoliert werden. Der Standard für ICC-Profile schreibt kein bestimmtes Interpolationsverfahren vor. Für die Implementation wurde eine tri-lineare Interpolation [3] gewählt:

$$\begin{aligned}
v_{x00} &= v_{000} + f_x(v_{100} - v_{000}) \\
v_{x01} &= v_{001} + f_x(v_{101} - v_{001}) \\
v_{x10} &= v_{010} + f_x(v_{110} - v_{010}) \\
v_{x11} &= v_{011} + f_x(v_{111} - v_{011})
\end{aligned}$$

$$\begin{aligned}
v_{xy0} &= v_{x00} + f_y(v_{x10} - v_{x00}) \\
v_{xy1} &= v_{x01} + f_y(v_{x11} - v_{x01})
\end{aligned}$$

$$v_{xyz} = v_{xy0} + f_z(v_{xy1} - v_{xy0})$$

Die Variablen haben die gleiche Funktion wie die bei der linearen Interpolation. Vor der eigentlichen Interpolation müssen die passenden acht Stützstellen (v_{000} bis v_{111}) ermittelt werden. Hierfür wird zuerst der Abstand zweier Stützstellen ermittelt:

$$d = \frac{\mathrm{nr}_{t_x} - 1}{\mathrm{nr}_{s_x} - 1}$$

nr_{s_x} sind hierbei die Anzahl der Stützstellen in eine Richtung. nr_{t_x} beschreibt die Zielauflösung. Bei 8 Bit wäre $\mathrm{nr}_{t_x} = 256$. Teilt man den zu verarbeitenden XYZ-Vektor durch d und speichert das Ergebnis als Ganzzahl, so erhält man die Position der Ausgangsstützstelle v_{000}. Neben der Position wird dann noch der Abstand des zu berechnenden Punktes von der Ausgangsstützstelle berechnet (f_x, f_y und f_z).

Die tri-lineare Interpolation muß für jeden Ausgangsfarbkanal pro Pixel einmal durchlaufen werden. Wird also z.B. Lab nach CMYK transformiert, finden pro Pixel vier tri-lineare Interpolation statt. Dieses ist natürlich ein ziemlicher Rechenaufwand und ist daher auch der Grund, warum die LUT-Methode soviel langsamer ist im Vergleich mit der TRC-Methode.

Das Ergebnis der Interpolation wird, falls es sich um den 8 Bit LUT-Typ handelt, mit 257,0 multipliziert, so daß die 8 und 16 Bit Methode beide den gleichen Wertebereich verwenden. Es wird dann noch einmal geprüft, ob der Wertebereich eingehalten wird.

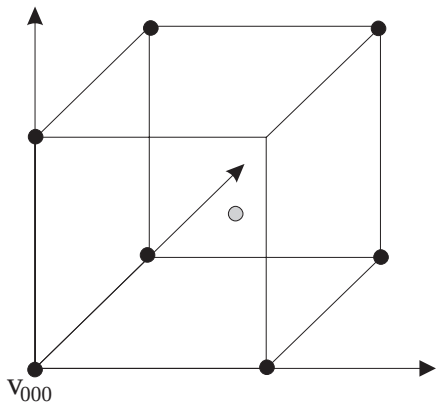

Abbildung 3.6: Tri-lineare Interpolation mit 8 Stützstellen

Schließlich wird jeder Ausgangskanal noch einmal mit einer eindimensionalen Tabelle behandelt. Bei den Ausgabetabellen ist keine Interpolation während der Transformation notwendig, da die Tabellen diskrete Ausgangswert (Ganzzahlen) liefern und deshalb bereits bei der Initialisierung passend interpoliert werden können.

ToPCS

Die Gegenrichtung der LUT-Methode ist im Prinzip genauso implementiert: Eingabetabellen, CLUT und Ausgabetabellen. Die e-Matrix entfällt.

Ein deutlicher Unterschied ergibt sich allerdings bei der Interpolation der CLUT. Während der Eingangsfarbraum vorher immer drei Dimensionen hatte, sind jetzt bis zu 15 erlaubt. Das bedeutet, daß eine Interpolation benötigt wird, die mit einer variablen Anzahl von Koordinaten umgehen kann. Hierfür wurde analog zur Herleitung der tri-lineare aus der lineare Interpolation [3] die Anzahl der Koordinaten erhöht.

Als erstes werden die Eingangstabellen angewendet. Die so entstandenen Ergebnissen werden durch die Distanz geteilt, um die Position der Hauptstützstelle im CLUT zu ermitteln. Dann werden mit

```
for (i = 0; i < cm_PowInt (2, hd->nr_input_ch); i++)
{
  for (i2 = 0; i2 < hd->nr_input_ch; i2++)
    add[i2] = (i & (cm_PowInt (2, hd->nr_input_ch-1) >> i2))  >>
              (hd->nr_input_ch - i2 -1);
  old[i] = hd->tb_clut8 [cm_ArrayPos (hd, pnt_pos, add) + color_o];
}
```

die Werte der Stützstellen, die für die Interpolation benötigt werden, in das Array `old[]` kopiert. Die Anzahl der benötigten Stützstellen ist gleich 2 hoch Anzahl der Eingangsfarbkanäle.

Um die Werte der Stützstellen ermitteln zu können, werden in dem Array `add[]` Nullen und Einsen abgelegt, die den Abstand der gesuchten Stützstelle von der Hauptstützstellen festlegen. Wenn die Hauptstützstelle z.B. (x, y, z) ist und die Stützstelle $(x + 1, y + 1, z)$ gesucht ist, so sähe das Array so

aus: $(1, 1, 0)$. Die Funktion `cm_ArrayPos()` besorgt dann mit diesem Array den Funktionswert der Stützstelle aus dem Handle.

Nach dieser Initialisierung der Arrays `old[]` mit den Stützstellen folgt dann die eigentliche Interpolation:

```
for (color_i = 0; color_i < hd->nr_input_ch; color_i++)
{
  f_i = pnt_zw[color_i] / hd->distance - pnt_pos[color_i];

  for (i = 0; i < cm_PowInt (2, hd->nr_input_ch - color_i - 1); i++)
    new[i] = old[i] + f_i * (old[i +
             cm_PowInt (2, hd->nr_input_ch - color_i - 1)] - old[i]);
  pnt_f = old;
  old = new;
  new = pnt_f;
}
zw = old[0];
```

Diese Schleife durchläuft jede Eingangsfarbkoordinate und halbiert dabei jeweils die Anzahl der Werte im Array `old[]`, bis schließlich nur noch ein Wert vorhanden ist: der gesuchte interpolierte Wert.

Nach der Interpolation folgt dann wie bei der Gegenrichtung die Wertebereichsprüfung, die Skalierung und die Ausgangstabelle für jeden Farbkanal.

3.4.4 add_tag.c

Diese Quelldatei kann einige Tags einlesen und verarbeiten, die nicht direkt der Transformation dienen, sondern vor allem Informationen für den Anwender bereithalten: Datum der Kalibrierung, Copyright, Hersteller und Typ des kalibrierten Gerätes, Luminance, Weiß- und Schwarzpunkt und Betrachtungsbedingungen.

Gerade die Beschreibung der Betrachtungsbedingungen ist sehr wichtig, wenn auch viele Profile diese nicht enthalten, da jedes Profil nur für eine Umgebung (Beleuchtung, Papierfarbe, Papierart, etc.) gilt.

3.5 Obere Schicht

3.5.1 api.c

Da der Standard für ICC-Profile nicht vorschreibt, ob ein Profil die TRC- oder LUT-Methode oder vielleicht sogar beide enthalten muß, ist es in der Regel wenig sinnvoll, daß der Programmierer einer Anwendung direkt auf die Transformationsroutinen der mittleren Schicht zurückgreift. Viel besser geeignet sind dafür die Routinen, die in der Quelldatei `api.c` implementiert sind.

Auch diese Quelldatei besteht aus drei Routinen für Initialisierung, Konvertierung und Freigabe des belegten Speichers. Mit der Initialisierung muß für jedes ICC-Profil, das bei der Transformation verwendet werden soll, ein Handle erzeugt werden. Ein Handle beschreibt jeweils eine Transformation zwischen geräteabhängigen Farbsystem zu einem geräteunabhängigen Farbsystem oder umgekehrt.

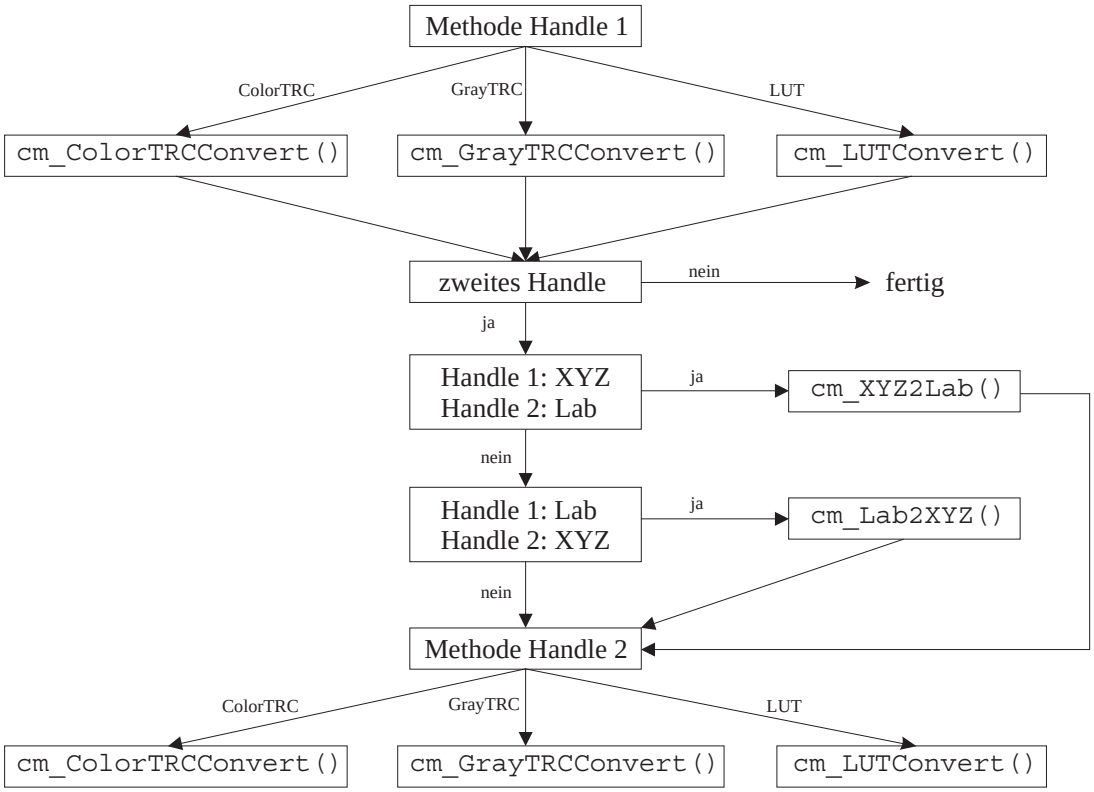

Abbildung 3.7: Flußdiagramm der Transformationsfunktion `cm_Convert()`

Bei der Initialisierung kann die Auflösung der Grafikdateien in Bit, die Optimierung, die Qualität und die Richtung der Transformation angegeben festgelegt werden.

An Hand der gewünschten Qualität der Transformation und der in dem verwendeten ICC-Profil enthaltenen Tags wählt die Bibliothek dann, ob die TRC- oder LUT-Methode benutzt wird. Reicht eine niedrige Qualität aus, wird in der Regel die TRC-Methode verwendet. Nur wenn das Profil keine Werte für die TRC-Methode enthält, wird die LUT-Methode benutzt. Bei einer hohen Qualität geht die Bibliothek genau entgegengesetzt vor. Es wird dann also die LUT-Methode bevorzugt verwendet.

Je nach gewählter Methode wird dann mit der Funktionen zur Initialisierung aus der mittleren Schicht ein passendes Handle erzeugt.

Die Funktion `cm_Convert()` führt die Konvertierung durch; siehe Abbildung 3.7. Es können der Funktion ein oder zwei mit `cm_Init()` erzeugte Handles übergeben werden. Das erste Handle wird in der Regel einen Scanner und das zweite, optionale Handle einen Monitor oder Drucker beschreiben. Intern ruft die Funktion die passenden Transformationsroutinen aus der mittleren Schicht auf. Falls die beiden Handles unterschiedliche geräteunabhängige Farbräume verwenden, findet eine Umrechnung zwischen diesen statt.

Nach getaner Arbeit werden mit `cm_Free()` die Handles gelöscht.

3.5.2 strings.h

Oftmals möchte eine Anwendung dem Benutzer einige Informationen zu den verwendeten ICC-Profilen anzeigen. Hierbei besteht jedoch das Problem, daß diese Informationen im Profil in der Regel mittels numerischer Konstanten oder gesetzten Bits abgelegt sind, mit denen der Benutzer nichts anfangen kann.

Die in der Quelldatei `strings.h` enthaltenen Funktionen können einige dieser Konstanten in für den Anwender verständliche Zeichenketten umwandeln. So wird z.B. aus dem Wert 0x73636E72 mit Hilfe der Funktion `cm_GetStringProfileClass()` die Zeichenkette »Input Device Profile«.

Durch die Verwendung des `gettext()`-Systemes, über das Linux und die meisten modernen Unix-Systeme verfügen, unterstützen die implementierten Funktionen beliebig viele Sprachen. Zur Zeit stehen die Texte in Englisch und Deutsch zur Verfügung.

3.6 Anwendungen

Neben der Bibliothek für das Farbmanagement mit ICC-Profilen wurden zwei Anwendungen erstellt, die die Verwendung dieser Bibliothek demonstrieren und zur Untersuchung der Qualität der Profile benutzt werden könnnen. Diese Anwendungen unterstützen wie die Bibliothek selbst auch eine Lokalisierung mittels `gettext()`.

3.6.1 iccinfo

Da ein Gerät mit mehreren Profilen ausgeliefert werden kann, ist es sehr praktisch, zuerst einmal einige Informationen über das Profil zu erhalten. Genau dieses erlaubt das Programm `iccinfo`.

Es liefert bei einem Aufruf eine solche Kurzbeschreibung:

```
Header:
  Profile Size: 3144 bytes
  CMM Type: 0x4C696E6F
  ICC Version: 2.1.0
  Profile/Device Class: Display Device Profile
  Color Space: RGB
  Profile Connection Space: XYZ
  Created: 1998/2/9 @ 6:49:00
  Primary Platform: Microsoft
  Device Manufacturer: 0x49454320
  Device Model: 0x73524742
  Attributes: Reflective, Glossy
  Rendering Intent: 0x0
  Illuminant: 0.964203 1.000000 0.824905
  Creator: 0x48502020
Device:
   Manufacturer: IEC http://www.iec.ch
   Model: IEC 61966-2.1 Default RGB colour space - sRGB
Copyright: Copyright (c) 1998 Hewlett-Packard Company
```

```
Media:
   White Point: 0.950455 1.000000 1.089050
   Black Point: 0.000000 0.000000 0.000000
Technology: CRT Display
Viewing Condition: Reference Viewing Condition in IEC61966-2.1
```

Wie man sehen kann, handelt es sich um ein Profil für einen Monitor, der den RGB-Farbraum verwendet. Als geräteunabhängiger Farbraum werden XYZ-Koordinaten benutzt. Gedacht ist das Profil für MS-Windows. Dieses ICC-Profil wurde von Hewlett-Packard erstellt und beschreibt den von dieser Firma geschaffenen Farbraum sRGB [14].

Dieses Programm ist ein gutes Beispiel für die Verwendung der Funktionen aus `strings.c`. Auch von diesem Programm werden mehrere Sprachen unterstützt.

3.6.2 icctiff

Zur Zeit wird ein Farbmanagement mit ICC-Profilen von keinem Programm oder Treiber für Linux unterstützt. Mit dem Programm `icctiff` kann die Zeit überbrückt werden, bis die Programme und Treiber diese Implementation des Farbmanagementes direkt benutzen.

Das Programm `icctiff` konvertiert die in Form einer TIFF-Datei übergebenen Grafikdaten mittels eines oder zweier ICC-Profile von einem Farbraum in einen anderen. Unterstützt werden dabei die Farbräume RGB, CMYK und Lab. Man kann die Grafikdaten also nicht nur zwischen zwei geräteabhängigen Farbräumen transformieren, sondern es ist auch z.B. möglich, eine TIFF-Datei im RGB-Format in eine TIFF-Datei im Lab-Format umzuwandeln.

Eine solche TIFF-Datei im Lab-Format wäre eigentlich ideal für den Datenaustausch geeignet. Wie Tests allerdings gezeigt haben, können die bekannten Grafikprogramm für Linux, wie z.B. The Gimp, xv und Electric Eyes, mit TIFF-Datei, die den Lab-Farbraum benutzen, nicht umgehen.

Auch der CMYK-Farbraum wird nur von Electric Eyes unterstützt. Hierbei gilt jedoch zu beachten, daß diese Programme intern weiterhin mit einem RGB-Farbraum arbeiten, selbst wenn sie Daten im CMYK-Farbraum lesen können. Es findet also in der Anwendung wiederum ein Konvertierung von CMYK nach RGB statt, was natürlich einem Farbmanagement im Wege steht.

Von daher ist es wenig sinnvoll, eingescannte Vorlagen mit `icctiff` in den Druckerfarbraum zu transformieren und dann mit einem normalen Grafikprogramm wie Electric Eyes auf einem Drucker auszugeben. Um die Umrechnungen in den Anwendungen bei einer Ausgabe auf einem Drucker zu umgehen, bietet `icctiff` die Möglichkeit, das Ergebnis statt in einer TIFF-Datei direkt in einer PostScript-Datei abzulegen, wobei dann für die Ausgabe die Farbräume RGB und CMYK zur Verfügung stehen.

Einen Nachteil hat aber auch diese Vorgehensweise. Die Farben der PostScript-Datei sind an einen bestimmten Druckprozeß angepaßt. Es ist also nicht sinnvoll, die so erzeugte PostScript-Datei auf einem anderen Drucker auszugeben oder einfach nur eine andere Papierart zu verwenden. Dieses widerspricht eigentlich dem Konzept von PostScript, nach dem jede PostScript-Datei auf jedem Drucker das gleiche Ergebnis erzeugen sollte, soweit dieses technisch möglich ist.

Für zukünftige Versionen des Programmes könnte es von daher sinnvoll sein, auch bei PostScript direkt den Lab-Farbraum zu unterstützen, so daß die Anpassung an den geräteabhängigen Farbraum des Druckers in diesem selbst stattfindet.

Kapitel 4

Ergebnisse

Die Qualität der entwickelten Bibliothek für ein Farbmanagement mit ICC-Profilen und die Qualität der Profile selbst wird in diesem Kapitel untersucht.

4.1 ICC-Profile in der Praxis

Als erstes wurde der Aufbau einiger ICC-Profile untersucht, um vorab möglichen Ursachen für eine schlechte Qualität der vom Farbmanagementsystem produzierten Ergebnisse zu finden.

Da für die Studienarbeit zwei Scanner von HP zur Verfügung standen, wurde als erstes das Profil für HP ScanJets näher betrachtet. Das von HP mitgelieferte Profil enthält Daten für die TRC- und die LUT-Methode. Die drei Linearisierungskurven für die drei RGB-Kanäle sind in Abbildung 4.1 zu finden. Das Profil definiert jede Kurve durch 256 Stützstellen. Die für die Gegenrichtung benötigte von der Bibliothek invertierte Kurve ist in Abbildung 4.2 zu sehen.

Die Mischmatrix wird von diesem Profil so definiert:

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} 0.479507 & 0.211273 & 0.279984 \\ 0.278992 & 0.526093 & 0.200272 \\ 0.065887 & 0.004135 & 0.840454 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

Vergleicht man diese Matrix mit der Transformationsmatrix für den CIE RGB-Farbraum, wie sie in Abschnitt 2.1 beschrieben wurde, so sind doch größe Unterschiede erkennbar. Diese Unterschiede machen sehr schön deutlich, warum überhaupt ein Farbmanagement benutzt werden muß.

Für die LUT-Methode enthält das Scannerprofil, wie es auch der Standard vorsieht, nur die Daten für die Transformation von geräteabhängigen nach geräteunabhängigen Koordinaten und diese auch nur für eine Optimierung.

Die Eingangstabellen der LUT-Methode sind für alle drei Farbkanäle linear und zeigen keine Clipping-effekte; die Tabellen haben folglich bei diesem Profil keine Funktion.

Die CLUT besteht aus 8 Stützstellen je Farbkanal, was recht wenig ist. Beim L-Kanal wird der gesamte Wertebereich benutzt, während die a- und b-Kanäle nur Werte aus dem Bereich 57 bis 184 nutzen. Die

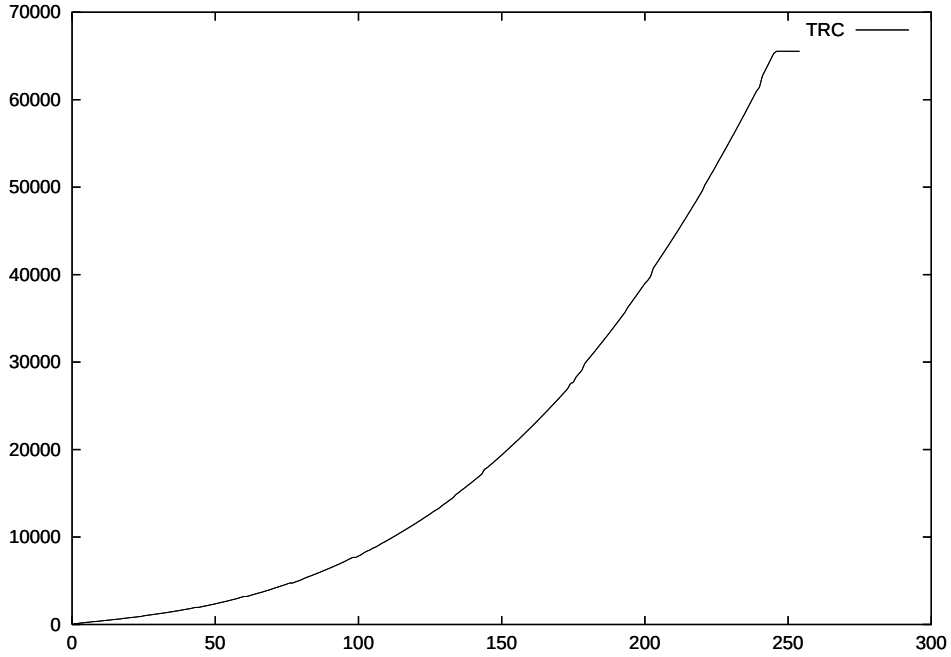

Abbildung 4.1: HP ScanJet: TRC-Kurve

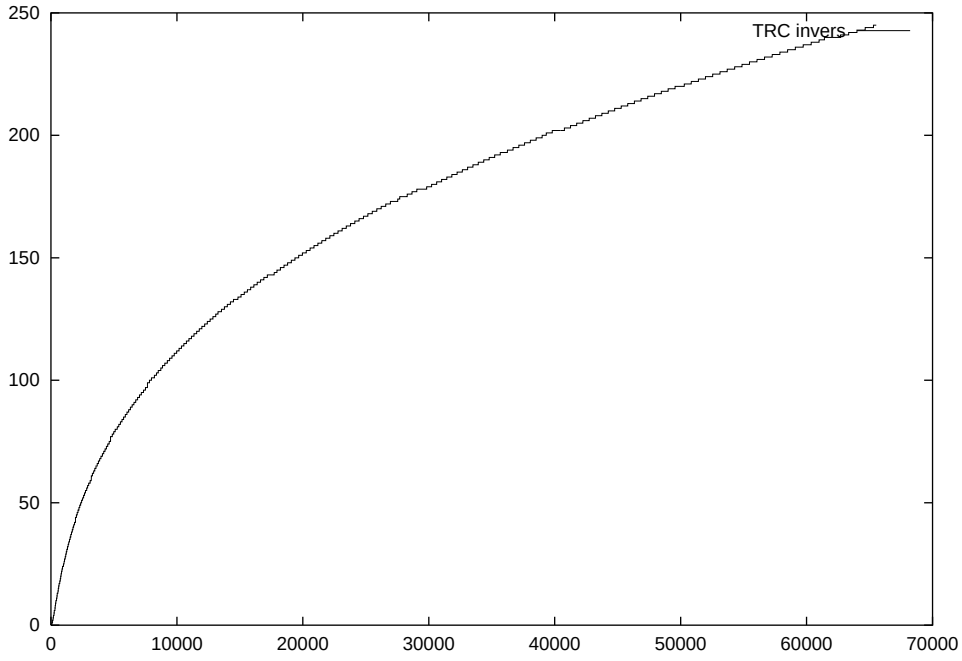

Abbildung 4.2: HP ScanJet: invertierte TRC-Kurve

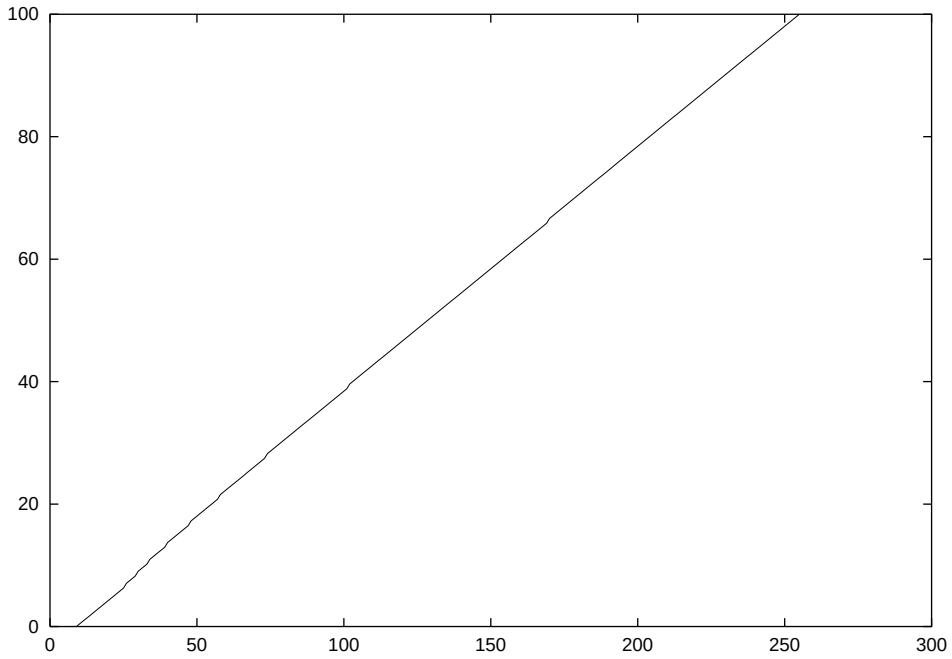

Abbildung 4.3: HP ScanJet: LUT-Ausgangstabelle für den L-Kanal

beiden letzten Kanäle arbeiten also mit einer verringerten Auflösung von 7 statt 8 Bit. Ein Effekt, der auch bei dem Profil für HP DeskJet Drucker beobachtet wurde; siehe Abbildung 2.1.

Auch die Ausgangstabellen nutzen nicht den gesamten Wertebereich aus. In den Abbildungen 4.3 und 4.4 sind die Ausgangstabellen des HP ScanJet Profiles dargestellt.

Insgesamt gewinnt man bei dem Profil den Eindruck, daß der Hersteller sich nicht sehr viel Mühe gegeben hat. Es ist zu vermuten, daß HP lieber seinen eigenen sRGB-Standard [14] unterstützt sehen möchte.

Als Drucker wurde für diese Studienarbeit ein Tektronix Phaser 740 verwendet. Sein ICC-Profil ist deutlich umfangreicher im Vergleich zu dem vom HP ScanJet. Dieses ist auch bereits an der Dateigröße ablesbar. Während das ScanJet Profil keine 10 KByte belegt, sind es beim Profil des Phasers bereits über 1 MByte.

Die Daten für die LUT-Methode liegen mit einer relativ hohen Auflösung vor. So besteht die CLUT in Ausgaberichtung aus 33 Stützstellen pro Koordinate. Für die andere Richtung, die zur Simulation des Druckers auf dem Bildschirm benutzt werden kann, werden dagegen nur 17 Stützstellen definiert. Die Ein- und Ausgabetabellen sind bis auf kleine Abweichungen linear und verwenden den gesamten Wertebereich, der zur Verfügung steht.

4.2 Farbabstand

Um die Qualität der Bibliothek in Verbindung mit den Profilen der verwendeten Geräte zu testen, wurden der Farbabstand und der Farbwiedergabe-Index bestimmt. Hierzu wurden die 17 CIE-Testfarben gemäß DIN 6169 verwendet [7]. Die Farben sind so definiert:

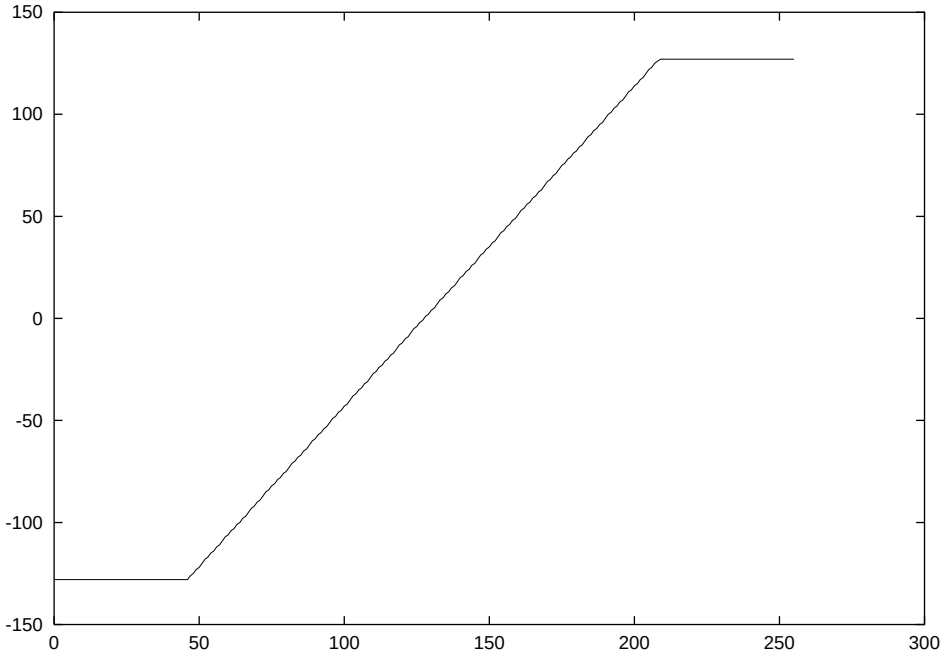

Abbildung 4.4: HP ScanJet: LUT-Ausgangstabelle für die a- und b-Kanäle

Nr.	X	Y	Z	L	a	b
1	0,3298	0,2976	0,2459	61,45	17,53	11,74
2	0,2749	0,2890	0,1501	60,69	0,08	28,92
3	0,2393	0,3043	0,0996	62,02	-20,59	44,41
4	0,2045	0,2948	0,2127	61,20	-33,17	17,07
5	0,2502	0,3087	0,4042	62,40	-17,48	-8,56
6	0,2826	0,2983	0,5791	61,51	-0,37	-28,40
7	0,3333	0,2939	0,5322	61,12	20,15	-24,56
8	0,3757	0,3131	0,4544	62,77	27,42	-13,64
9	0,2048	0,1120	0,0436	39,92	58,74	27,99
10	0,5487	0,5894	0,1208	81,26	-2,90	71,56
11	0,1212	0,2035	0,1533	52,23	-42,43	13,60
12	0,0628	0,0647	0,2773	30,57	1,41	-46,48
13	0,5885	0,5709	0,4139	80,23	11,37	21,04
14	0,0935	0,1171	0,0543	40,75	-13,81	24,23
15	0,0342	0,0359	0,0394	22,27	0,12	-0,17
16	0,1885	0,1983	0,2157	51,65	0,00	0,04
17	0,7239	0,7615	0,8289	89,93	0,02	0,03

Das Programm `din6169` wurde verwendet, um eine PostScript-Datei zu erzeugen, die eine Testtafel mit obigen Farben enthält. Eine solche Tafel ist in Abbildung 4.5 dargestellt. Um die Tafel zu erzeugen, transformiert das Programm die obigen Werte mit dem ICC-Profil des Druckers in dessen Farbraum und legt diese Werte dann direkt im RGB- bzw. CMYK-Farbraum in einer PostScript-Datei ab. Wie schon bei dem Programm `icctiff` war es notwendig, die PostScript-Datei direkt zu erzeugen, um jegliche zusätzliche Fehler durch irgendwelche internen Umrechnungen in den Anwendungen zu vermeiden.

Die so für die verschiedenen Optimierungen erzeugten PostScript-Dateien wurden dann auf einem Tektronix Phaser 740 ausgegeben.

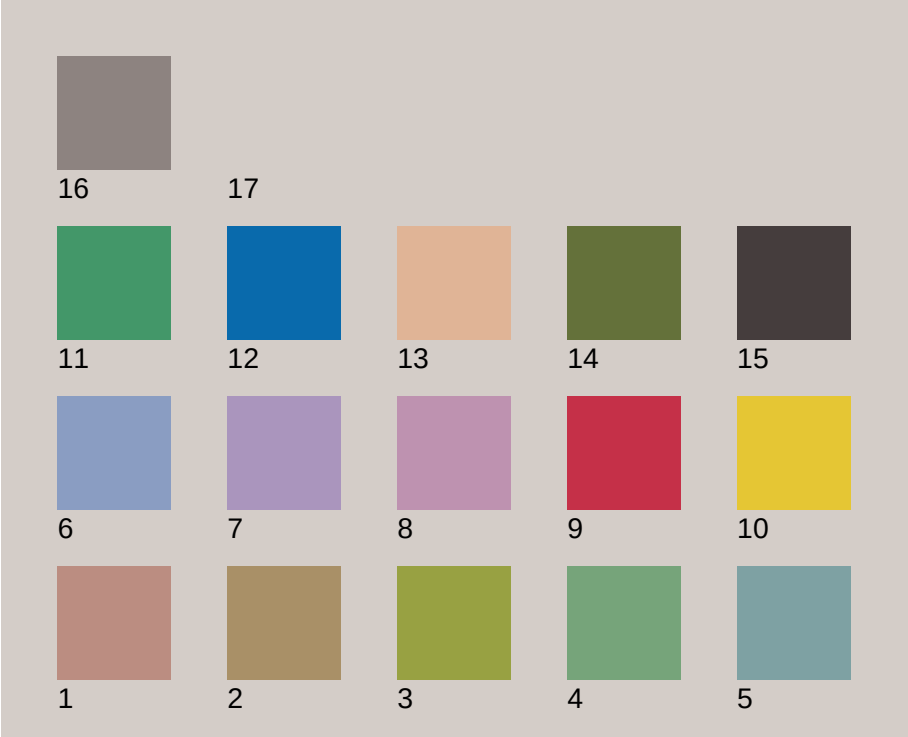

Abbildung 4.5: 17 CIE-Testfarben nach DIN 6169

Mit einem HP ScanJet wurden die so erstellten Ausdrucke wieder in den Rechner eingelesen. Hierbei wurde allerdings nicht der Treiber von HP für MS-Windows benutzt, da zu vermuten ist, daß dieser die Rohdaten des Scanners bereits verändert. Vielmehr fand das Linux-Programm `hpscanpbm` Verwendung, das im Source Code vorliegt, so daß sichergestellt werden konnte, daß hier wirklich die Rohdaten geliefert werden.

Um die Istwerte der eingescannten Tafeln mit den Sollwerten vergleichen zu können, wurden die Dateien mit den Tafeln in The Gimp geladen und die Istwerte mittels der Pipette ermittelt. Problematisch ist hierbei, daß die Pipette immer nur genau einen Pixel ausmißt, was durch die Rasterung des Druckers und dem Rauschen des Scanners zu sehr großen Abweichungen innerhalb eines Farbfeldes führt. Deshalb wurden die einzelnen Felder vor dem Ausmessen mit einem Unschärfefilter behandelt, um nicht den Wert eines einzelnen Pixels sondern den Mittelwert aller Pixel zu messen.

Die so ermittelten Istwerte wurden dann mit dem Programm `din6169_res` und dem Profil des Scanners in den Lab-Farbraum transformiert und mit den Sollwerten verglichen. Dabei wurde für jeden Farbwert der Farbstand nach dieser Formel berechnet:

$$\Delta E = \sqrt{(\Delta L)^2 + (\Delta a)^2 + (\Delta b)^2}$$

Aus den einzelnen Farbabständen wurde dann ein Mittelwert berechnet, aus dem dann der Farbwiedergabe-Index folgt:

$$R_m = 100 - 4,6\,\Delta E$$

Für den Prozeß Tektronix Phaser 740 mit HP ScanJet 4p ergeben sich dann folgende Fehler für die verschiedenen Optimierungen:

Optimierung	Farbabstand
fotografisch	29,39
relativ farbmetrisch	28,77
sättiungsbasiert	33,08
absolut farbmetrisch	28,59

Gemäß [1] läßt sich der Farbabstand so bewerten:

ΔE	Bewertung
bis 0,2	nicht wahrnehmbar
0,2 bis 0,5	sehr gering
0,5 bis 1,5	gering
1,5 bis 3,0	deutlich
3,0 bis 6,0	sehr deutlich
6,0 bis 12,0	stark
über 12,0	sehr stark

Es liegt folglich bei allen vier Optimierungen ein sehr starker Unterschied vor, der auch einem ungeschultem Beobachter sofort auffallen würde.

Es stellt sich folglich die Frage, wodurch diese sehr starken Abweichungen verursacht werden und in wie weit diese Abweichungen normal sind. In [13] wurden einige Profiscanner getestet, die durchschnittliche Farbabstände von 6,5 bis 10,0 aufwiesen. Die maximalen Farbabstände lagen bei 28,2 bis 39,5. Der für die Tests verwendete HP ScanJet 4p ist im Gegensatz zu den getesteten Geräten schon einige Jahre alt und auch eher im Consumerbereich angesiedelt.

Bedacht werden muß außerdem, das im Rahmen dieser Studienarbeit ja nicht der Farbabstand eines einzelnen Gerätes gemessen wurde. Vielmehr wurde der Farbabstand von Drucker und Scanner gemeinsam ermittelt.

Viele Hersteller kalibrieren mit ihren ICC-Profilen anscheinend nicht nur das Gerät sondern die Kombination aus Treiber und Gerät, wobei auch bereits im Treiber Manipulation an den Rohdaten durchgeführt werden. Unterstützt wird diese These durch die Tatsache, daß der Windows-Treiber für den ScanJet die Möglichkeit bietet, den Scan ohne oder mit einer Gamma-Korrektur durchzuführen.

Um dieser Vermutung genauer nachzugehen, wurde das Programm `din6169_res` dann so umgeschrieben, daß es vor der Transformation der Rohdaten in den Lab-Farbraum eine Gamma-Korrektur durchführt. Es wurden dann verschiedene Werte für die Gamma-Korrektur ausprobiert.

Wie man sehen in Abbildung 4.6 sehen kann, kann durch eine simple Gamma-Korrektur bei allen vier Optimierungen eine deutliche Verbesserung des Farbabstandes erreicht werden. In der nachfolgenden Tabelle sind die minimalen Farbabstände und die Gamma-Werte, bei denen diese auftreten, aufgeführt:

Optimierung	Gamma-Korrektur	Farbabstand	R_m
fotografisch	2,44	13,27	38,95
relativ farbmetrisch	2,56	13,35	38,59
sättigungsbasiert	2,57	15,15	30,33
absolut farbmetrisch	2,70	13,59	37,49

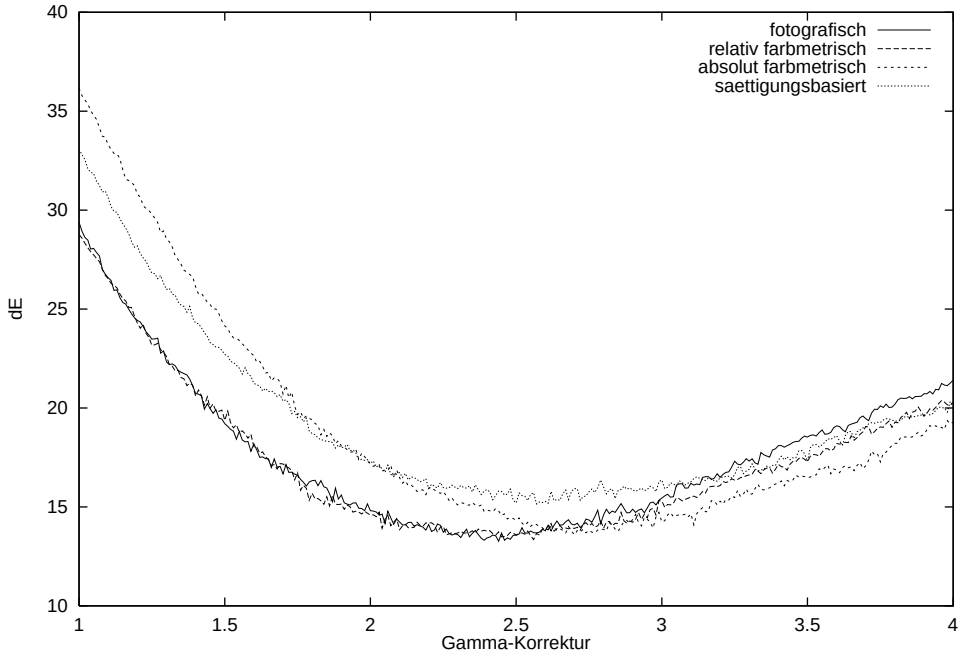

Abbildung 4.6: Gamma-Korrektur der 17 CIE-Testfarben

Die Vermutung, daß das ICC-Profil des Scanners nicht für die Rohdaten des Scanners gedacht ist, hat sich also mehr oder weniger bestätigt. Das mit dem Gerät gelieferte ICC-Profil funktioniert folglich nur mit Einschränkungen unter anderen Betriebssystemen bzw. mit anderen Treibern, die nicht direkt vom Hersteller stammen. Die Trennung zwischen Ansteuerung der Hardware durch einen Treiber und das Farbmanagement durch ein ICC-Profil ist hier also noch nicht vollzogen worden. Der Treiber verändert weiterhin die Rohdaten des Scanners und übernimmt damit Teile des Farbmanagements.

Die Fehler der einzelnen Farbwerte sind in Abbildung 4.7 für die vier Optimierungen grafisch dargestellt. Hierbei wurden die Rohdaten des Scanners vor der Transformation in den Lab-Farbraum einer Gamma-Korrektur unterzogen, wobei die vorher berechneten optimalen Gamma-Werte benutzt wurden.

4.3 Visuelle Kontrolle

Nach der Bestimmung des Farbabstandes erfolgte eine visuelle Kontrolle der mit der entwickelten Bibliothek und den vorhandenen ICC-Profilen erreichbaren Qualität. Hierzu wurden Fotos auf glänzendem Papier mit dem HP ScanJet 4p eingelesen und auf dem Sony Monitor PS-200 und dem Drucker Tektronix Phaser 740 ausgegeben. Für die Farbanpassung wurde das entwickelte Programm `icctiff` und die mit den Geräten gelieferten ICC-Profile verwendet.

Fotos auf Fotopapier wurden deshalb benutzt, da die von den Herstellern zur Kalibrierung verwendeten Vorlagen wie z.B. IT-8 [2] in der Regel auf glänzendem Fotopapier vorliegen und deshalb das ICC-Profil im Prinzip auch nur für solches Papier gültig ist.

Alle Bilder wurden jeweils mit und ohne Gamma-Korrektur und mit und ohne ICC-Farbmanagement ausgegeben. Für die Wiedergabe auf dem Monitor standen drei ICC-Profile zur Verfügung: Sony PS-200 bei 5000 K und 9300 K und HP sRGB [14].

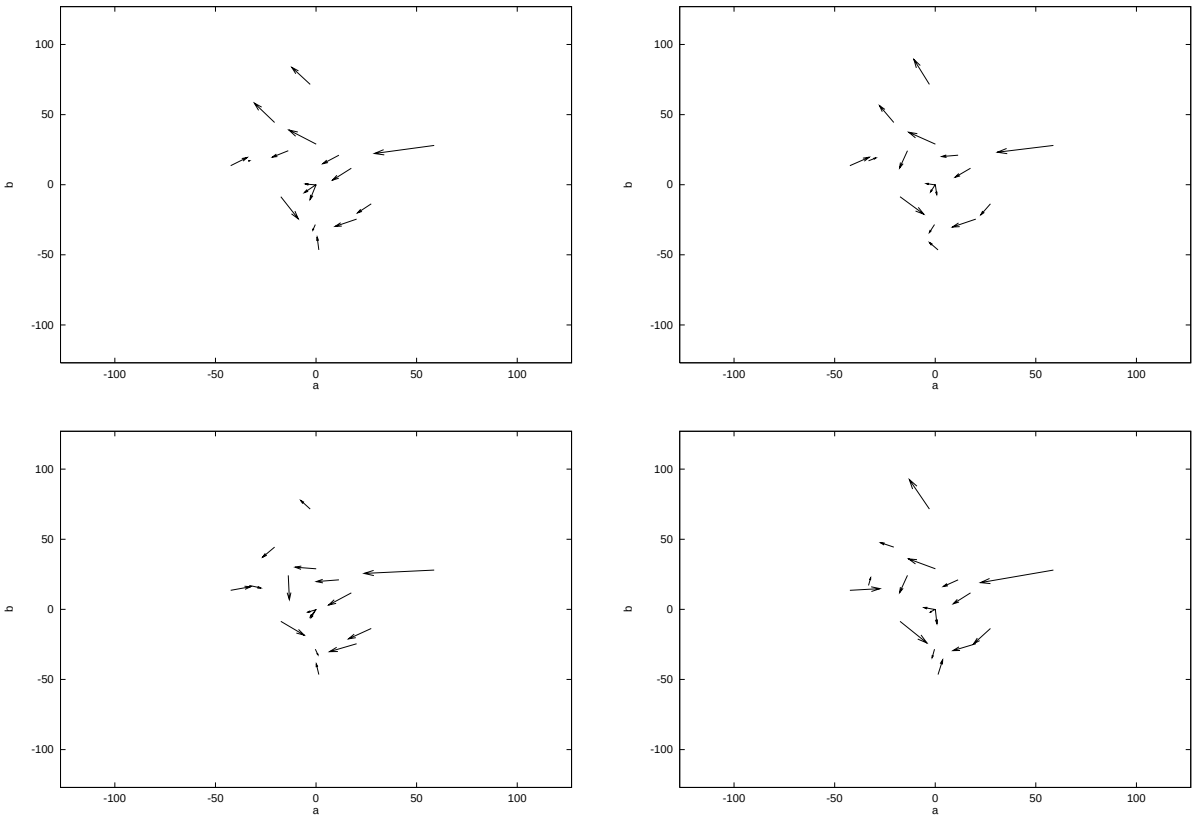

Abbildung 4.7: Fehler der 17 CIE-Testfarben für folgende Optimierungen: fotografisch, relativ und absolut farbmetrisch, sättigungsbasiert

Die Darstellung der Bilder auf dem Monitor war viel zu dunkel. Auch die Anwendung der ICC-Profile änderte daran nichts. Vielmehr mußten die Bilder manuell einer Gamma-Korrektur unterzogen werden, um eine akzeptable Helligkeit zu erzielen.

Die Gamma-Korrekturen führten allerdings, da es sich nur um einen 24 Bit-Scanner handelt, zu einem deutlich sichtbaren Rauschen. Teilweise entstanden sogar Artefakte in den sehr dunklen Bereichen.

Vor der Farbanpassung mittels des entwickelten Programmes zeigten die Bilder einen mehr oder weniger stark ausgebildeten Rotstich. Nach der Anwendung der drei ICC-Profile wurde dieser zwar reduziert, dafür war dann aber in weißen Flächen ein leichter Blaustich zu erkennen. Teilweise schien auch die Sättigung der Farbtöne leicht zu stark zu sein.

Wie zu erwarten war, lieferte das Profil, wenn der Monitor auf 9300 K eingestellt war, für das D93-Profil rotere Farben wie für das D50-Profil, so daß man bei Übereinstimmung der Farbtemperaturen von Monitor und ICC-Profil bei beiden Farbtemperaturen ungefähr die selben Farben erhält.

Insgesamt kann man sagen, daß die Qualität des Farbmanagements nicht überzeugt, was vor allem auf die verwendeten ICC-Profile zurückzuführen ist. Denn die Profile von Sony bestehen ausschließlich aus einer 3x3 Mischmatrix und einem Gamma-Wert für die TRC-Methode. Zusammen mit den schon vorher beschriebenen Problemen des Profiles für den ScanJet ergibt sich so ein recht schlechtes Ergebnis.

Auch die Ausgabe auf dem hochwertigen Tektronix Phaser 740 führte trotz des sehr großen ICC-Profiles nicht zu überzeugenden Ergebnissen. Die Bilder waren wie schon bei der Ausgabe auf dem Monitor viel zu dunkel und zeigten deutliche Farbstiche. Die Hauptursache für die schlechte Qualität dürfte auch

hier bei dem ICC-Profil für den verwendenten HP-Scanner zu suchen zu sein.

4.4 Geschwindigkeit

Um die Praxistauglichkeit des entwickelten Systemes zu überprüfen, wurden einige Geschwindigkeitsmessungen durchgeführt. Als erstes wurde mit folgendem Programm überprüft, wie lange es dauert, ein Handle für eine bestimmte Methode der Transformation zu erzeugen:

```
#include <stdio.h>
#include "color.h"

int main (void)
{
  cmt_Handle     hd_in;
  FILE           *in;
  unsigned int   i;

  in = fopen ("in.icm", "rb");

  for (i=0; i < 500; i++)
  {
    cm_Init (&hd_in, in, 8, cme_Perceptual, cme_LowQuality,
             cme_FromPCS);
    cm_Free (&hd_in);
  }
  fclose (in);
  return (0);
}
```

Das Programm erzeugt 500 mal ein Handle und gibt es wieder frei. Verwendet wurde für den Test ein PC mit einer Celeron 366 CPU und 128 MB Hauptspeicher. Die Ergebnisse sind in der nachfolgenden Tabelle zu sehen.

Methode	Richtung	Auflösung	Initialisierung/Sekunde
TRC	ToPCS	8	3125,0
		16	19,2
	FromPCS	8	17,9
		16	17,2
LUT	ToPCS	8	83,3
		16	13,5
	FromPCS	8	14,7
		16	14,7

Wie man sehen kann, ist die Initialisierung bei beiden Methoden ungefähr gleich schnell. Die 3125 Initialisierung pro Sekunde bei der TRC-Methode entstehen dadurch, daß das verwendete Profil (HP ScanJet) die eindimensionalen Tabellen bereits in einer Auflösung von 8 Bit enthält, so daß hier keine Interpolation stattfindet. Auch die 83,3 bei der LUT-Methode entstehen dadurch, daß einige der eindimensionalen Tabellen nicht interpoliert werden müssen.

Die Initialisierung muß in der Regel nur sehr selten durchgeführt werden, wenn z.B. die Anwendung gestartet wird oder eine neue Grafikdatei geöffnet wird. Ganz anders sieht das bei der eigentlichen Konvertierung aus. Sobald der Anwender eine Grafik bearbeitet, muß diese ganz oder in Teilen neu konvertiert werden. Von daher werden an die Konvertierungsroutinen ganz andere Geschwindigkeitsanforderungen gestellt als an die Initialisierungsroutinen.

Als nächste wurde deshalb mit nachfolgendem Programm getestet, wieviel Pixel pro Sekunde die unterschiedlichen Methoden transformieren können:

```
#include <stdio.h>
#include <stdlib.h>
#include "color.h"


int main (void)
{
  cmt_Handle     hd_in;
  FILE           *in;
  unsigned int   i;
  unsigned char  *pnt;


  in = fopen ("in.icm", "rb");
  cm_Init (&hd_in, in, 8, cme_Perceptual, cme_HighQuality,
           cme_ToPCS);
  fclose (in);

  pnt = malloc (3 * 10000);

  for (i=0; i < 50; i++)
  {
    free(cm_Convert (&hd_in, NULL, pnt, 10000));
    printf ("%u \n", i);
  }

  cm_Free (&hd_in);
  return (0);
}
```

Dabei ergaben sich folgende Geschwindigkeiten:

Methode	Richtung	Auflösung	Pixel/Sekunde
TRC	ToPCS	8	1.250.000
		16	1.250.000
	FromPCS	8	1.250.000
		16	1.250.000
LUT	ToPCS	8	16.667
		16	16.129
	FromPCS	8	227.273
		16	227.273

Wie zu erwarten war, ist die relativ simple TRC-Methode mit Abstand am schnellsten. Die Richtung der Transformation spielt bei der TRC-Methode keine Rolle. Ganz anders sieht das bei der LUT-Methode aus, wo die Geschwindigkeit der Transformation in Richtung der geräteunabhängigen Koordinaten deutlich kleiner ist wie bei der Gegenrichtung.

Dieser große Unterschied läßt sich dadurch erklären, daß die Interpolation im CLUT bei beiden Richtungen unterschiedlich implementiert werden mußte; siehe Abschnitt 3.4.3. Während bei der Transformation in Richtung geräteabhängiger Koordinaten einfach eine tri-lineare Interpolation verwendet werden konnte, mußte diese für die Gegenrichtung verallgemeinert werden, so daß zwei bis fünfzehn Eingangskoordinaten benutzt werden können. Diese allgemeine Implementation der Interpolation ist deshalb nicht so effektiv.

Wenn man bedenkt, daß ein Monitor bei einer Auflösung von 1024x768 bereits aus fast 800.000 Pixeln besteht, so muß man beim Vergleich mit der Geschwindigkeit der Transformation feststellen, daß die entwickelte Bibliothek selbst mit schnellen PCs zur Zeit eher nicht für Echtzeitanwendungen geeignet ist. Denn die wenigsten Anwender werden bereit sein, 5 Sekunden auf den Bildaufbau zu warten.

Bei eingescannten Vorlagen, die später ausgedruckt werden sollen, sieht die Situation noch schlechter aus, da diese Vorlagen schnell aus 10 Millionen Pixeln und mehr bestehen.

Von daher ist eine weitere Optimierung der Bibliothek notwendig. Die LUT-Methode für die Transformation in Richtung von geräteunabhängigen Koordinaten sollte sich recht leicht so optimieren lassen, daß sie wenigstens die Geschwindigkeit der Gegenrichtung erreicht.

Hierfür müßte lediglich die Implementation der n-linearen Interpolation um den Spezialfall der trilinearen Interpolation erweitert werden. Dieses ist deshalb sinnvoll, da die Rohdaten meistens im RGB-Farbraum vorliegen und so sowieso meistens die tri-lineare Interpolation benötigt wird.

Zusätzlich könnte eventuell noch der Spezialfall von vier Koordinaten berücksichtigt werden, wie er für den CMYK-Farbraum benötigt wird. Farbräume mit mehr als vier Koordinaten sind im Consumerbereich eher unüblich, so daß dann die langsame n-lineaere Interpolation garnicht zum Einsatz kommen müßte.

Kapitel 5

Zusammenfassung

Ziel dieser Studienarbeit war es, ein System für ein Farbmanagement auf Basis von ICC-Profilen zu entwickeln. Dieses Ziel wurde erreicht. Allerdings traten bereits während der Implementation einige Probleme auf, die auf die ungenaue Beschreibungen des Formates der Profile im ICC-Standard [4, 5] zurückzuführen sind. Hierbei fielen vor allem zwei Hauptprobleme auf.

Das erste Problem sind die Wertebereiche, die die geräteunabhängigen Farbräumen XYZ und Lab verwenden. Weder bei der TRC- noch bei der LUT-Methode legt der Standard fest, wie diese aussehen, was insbesondere dann zu Problemen führt, wenn beide Methoden an einer Transformation beteiligt sind.

Zuerst war ich davon ausgegangen, daß bei der LUT-Methode z.B. für XYZ-Koordinaten der »normale« Wertebereich von 0,0 bis 1,0 benutzt wird. Später viel mir dann im Anhang von [4] eine Beschreibung auf, die andere Wertebereiche für die geräteunabhängigen Farbräume vorschlägt. Allerdings wird im Anhang nicht klar, daß diese Kodierung für die TRC- und LUT-Methode verwendet werden soll. Die im Anhang vorschlagene Kodierung wurde schließlich implementiert.

Aber auch nach dieser Änderung blieb die Qualität des Farbmanagements unter Verwendung der von den Herstellern der Geräte gelieferten ICC-Profile weit hinter den Erwartungen zurück. Dieses ist aber auch nicht weiter verwunderlich, da der ICC-Standard nicht genau festlegt, wo das Farbmanagement überhaupt ansetzen soll.

Dieses scheint dazu zu führen, daß viele Gerätehersteller die ICC-Profile nicht für das Gerät sondern für die Kombination aus bestehendem Treiber und Gerät erstellen. Da viele Treiber noch aus einer Zeit stammen, wo MS-Windows noch nicht über ein System für ein Farbmanagement verfügte, finden auch in den Treibern Manipulationen der Farben statt.

Im Gegensatz zu den Treiber für MS-Windows liefern die meisten Treiber für Linux und Unix die Rohdaten der Geräte. Aus diesem Grund funktionieren die meisten ICC-Profile, die mit den Geräten ausgeliefert werden, nur mit MS-Windows und den Originaltreibern der Gerätehersteller.

Das *Internation Color Consortium* hat dieses Problem zwar anscheinend erkannt, aber schreibt bisher leider nicht vor, was ein Profil genau kalibrieren soll. Bisher existiert im Header von ICC-Profilen lediglich eine Kennzeichnung dafür, mit welchem Betriebssystem das Profil benutzt werden soll. Im Idealfall sollte sich ein Gerätetreiber ausschließlich um die Ansteuerung des Gerätes kümmern, während ein ICC-Profil die Anpassung der Farben an das Gerät vornimmt.

Sinnvoll wäre es sicherlich außerdem, im Standard für ICC-Profile das zu verwendende Interpolations-

verfahren genau festzulegen, da die Resultate der verschiedenen Verfahren sich doch deutlich unterscheiden können, was dann zum Problem wird, wenn Profil-Generator und ICC-Bibliothek verschiedene Verfahren verwenden.

Ein Problem dieser Implementierung eines Systemes für ein Farbmanagement ist sicherlich die benötigte Rechenzeit für die Transformation. Insbesondere die LUT-Methode müßte für einen Echtzeiteinsatz in der Praxis deutlich optimiert werden. Bisher ist lediglich die Geschwindigkeit der TRC-Methode für dieses Einsatzgebiet akzeptabel. Dafür ist allerdings die von der TRC-Methode gelieferte Qualität bei weitem nicht überzeugend.

Kapitel 6

Ausblick

Bevor die entwickelte Bibliothek für ein Farbmanagement in der Praxis sinnvoll eingesetzt werden kann, sind einige weitere Schritt notwendig.

Zum einen müssen Standards geschaffen bzw. ergänzt werden, wie Anwendungen an die ICC-Profile der beteiligten Geräte kommen.

Hiervon ist z.B. die SANE-API betroffen, die sich wie die TWAIN-API unter Windows um die Ansteuerung von Scannern bei Unix-/Linux-Systemen kümmert. Da der Treiber des Scanners in der Regel am besten weiß, welches ICC-Profil für den Scanner benutzt werden soll, muß eine Möglichkeit in der SANE-API beschaffen werden, um zwischen Scanner-Treiber und der Anwendung, die mittels dieses Treibers Vorlagen einscannt, die passenden ICC-Profile austauschen zu können. Das Scanprogramm könnte das so erhaltene ICC-Profil dann in die Grafikdatei einbetten.

Zur Ausgabe einer so erstellten Grafikdatei benötigt die Anwendung das jeweilige ICC-Profil des Monitores. Da Unix/Linux ein verteiltes System ist, bei dem das Programm nicht unbedingt auf dem Rechner laufen muß, auf die Ausgabe erfolgt, und jeder Rechner außerdem mehrere Monitore besitzen kann, ist es wenig sinnvoll, per definierter Konfigurationsdatei festzulegen, welches Profil benutzt werden soll. Besser geeignet hierfür dürften die X11-Ressourcen sein oder ein spezielles Netzwerkprotokoll für den Austausch von Profilen.

Um den Druckprozeß mit einem Farbmanagementsystem zu versehen, könnte man das Programm Ghostscript, das in der Regel als Software-RIP eingesetzt wird, so erweitern, daß es die entwickelte ICC-Bibliothek verwendet.

Neben diesen Anpassungen müssen natürlich auch die eigentlichen Anwendungen so geändert werden, daß sie die entwickelte Bibliothek für ein farbsicheres Arbeiten verwenden.

Da die mit den Geräten mitgelieferten ICC-Profile in vielen Fällen nicht geeignet sind, mit Treibern verwendet zu werden, die nicht vom Hersteller des Gerätes stammen, wird es für den Praxiseinsatz unerläßlich sein, einen eigenen ICC-Profil-Generator zu entwickeln, der z.B. bei Scannern eine IT-8-Vorlage einliest und aus dem Vergleich der Ist- mit den Sollwerten ein ICC-Profil für das Gerät erzeugt.

Anhang A

Farbräume

In diesem Kapitel werden einige Farbräume vorgestellt.

A.1 RGB

Die Farben Rot, Grün und Blau bilden die drei Primärvalenzen dieses geräteabhängigen Farbraumes. Die genauen Werte der Primärvalenzen sind allerdings nicht normiert, sondern hängen von dem jeweiligen Gerät ab. Bei einem Monitor/Fernseher werden sie durch das verwendete Phosphor bestimmt und bei Scannern durch die verwendeten Filter.

Die NTSC-Norm schreibt z.B. folgende Primärvalenzen vor: Rot (610 nm), Grün (537 nm) und Blau (472 nm) [1]. Die CIE-Norm definiert stattdessen folgende Primärvalenzen: Rot (700 nm), Grün (546,1 nm) und Blau (438,8 nm).

Mit den drei Primärvalenzen der RGB-Farbräume lassen sich nicht alle möglichen Farben darstellen, da dazu teilweise negative Anteile der Primärvalenzen notwendig wären, was sich praktisch natürlich nicht realisieren läßt. Durch die fehlende Normung der Primärvalenzen lassen sich Daten zwischen zwei Geräte, die unterschiedliche RGB-Primärvalenzen verwenden, nicht austauschen.

A.2 CIE XYZ

Dieser Farbraum wird durch drei linear unabhängige Primärvalenzen gebildet, die als XYZ bezeichnet werden. Die drei Primärvalenzen sind so gewählt, daß zum einen das von ihnen gebildete gleichseitige Dreieck den Spektralfarbenzug möglichst eng umschließt und zum zweiten X und Z keinen Beitrag zur Helligkeit liefern [1]. Diese Primärvalzenzen sind in der Praxis nicht realisierbar.

Da das Dreieck alle Farben des Spektralfarbenzuges umschließt, können mit einer Linearkombination der drei Primärvalzenzen alle Farben erzeugt werden. Es handelt sich folglich um einen geräteunabhängigen Farbraum.

Der CIE RGB-Farbraum kann so nach XYZ transformiert werden:

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} 0,490 & 0,310 & 0,200 \\ 0,177 & 0,813 & 0,011 \\ 0,000 & 0,010 & 0,990 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

A.3 CIE Lab

Ein zweiter geräteunabhängiger Farbraum ist CIE Lab, der 1976 verabschiedet wurde und vom Aufbau her formal den Hunter-Formel entspricht. Bei diesem System gibt die L-Koordinate die Helligkeit wieder. Sind a und b gleich Null, so erhält man für L einen Farbverlauf von Weiß bis Schwarz.

L bildet die Grauachse und steht senkrecht auf der Fläche, die durch a und b aufgespannt wird. Die -a-Achse entspricht einem Blaugrün und die a-Achse einem roten Purpur. Die b-Achse läuft von Blau nach Gelb.

Betrachtet man diesen Farbraum in Zylinderkoordinaten statt in rechtwinkligen Koordinaten, so erhält man den LCH-Farbraum. Der Winkel um die L-Achse stellt dann die Farbart dar. Die Sättigung einer Farbe wird durch den Radius von der L-Achse bestimmt.

Bei dem Lab-Farbraum soll der Abstand zwischen zwei Farborten proportional zu dem vom Betrachter empfundenen Farbunterschied sein. Dieses gelingt jedoch nur teilweise.

Literaturverzeichnis

[1] Hansl Loos: *Farbmessung - Grundlagen der Farbmetrik und ihre Anwendungungsbereiche in der Druckindustrie*, Verlag Beruf + Schule, ISBN 3-88013-380-8, 1989

[2] Jan-Peter Homann: *Digitales Colormanagement - Farbe in der Publishing-Praxis*, Springer, ISBN 3-540-60724-2, 1998

[3] Steve Hill: *Tri-linear Interpolation*, Seite 521-525 aus Paul S. Heckbert: *Graphics Gems IV*, Academic Press, ISBN 0-12-336155-9, 1994

[4] International Color Consortium: *File Format for Color Profiles*, Specification ICC.1:1998-09, http://www.color.org, 1998

[5] International Color Consortium: *Addendum 2 to Spec. ICC.1:1998-04*, Document ICC.1A:1999-04, http://www.color.org, 1999

[6] Michael Has: *Color Management: Current Practice and The Adoption of a New Standard*, http://www.color.org

[7] Klaus Richter: *Computergrafik und Farbmetrik*, VDE-Verlag, ISBN 3-8007-1775-1, 1996

[8] Manfred Richter: *Einführung in die Farbmetrik*, Walter de Gruyter, ISBN 3-110-08209-8, 1981

[9] R.W.G. Hunt: *Measuring Color*, Ellis Horwood, ISBN 0-13-567686-X, 1991

[10] Adobe Systems Inc.: *PostScript Handbuch*, Addison-Wesley, ISBN 3-89319-267-0, 1989

[11] Aldus: *TIFF Revision 6.0*, 1992

[12] Dr. Jörn Loviscach, Wolfgang Fastenrath: *Villa Kunterbunt - Farbkorrektur mit Color-Management-Systemen*, c't 10/1996, S.180 ff

[13] Ulrich Hilgefort, Dr. Wolfgang Stieler, Carsten Meyer, Stefan Labusga: *Guckkästen - Flachbettscanner der DTP-Klasse bis 5000 Mark*, c't 16/1999, S.104 ff

[14] Hewlett-Packard: *sRGB*, http://www.srgb.com