Positionierung von Farbauszügen

Diplomarbeit
Fakultät für Physik
Universität Bielefeld
Carsten Hammer
Schwindstr.7
33615 Bielefeld

14. Mai 1997

Inhaltsverzeichnis

1 Einführung

Die Aufgabenstellung der folgenden Arbeit ist, Farbauszüge anhand ihres bildlichen Inhalts gegeneinander zu verschieben und zu drehen. Die Farbauszüge sollen am Ende paßgenau übereinanderliegen. Diese Aufgabenstellung ergibt sich aus einer Druckvorstufe beim Offsetdruck. Dabei werden Farbauszüge bis heute manuell justiert.

Die einzelnen Arbeitsschritte beim Offsetdruck sind:

- **Rasterung:** Die Vorlage wird gerastert. Zu diesem Zeitpunkt liegt verschiedenes Material in Form von Textblöcken, Farbdias oder Farbfotos vor. Durch Farbkanaltrennung werden Farbauszüge erzeugt, die dem subtraktiven CMYK-Modell entsprechen.

- **Montage:** Die Folien werden paßgenau auf vier Trägerfolien montiert. Die korrekte Position der Trägerfolien wird duch eine genormte Lochung sichergestellt. Die Montage der Folien auf den Trägerfolien ist kritisch hinsichtlich Offset und Winkel und wird mit großer Konzentration in Handarbeit erledigt. Die Farbfolien müssen auf $3\mu m$ genau passen.

- **Kopie:** Die fertig montierten Seiten werden auf Druckformen belichtet und die Formen werden geätzt.

- **Druck:** Die Druckformen werden ihrer Druckfarbe gemäß auf den dafür vorgesehenen Druckzylinder gespannt. Das Papier läuft über alle vier Druckzylinder und die Farben werden schrittweise übereinander gedruckt.

Ziel dieser Arbeit ist die Entwicklung von Algorithmen damit die zeitaufwendige Montage vom Computer erledigt werden kann. Die hier bearbeite Aufgabenstellung ist allerdings insofern nicht exakt die Aufgabenstellung aus dem Offsetdruck, als daß die Betrachtung der nötigen Pixelauflösung keine Rolle gespielt hat und auch der Schritt der Erfassung der Farbfolien zur Weiterverarbeitung im Computer nicht Gegenstand der Untersuchung war. Vielmehr wurde davon ausgegangen, daß die gerasterten Farbfolien in geeigneter Weise in eine Graustufenform umgewandelt worden sind und in dieser Form im Computer vorliegen. Neben der Genauigkeit und Korrektheit ist außerdem die Rechenzeit ein wichtiger Punkt bei der Implementation entsprechender Algorithmen.

Das Farbmodell im Druck ist in der Regel CMYK. Bei der Bearbeitung dieser Arbeit wurde RGB zugrundegelegt.

In der methodisch anders vorgehenden Arbeit [9] von Carsten Hellmich findet man eine ausführliche Erklärung der relevanten Schritte der Druckverfahrens.

Eine Gemeinsamkeit der beiden Arbeiten ist die Entkopplung von Winkel und Offsetbestimmung. In dieser Arbeit vorgestellte Verfahren kommen ohne weitere Kenntnis über Form und Inhalt der Farbauszüge aus. Der Winkel wird in [9] über die Rasterneigung der Farbauszüge bestimmt. Eine Zeile im Bild weist einen für einen bestimmten Rasterneigungswinkel charakteristischen Abstand der Rasterpunkte auf. Durch eine FFT der Zeile kann dieser Abstand gewonnen werden. Berücksichtigt werden muß dabei die Rasterneigung der verschiedenen Farbauszüge. Die daran anschließende Bestimmung des Offsets wurde mit Hilfe von

neuronalen Netzen versucht. Die Ausrichtung hing dabei aber von dem Erfolg der vorgeschalteten Merkmalsextraktion ab. Es ist aber schwer, für diese Merkmalsextraktion ein Verfahren anzugeben, das auf Bildern unabhängig von ihrem Inhalt funktioniert. Im hier vorgestellten Verfahren tritt diese Schwierigkeit nicht auf, da es im Gegensatz zum merkmalsorientierten Verfahren in [9] flächenorientiert ist.

Unter optischem Fluß („optical flow“) versteht man die Bewegung von Strukturen in Bildfolgen. Es gibt eine Reihe von Verfahren, etwa Gradientenverfahren oder Korrelationsverfahren, die dabei benutzt werden, um aus diesen Bildfolgen Verschiebungsvektorfelder zu berechnen. Der Unterschied zum hier bearbeiteten Problem ist, daß im Gegensatz zur lokalen Ausrichtung dort, hier eine globale Ausrichtung berechnet werden muß.

Die Reihenfolge in der vorliegenden Arbeit wurde folgendermaßen gewählt. In Kapitel 2 sollen zunächst alle benötigten Verfahren und mathematischen Gesetzmäßigkeiten, soweit das notwendig erscheint, vorgestellt werden.

Im Kapitel 3 gehe ich auf Aspekte der Implementierung der Algorithmen ein. Das bedeutet, die im einzelnen erklärten Verfahren und Gesetzmäßigkeiten werden hier zu einem ganzen Programm zusammengefügt.

Das Kapitel 4 zeigt dann anhand von konkreten Beispielen die Robustheit des Verfahrens und verschiedene praktische Aspekte, wie Rechenzeitbedarf und Einfluß der Eingabedaten.

2 Grundlagen

Zur Einführung in einige wichtige, hier angewendete Bildverarbeitungsverfahren seien [2][20][15] und [10] empfohlen.

2.1 Farben in der Bildverarbeitung

Das menschliche Auge nimmt elektromagnetische Wellen im Bereich zwischen 400nm und 700nm mit 3 Rezeptoren unterschiedlicher spektraler Empfindlichkeit wahr, 4 wenn man einen nur bei extrem schlechten Lichtverhältnissen wichtigen Rezeptor berücksichtigt. Weil nur drei Rezeptoren existieren, sind drei Werte notwendig und hinreichend für die Beschreibung jeder für das Auge sichtbaren Farbe.

Es gibt aber mehrere Farbmodelle für lineare, verschiedene nichtlineare, additive und subtraktive Farbmischungen. Additiv ist RGB, subtraktiv CMY daneben gibt es noch YUV, YCC, HSB, HLS und eine Anzahl anderer ([5]).

2.1.1 RGB-Farbmodell

Die meisten Experimente mit nachfolgenden Algorithmen wurden auf Farbauszügen im RGB-Farbmodell gemacht. Dabei liegen 3 Farbauszüge vor in rot, grün und blau. Mittels additiver Farbmischung entsteht daraus bei der Darstellung das ganze Farbspektrum.

2.1.2 CMYK-Farbmodell

Im Druck wird das subtraktive CMYK-Farbmodell benutzt, bei dem aus Cyan, Magenta, Yellow und Black alle Farben gemischt werden. CMYK ist dabei nur eine Erweiterung des CMY Farbmodells um den Schwarzanteil. Einmal hat das den Vorteil, die teure bunte Farbe zu sparen und zum anderen ist die Farbmischung „schwarz“ im Druck durch allerlei Nichtlinearitäten leider nur braun. Außerdem wird das Papier nicht so naß beim Druck und mechanische Toleranzen führen nicht zu farbigen Rändern bei schwarzen Punkten, sind also leichter tolerierbar.

2.1.3 CMYK RGB Farbkonvertierung

Ein einfacher Weg führt über die folgenden beiden Schritte zu einem für unsere Betrachtungen ausreichenden Ergebnis. Dabei werden sämtliche Nichtlinearitäten vernachlässigt:

CMYK nach CMY

$$\begin{aligned} Cyan &= Min(1, Cyan \cdot (1 - Black) + Black) \\ Magenta &= Min(1, Magenta \cdot (1 - Black) + Black) \\ Yellow &= Min(1, Yellow \cdot (1 - Black) + Black) \end{aligned} \qquad \text{mit } Min(a,b) = \begin{cases} a \text{ für } a < b, \\ b \text{ für } a > b. \end{cases} \tag{1}$$

CMY nach RGB

$$\begin{aligned} Red &= 1 - Cyan \\ Green &= 1 - Magenta \\ Blue &= 1 - Yellow \end{aligned} \tag{2}$$

Unter Red, Green, Blue, Cyan, Magenta, Yellow sind die entsprechenden Farbanteile zu verstehen. Wie man sieht, sind im RGB-Farbmodell die Intensitäten invers gegenüber dem CMY-Modell (und CMYK). Um die hier vorgestellten Ergebnisse auf CMYK übertragen zu können, muß man die Eigenschaften der Konvertierung berücksichtigen. Die Ermittlung geeigneter Maßnahmen, um die hier vorgestellten Algorithmen auf CMYK direkt anwenden zu können, war nicht Gegenstand der Untersuchung dieser Arbeit.

2.2 Bildvorverarbeitung (Bildfilterung)

Eine Faltung ist eine Operation mit folgender Rechenvorschrift:

$$g(\boldsymbol{x}) = \int_{-\infty}^{\infty} d\boldsymbol{x}' g'(\boldsymbol{x}') h(\boldsymbol{x} - \boldsymbol{x}') = g'(\boldsymbol{x}) * h(\boldsymbol{x}) \tag{3}$$

Jeder Punkt x' in der Objektebene gibt eine Intensitätsverteilung $g'(x')h(x - x')$ in der Bildebene. An einem Punkt x sind die Beiträge von allen Punktantworten aufzusummieren. Das entspricht einer Integration über x' und ergibt $g(x)$.

In Gleichung 4 ist die Rechenvorschrift für eine Filterung eines Bildes G angegeben. Die Faltungsmaske ist H und i und j geben die Indices von Punkten in dieser Faltungsmaske an.

$$G'_{m,n} = \sum_{i=-r}^{r} \sum_{j=-r}^{r} H_{i,j} G_{m-i,n-j} \tag{4}$$

Man sieht an dieser Gleichung, daß aus Symmetriegründen Masken ungerader Größe bevorzugt werden ([10]). Es ist auch möglich die Faltung im Frequenzraum vorzunehmen. Dann wird daraus eine Multiplikation. Das lohnt sich allerdings nur für große Masken.

2.2.1 Sobeloperator

$$S_x = \frac{1}{8}\begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix} \quad S_y = \frac{1}{8}\begin{pmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{pmatrix} \tag{5}$$

$$S = \sqrt{S_x^2 + S_y^2}$$

Der Operator, der hier benutzt wurde, ist ein Sobeloperator in x und y Richtung. Der Sobeloperator verstärkt die Kanten.

Abbildung 1: Zur Wirkung des Sobeloperators (Grünauszug)

Abbildung 1 zeigt die Wirkung dieses Operators S auf den Grünauszug in Abbildung 22 auf Seite 35 (Für weitere Hinweise dazu siehe [1][10][15]).

2.2.2 Laplaceoperator

$$L_1 = \begin{pmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{pmatrix} \quad L_2 = \frac{1}{4}\begin{pmatrix} 1 & 2 & 1 \\ 2 & -12 & 2 \\ 1 & 2 & 1 \end{pmatrix} \tag{6}$$

L_1 und L_2 sind zwei Varianten des Laplace-Operators. Der Laplaceoperator unterdrückt konstante Bereiche des Bildes und ist in dieser Form eine diskrete Version von ([15] Seite 54)

$$\triangle f(x,y) = \frac{\partial^2 f(x,y)}{\partial x^2} + \frac{\partial^2 f(x,y)}{\partial y^2} = \nabla^2 f(x,y)$$

Abbildung 2 zeigt die Wirkung dieses Operators L_2 auf den Grünauszug in Abbildung 22 auf Seite 35.

Abbildung 2: Zur Wirkung des Laplaceoperators L_2 (Grünauszug)

2.2.3 Gaußoperator

$$G = \frac{1}{16}\begin{pmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{pmatrix} \tag{7}$$

In Abbildung 4 kann man die Wirkung einer Gaußfilterung beobachten.

Abbildung 3: Original zum Vergleich (Grünauszug)

Abbildung 4: Zur Wirkung des Gaußoperators (Grünauszug)

Bild-Details gehen verloren, weil dieser Filter eine Mittelung über Nachbarpunkte durchführt.

2.2.4 Normierung

Um Probleme mit Bereichsüberschreitung von benutzten Datentypen zu vermeiden, wird nach jeder Filterung eine Normierung vorgenommen, die die Grauwerte der Bilder in den Bereich zwischen 0 und $2^{16} - 1$ (bzw. $2^8 - 1$, je nach verwendeter Farbtiefe) bringt ([15] Seite 69).

$$h(x,y) = \frac{f(x,y) - f_{min}}{f_{max} - f_{min}}(2^{16} - 1) \tag{8}$$

$$f_{min} = min(f(x,y)) \quad und \quad f_{max} = max(f(x,y))$$

Will man den Drehwinkel zweier Farbauszüge gegeneinander bestimmen, so darf man die berechneten Kennziffernbilder für die verschiedenen Drehwinkel nicht einzeln normieren. In dem Fall ist die Tiefe des Minimums in Abhängigkeit von dem Drehwinkel zu untersuchen. Das Minimum dieser Funktion ist dann der korrekte Drehwinkel (und der korrekte Offset).

2.3 Ausrichtung über Schwerpunktbestimmung

Zur Initialisierung der nachfolgenden Extremwertbestimmung wird als erstes auf allen Farbauszügen der jeweilige Schwerpunkt bestimmt. Dazu wird der Ortsvektor jeden Punktes - mit dem entsprechenden Grauwert gewichtet - aufsummiert und anschließend durch die Summe der Grauwerte geteilt.

$$\vec{\mathcal{S}} = \frac{\sum\limits_{\vec{x} \in \mathbb{D}} f(\vec{x}) \cdot \vec{x}}{\sum\limits_{\vec{x} \in \mathbb{D}} f(\vec{x})} \tag{9}$$

f ist der Grauwert des jeweiligen Farbauszuges in Abhängigkeit vom Ortsvektor. Für jeweils zwei Farbauszüge können die Schwerpunkte dann zur Deckung gebracht werden, indem die Farbauzüge entsprechend gegeneinander verschoben werden. Damit hat man einen Startwert für die nachgeschaltete Minimierung.

Die Berechnung mit Hilfe von Gleichung (9) ergibt bei Bildern gute Resultate, solange keine Teile in einzelnen Farbauszügen abgeschnitten sind. Bei Bildern mit deutlich getrennten Farbfeldern ist es aber auch gar nicht nötig, eine Justierung vorzunehmen. Bei Bildern, die die wesentlichen Strukturmerkmale in allen Farbauszügen zeigen, liegen die Schwerpunkte der verschiedenen Farbauszüge nicht weit auseinander.

2.4 Kennziffernberechnung

Bei einer gegebenen Verschiebung zweier Farbauszüge kann numerisch eine Kennziffer berechnet werden, die ein Maß für die Übereinstimmung der beiden Farbauszüge ist. Bei dem Verfahren des „optischen Flusses“ [18][11] werden ähnliche Methoden benutzt. Bei der Bestimmung von Verschiebungsvektorfeldern haben Korrelationmethoden einen festen Platz. Dabei werden Bilder oder Bildausschnitte genommen und ein Operator auf diesen Ausschnitten angewendet. Die Funktion, die man damit bekommt, hat an dem Ort einer optimalen Verschiebung ein Extremum.

2.4.1 Kennziffer Produktformel

Die Grauwerte der zwei Farbauszüge an der durch (x,y) gegebenen Position werden multipliziert. Der Wert dieses Produkts wird über alle Bildpunkte im Überlappungsbereich aufsummiert. Diese Summe wird anschließend durch die Fläche des Überlappungsbereichs geteilt. Die hier verwendete Produktformel entspricht bis auf diese Flächennormierung der in [10] beschriebenen Kreuzkorrelation.

$$\mathcal{K}_p(ro_x, ro_y) = \frac{\sum_{x=0}^{ro_x}\sum_{y=0}^{ro_y} \frac{f_1(x-ro_x, y-ro_y)\cdot f_2(x,y)}{(2^{16}-1)\cdot(2^{16}-1)}}{ro_x \cdot ro_y} = \frac{\sum_{(x,y)\in\mathbb{D}} \frac{f_1(x-ro_x, y-ro_y)\cdot f_2(x,y)}{(2^{16}-1)\cdot(2^{16}-1)}}{ro_x \cdot ro_y} \tag{10}$$

mit

$$0 < x < ro_x; \quad 0 < y < ro_y; \quad 0 < f_{1,2,3}(x,y) < 2^{16}-1; \quad \text{für die RGB-Auszüge } f_1, f_2, f_3$$

2.4.2 Kennziffer Differenzformel

Die Grauwerte der zwei Farbauszüge an der durch (x,y) gegebenen Position werden abgezogen. Der Absolutwert dieser Differenz wird über alle Bildpunkte im Überlappungsbereich aufsummiert. Diese Summe wird anschließend durch den Flächeninhalt des Überlappungsbereichs geteilt. Hierbei handelt es sich um das „Integral des (Betrags-) Differenzbildes“.

$$\mathcal{K}_d(ro_x, ro_y) = \frac{\sum_{(x,y)\in\mathbb{D}} \frac{|f_1(x-ro_x, y-ro_y) - f_2(x,y)|}{2^{16}-1}}{ro_x \cdot ro_y} \tag{11}$$

2.4.3 Überlappungsbereich

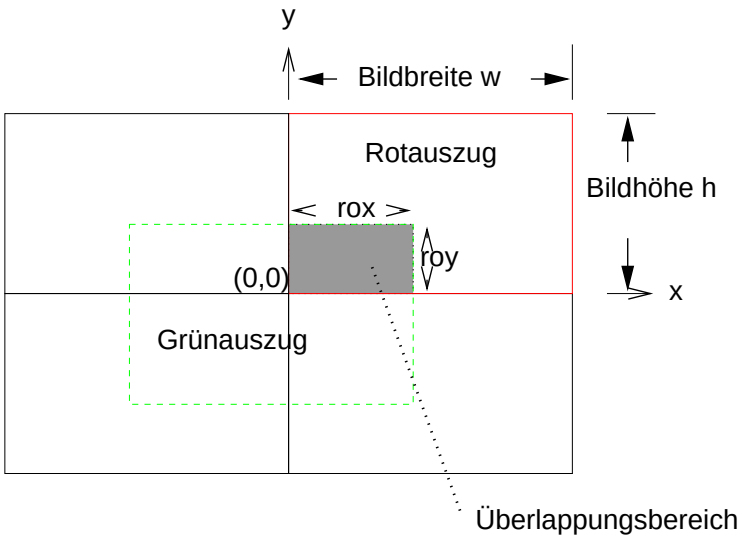


Abbildung 5: Zur Berechnung der Kennziffer

In Abbildung 5 ist schematisch dargestellt, welche Positionen die Farbauszüge zueinander einnehmen können. Der eine (Rotauszug) liegt fixiert mit der linken unteren Ecke an der Position (0,0). Der andere (Grünauszug) kann verschoben werden, so daß die rechte obere Ecke irgendwo im Bereich zwischen (0,0) und (2·w,2·h) liegt. Der überlappende Bereich, die „Schnittfläche“, wird für die Kennziffernberechnung genutzt. Bei der Drehung wird für jeden Farbauszug in einem zusätzlichen Bild abgelegt, ob an dem entsprechenden Bildpunkt ein gültiger Bildpunkt vorhanden ist, oder ob er durch Drehung aus dem außerhalb des Bildes liegenden Bereichen bevölkert wurde.

2.5 Rotation von Bildern

Die Drehung eines Graustufenbildes wirft gewisse Schwierigkeiten auf, die auf dem zugrundeliegenden Gitter mit ganzzahligen Koordinaten beruhen. Eine Drehung auf solchen Gittern kann nur für Vielfache von 90^o exakt sein, da der gedrehte Punkt dann wieder auf einen Gitterpunkt fällt. Bei allen anderen Drehwinkeln sind die Nachbarpunkte in die Rechnung mit einzubeziehen (z.B. Interpolation). Das Bild wird verändert, eine Drehung ist im allgemeinen nicht reversibel. Diese Tatsache ergibt sich aus dem Abtasttheorem, denn bei einem gedrehten Gitter verkürzen sich stellenweise die Abstände der Gitterknoten des gedrehten Gitters im Vergleich zum ungedrehten Gitter, und damit kann die Funktion im Ortsraum aus den Abtastwerten nicht mehr exakt rekonstruiert werden.

Die an eine Drehung zu stellenden Forderungen sind die folgenden:

- stetig: Es sollten keine Sprünge in den Graustufenwerten auftreten, die über die zwingende Typquantisierung hinausgehen.

- normerhaltend: Da Bilder unter verschiedenen Drehwinkeln miteinander verglichen werden, darf das Integral über die Intensität sich nicht ändern.

Je besser beide Forderungen erfüllt sind, desto vergleichbarer sind die Kennzifferwerte.

2.5.1 Vorwärtsdrehung

Bei dieser Drehung aus [?] wird ein Punkt mit den Koordinaten (i,j) gedreht und landet im Raster (i',i'+1 und j',j'+1). Die Numerierung der Gitterpunkte steigt an von links nach rechts und von oben nach unten. Es werden neue Intensitäten ausgerechnet an den 4 neuen Gitterpunkten.

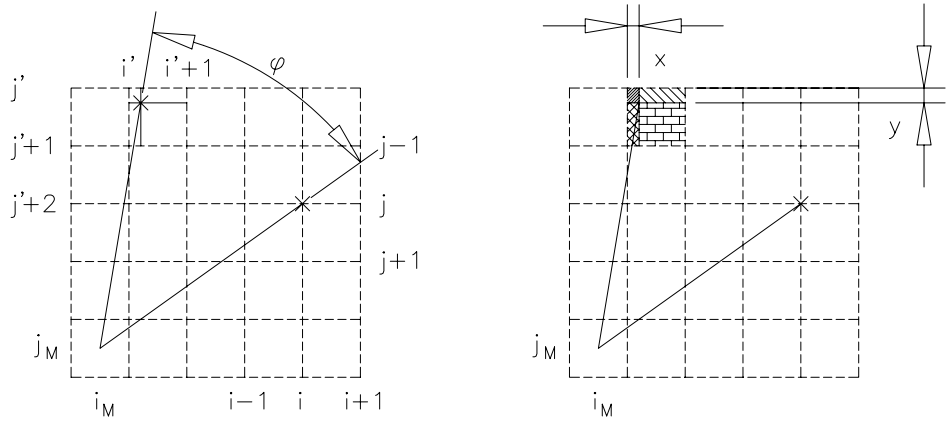


Abbildung 6: Vorwärtsdrehung, Flächen zur Wichtung

$$\begin{aligned} px = & i - i_M;\ py = j - j_M;\ x = \quad px * cos\varphi + py * sin\varphi \\ & y = -px * sin\varphi + py * cos\varphi \end{aligned} \tag{12}$$

Gleichung 12 beschreibt die Drehung eines Punktes (i,j) mit dem Winkel φ um das Zentrum $i_M; j_M$.

Indices des gedrehten Punktes, Flächen zur Intensitätsumrechnung:

$$\begin{aligned} i' = & ent(x + i_M);\ x := x - i' + i_M \\ j' = & ent(y + j_M);\ y := y - j' + j_M \end{aligned} \tag{13}$$

In Gleichung 13 werden die Indices des gedrehten Punktes sowie die Flächen zur Intensitätsumrechnung beschrieben. Unter der Funktion ent(a) wird die größte ganze Zahl $\leq a$ verstanden.

Intensitäten des Punktes im gedrehten Bild

$$\begin{aligned} e_{i',j'} &:= e_{i',j'} &+ent[d_{i,j}\ (1-x)(1-y)] \\ e_{i'+1,j'} &:= e_{i'+1,j'} &+ent[d_{i,j}\ x(1-y)] \\ e_{i',j'+1} &:= e_{i',j'+1} &+ent[d_{i,j}\ (1-x)y] \\ e_{i'+1,j'+1} &:= e_{i'+1,j'+1} &+ent[d_{i,j}\ xy] \end{aligned} \tag{14}$$

Gleichung 14 beschreibt die Intensitätsumrechnung bei der Drehung. Die Intensitätsbeiträge des gedrehten Bildes werden im Feld $e_{i,j}$ aufsummiert. Jeder Punkt erhält den Intensitätsanteil aus dem gedrehten Bildpunkt, der aus der Wichtung mit den gegenüberliegenden Flächen entsteht. Durch diese Wichtung bleibt die Intensität als Funktion des Drehwinkels stetig. Der Gitterpunkt erhält den größten Anteil, der am dichtesten an dem gedrehten Punkt liegt. Landet der gedrehte Punkt auf einen Gitterpunkt, geht die Intensität voll an ihn über. Da die Flächensumme (1-x)(1-y)+x(1-y)+(1-x)y+xy=1 konstant ist, erzeugt jeder Punkt $d_{i,j}$ im neuen Bild an 4 Punkten Intensitäten, deren Summe der Intensität des ursprünglichen Bildpunktes entsprechen. Da das für jeden Punkt gilt, bleibt die Intensität des Gesamtbildes erhalten.

2.5.2 Rückwärtsdrehung

Hier wird ein Punkt mit den Koordinaten (i,j) im gedrehten Bild zurückgedreht ($-\varphi$) und landet im Raster (i',i'+1 und j',j'+1) des Originalbildes. Die Intensität an der gedrehten Stelle (i,j) setzt sich additiv zusammen aus 4 Punkten auf dem alten Gitter.

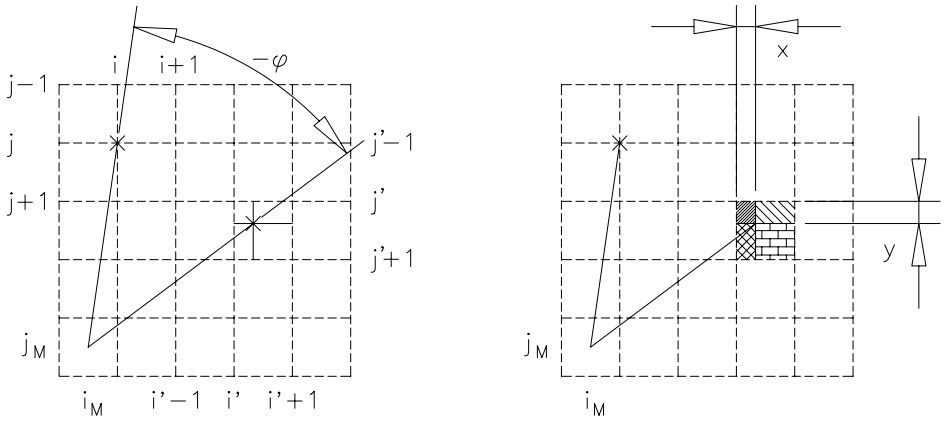


Abbildung 7: Rückwärtsdrehung, Flächen zur Wichtung

$$\begin{aligned} px = i - i_M;\ py = j - j_M;\ x &= px * cos\varphi - py * sin\varphi \\ y &= px * sin\varphi + py * cos\varphi \end{aligned} \tag{15}$$

Gleichung 15 beschreibt die Drehung eines Punktes (i,j) mit dem Winkel $-\varphi$ um das Zentrum $i_M; j_M$.

Indices des gedrehten Punktes, Flächen zur Intensitätsumrechnung:

$$\begin{aligned} i' &= ent(x + i_M);\ x := x - i' + i_M \\ j' &= ent(y + j_M);\ y := y - j' + j_M \end{aligned} \tag{16}$$

In Gleichung 16 werden die Indices des gedrehten Punktes sowie die Flächen zur Intensitätsumrechnung beschrieben.

Intensitäten des Punktes im gedrehten Bild

$$\begin{aligned} e_{i,j} := ent[& d_{i',j'} \quad (1-x)(1-y) + d_{i_{i'+1,j'}} \quad x(1-y) \\ & + d_{i',j'+1}\ (1-x)y \quad + d_{i'+1,j'+1}\ xy \quad] \end{aligned} \tag{17}$$

Die Intensität des gedrehten Bildes setzt sich nach Gleichung 17 wieder aus 4 Einzelbeiträgen zusammen, die aber im Gegensatz zum 1. Verfahren nicht mehr die gleiche Intensität enthalten.

2.5.3 Vergleich

Die Verfahren stimmen nur näherungsweise überein. Für ein Testbild ergab sich ein maximaler Fehler von 4.5% und ein mittlerer von 1.4%. Im 1. Verfahren erfolgt die exakte Drehung von 4 Punkten auf 4 verschiedenen Kreisen richtig, im 2. Verfahren wird die Drehung näherungsweise auf einem mittleren Kreis durchgeführt. Der Nachteil des 1. Verfahrens liegt in der etwas größeren Ungenauigkeit wegen der Festkomma-Umwandlung in 4 Schritten. Der Nachteil des 2. Verfahrens liegt darin, daß es nur näherungsweise stimmt.
Beide Verfahren ersetzen bei der Drehung im allgemeinen 1 Bildpunkt durch 4 Punkte, damit werden durch die Drehung Konturen verschmiert. Nur bei der Drehung um einen Drehwinkel von $n\ 90^o;\ n = 1, 2, ...$ erfolgt eine Punkt zu Punkt Abbildung für alle Bildpunkte. Für Drehwinkel $\neq 90^o$ kann das für wenige Bildpunkte auch gelten.

2.5.4 Ränder

Besondere Schwierigkeiten ergeben sich auf den Rändern. Es ist zwar einfach, einen Algorithmus anzugeben, bei dem über die Flächen zwischen einem Punkt des Ziel- und des Originalgitters die Normerhaltung auf den innenliegenden Punkten gewährleistet ist, die Ränder sind dabei aber möglicherweise nicht richtig. Die Schwierigkeit wird behoben, wenn die Kennzifferberechnung auf den inneren Kreis beschränkt wird.

Ein anderer Effekt bei der Drehwinkelbestimmung ist der, daß die FFT eines gedrehten Bildes deutliche Maxima durch die Bildränder bekommt. Betrachtet man einfach nur die innere Kreisfläche bzw. setzt man die außerhalb des inneren Kreises liegenden Flächen auf Null, so tritt dieser Effekt nicht auf. Die kreisförmige Kante gibt dann rotationssymetrische Maxima im Frequenzraum und mittelt sich dann bei der Kennzifferberechnung für verschiedene Drehwinkel heraus.

2.5.5 Anfangsdrehung

Die Tatsache einer Verschmierung schon für kleine Drehwinkel hat die Konsequenz, daß beim Vergleich von 2 Bildern zunächst eine Drehung um einen Anfangsdrehwinkel nötig ist. Sonst entartet das Minimum der Kennziffer zu einer so scharfen Nadel, daß die Rückdrehung durch eine Minimumsuche nicht mehr möglich ist. Als Anfangsdrehwinkel wird in der Regel $\pi/4$ verwendet.

2.6 Numerische Extremwertfindung

Nachdem wir eine Funktion gefunden haben, die ein Minimum an der gesuchten Sollposition der Farbauszüge hat, brauchen wir einen Algorithmus, der dieses Minimum findet. Da der Bereich außerhalb der Überlappung durchaus von den Algorithmen benötigt werden kann, wird er bei Funktionsaufrufen auf den höchstmöglichen Wert gesetzt.

Hier wurden zwei Algorithmen aus [17] („Numerical Recipes“) für die Kennzifferminimierung verwendet. Dabei waren Anpassungen nötig, um mit ihnen auch nicht stetige Funktionen effizient verarbeiten zu können. ro_x, ro_y verändern sich nur diskret um 1. Die Algorithmen schlagen aber Zwischenwerte vor, an denen sie das Minimum vermuten. Ein weiterer wichtiger Punkt zur Verkürzung der Rechenzeit ist, eine Pufferung für ermittelte Kennziffern einzuführen. Es gibt zwar in einem gewissen Umfang in den Extremwertalgorithmen eine Pufferung, aber erstens vermeidet diese nicht unter allen Umständen eine Mehrfachberechnung und zweitens haben die Algorithmen selbst keine Kenntnis davon, daß die Kennzifferfunktion eine Stufenfunktion ist, die für verschiedene Ortsvektoren innerhalb desselben Quadrats immer zur gleichen Kennziffer führt.

Die Minimierung der Kennziffer in Abhängigkeit von dem Drehwinkel wurde mit der Methode der inversen parabolischen Interpolation (Brent-Algorithmus) realisiert.

Die Zeit, die innerhalb der Extremwertalgorithmen verbracht wird, ist vernachlässigbar gegenüber der Zeit, die zur Kennziffernberechnung gebraucht wird. Ziel ist es daher, mit möglichst wenigen Kennziffernberechnungen das Minimum der Kennziffernfunktion zu finden. Bei beiden benutzten Algorithmen zeigt sich, daß sie sehr schnell in die Nähe der Sollposition laufen. In der unmittelbaren Nähe der Sollposition werden dagegen in der Regel alle Kennziffern berechnet. Es könnte durchaus sein, daß sich das durch Optimierung der Abbruchkriterien vermeiden läßt. Das würde die Rechenzeit weiter stark reduzieren. Im Rahmen der Diplomarbeit wurde darauf verzichtet.

2.6.1 Extremwertfindung mit der „Powell“-Methode

Der Powell-Algorithmus ist ein Verfahren, das mit Hilfe zweier drehbarer Vektoren den Weg zum Minimum „suchen“ kann. Entlang eines Vektors wird das Minimum eines durch 3 Punkte festgelegten Polynoms 2. Grades bestimmt. Initialisiert wird das Verfahren mit zwei orthogonalen Vektoren. Im Laufe der Suche haben die Vektoren einen Hang zur Parallelstellung und damit zur linearen Abhängigkeit. Die Programme überwinden das Problem mit vielen numerischen Kniffen.

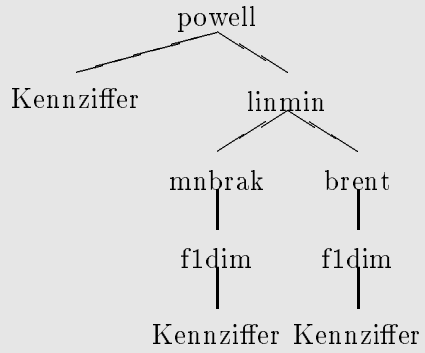


Abbildung 8: Die Schachtelung der Funktionsaufrufe

2.6.2 Extremwertfindung mit der „Downhill-Simplex“-Methode

Die „Downhill-Simplex“-Methode („Amoeba“) von Nelder und Mead ([14]) ist eine Methode, bei der sich ein aus 4 Punkten bestehender Simplex durch Reflexion, Expansion und Kontraktion seiner Punkte auf ein Minimum zubewegt. Der Programmname „Amoeba“ aus [17] erinnert an das Verhalten einer Amöbe.

Das Verfahren benötigt folgende Schritte:

- Eine Reflexion des Punktes P_h mit dem höchsten Funktionswert am Schwerpunkt $\bar{P}$ der 3 Punkte ($\alpha = 1$). Ist der Funktionswert auch dann niedriger, wird eine Expansion ($\gamma = 2$) versucht: Schrittweitenkontrolle.

- Eine Kontraktion des höchsten Punktes P_h zum Schwerpunkt $\bar{P}$ hin wird versucht, wenn die Reflexion fehlschlägt ($\beta = \frac{1}{2}$). Eine Kontraktion in Richtung des tiefsten Wertes ist auch vorgesehen.

$$P^* = \bar{P} - \alpha(P_h - \bar{P}) \qquad \text{Reflexion} \qquad (\alpha = 1)$$

$$P^{**} = P^* - \gamma(\bar{P} - P^*) \qquad \text{Expansion} \qquad (\gamma = 2)$$

$$P^{**} = \bar{P} - \beta(P_h - \bar{P}) \qquad \text{Kontraktion} \qquad (\beta = \frac{1}{2})$$

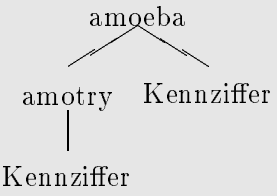


Abbildung 9: Die Schachtelung der Funktionsaufrufe

2.6.3 Extremwertfindung mit der Methode der „inversen parabolischen Interpolation“

Der Brent-Algorithmus aus [17] stellt ein Verfahren zur Minimierung dar, das ohne Ableitungen auskommt. Mit Hilfe von quadratischer Interpolation werden neue Stützstellen berechnet und auf Konvergenz geprüft. Damit eine neue Stützstelle akzeptiert wird, muß sie innerhalb der Schranken sein und weniger als halb so weit vom bis jetzt besten Wert entfernt sein. Wenn die quadratische Interpolation keinen akzeptablen neuen Wert hervorgebracht hat, wird ein Schritt über einen goldenen Schnitt berechnet.

Abbildung 10: Die Schachtelung der Funktionsaufrufe

2.7 Fouriertransformation

In unserem Fall benutzen wir eine Fouriertransformation ([4][22][23][12][7][16]), um ein Bild f in ein neues Bild F zu transformieren, daß unabhängig von der Lage des ursprünglichen Bildes den gleichen Inhalt hat. Durch diese Eigenschaft ist es möglich, das Problem der Kennzifferminimierung von 3 Dimensionen (X-Koordinate,Y-Koordinate,Winkel) auf eine Dimension (Winkel) zu reduzieren. Dadurch erreicht man eine dramatische Reduzierung der Rechenzeit.

Diese Reduzierung gilt allerdings nicht für das komplexe Ergebnis F einer Fouriertransformation einer reellen Funktion wie unserer Grauwertfunktion, sondern für die Amplitude $|F|$.

Es ist für die Winkelsuche nicht unbedingt notwendig, die volle Bildauflösung zu transformieren, sondern man kann auch mit einer niedrigeren Auflösung rechnen. Darauf beruht die Anwendung der FFT aus [17]. Wir skalieren das Eingabebild auf eine niedrigere Auflösung und bestimmen den Drehwinkel. Die FFT aus [17] basiert nur auf Arrays, deren Abmessungen Potenzen von 2 sind. Die Bilder werden deshalb auf diese Auflösung umgerechnet.

Da unsere Grauwertfunktion reell ist, gilt für die Fouriertransformierte – abgesehen von der Periodizität – folgende Symmetrie:

$$H(-\vec{n}) = H^*(\vec{n}) \tag{18}$$

Dabei ist H die Fouriertransformation und $\vec{n}$ ein Vektor mit Werten auf dem Gitter der ganzen Zahlen.

Zu einer tiefergehenden Betrachtung der Theorie der Fouriertransformationen sei auf [19] Seite 158 verwiesen. Die Fouriertransformierte $F(k_x, k_y)$ von f berechnet sich folgendermaßen:

$$F(k_x, k_y) = \int_x \int_y f(x,y) e^{-i(k_x x + k_y y)} dy dx \qquad mit \quad i = \sqrt{-1} \tag{19}$$

Im diskreten Fall geht das Integral in die Summe über. Bei einem Eingabebild von den Abmessungen $N_x \cdot N_y$ berechnet sie sich als:

$$F(k_x, k_y) = \frac{1}{N_x N_y} \sum_{x=0}^{N_x - 1} \sum_{y=0}^{N_y - 1} f(x,y) e^{-i(k_x x + k_y y)} dy dx \tag{20}$$

Der Betrag $|F(k_x, k_y)| \in \mathbb{R}$, das Amplitudenspektrum, ergibt sich einfach zu:

$$S(k) = \sqrt{Re^2(F(k_x, k_y)) + Im^2(F(k_x, k_y))} \tag{21}$$

Eine Fouriertransformation, wie sie bis in die 70er Jahre angewendet wurde, ist viel zu langsam. Sie benötigt $O(N^2)$ Multiplikationen für $H_n = \sum_{k=0}^{N-1} W^{kn} h_k$. Deshalb verwenden wir eine FFT (Fast-Fourier-Transformation), die mit einem viel geringeren Aufwand [$O(N \log N)$ arithmetische Operationen] auskommt. Das Verfahren beruht auf einem Beweis des Danielson-Lanczos Lemmas, der es gestattet, eine diskrete Fouriertransformation der Länge N (Voraussetzung: N ist gerade) als Summe zweier diskreter Fouriertransformationen der Länge $N/2$ zu schreiben.

FFT's, die dieser Einschränkung unterliegen, heißen „Cooley-Tukey-Radix-2-FFT-Algorithmen". FFT's, die auch auf $N \neq 2^n$-Gittern arbeiten, heißen „Mixed-Radix-FFT-Algorithmen".

Die FFT, die hier aus [17] angewendet wurde, besteht aus zwei Funktionen:

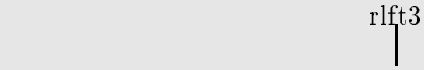


Abbildung 11: Die Schachtelung der Funktionsaufrufe bei rlft3

Diese FFT arbeitet auf dreidimensionalen Datenfeldern. Zur Benutzung auf zweidimensionalen Datenfeldern setzt man die äußere Arraygröße 1. Dann wird die äußere Schleife nur 1x durchlaufen.

Generell kann eine zweidimensionale FFT einfach unter Benutzung einer eindimensionalen FFT abgeleitet werden, indem zunächst der Reihe nach die Spalten eines zweidimensionalen Zahlenfeldes mit Hilfe einer eindimensionalen FFT transformiert werden. Anschließend werden von der entstandenen zweidimensionalen Zahlenmatrix die Zeilen nacheinander fouriertransformiert. Die Ergebnisse der eindimensionalen Fouriertransformationen müssen jeweils in das zweidimensionale Zahlenfeld zurückgeschrieben werden.

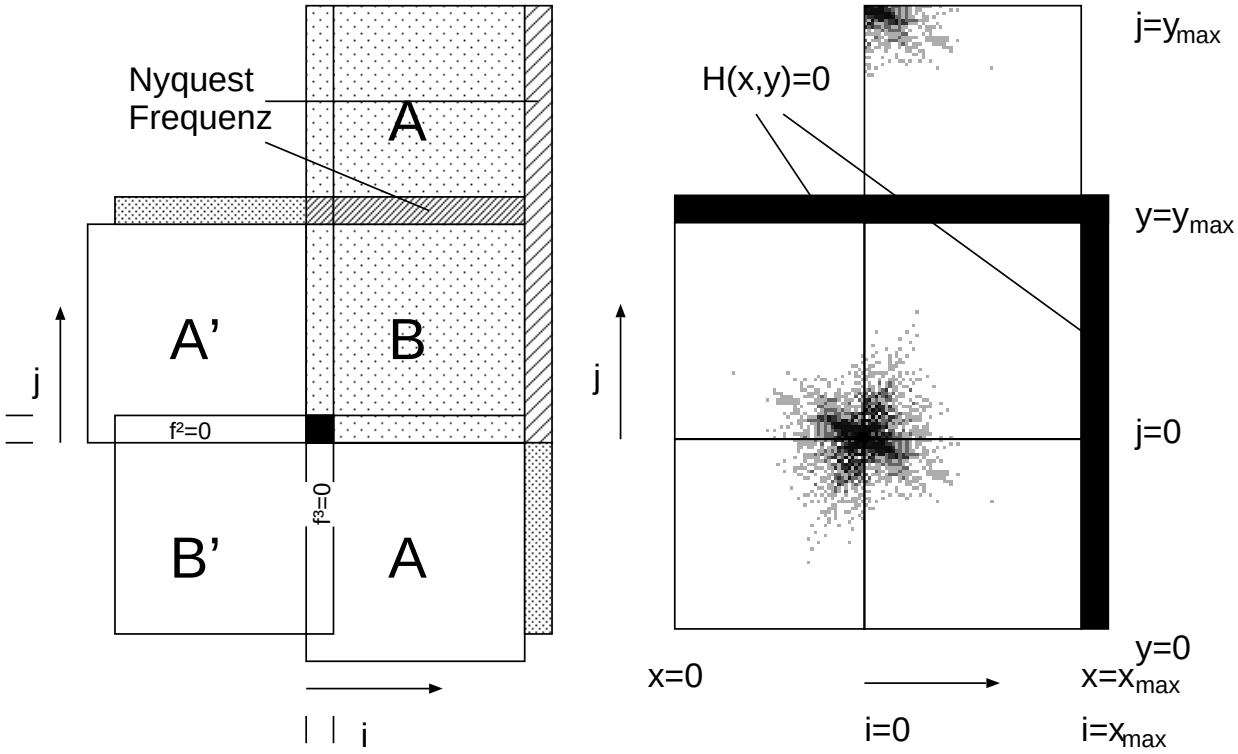
Drei Dinge gibt es allerdings zu beachten:

- Bei einigen Algorithmen stimmt die Normierung der FFT nicht. Es handelt sich dabei allerdings nur um konstante Faktoren. Die FFT-Werte des Programms rlft3 aus [17] sind noch durch N^2 zu dividieren. Die FFT-Rechnung von Mathematica dagegen hat diesen Faktor bereits berücksichtigt.

- Die Nullfrequenzen liegen bei den meisten Algorithmen nicht in der Mitte sondern am Rand. Deshalb ist eine Umordnung nötig.

- Das Ausgabe-Feld der FFT ist um eine Zeile und eine Spalte größer als das Eingabe-Feld. Die zusätzlichen Feldelemente enhalten die Werte für die maximalen Frequenzen. Das sind die gleichen, wie an der entsprechenden Stelle mit der negativen Frequenz. In rlft3 aus [17] werden diese zusätzlichen Werte in einem zusätzlichen Feld abgelegt. Hier werden diese Werte weggelassen. Das Ausgabe-Feld ist dadurch symetrisch um einen Punkt, der einen halben Bildpunkt vertikal und horizontal verschoben ist. Der Vorteil ist, daß die FFT in ein Feld der gleichen Größe paßt. Bei der Drehung im Frequenzraum muß diese Asymetrie berücksichtigt werden.

Zu der Umordnung ist in Abbildung 12 für das Programm rlft3 [17] dargestellt, wie man durch Ergänzung, Umordnung und Punktspiegelung ein komplettes Frequenzraumbild gewinnt.



(a) Schema (b) Beispiel

Abbildung 12: Um eine Darstellung mit den Nullfrequenzen in der Mitte zu erhalten, müssen die Quadranten umgeordnet und eine Punktspiegelung durchgeführt werden. Die große, grau unterlegte Fläche markiert die Feldausgabe von rlft3 nach [17] [G(i,j)]. A' und B' werden durch Punktspiegelung aus A und B gewonnen. Das Beispielbild zeigt die Amplitude im Frequenzraum. Die Feldelemente mit der Nyquistfrequenz werden hier auf Null gesetzt. Die schraffierte Spalte am rechten Rand steht für ein zusätzliches Feld, in dem diese Feldelemente mit Nyquistfrequenz von rlft3 zurückgegeben werden. Rechts oben im Schema ist der schraffierte Bereich der Nyquistfrequenzen zu erkennen. Das Beispiel und die Schemazeichnung sind nicht maßstabsgetreu. Das Feld G(i,j) geht von 0 bis xmax, weil der reelle und der imaginäre Anteil der komplexen Ausgabe im Feld immer aufeinander folgen.

- G(i,j):Feld aus FFT mit $1 \leq i \leq \mathit{bildbreite}, 1 \leq j \leq \mathit{bildhöhe}$ und mit $i = j = 0$ im Koordinatenursprung der Frequenzkoordinaten
- H(x,y):Ausgabe-Feld mit $x = y = 0$ in der linken unteren Ecke

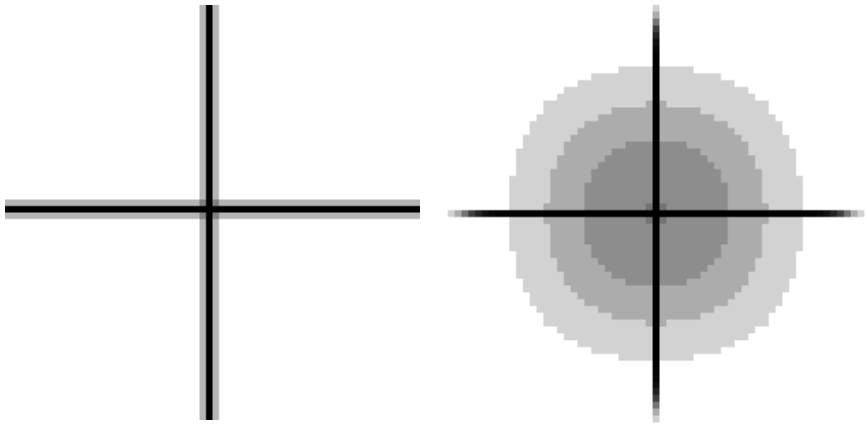
In den angegebenen Gleichungen kann man die Transformationen ablesen, die in Abbildung 12 angedeutet sind. „xmax“ und „ymax“ sind Bildbreite und Bildhöhe.

Quadrant			Bereich
B’		$H(\frac{xmax}{2}-(x-1),\frac{ymax}{2}-(y-1))$	
	$=$	$\sqrt{G^2(2x-1,y)+G^2(2x,y)}$	$\binom{1\le x\le xmax}{1\le y\le ymax}$
B		$H(\frac{xmax}{2}+(x-1),\frac{ymax}{2}+(y-1))$	
	$=$	$\sqrt{G^2(2x-1,y)+G^2(2x,y)}$	$\binom{1\le x\le xmax}{1\le y\le ymax}$
A		$H(\frac{xmax}{2}+(x-1),(y-1))$	
	$=$	$\sqrt{G^2(2x-1,\frac{ymax}{2}+y)+G^2(2x,\frac{ymax}{2}+y)}$	$\binom{1\le x\le xmax}{2\le y\le ymax}$
A’		$H(\frac{xmax}{2}-(x-1),2\frac{ymax}{2}-(y-1))$	
	$=$	$\sqrt{G^2(2x-1,\frac{ymax}{2}+y)+G^2(2x,\frac{ymax}{2}+y)}$	$\binom{2\le x\le xmax}{1\le y\le ymax}$

Tabelle 1: Die Transformationsgleichungen im Überblick

Zur Fehlerkontrolle wurden die Ergebnisse der FFT mit entsprechenden Ausgaben von Mathematica verglichen. Um sicher zu sein, daß beide Programme mit denselben Daten rechnen, wurde einfach eine Funktion geschrieben, die einen Farbauszug in einem Mathematica-kompatiblen Programm als Liste ausgibt.

Um einen kleinen Eindruck davon zu bekommen, wie sich die Fouriertransformation auswirkt, sind im folgenden ein paar Beispiele abgebildet.



(a) Ortsraum (b) Frequenzraum

Abbildung 13: Wirkung der Fouriertransformation, Fadenkreuz

In Abbildung 13 ist gezeigt, wie ein Bild eines Kreuzes im Ortsraum aussieht, wenn man

es in den Frequenzraum transformiert. Das Bild im Frequenzraum zeigt dabei den Betrag des FFT-Bildes.

(a) Ortsraum (b) Frequenzraum

Abbildung 14: Zur Wirkung der Fouriertransformation, Streifenbild um $6,6^o$ gedreht

In Abbildung 14 ist gezeigt, wie ein Bild eines Streifenmusters im Ortsraum aussieht, wenn man es in den Frequenzraum transformiert.

(a) Ortsraum (b) Frequenzraum

Abbildung 15: Zur Wirkung der Fouriertransformation, Streifenbild um $6,6^o$ gedreht, x2

In Abbildung 15 ist gezeigt, wie ein Bild eines Streifenmusters im Ortsraum aussieht, wenn man es in den Frequenzraum transformiert. Das Streifenmuster ist um den Faktor 2 größer als in Abbildung 14. Der Abstand der Peaks im Frequenzraum hat sich halbiert.

2.7.1 Translationsinvarianz des Frequenzraumbildes

Damit ist die Invarianz der Abbildung im Frequenzraum unter Translationen des Urbildes im Ortsraum gemeint ([19], Seite 158). Eine Translation läßt sich durch folgende Abbildung beschreiben:

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} x \\ y \end{pmatrix} - \begin{pmatrix} x_0 \\ y_0 \end{pmatrix}$$

Durch Einsetzen erhalten wir:

$$\begin{aligned} F'(k_{x'}, k_{y'}) &= \int_x \int_y f(x,y) \exp[-jk_{x'}(x - x_0) - jk_{y'}(y - y_0)] dy dx \\ &= \exp[j(k_{x'}x_0 + k_{y'}y_0)] \int_x \int_y f(x,y) \exp[-jk_{x'}x - k_{y'}y] dy dx \end{aligned}$$

Damit haben wir:

$$F'(k_{x'}, k_{y'}) = \exp[j(k_{x'}x_0 + k_{y'}y_0)] F(k_x, k_y)$$

und:

$$\begin{aligned} F'^* F' &= (\exp[jk]F)^* (\exp[jk]F) \\ &= \exp[jk]^* \exp[jk] F^* F \\ &= F^* F \end{aligned}$$

Daher gilt im Frequenzraum:

$$\begin{pmatrix} k_{x'} \\ k_{y'} \end{pmatrix} = \begin{pmatrix} k_x \\ k_y \end{pmatrix}$$

2.7.2 Rotationsinvarianz des Frequenzraumbildes

Damit ist die Invarianz der Abbildung im Frequenzraum unter Rotationen des Urbildes im Ortsraum gemeint. Eine Rotation läßt sich durch folgende Abbildung beschreiben:

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

Mit der Drehmatrix Θ und dem Drehwinkel θ:

$$J(x,y) = ||\Theta|| = |\cos^2\theta + \sin^2\theta| = 1$$

$$\mathbf{r}' \equiv \Theta\mathbf{r}$$

Durch Einsetzen erhalten wir:

$$\begin{aligned} F'(\mathbf{k}_{\mathbf{r}'}) &= \int_{\mathbf{r}'} f'(\mathbf{r}') \exp[-\mathbf{j}\mathbf{k}_{\mathbf{r}'}^{\mathbf{T}}\mathbf{r}'] \mathbf{d}\mathbf{r}' \\ &= \int_{\mathbf{r}} f(\mathbf{r}) \exp[-\mathbf{j}\mathbf{k}_{\mathbf{r}'}^{\mathbf{T}}\Theta\mathbf{r}] \mathbf{d}\mathbf{r} \end{aligned}$$

Wir bezeichnen den räumlichen Frequenzbereich, der im obigen Exponenten vorkommt, wie folgt:

$$k_{\mathbf{r}}^T \equiv k_{\mathbf{r}'}^T \Theta$$

Wenn wir das in das Integral einsetzen, bekommen wir folgendes:

$$\begin{aligned} F'(\mathbf{k}_{\mathbf{r}'}) &= \int_{\mathbf{r}} f(\mathbf{r}) \exp[-\mathbf{j}\mathbf{k}_{\mathbf{r}}^{\mathbf{T}}\mathbf{r}]\mathbf{dr} \\ &= F(\mathbf{k}_{\mathbf{r}}) \end{aligned}$$

Transponiert man nun Gleichung (2.7.2) auf beiden Seiten, kann man schreiben:

$$k_{\mathbf{r}} = \Theta^T k_{\mathbf{r}'}$$

Eine der Eigenschaften der Drehmatrix ist nun, daß ihre Inverse gleich ihrer Transponierten ist:

$$\Theta^T = \Theta^{-1}$$

$$k_{\mathbf{r}'} = \Theta k_{\mathbf{r}}$$

Das bedeutet, daß die Fouriertransformation einer um einen bestimmten Winkel gedrehten zweidimensionalen Funktion gleich der um den gleichen Winkel gedrehten Fouriertransformation einer ungedrehten zweidimensionalen Funktion ist.

Damit haben wir:

$$F'(k_{x'}, k_{y'}) = F(k_x, k_y)$$

Daher gilt im Frequenzraum:

$$\begin{pmatrix} k_{x'} \\ k_{y'} \end{pmatrix} = \begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} k_x \\ k_y \end{pmatrix}$$

2.7.3 Skalierung des Frequenzraumbildes

Mit a,b konstant:

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} ax \\ by \end{pmatrix}$$

und:

$$J(x, y) = |ab|$$

Damit haben wir:

$$F'(k_{x'}, k_{y'}) = |ab| F(k_x, k_y)$$

$$\begin{pmatrix} k_{x'} \\ k_{y'} \end{pmatrix} = \begin{pmatrix} \frac{k_x}{a} \\ \frac{k_y}{b} \end{pmatrix}$$

Für die Skalierung (Vergrößerung oder Verkleinerung) eines Bildes im Ortsraum gilt, daß das Frequenzraumbild entsprechend mit dem Kehrwert skaliert, d. h. bei einer Vergrößerung des Ortsraumbildes wird das Frequenzraumbild verkleinert.

2.7.4 Weitere Anwendungen der FFT

Neben der Eigenschaft der Translationsinvarianz läßt sich die FFT auch noch zum Skalieren von Bildern nutzen. Man bekommt eine Skalierungsoperation sehr hoher Qualität. Angenommen man hat ein Bild von 256x256 Punkten und will dieses Bild auf die Maße 128x128 bringen. Dann kann man im FFT-Bild die hochfrequenten Anteile unberücksichtigt lassen, so daß das Restbild genau 128x128 Punkte hat. Eine FFT-Rücktransformation dieses Teilbildes transformiert dieses kleinere Bild in den Ortsraum zurück. Genauso ist auch eine Vergrößerung durch Ergänzen der hochfrequenten Anteile im Frequenzraum mit Nullen und anschließender Rücktransformation möglich.

Zu beachten ist, daß man bei Skalierungen um wenige Zeilen zu einer FFT eines Feldes mit den Abmessungen der nächstgrößeren Zweierpotenz greifen muß. Diese Verschwendung von Speicherplatz ist aber normalerweise ein geringes Problem.

Klassische Anwendungen sind Hoch- und Tiefpassfilter und das gezielte Ausblenden von periodischen Störungen. Außerdem haben vor allem in den letzten Jahren ihre Bedeutung bei der Datenkompression (JPEG) und bei der Kryptologie zugenommen. Bei der Faltung bringt der Übergang in den Frequenzraum im Fall großer Masken zum Teil erhebliche Rechenzeitvorteile. In Abbildung 16 sieht man durch Pfeile angedeutet die Fouriertransformation.

$$\begin{array}{lccccc} Ortsraum & g(\boldsymbol{X}) & = & \int h(\boldsymbol{X}-\boldsymbol{\tau}) & & g'(\boldsymbol{X})d\tau \\ & \uparrow & & \downarrow & & \downarrow \\ Frequenzraum & G(\boldsymbol{k}) & = & H(\boldsymbol{k}) & & G'(\boldsymbol{k}) \end{array}$$

Abbildung 16: Aus der Faltung wird beim Übergang in den Frequenzraum eine Multiplikation. Dabei wird zunächst h und g' fouriertransformiert, das Produkt $G = H * G'$ gebildet und das FFT-Bild G rücktransformiert.

2.8 Zusammenfassung

In diesem Kapitel wurden eine Reihe von gebräuchlichen Filtern vorgestellt. Darunter waren der Laplace-, der Sobel- und Gaußfilter. Ein weiterer wichtiger Punkt war die Schwerpunktbestimmung. Sie ist bei der Offsetbestimmung nicht sehr genau, liefert aber in der Regel brauchbare Anfangswerte. Für eine abschließende genaue Bestimmung der Offsets wurden zwei Operatoren, die Differenzformel und die Produktformel vorgestellt. Außerdem wurden zwei einfache Verfahren zur Drehung von Bildern erklärt. Die für die Drehwinkelanalyse entscheidende FFT und die in diesem Zusammenhang relevanten Eigenschaften wurden kurz zusammengestellt und exemplarisch erläutert.

3 Implementation

Alle vorgestellten Methoden wurden wegen ihrer rechenzeitintensiven Algorithmen in C unter Unix implementiert. Um eine angemessene Zykluszeit bei Experimenten mit verschiedenen

Parametern zu erhalten, wurde ein einfaches interaktives Benutzerinterface in Motif ([3]) implementiert. Zur Versionsverwaltung und der Erleichterung bei der Entwicklung auf mehreren Maschinen wurde CVS eingesetzt. Dadurch ist ein hoher Grad an Portierbarkeit zu jedem Zeitpunkt sichergestellt.

Die zwei wichtigen Schritte, Drehwinkelbestimmung und Offsetberechnung sowie die komplette Rejustierung eines Eingabebildes liegen jeweils auch getrennt als Programm vor. Die implementierten Algorithmen wurden dabei in Form einer Linkbibliothek realisiert, um ein gewisses Maß an Trennung von Form und Funktion zu erreichen.

An einigen Stellen der Implementation wurde ein Kompromiß zwischen der Rechengenauigkeit und der Rechenzeit gewählt, der unter geringfügig anderen Vorgaben auch anders aussehen kann. Zum Beispiel wird im Eingabebild bei der hier vorliegenden Implementation eine Auflösung von mindestens 16 Millionen Farben (8 Bit pro Farbauszug) vorausgesetzt. Maximal werden 16 Bit pro Farbauszug verarbeitet. Reduziert man diese Farbtiefe, wird der Algorithmus ab einer bestimmten Farbtiefe nicht mehr in der Lage sein, das Minimum zu finden.

Damit in der Kennzifferberechnung kein Überlauf auftritt bzw. kein erheblicher Anteil an Genauigkeit verloren geht, wird die Kennziffer vom Typ Fließkommazahl berechnet. Die Graustufen der Eingabebilder sind aber vom Typ Ganzzahl.

3.1 Grob-Konzept

In diesem Kapitel sollen zuerst die Teilschritte des Programms und dann am Ende der Prozeß der Ausrichtung im ganzen Programm erläutert werden. Damit die Teilschritte aber jetzt schon grob eingeordnet werden können in wenigen Sätzen das Konzept:

Die Ausrichtung wird in zwei Schritten mit jeweils zwei Farbauszügen durchgeführt. Der Rotauszug dient als Referenz, alle Offsets von Grün- und Blauauszügen beziehen sich auf den Rotauszug.

Da, wie man an den in Kapitel 2 betrachteten Eigenschaften der FFT sieht, die Drehung und die Offsetbestimmung trennen kann, soll zunächst die Drehwinkel-Bestimmung erfolgen und anschließend unter Berücksichtigung des bestimmten Drehwinkels die Offset-Bestimmung.

Vor der Offsetbestimmung mit Hilfe eines flächenorientierten Minimierungsverfahrens wird ein Startwert bestimmt, indem die Schwerpunkte der Farbauszüge miteinander verglichen werden.

3.2 Drehwinkelbestimmung

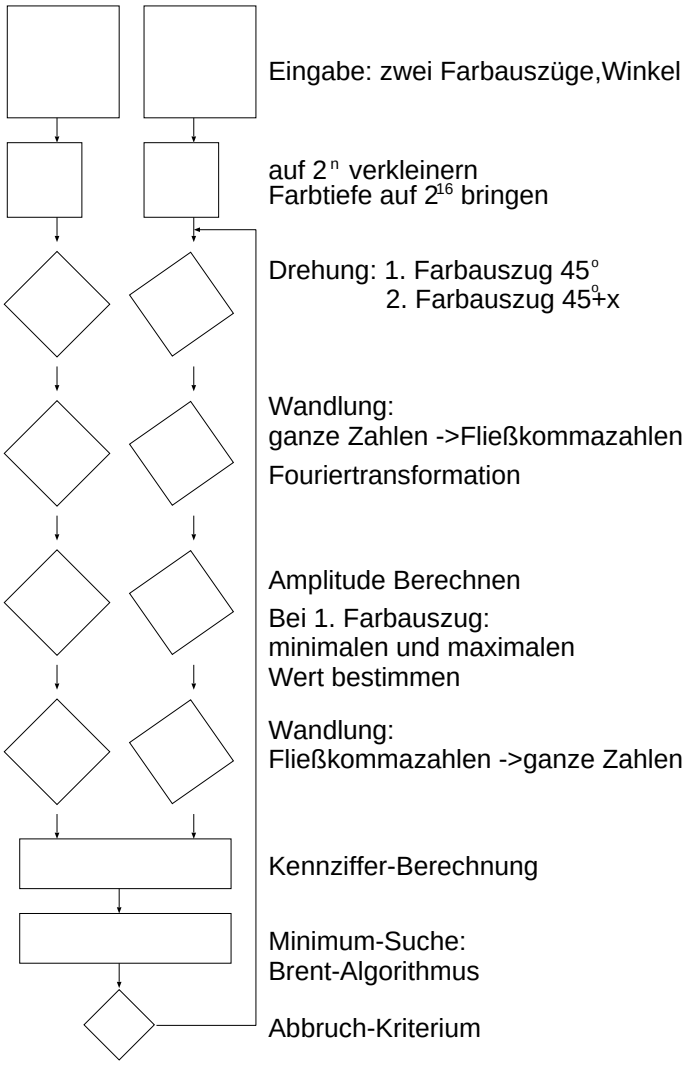


Abbildung 17: Kennzifferberechnung für Drehwinkelbestimmung

In Abbildung 17 ist gezeigt, welche Transformationen bei einer Kennzifferberechnung für einen speziellen Drehwinkel zweier Farbauszüge zueinander durchlaufen werden. Zusätzlich werden beide Farbauszüge gemeinsam um 45^o gedreht. Dies liegt daran, daß ein ungedrehtes, scharfes Bild schon bei einer Drehung um Bruchteile von 1^o über 4 Nachbarpunkte verschmiert. Diese Verschmierung führt zu einer Spitze in der Kennzifferfunktion in Abhängigkeit von dem Drehwinkel, die die Minimumsuche erschwert. Werden beide Bilder vorher um einen zusätzlichen Winkel gedreht, tritt diese Spitze nicht im Drehwinkelbereich, der bei der Minimumsuche durchlaufen wird, auf.

Besondere Aufmerksamkeit verdient hier die Normierung. In der gewählten Lösung ist ein systematischer Fehler, der allerdings in den Versuchen keine spürbaren Auswirkungen hat. Der Farbauszug liegt nach der FFT als Fließkommafeld vor. Die Feldwerte werden mit Hilfe des minimalen und maximalen Wertes des Fließkommafeldes in ein Ganzzahlfeld transformiert. Dabei geht natürlich ein Teil der Information verloren.

Wichtig bei der vorliegenden Implementation ist allerdings, daß die minimalen und maximalen Werte nur auf dem FFT-Feld des ungedrehten Farbauszuges bestimmt werden. Dies ist nötig, weil die durch die Normierung entstehenden Farbauszüge unbedingt vergleichbar sein müssen und daher bezüglich des gleichen minimalen und maximalen Wertes normiert werden müssen. Sonst wäre es sogar günstiger, auf die Wandlung in ein Ganzzahlfeld zu verzichten und ganz im Fließkommafeld zu rechnen mit allen Nachteilen:

- großer Speicherbedarf,
- erheblich langsamere Berechnung.

Wie in der theoretischen Betrachtung gezeigt wurde, ist es nicht unbedingt notwendig, erst die Farbauszüge zu drehen und dann die Fouriertransformation vorzunehmen. Genauso ist es möglich, erst die Fouriertransformation durchzuführen und im zweiten Schritt die Frequenzraumbilder zu drehen. Das reduziert die Rechenzeit ganz erheblich, da die rechenzeitintensive FFT-Berechnung für jeden neuen Winkel entfällt.

3.3 Offsetberechnung

Zunächst wird der Schwerpunkt der Farbauszüge bestimmt. Daraus ergibt sich ein Offset, der als Startwert für die folgende Offsetbestimmung dient.

Vor jeder Offsetberechnung werden einige Datenstrukturen verwaltet. Diese Verwaltung benötigt unabhängig von der Minimierung Rechenzeit und Ressourcen. Im einzelnen gehören zu dieser Verwaltung folgende Schritte:

- Anlegen und Initialisiern des Puffers für die Kennzifferberechnung,
- Anlegen zweier Farbauszugkopien,
- Kopieren und Verkleinern der Farbauszüge,
- Farbtiefe auf 16 Bit pro Farbauszug umrechnen,
- jeweils mit dem vorher bestimmten Winkel die Drehung vornehmen,
- Maske jeweils initialisieren.

Der Puffer der Kennzifferberechnung ist durch ein auf Null initialisiertes Fließkommafeld realisiert. Jede berechnete Kennziffer ersetzt an der entsprechenden Stelle die Null. Dieses Vorgehen zeichnet sich vor allem durch die Einfachheit seiner Implementierung aus. Zwar ist es auch ein sehr schneller Algorithmus, aber da der größte Teil der Rechenzeit in der Kennzifferberechnung verbraucht wird, wirkt sich ein langsamerer Algorithmus an dieser Stelle kaum aus. Wenn eine Liste von berechneten Koordinaten verwaltet wird, kann der Speicherbedarf

sehr stark schrumpfen. Eine solche Verbesserung an dieser Stelle ist deshalb wünschenswert, in der Diplomarbeit wurde aber darauf verzichtet.

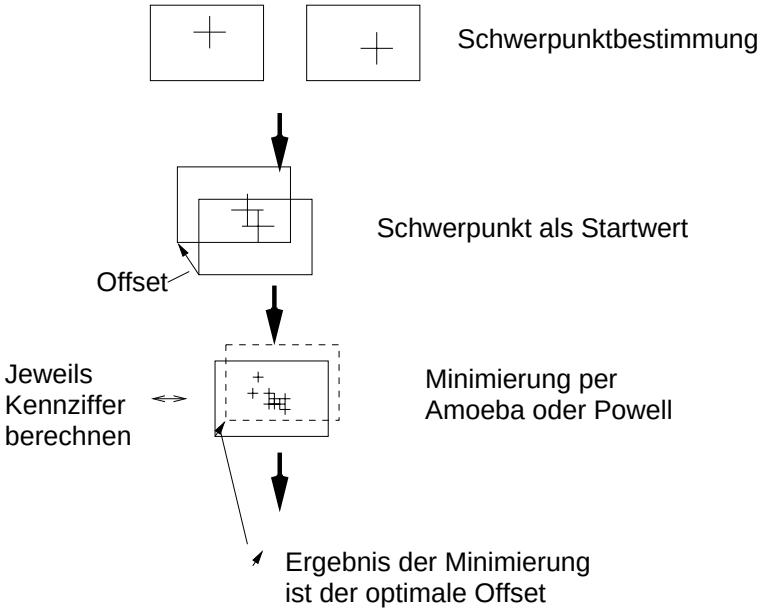


Abbildung 18: Offsetberechnung durch Minimierung der Kennziffer

Abbildung 18 zeigt die Berechnung des Offset in ihren groben Zügen. Ein RGB-Bild wird als JPEG oder PPM Bild eingelesen und in seine Anteile zerlegt. JPEG erlaubt nur eine Farbtiefe von 8 Bit pro Farbauszug, PPM erlaubt auch 16 Bit. Bei JPEG kann es auch zu Einflüssen der Farbauszüge gegeneinander kommen. Zwei Farbauszüge werden gefiltert oder ungefiltert dem Kennzifferalgorithmus zugeführt. Der bekommt einen Startwert von der Schwerpunktbestimmung und liefert daran anschließend die Kennziffern für den Minimierungsalgorithmus.

3.3.1 Funktion der Maske

Zusätzlich zu den eigentlichen Farbauszügen werden in dieser Implementation noch Bildebenen gleicher Größe verwaltet, die folgende Aufgaben haben:

- Ausblenden der äußeren Bildränder nach bestimmten Filteroperationen (Ausblenden nicht definierter Bildpunkte).
- Ausblenden von durch Drehungen ins Bild geratenen Teilen der immer rechteckigen Farbauszüge, die keine echten Bildinformationen tragen.

Die Drehwinkelbestimmung liefert einen Winkel. Um diesen Winkel wird der Farbauszug gegenüber dem Rotauszug zurückgedreht. Erst danach kann die Offsetbestimmung auf dem Rotauszug und dem verdrehten zweiten Farbauszug beginnen. In der aktuellen Form wird

beim Drehen der Farbauszug in ein Bild gedreht, das die gleichen Abmessungen wie der Farbauszug hat. Daher werden überstehende Teile abgeschnitten. Im idealen Fall sollte kein Teil des Bildes verloren gehen, sondern der Farbauszug sollte nach dem Drehen in ein entsprechend großes Zielbild gerade ganz hineinpassen. Bei der Kennzifferberechnung müssen aber die Bildteile, die nach der Drehung mit Nullen gefüllt sind, unberücksichtigt bleiben. Das ist mit einer Maske realisiert, die die Flächen ohne Bildinformation markiert.

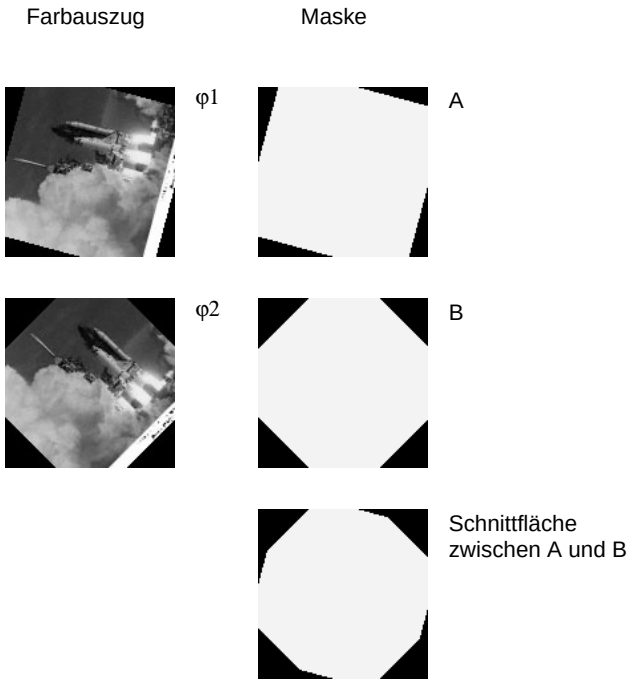


Abbildung 19: Um die korrekte Schnittfläche zu bestimmen, wird mit jedem Farbauszug eine Maske verwaltet. Muß ein Farbauszug gedreht werden, so wird die Maske mitgedreht.

Nach Abbildung 19 existiert zu jedem Farbauszug eine Maske gleicher Größe. Diese Maske wird zu Anfang auf jedem Bildpunkt mit 1 initialisiert und anschließend beim Drehen des Farbauszuges mitgedreht. Jeder Punkt der Maske wird um den Drehwinkel gedreht. Liegt der gedrehte Punkt außerhalb des Bildrechteckes wird der Punkt des Maskenbildes auf 0 gesetzt. Die Schnittfläche ergibt sich als Und-Verküpfung aus den Maskenfeldern. Das Verfahren hat den Vorteil, daß das verdrehte Bild und die zugehörige Maske völlig kongruent sind. Die Kennzifferberechnung wertet nur jene Bildpunkte aus, für die auf beiden Masken die korrespondierenden Bildpunkte gesetzt sind. So wird verhindert, daß ins Bild gedrehte leere Bildteile die Kennzifferberechnung beeinflussen.

Das gleiche Verfahren findet bei Faltungen Verwendung, bei denen aufgrund der Maskengröße die Bildränder eines gefilterten Bildes nicht definiert sind (oder entsprechend explizit definiert werden müssen). Bei der Anwendung auf Faltungen werden einfach die Bildränder auf der Maske gleich Null gesetzt.

3.4 Gesamtprozeß

Die wichtigsten Parameter des Verfahrens sind:

- Abmessungen des FFT-Bildes
- Startwerte der Drehwinkelminimierung
- Brent-Algorithmus:
  - Maximale Iterationszahl
  - Genauigkeit
- gemeinsame Vordrehung
- Startwerte des Amoeba- und Powell-Algorithmus
- Schrittweite beim Auslassungs-Verfahren
- Anzahl der Stufen beim Pyramiden-Verfahren

Die aufgeführten Parameter sind nicht alle gleichermaßen für eine Variation geeignet. Der erste Parameter (die Abmessungen des FFT-Bildes) bezieht sich auf die Größe des Feldes, auf dem die FFT stattfindet. Nicht jedes Bild hat Abmessungen von der Größe einer Zweierpotenz. Deshalb muß das Bild in der Regel skaliert werden. Von einer Feldgröße oberhalb der Bildgröße erwarten wir vor allem drastisch steigende Rechenzeiten. Interessant ist aber, ob wir ein wesentlich kleineres Feld nehmen können mit entsprechenden Vorteilen in der Rechenzeit und trotzdem noch eine ausreichende Genauigkeit bei der Winkelbestimmung bekommen.

Der zweite Parameter ist ebenfalls ein Parameter der Winkelbestimmung und bezieht sich auf die Startwerte des Brent-Algorithmus. Der Brent-Algorithmus verlangt 3 Startwerte. 2 davon liegen außen und wurden in den Untersuchungen meistens auf -5^o und $+5^o$ gesetzt. Der dritte Wert liegt dazwischen und möglichst in der Nähe des Minimums. Eine ungünstige Wahl der Startwerte kann durchaus den Erfolg beeinflussen, da durch die quadratische Interpolation relativ weite Schritte möglich sind und auch bei der Drehwinkelbestimmung Nebenminima existieren können.

Die maximale Iterationszahl und die Genauigkeit sind Parameter in der internen Verarbeitung im Brent-Algorithmus.

Die gemeinsame Vordrehung sollte außerhalb des Winkelbereichs liegen, den der Brent-Algorithmus abtastet. Wenn man allerdings immer nur wenige Grad zu justieren hat, kann immer eine Vordrehung von 45^o gewählt werden.

Bei der Offsetbestimmung werden die Koordinaten der Schwerpunktbestimmung als Startwert genommen. Der Amoeba-Algorithmus erwartet allerdings drei Koordinaten und infolgedessen wird aus der einen Koordinate ein Tripel von drei Koordinaten um den Schwerpunkt generiert. Auch hier gibt es einen Spielraum, der Einfluß auf Rechenzeit und Erfolg hat. Bei dem Powell-Algorithmus gilt das gleiche mit dem Unterschied, daß nicht ein Tripel von Koordinaten, sondern ein Paar von Vektoren gewählt werden muß.

Die letzten beiden Punkte ergeben sich aus der Untersuchung von Optimierungen und erlauben eine Rechenzeitverkürzung. Ob dabei der Zeitgewinn mit einer Verschlechterung der Genauigkeit oder Erfolgsquote einhergeht, wird im nächsten Kapitel untersucht.

Das Flußdiagramm in Abbildung 20 zeigt einen Überblick über nahezu alle Stufen des Verfahrens.

Die große Schleife deutet an, daß jeweils für zwei Farbauszüge getrennt gegenüber einem dritten die Rejustierung nacheinander vollzogen wird. In unseren Verfahren wird also konkret zunächst der Grünauszug gegenüber dem Rotauszug rejustiert.

Die beiden innen liegenden Teilverfahren auf Abbildung 20 zur Drehwinkel- und Offsetbestimmung werden nacheinander durchgeführt. Danach wird dasselbe mit dem Blauauszug gegenüber dem Rotauszug durchgeführt. Auf der linken Seite sind zur Veranschaulichung zwei Grafiken eingezeichnet, die die Kennzifferfunktion für den Verfahrensschritt zeigen. Bei der Drehwinkelbestimmung haben wir eine 1-dimensionale Kennzifferfunktion und bei der Offsetbestimmung eine 2-dimensionale Kennzifferfunktion.

Nicht aufgenommen worden sind die Verfahrensschritte zur Beschleunigung des Auslassungs-Verfahrens.

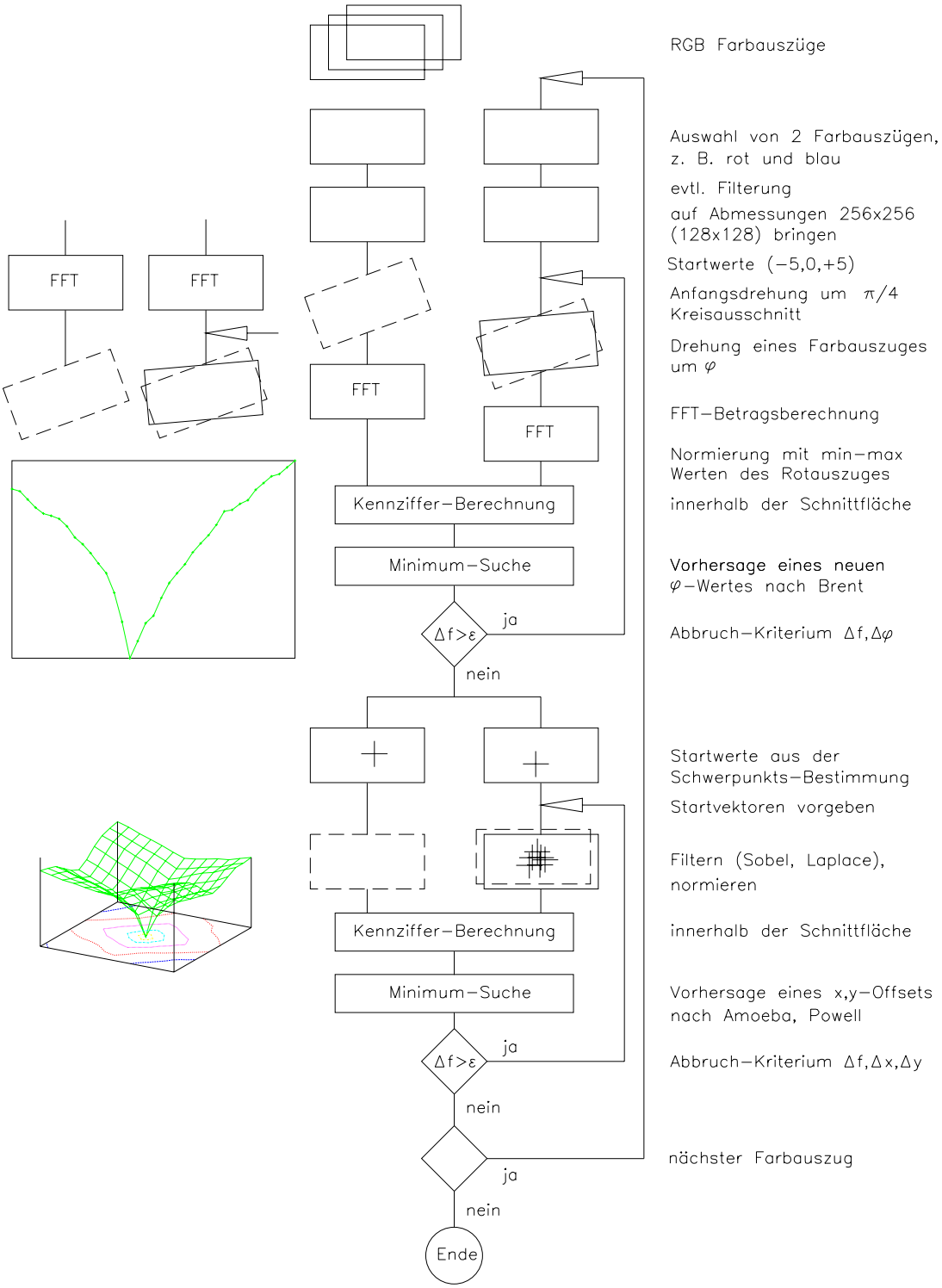


Abbildung 20: Rejustierung-Gesamtüberblick

4 Ergebnisse

Im folgenden Teil werden die angeführten Verfahren auf Brauchbarkeit hin untersucht. Zunächst wird die Kennziffer an sich auf Eignung bei der Bestimmung des Drehwinkels untersucht. Breiten Raum wird auch die Optimierung in Hinblick auf den Rechenzeitbedarf einnehmen. Zum Schluß kommen ein paar Beispiele, an denen man den Erfolg des Verfahren anhand von positionierten Bildern ablesen kann.

(a) 192x128 (b) 768×512

Abbildung 21: Eingabebilder

An den Beispielbildern in Abbildung 21 werden einige Ergebnisse gezeigt. Abbildung 22 zeigt die dazugehörigen Rot-, Grün und Blau-Auszüge. Der abgebildete Transporter hat einen sehr großen Blauanteil. Außerdem bedeckt diese blaue Fläche einen großen Teil des Bildes. Es zeigt sich, daß dies unproblematisch für das Verfahren ist. Auch sehr bunte Bilder zeigen eine Ähnlichkeit in den verschiedenen Farbauszügen. Schon die Schwerpunkte der RGB-Auszüge differieren nur noch wenig. Anschaulich heißt das: keine Ähnlichkeit bedeutet, daß sich die Farben von weiten Bildbereichen auf 3 Linien im RGB-Farbwürfel befinden müssen und nirgendwo anders (bei anderen Farbmodellen entsprechend). Dann besteht aber auch nicht das Problem der Justierung der RGB-Auszüge. Es besteht daher die begründete Hoffnung, daß natürliche Bilder von diesem Verfahren erfolgreich bearbeitet werden können.

(a) Rot (b) Grün (c) Blau

Abbildung 22: RGB-Auszüge(192×128)

4.1 Drehung und Kennziffer im Frequenzraum

Abbildung 23 zeigt ein einfaches Testbild mit 5x5 schwarzen Linien. Die Linien haben die maximal mögliche Schwärzung und der Hintergrund hat die Intensität Null. Im Bild ist er allerdings etwas abgedunkelt dargestellt, um einen Kontrast zum weißen Papier zu haben.

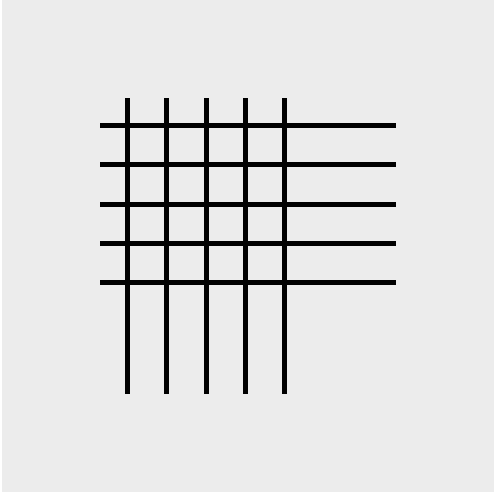


Abbildung 23: Das Beispielbild (100x100)

Die Symetrie des Bildes findet man in Abbildung 24 wieder, das die Kennziffer als Funktion des Drehwinkels (X-Achse) zeigt. Die Berechnung erfolgte für zwei Bilder nach Abbildung 23, wobei das eine um den Winkel φ gedreht ist. Alle 90 Grad findet man ein Minimum. Das Minimum bei 0 Grad geht aber als einziges auf Null herunter. Die beiden Bilder waren bei dieser Berechnung um 45^o vorgedreht. Die Differenzformel wurde als Kennzifferfunktion benutzt.

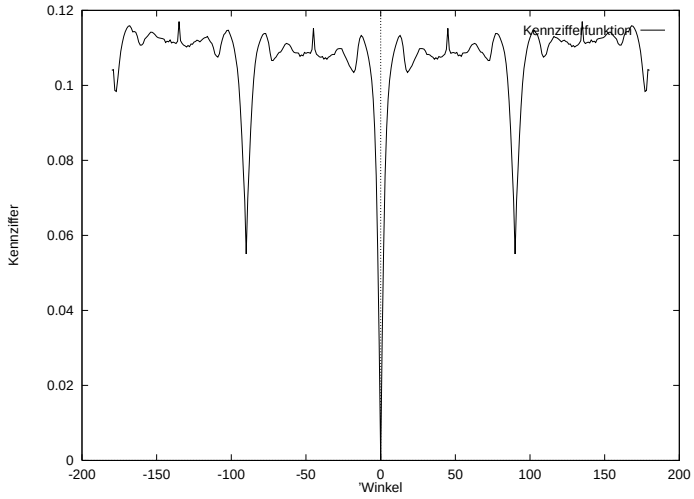


Abbildung 24: Die Kennziffer des Beispielbildes im Bereich -180 bis $+180$ Grad

Insgesamt haben wir auf einem Farbauszug 575 Punkte. Das Differenzbild sollte also zwischen 0 und $\approx$ 1200 Punkte haben. Dieses hat im Mittel 900 Punkte. Die Schnittfläche zwischen den beiden Farbauszügen hat bei unterschiedlichen Winkeln im Mittel 80 % der Fläche eines Auszuges. Ein Auszug hat 100x100=10000 Punkte. Die Kennziffer ergibt sich damit zu $\frac{900}{8000} \approx 0.11$.

Abbildung 25 zeigt vergrößert den 10^o Ausschnitt um 0 herum. Dort läuft in diesem Fall eine Minimierung nicht Gefahr, in ein Nebenminimum zu geraten.

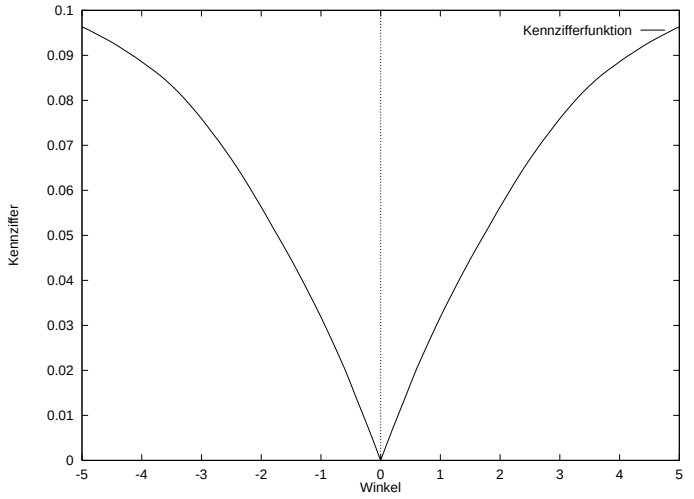


Abbildung 25: Die Kennziffer des Beispielbildes im Bereich -5 bis $+5$ Grad

Bei der Ausrichtung des Drehwinkels mit Hilfe der Fouriertranformierten wird der eine

Farbauszug festgehalten, ein kreisförmiger Bildbereich ausgeschnitten und fouriertransformiert. Der andere Farbauszug wird gedreht, ein kreisförmiger Bildbereich ausgeschnitten und fouriertransformiert und von den beiden Farbausügen dann die Kennziffer gebildet.

Zur Vermeidung von Artefakten der Bildränder wird die Berechnung der Kennziffer auf den Kreisausschnitt beschränkt. Die Artefakte würden sonst jeden Bildinhalt durch ausgeprägte Peaks im Frequenzraum übertreffen.

Das folgende Beispiel zeigt, wie der Verlauf der Kennziffer aussieht, wenn man sie im Frequenzraum bildet. Abbildung 26 zeigt den Verlauf der Kennziffer (Differenzformel) des Lenabildes im gesamten Bereich von -180 bis $+180$ Grad. Bei ± 45 Grad erkennt man die Spitzen, die von der Rotation herrühren. Beim Übergang in den Frequenzraum treten unvermeidliche Symmetrien (Abbildung 26) auf, die neben dem Hauptminimum immer ein weiteres bei 180^o zeigen.

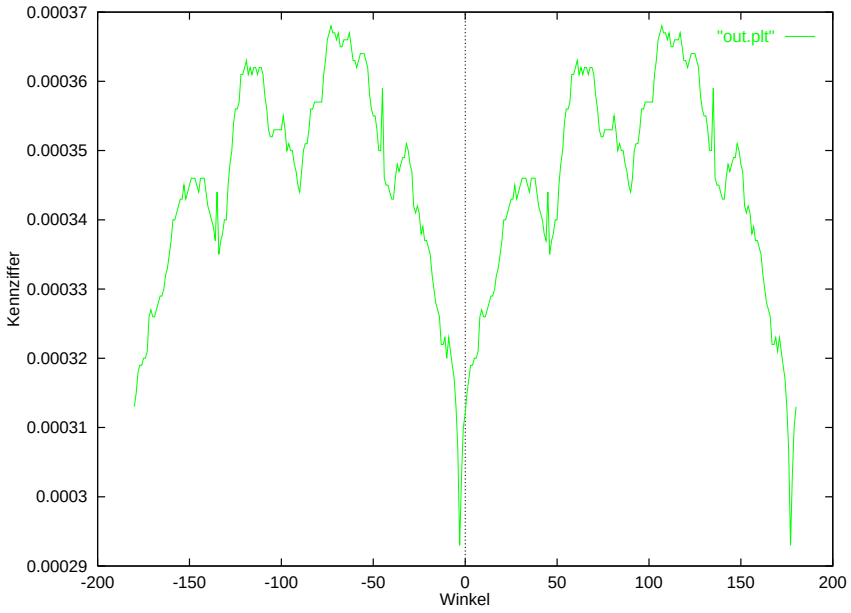


Abbildung 26: Die Kennziffer für das Lenabild im Bereich -180 bis $+180$ Grad

In Abbildung 27 ist wieder ein Ausschnitt vergrößert.

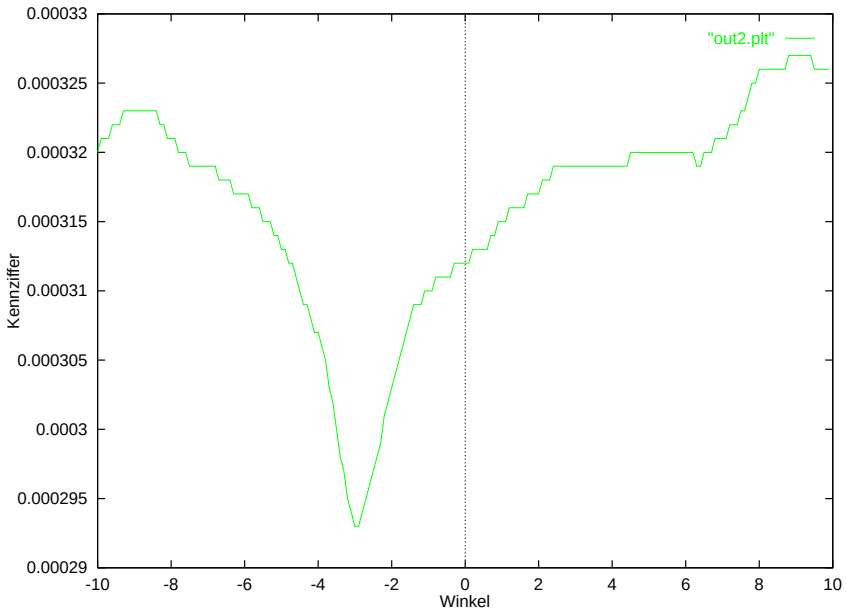


Abbildung 27: Die Kennziffer für das Lenabild im Bereich -10 bis $+10$ Grad

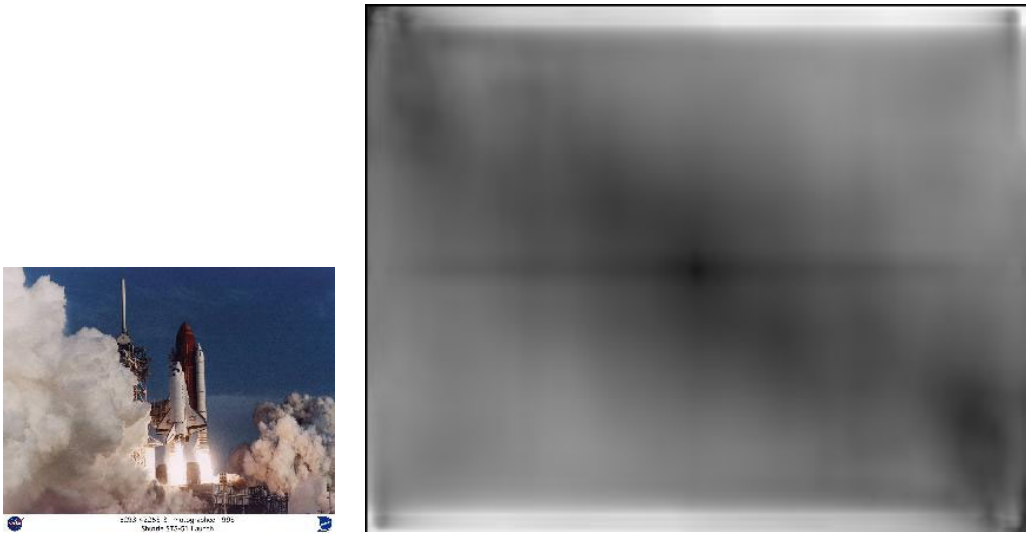
Deutlich erkennt man das Minimum bei -3^o. Das entspricht genau dem Winkel, mit dem der Rot- und der Grünauszug gegeneinander verdreht wurden.

4.2 Eindeutigkeit der Kennzifferminima

Eine ganz wichtige Frage bei der Untersuchung des beschriebenen Verfahrens ist natürlich, inwieweit die verwendeten Algorithmen abhängig von dem verwendeten Bildmaterial sind und in welchen Fällen sie nicht zum Erfolg führen.

Bei den getesteten Eingabebildern führten die Verfahren in fast allen Fällen zum Erfolg. Ein wichtiger Parameter war dabei der Startwert der Minimierung, der in unserem Verfahren durch eine Schwerpunktbestimmung gefunden wird. Im folgenden Beispiel ist ein Kennzifferverlauf gezeigt, der über alle möglichen Positionen als Grauwert aufgezeichnet wurde.

Die positionierten Bilder haben natürlich eine andere Auflösung als die Eingabebilder. Da die Farbauszüge jede beliebige Lage zueinander einnehmen können, bei der noch eine Überlappung vorhanden ist, kann bei drei Farbauszügen aus einem 100x100 Punkte-Bild ein 300x300 Punkte-Bild werden (Dann gäbe es überhaupt keine Überlappung mehr und die Farbauszüge liegen einfach nebeneinander). Ein großer Offset ist allerdings verbunden mit einem entsprechend kleinen Überlappungsbereich, darunter leidet die Zuverlässigkeit des Verfahrens.



(a) Eingangsbild (b) Kennzifferbild

Abbildung 28: Das Verfahren ist nur in der Nähe der Sollposition eindeutig (Auflösung des Eingabebildes: 45×33)

An Abbildung 28 kann man sehen, daß im Kennzifferbild Nebenminima existieren können. Die Kennziffern für den ganzen möglichen Bereich sind hier aufgetragen.

Generell gilt deshalb: Je weiter die Farbauszüge auseinandergezogen werden, desto schwieriger ist die Rejustierung. Schließlich ist der außerhalb des Überlappungsbereichs liegende Teil einfach nicht mehr berücksichtigt. Im folgenden sind Ausschnitte von ±10% um die Sollposition wiedergegeben. Dort ist die Kennzifferfunktion eindeutig.

Die bei Analyse der Rot- und Grün-Auszüge erhaltenen Kennziffern sind in Abbildung 29 so gezeichnet, daß die Höhe den Wert der Kennziffer angibt. Der Ort, also die (x,y) Koordinate entspricht dem Ort des Grünauszuges bei festgehaltenem Rotauszug. Die Koordinate (0,0) ist 10% nach links unten gegenüber der Sollposition verschoben. Man erkennt deutlich das Minimum in der Mitte. Die Sollposition ist im Kennzifferbild deutlich ausgezeichnet.

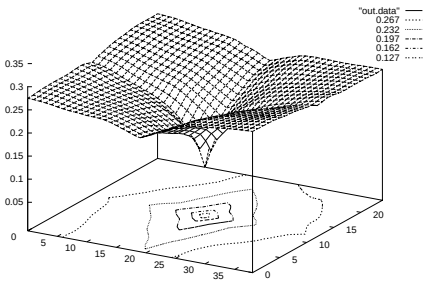


Abbildung 29: Kennziffern im 10% Bereich um die Sollposition

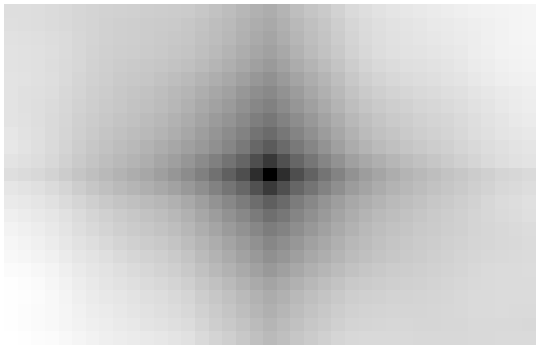
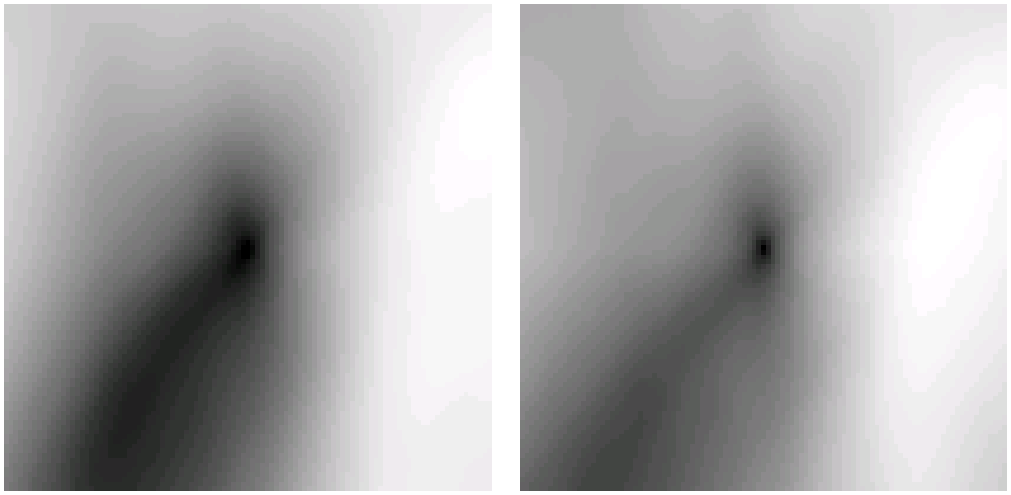


Abbildung 30: Kennziffern im 10% Bereich um die Sollposition als Graustufenbild

In Abbildung 30 ist dasselbe als Graustufenbild dargestellt. Wieder entspricht die Mitte der Sollposition, die Eckpunkte entsprechen ±10% Bildverschiebung.

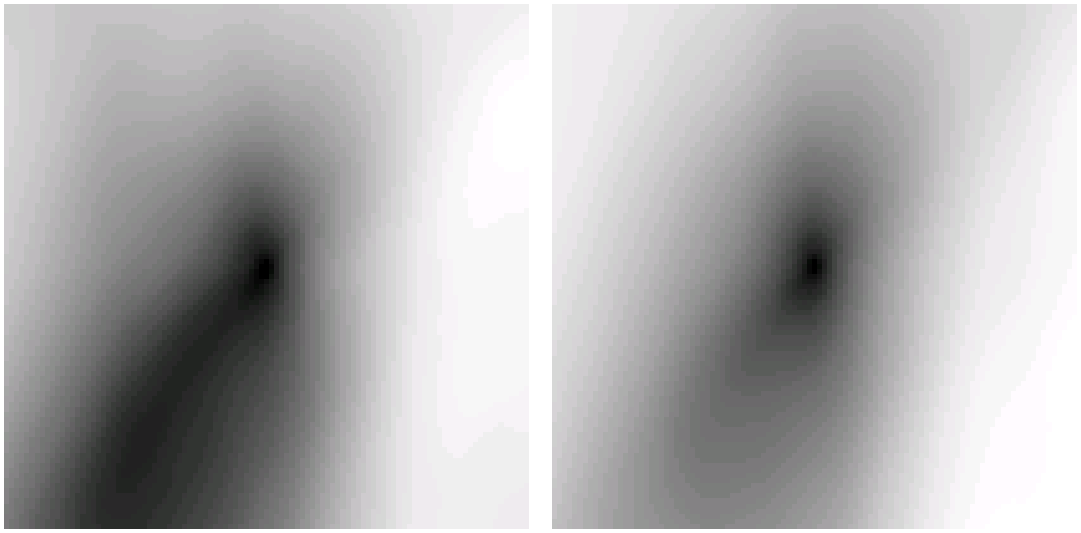
Kleine Graustufenunterschiede sind für das Auge schwer zu unterscheiden. Daher ist eine solche Darstellung bei feinen Strukturen einem Höhenbild wie Abbildung 29 unterlegen. Oft interessiert aber nicht nur die Feinstruktur sondern das Verhalten der Kennziffer über große Bereiche. Deshalb werde ich je nach Bedarf die eine oder die andere Darstellung bevorzugen.



(a) Rot Grün (b) Rot Blau

Abbildung 31: Das linke Kennzifferbild zeigt ein deutlich langgestrecktes Minimum. Unter bestimmten Umständen läuft die Minimierung in das Nebenminimum links unten. Dargestellt ist der 10 % Bereich um die Sollposition.

4.3 Unterdrückung von lokalen Minima



(a) Ohne Filterung (b) Gaußfilter

Abbildung 32: Filtert man hochfrequente Anteile in den zugrundeliegenden Farbauszügen weg, so können kleine Nebenminima unterdrückt werden. Hier wurden die Farbauszüge für die Präparation des rechten Kennzifferbildes mit einem Gaußfilter vorverarbeitet.

Mitunter weist die Kennzifferfunktion ausgeprägte Nebenminima auf. In vielen Fällen kann eine Verbesserung in Form von einer Abschwächung der Nebenminima dadurch erreicht werden, daß die Farbauszüge vor der Minimierung gefiltert werden. Im Gegensatz zu einer differenzierenden Filterung mit Sobel- (∇) oder sogar Laplace- ($\triangle = \nabla^2$) filtern erreicht man mit einem Gaußfilter eine Verschmierung der Bildstrukturen anstatt einer Kantenverstärkung. Läßt man das komplette Verfahren durchlaufen, so wird das Lenabild korrekt ausgerichtet, obwohl eine reine Offsetbestimmung ohne Filterung etwas Probleme bereitet. Das liegt daran, weil bei einer kompletten Ausrichtung im ersten Schritt Winkel bestimmt werden, und im zweiten Schritt die Farbauszüge um die gefundenen Winkel gedreht werden. Da die Drehung aber mit einer Verschmierung über Nachbarpunkte einhergeht, ist mit der Drehung schon eine Filterung verbunden.

Wie man an dem Beispiel sieht, kann derselbe Effekt auch durch eine Filterung mit dem Gaußfilter erreicht werden. Die Rechenzeit dieser Filterung ist dabei vernachlässigbar, weil diese Filterung nur einmal vor der Offsetbestimmung für jeweils beide Farbausüge vorgenommen werden muß.

4.4 Einfluß der Bildfilterung

Eine Bildvorverarbeitung in Form von verschiedenen differenzierenden Bildfiltern zur Kantenverstärkung führt zu einer Verstärkung des Minimums der Kennzifferfunktion. Allerdings tritt diese Verstärkung nur in der unmittelbaren Umgebung der Sollposition auf. Nebenminima rücken näher an die Sollposition. Deshalb sind die angegebenen Verfahren zur Bildvorverarbeitung in Abbildung 34 und Abbildung 35 in dieser Form wenig hilfreich. Alle Kennziffernplots wurden für den Rot- und den Grün-Auszug von Abbildung 22 berechnet. Dargestellt sind jeweils die 10% Bereiche um die Sollposition. In den angeführten Beispielen wurde die Differenzformel benutzt. Zum Vergleich zeigt Abbildung 33 einen Kennziffernplot ohne Filterung.

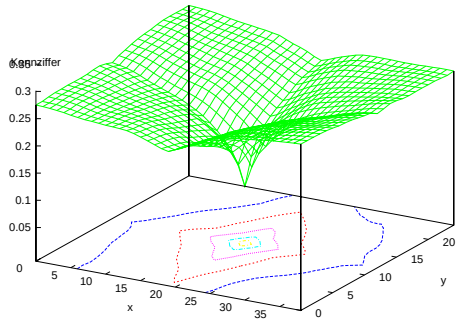


Abbildung 33: Kennzifferfunktion in der Nähe der Sollposition, Farbauszüge ungefiltert.

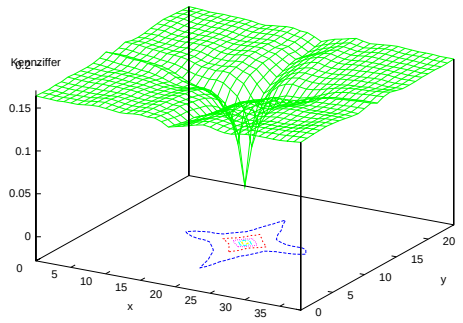
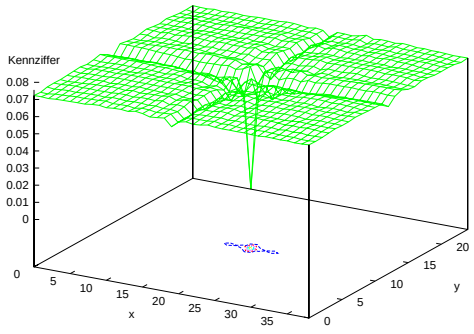
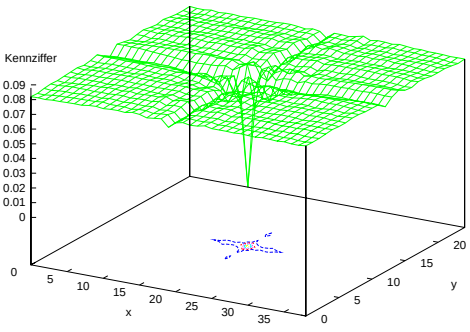


Abbildung 34: Kennzifferfunktion in der Nähe der Sollposition, Farbauszüge mit dem Sobeloperator gefiltert.



(a) Variante L_1



(b) Variante L_2

Abbildung 35: Kennzifferfunktion in der Nähe der Sollposition, Farbauszüge mit dem Laplaceoperator gefiltert.

Man erkennt deutlich, daß das Minimum zu einer scharfen Spitze degeneriert. Außerdem treten bei allen differenzierenden Bildfiltern Nebenminima wesentlich näher an der Sollposition auf.

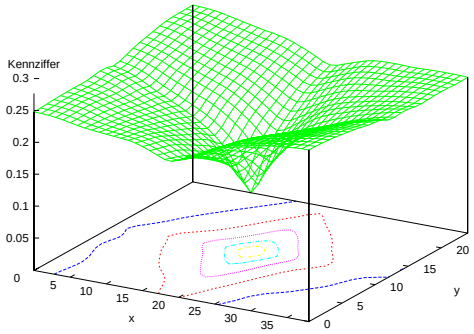


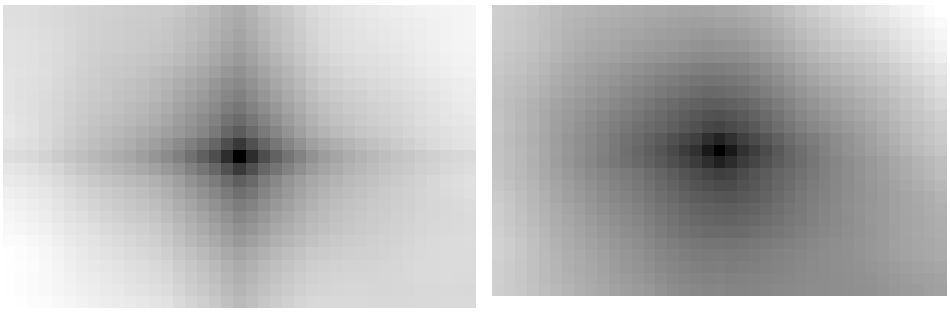
Abbildung 36: Kennzifferfunktion in der Nähe der Sollposition, Farbauszüge mit dem Gaußoperator gefiltert.

Der Gaußoperator verschmiert die Farbauszüge und führt zu einem breiteren Verlauf des Minimums. Die Nebenminima rücken weiter weg von der Sollposition. In Versuchen zeigt sich, daß dieser Filter die Minimumfindung positiv beeinflußt.

4.5 Verfahren zur Rechenzeitverkürzung

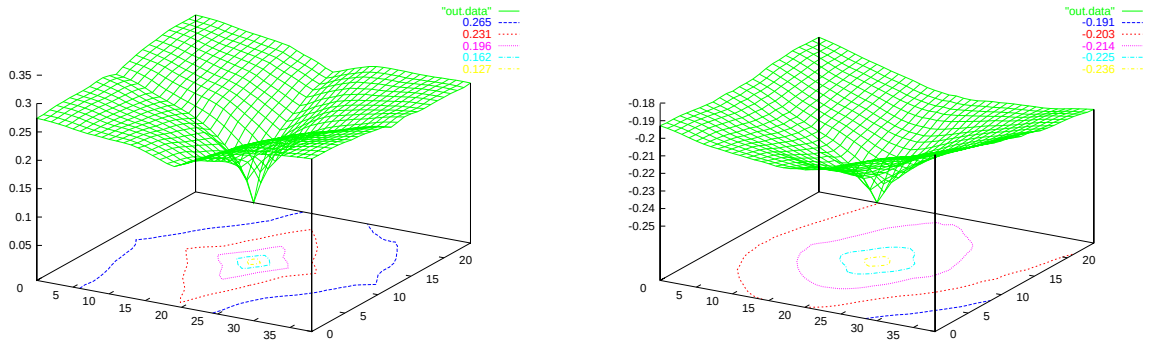
Im folgenden sind einige Verfahren und Faktoren beschrieben, die einen Einfluß auf die benötigte Rechenzeit haben.

4.5.1 Vergleich von Differenz und Produktformel



(a) Differenz (b) Produkt

Abbildung 37: Kennziffern als Graustufenbild (10% 192×128)



(a) Differenz (b) Produkt

Abbildung 38: Kennziffern als Plot (10% 192×128)

Die Produktformel ist auf Computern etwas langsamer als die Differenzformel, weil das Produkt auf Maschinensprachenebene einen sehr langsamen Befehl darstellt.

	Differenzformel Schrittzahl	Produktformel Schrittzahl
x1	17/34	17/34
Rechenzeit P133	5.97 sec	7.53 sec

Tabelle 2: 768x512 Punkte Bild; Vergleich Minimierung in einem Schritt: Differenz- und Produktformel. Die Produktformel ist auf Computern etwas langsamer als die Differenzformel, weil das Produkt auf Maschinensprachenebene einen sehr langsamen Befehl darstellt.

In Tabelle 2 kann man sehen, wie sich dieser Geschwindigkeitsunterschied in der Praxis auswirkt. Auch bei der gleichen Anzahl von Schritten [1], die die Minimierung benötigt, unterscheiden sich die Rechenzeiten sehr deutlich.

4.5.2 Vergleich von Amoeba und Powell

Da ein wichtiger Gesichtspunkt die Geschwindigkeit der Positionierung ist, liegt ein Vergleich der beiden verwendeten Minimierungsalgorithmen nahe.

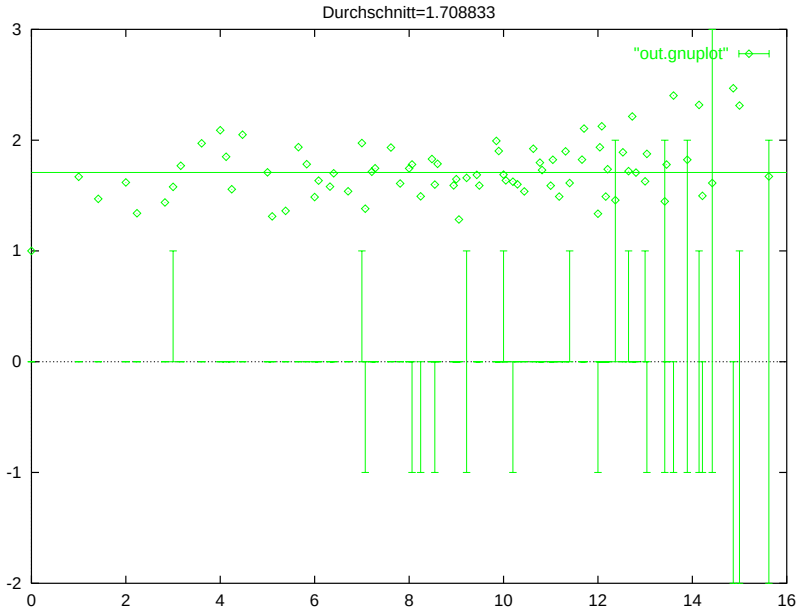


Abbildung 39: Zum Vergleich von Amoeba und Powell; kein Caching

Die X-Achse in Abbildung 39 stellt den Abstand des Startpunktes vom Minimum dar.

[1] Wenn die Anzahl der Schritte zur Minimierung die gleiche ist, ist der Rechenaufwand nur annähernd derselbe. Wir hatten ja schon gesehen, daß die Schnittfläche in die Rechenzeit zur Berechnung der Kennziffer eingeht. Diese ist aber für unterschiedliche Abstände von der Sollposition unterschiedlich groß.

Die Y-Achse stellt das Verhältnis der nötigen Schritte von „Powell“ zu „Amoeba“ dar. Die zugrundegelegte Schrittzahl ist dabei die Anzahl der Aufrufe der Kennziffernberechnung aus „Amoeba“ und „Powell“ (bzw. den darin aufgerufenen Algorithmen). Es sind daher durchaus Aufrufe in den Schrittzahlen enthalten, die nicht zu einer erneuten Kennziffernberechnung führen, weil der Algorithmus dieselbe Position noch einmal vorschlägt. Eine Kennziffernberechnung von (1.2,2.5) und (1.3,2.5) führt zum selben Pixeloffset (1,2); Pixeloffsets sind ja immer nur ganzzahlig.

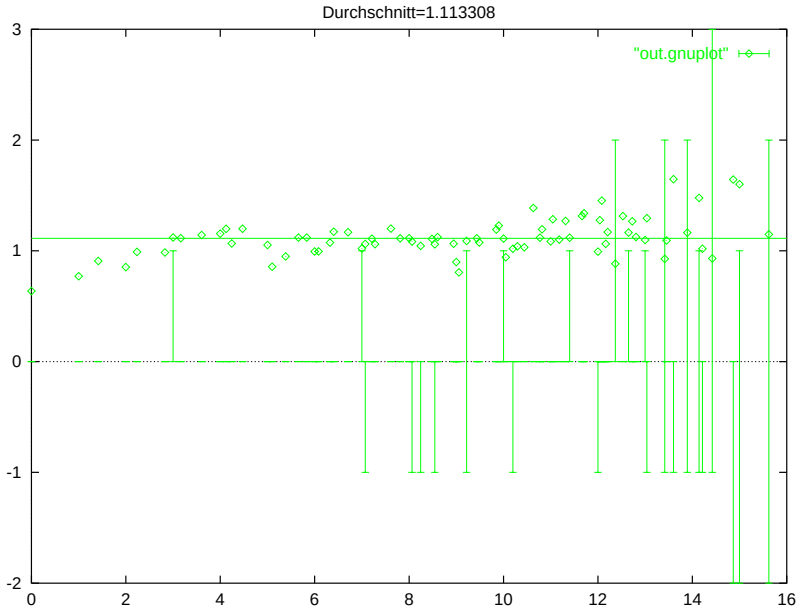


Abbildung 40: Zum Vergleich von Amoeba und Powell; mit Caching. Aufgetragen ist der Quotient von Anzahl der Schritte mit „Powell“ und Anzahl der Schritte mit „Amoeba“. Die Balken zeigen die Anzahl der fehlerhaften Minimierungen (nach oben: „Powell“, nach unten: „Amoeba“)

Abbildung 40 zeigt die entsprechenden Kurven für die Schrittzahlen ohne Mehrfachaufrufe. Der Powell-Algorithmus liegt hier nicht viel ungünstiger als der Amoeba-Algorithmus.

Die nach oben gerichteten Balken zeigen die Anzahl der fehlerhaften Minimierungen des Powell-Algorithmus jeweils für einen bestimmten Abstand und die nach unten gerichteten Balken zeigen die entsprechenden Fehlminimierungen für Amoeba. Wie man auch erwartet, steigt die Häufigkeit von Fehlern mit steigendem Abstand zur Sollposition. Die Wahrscheinlichkeit, daß die Minimumsuche in ein Nebenminimum läuft, steigt mit wachsendem Abstand zur Sollposition und damit zum gesuchten Minimum.

Die Vektoren, mit denen Amoeba initialisiert wird, waren (2.0,0.0), (-2.0,-2.0) und (0.0,2.0). Diese Vektoren spannen vom Ort des Startpunktes ein Dreieck auf, das den Startpunkt umspannt.

Powell wurde mit den zwei orthogonalen Vektoren (1.0,0.0) und (0.0,1.0) initialisiert.

4.5.3 Pyramiden-Verfahren

Eine weitere Reduzierung der Anzahl der nötigen Kennzifferberechnungen ist möglich, wenn das Eingabebild durch Mittelung über zB. 16 Nachbarpunkte auf eine kleinere Auflösung umgerechnet wird. In diesem gemittelten Bild erfolgt die Minimumsuche. Ist das Minimum gefunden, wird die Auflösung gesteigert, indem nur noch über 4 Nachbarpunkte gemittelt wird. Der Startpunkt für die Minimumsuche ist jeweils das Minimum aus dem vorhergehenden Iterationsschritt. Um im Bild einer Pyramide zu bleiben: Ausgehend von der Pyramidenspitze (kleiner Auflösung) arbeitet sich der Algorithmus also durch die Pyramide zur großen Auflösung am Pyramidenboden. Besonders vorteilhaft erweist sich hierbei, daß bei den gemittelten Bildern die Punktzahl wesentlich kleiner ist. Die Rechenzeit für die einzelne Kennzifferberechnung wird reduziert. Eine Halbierung der Auflösung in X und Y Richtung führt ja zu einer viermal schnelleren Kennzifferberechnung.

Durch etwas Overhead beim Herunterrechnen wirkt sich das zwar nicht voll auf die Geschwindigkeit aus, dieser Overhead bedingt durch Speicherverwaltung und ähnliches ist aber bei optimierten Algorithmen vernachlässigbar.

Das Ergebnis einer Minimierung bei halbierter Auflösung ist dann zwar in X und Y Richtung mit einer Unsicherheit von ± 1 behaftet. Da dieses Vorgehen aber zu einer drastischen Reduzierung der nötigen Berechnungen im rechenintensiven letzten Schritt bei Originalauflösung führt, ist dieser Aufwand gerechtfertigt.

Das Vorgehen ist dadurch begrenzt, daß bei zu kleiner Auflösung das Minimum im Kennzifferbild nicht mehr deutlich genug ist. Außerdem verdoppelt sich die Ungenauigkeit mit jeder Halbierung der Auflösung und es wird immer wahrscheinlicher, daß der Algorithmus in ein Nebenmimimum läuft. Rechenzeit und Anzahl der Kennziffer-Berechnungen gehen aus folgender Tabelle hervor:

	Direkt Schrittzahl	Pyramiden Verfahren Schrittzahl	Pyramiden Verfahren Schrittzahl
x4	-	-	8/23
x2	-	11/22	9/23
x1	16/34	10/23	8/23
Rechenzeit P133	0.35 sec	0.27 sec	0.23 sec

Tabelle 3: 192x128 Punkte Bild; Vergleich Minimierung in einem/mehreren Schritten. Im linken Fall wird der Schwerpunkt auf der Originalauflösung berechnet und ebenfalls auf dieser Auflösung minimiert. Im rechten Fall wird das Bild erst bei $\frac{1}{16}$ der Punktzahl minimiert, dann mit $\frac{1}{4}$ und schließlich mit der Originalauflösung. Die Ergebnisse der vorherigen Minimierungsstufe werden immer an die nächste weitergereicht. Man erkennt deutlich, daß schrittweises Vergößern eine kürzere Rechenzeit bringt.

Alle Zeiten wurden mit der Funktion getrusage()[2] ermittelt. Es ist dabei nur die Zeit

[2]Die Funktion getrusage() wurde mit 4.2BSD eingeführt ([21][13]).

zusammengerechnet, die der Algorithmus in der (Powell-) Minimierung verbringt. Berücksichtigt wird außerdem nur die „user time“. Damit ist die Zeit gemeint, die das Programm im Objektcode des Programms verbringt. Die Zeit, die in Systemfunktionen verbraucht wird, ist dabei unberücksichtigt und eine möglicherweise ungünstigere Häufigkeit oder Dauer von solchen Funktionsaufrufen kann anhand der vorgelegten Zahlen nicht beurteilt werden. Die in allen Fällen auf dem Originalbild durchgeführte Schwerpunktberechnung benötigt in diesem Fall etwa 0.01 Sekunden. Die Vorbereitung des Pufferspeichers, das Angleichen der Bildtiefe und das Umkopieren bzw. Verkleinern der Farbauszüge zusammen benötigen bei nicht verkleinertem Bild 0.05 Sekunden und bei halbierter Auflösung noch 0.03 Sekunden. Der erste Wert bei der Schrittzahl ist die Anzahl der Kennziffern, die tatsächlich berechnet werden müssen, der zweite Wert die Anzahl von Kennzifferaufrufen im Powell-Algorithmus. Dies ist nicht das gleiche, weil der Powell-Algorithmus einige Stützpunkte mehrmals zur Berechnung aufruft. Mit Hilfe eines einfachen Pufferverfahrens wird die erneute Berechnung verhindert.

	Direkt Schrittzahl	Pyr. Verf. Schrittzahl	Pyr. Verf. Schrittzahl	Pyr. Verf. Schrittzahl
x8	-	-	9/24	-
x4	-	11/22	-	11/22
x2	-	10/22	11/22	-
x1	17/34	10/22	9/23	18/41
Rechenzeit P133	5.98 sec	4.89 sec	4.40 sec	6.58 sec

Tabelle 4: 768x512 Punkte Bild; Minimierung in mehreren Schritten, Varianten. Manchmal ist es besser eine Stufe auszulassen.

Bei dem zweiten Lauf des Verfahrens wurde statt der Auflösung 4/2/1 die Folge 8/2/1 gewählt. Das Endergebnis ist dasselbe, aber die Rechenzeit ist noch kürzer. Der dritte Lauf besteht nur aus 4/1. Der Test soll der Ermittlung einer optimalen Kombination von Auflösungen dienen.

Die Bilder wurden in Tabelle 4 durch Mittelwertbildung auf niedrigere Auflösungen heruntergerechnet. Die Zeit der Berechnung des Bildes mit geringerer Auflösung aus einem Bild mit höherer Auflösung ist im allgemeinen gering. Der Unterschied der Berechnungszeit für das untersuchte Bild kommt in diesem Fall allerdings alleine dadurch zustande, daß die Berechnung des Mittelwertes geringfügig langsamer ist, als wenn man einfach nur einen Grauwert herausgreift. In Tabelle 5 sind deshalb die Minimierungszeiten den Zeiten gegenübergestellt, die der Algorithmus für das Kopieren und Initialisieren der benötigten Strukturen und Felder braucht. Diese für die Initialisierung benötigte Zeit wächst mit der Anzahl der auf den untersuchten Farbauszügen liegenden Bildpunkte etwa linear.

Die Zeiten für x1 sollten gleich sein. Hier zeigen sich die Grenzen der Rechenzeitberechnung unter Unix. Grundsätzlich zeigen nahezu keine zwei Aufrufe einer Zeitmessung dasselbe Ergebnis. Trotzdem haben die mit getrusage() bestimmten Zeiten in der Regel nur einen Fehler von wenigen Prozent.

	Init:Auslassung	Init:Mittelwert	Minimierung
x4	0.07	0.19	0.23
x2	0.28	0.49	0.99
x1	0.79	0.80	3.63

Tabelle 5: 768x512 Punkte Bild; Die Initialisierung hat bei Mittelwertbildung einen größeren Anteil an der Rechenzeit

4.5.4 Auslassungs-Verfahren

Eine weitere Möglichkeit zur Reduzierung der Rechenzeit ist einfach, nur jeden 2. (3. ,4. ,..) Punkt zur Berechnung der Kennzifferberechnung heranzuziehen.

Der Unterschied zu der vorher beschriebenen Möglichkeit ist der, daß hierbei nicht in einem Schritt das Bild auf eine niedrigere Auflösung heruntergerechnet wird, sondern der Überlappungsbereich von zwei Farbauszügen in der vollen Auflösung benutzt wird. Damit läuft das Minimierungsverfahren auf dem vollen Wertebereich. Bei dem vorher beschriebenen Verfahren fallen $\frac{3}{4}$ der Stüzstellen sofort raus. Hier wird nur die einzelne Stützstelle bzw. Kennziffer mit $\frac{1}{4}$ ($\frac{1}{9}$,$\frac{1}{16}$) der Bildpunkte berechnet.

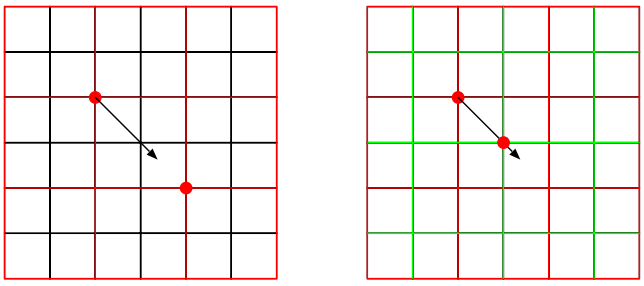
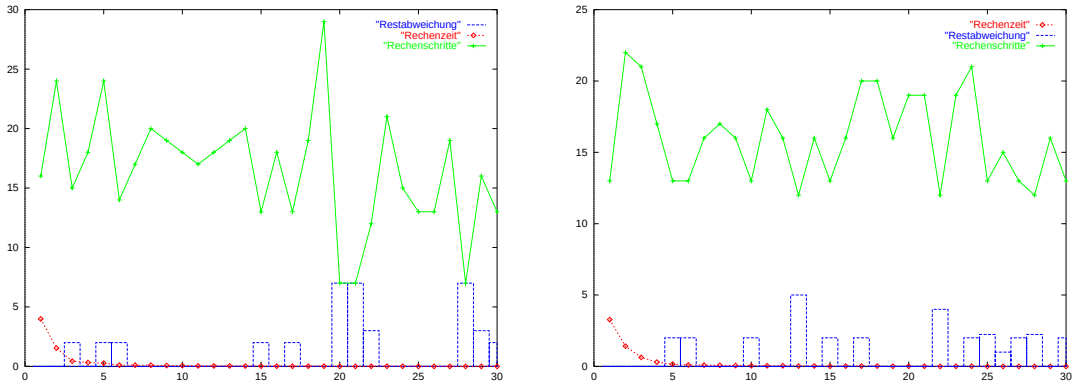


Abbildung 41: Zum Vergleich der Minimierung mit Auflösungspyramide und mit dem Auslassungsverfahren. Bei der Minimierung über eine Auflösungspyramide werden nur auf Gitterplätzen auf dem roten Gitter Kennziffern berechnet. Bei dem Auslassungs-Verfahren sind auch alle übrigen Gitterplätze möglich. Die Anzahl der für die Kennzifferberechnung zu berücksichtigenden Gitterplätze ist aber gleich. Darum ist das Auslassungs-Verfahren bei verleichbarem Rechenaufwand genauer.

	Schrittzahl mit/ohne Pufferung	Rechenzeit P133 sec	Restabweichung x,y
1x	16/36	3.99	0,0
2x	24/64	1.54	0,0
3x	15/32	0.44	0,2
4x	18/43	0.32	0,0
5x	24/47	0.27	0,2
6x	14/25	0.11	0,2
7x	17/40	0.10	0,0
8x	20/42	0.10	0,0
9x	19/41	0.07	0,0
10x	18/36	0.06	0,0

Tabelle 6: 512x512 Punkte Pepper-Bild; Vergleich Minimierung mit Auslassungs-Verfahren. Obwohl schon bei 2x nur noch ein Viertel der Punkte zur Berechnung der Kennziffer herangezogen werden, findet die Minimierung genau die Sollposition.

Die Schwerpunktberechnung dauert in diesem Beispiel 0.2 Sekunden und liegt um (5,2) in X und Y Richtung neben der Sollposition. Bei dem untersuchten Bild führt die Minimierung sogar ohne ein Pyramidenverfahren in fast allen Fällen zum Erfolg. In jedem Fall ist die Rechenzeit deutlich geringer als bei direkter Berechnung.



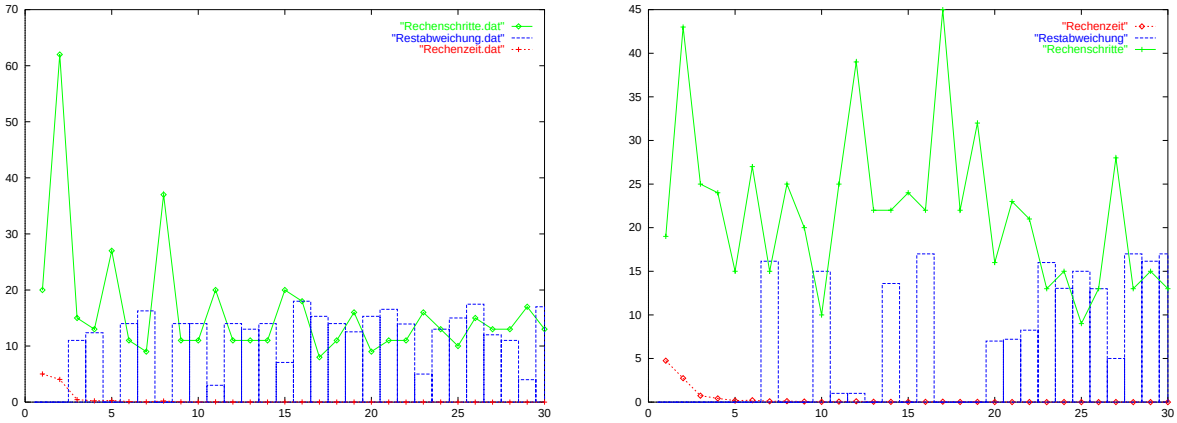
(a) ohne Gaußfilter

(b) mit Gaußfilter

Abbildung 42: Einfluß der Auslassung auf die Rechenzeit; Jeder 1te bis 30te Punkt wird berücksichtigt. Die Rechenzeit sinkt drastisch, aber die Zuverlässigkeit des Minimierungsverfahrens ist trotzdem hoch. Je nach Kurve ist auf der Y-Achse Rechenzeit in Sekunden, Restabweichung in Pixeln oder Anzahl der Rechenschritte aufgetragen. Pepper-Bild: Rot-Grün

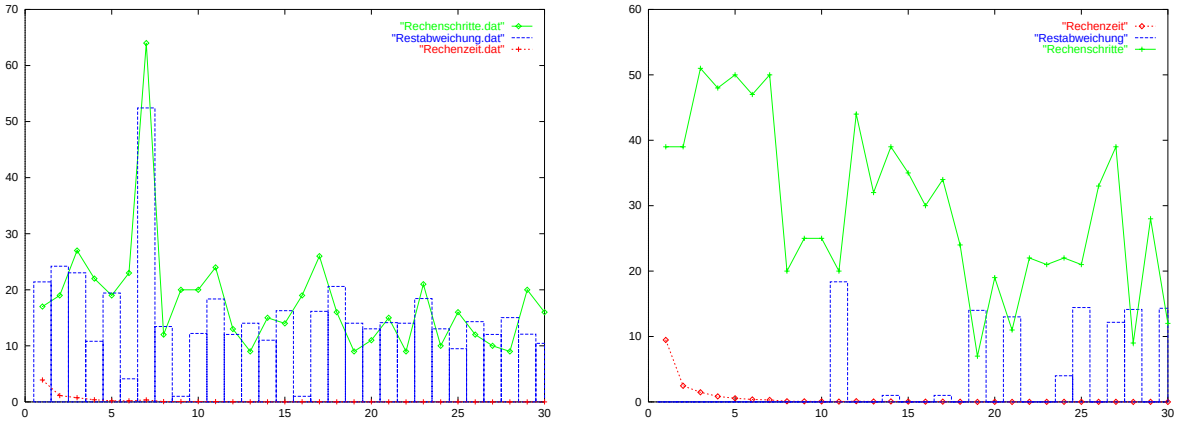
Bemerkenswert ist in Abbildung 42 die Tatsache, daß falsche Minimierungen mit einer

geringen Anzahl von Minimierungsschritten einhergehen.



(a) ohne Gaußfilter (b) mit Gaußfilter

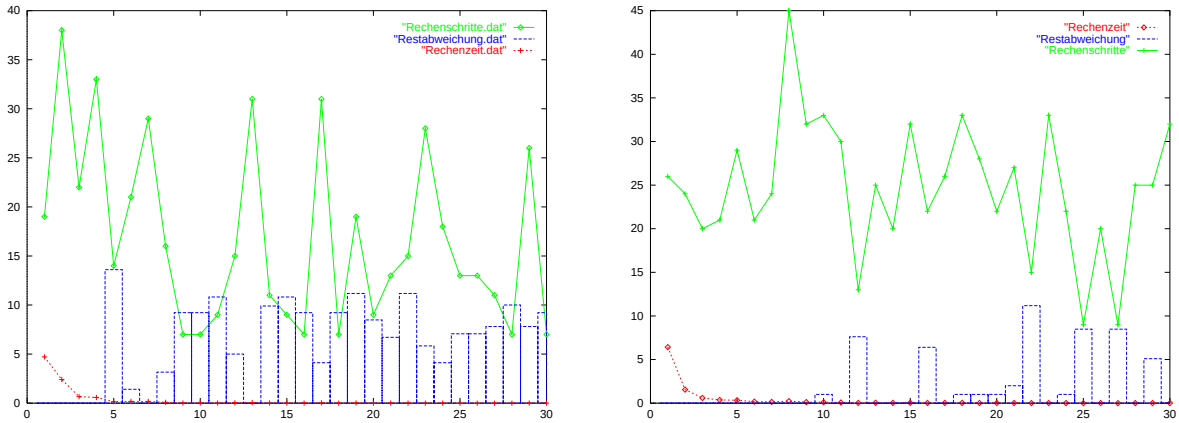
Abbildung 43: Pepper-bild: Rot-Blau



(a) ohne Gaußfilter (b) mit Gaußfilter

Abbildung 44: Lenabild: Rot-Grün

Abbildung 44 zeigt einen Fall von fehlerhafter Minimierung. Abbildung 31 zeigt den dazugehörigen Kennzifferplot. Der Algorithmus läuft in ein Nebenminimum. Durch eine vorherige Filterung kann das dennoch tiefe Minimum an der Sollposition jedoch verbreitert werden. Dann findet der Algorithmus das korrekte Minimum.



(a) ohne Gaußfilter

(b) mit Gaußfilter

Abbildung 45: Lenabild: Rot-Blau

4.5.5 Optimierung bei Ganzzahlgittern

Die Kennzifferfunktion ist nur auf ganzzahligen Koordinaten definiert. Die verwendeten Minimierungsalgorithmen dagegen erlauben stetige Eingangsfunktionen. Daher ist es notwendig, die Eingangskoordinaten der Kennzifferberechnung zu runden. Einfaches Runden vor dem Aufruf der Kennzifferberechnung führt dazu, daß die Ortsinformation dem Minimierungs-Programm verloren geht.

Eine Programmodifikation, die die Rundung in dem Minimierungs-Programm vornimmt, wurde deshalb nötig. Das wirkt sich positiv auf die Rechenzeit aus.

	Version A Powell	Version B Powell	Version A Amoeba	Version B Amoeba
Schrittzahl	17/34	20/38	19/24	20/27
Rechenzeit P133	6.01 sec	7.12 sec	6.70 sec	7.00 sec
Restabweichung	0,0	0,1	0,0	1,0

Tabelle 7: 768x512 Punkte Bild; Minimierung in einem Schritt im Vergleich. Die Algorithmen aus [17] können für diese spezielle Anwendung durch Ganzzahlwandlung im Algorithmus selbst optimiert werden. Version A ist optimiert.

In der Version A wird die Rundung schon im Algorithmus selbst vorgenommen, bei der Version B nur vor der Berechnung der Kennziffer. In Abbildung 8 und 9 kann man sehen, wie die Daten weitergereicht werden. In Version A wird die Rundung in „powell“ und „f1dim“ bzw. in „amoeba“ und „amotry“ vorgenommen statt erst bei der Berechnung der Kennzif-

fer. In der Zeile „Abweichung“ ist die verbleibende Restabweichung in X und Y Richtung aufgeführt. Die angegebenen Rechenzeiten sind reine Minimierungszeiten. Der Overhead bei der Ermittelung der Zeiten in 8 für die Initialisierung, Speicherverwaltung etc. ist etwa 1.5 Sekunden.

Alle Werte wurden mit der Differenzformel ermittelt. Zur Initialisierung wurden die Schwerpunkte auf der vollen Auflösung berechnet. Bei der Reduzierung der Auflösung wurde die Mittelwertberechnung verwendet.

	Version A Schrittzahl	Version B Schrittzahl
x4	11/22	11/23
x2	10/22	12/24
x1	10/22	12/24
Rechenzeit P133	4.88 sec	5.75 sec
Restabweichung	0,0	1,1

Tabelle 8: 768x512 Punkte Bild; Minimierung in mehreren Schritten.

Der Unterschied zur Tabelle 4.5.5 ist, daß hier das Pyramidenverfahren benutzt wurde.

Bei gleichzeitiger Nutzung des Pyramidenverfahrens ist der Geschwindigkeitsgewinn nicht sehr groß. Es treten aber außerdem in der Version B Fehler auf, die in der Version A vermieden werden.

4.5.6 Reihenfolge von FFT und Drehung

Die für die Funktionstüchtigkeit der Winkelbestimmung notwendigen mathematischen Beziehungen wurden in Kapitel 2.7.1 und Kapitel 2.7.2 erläutert. In Abbildung 20 ist eine Variante der Drehwinkelbestimmung im Bild links dargestellt, die nur eine rechenzeitintensive FFT pro Farbauszug benötigt. Da die entsprechenden Operatoren kommutieren, kann auch im Frequenraum gedreht werden. Mit einer entsprechenden Hardwareimplementation der FFT würde die Umkehrung der Reihenfolge nichts mehr bringen. Hier wird die FFT aber auf dem Computer als simples Unterprogramm ausgeführt und diese Umkehrung verkürzt entsprechend die Rechenzeit.

Reihenfolge Auflösung	Erst FFT dann Drehen	Rest- abweichung [^{o}]	Erst Drehen dann FFT	Rest- abweichung [^{o}]
128	9/1.09	0.42	10/1.77	2.05
256	10/4.56	0.06	9/6.59	0.16
512	9/17.23	0.01	11/33.57	0.04

Tabelle 9: 557x537 Punkte Pepper-Bild mit um 4^o gedrehten Grünauszug. Aufgeführt sind die Minimierungsschrittzahlen und -zeiten für den Drehwinkel des Grünauszuges gegenüber dem Rotauszug. Startwerte für den Brent-Algorithmus waren in allen Fällen (-5,-1,5).

4.6 Beispiel einer erfolgreichen Positionierung

Im folgenden sind exemplarisch Rejustierungen mit den angegebenen Verfahren aufgeführt. Da eine Drehwinkelbestimmung kein Bild liefert, dem man den Erfolg dieses Rejustierungsschrittes ansieht, wurde auf einen Abdruck dieser Ergebnisse verzichtet.

4.6.1 Nur Schwerpunktbestimmung

Um die Wirkung der Schwerpunktbestimmung zu demonstrieren, ist in Abbildung 46 die Abbildung 21 mit verschobenen Grün und Blauauszügen in die Schwerpunktbestimmung gegeben worden. Es wurde ein Offset von (20,20) (Grün) und (20,-20) (Blau) gewählt.

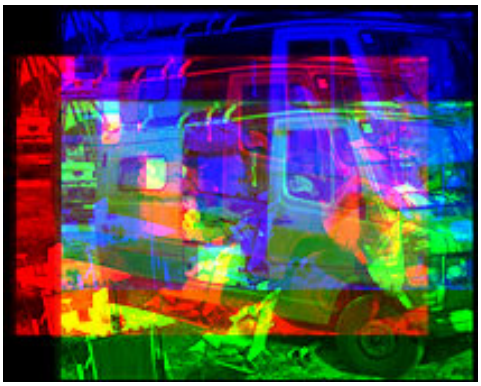


Abbildung 46: Positionierung der Farbauszüge vor Ausrichtung über Schwerpunktbestimmung

Obwohl Abbildung 21 große Blau- und Rotanteile enthält, liegen die Schwerpunkte nicht weit auseinander.

Abbildung 47: Positionierung der Farbauszüge nach Ausrichtung über Schwerpunktbestimmung

Bei diesem Beispielbild beträgt die Restabweichung nach der Schwerpunktbestimmung lediglich (-2,4) für Grün-Rot und (-3,7) für Blau-Rot in Pixeln. Der Rechenaufwand der Schwerpunktberechnung hängt außerdem lediglich von der Bildfläche ab und ist im Vergleich zu den übrigen Schritten relativ klein.

4.6.2 Nur Offsetbestimmung

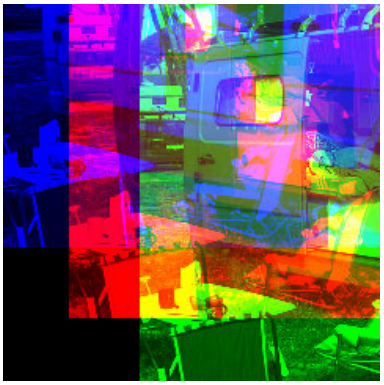


Abbildung 48: Positionierung der Farbauszüge vor Ausrichtung

Bei der Durchführung nach den in Kapitel 2 angeführten Methoden auf einem RGB-Bild mit verschobenen und abgeschnittenenen Farbauszügen wie Abbildung 48 erhalten wir gute Ergebnisse (Abbildung 49). Die RGB-Auszüge sind gegeneinander verschoben und abgeschnitten. Um die Robustheit des Verfahrens zu demonstrieren, sind von den Farbauszügen an unterschiedlichen Seiten Teile abgeschnitten, damit die Schwerpunktbestimmung nicht erfolgreich die Farbauszüge positionieren kann. Erst die darauf folgende Extremwertsuche im Kennzifferbild kann dieses leisten.

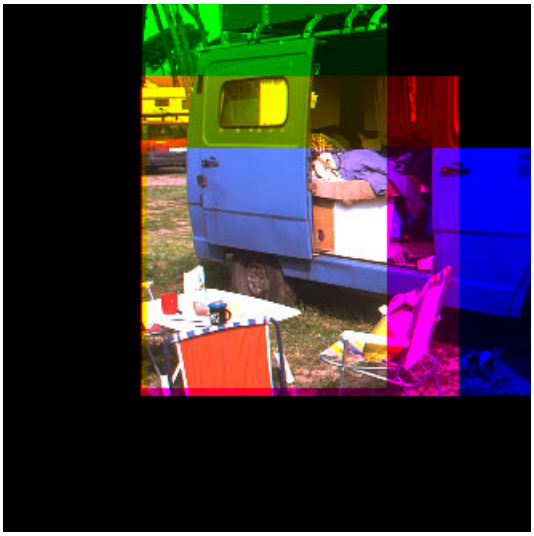


Abbildung 49: Das Ausgabebild nach der Ausrichtung

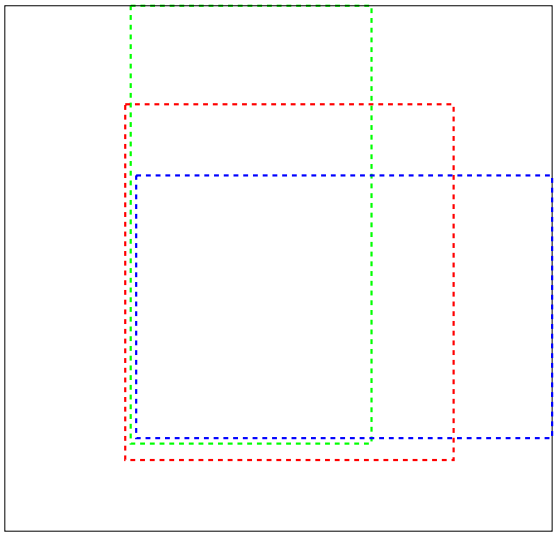


Abbildung 50: Das Ausgabebild nach der Ausrichtung (Schema)

Die RGB-Auszüge sind in Abbildung 49 korrekt positioniert. Die Erfolgskontrolle ist dadurch gewährleistet, daß die Farbauszüge des Eingabebildes mit definierten Offsets verschoben wurden. Diese Offsets kamen bei der Kennzifferminimierung in diesem Fall exakt heraus. Außerdem erkennt man auch so schon, daß die Bilder korrekt justiert wurden, da keine Farbsäume in Abbildung 49 zu sehen sind.

4.6.3 Offset und Drehwinkel: Lena Bild

Im folgenden Beispiel ist der Verlauf der Kennziffer als Funktion des Drehwinkels φ eingezeichnet. Außerdem wurden die Winkel markiert, die der Brent-Algorithmus der Reihe nach zur Berechnung aufruft. Die Startwerte waren $(-5, 0, 5)$. Man sieht deutlich, daß der Brent-Algorithmus sehr schnell konvergiert.

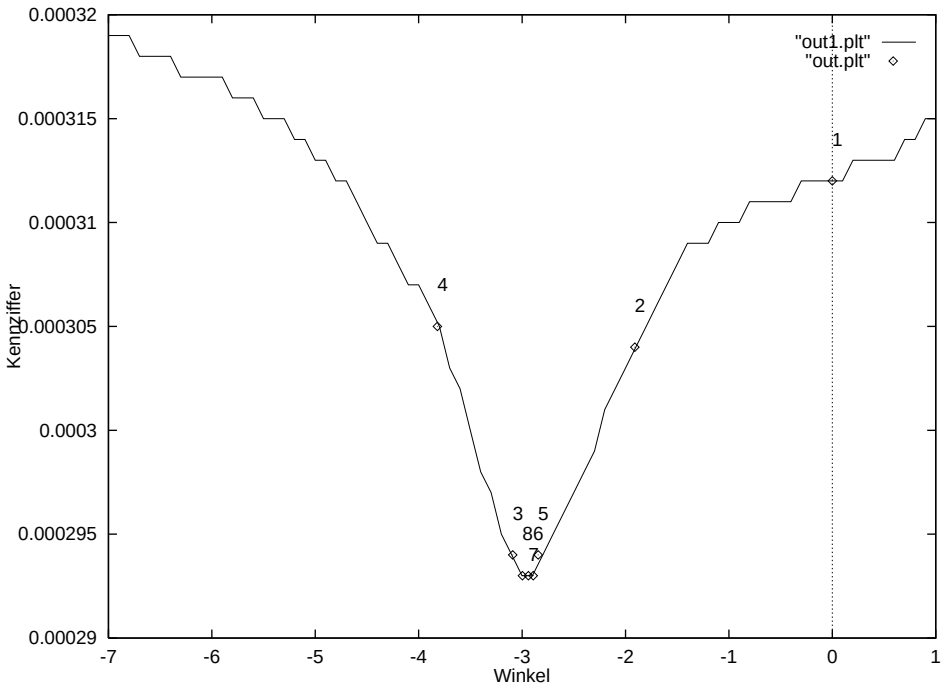


Abbildung 51: Kennziffern im Bereich -7^o und $+1^o$, Sollposition ist -3^o, Schritte des Brent-Algorithmus sind durchnummeriert

Das Minimum bei -3^o wird mit einer Genauigkeit von $\approx \frac{1}{4}^o$ gefunden. Das Abbruchkriterium im Original Brent-Algorithmus aus [17] führt zu einer zu großen Zahl von Aufrufen in der Nähe des Minimums. Das liegt unter anderem daran, daß dort ein Abbruchkriterium nur über die φ-Achse definiert ist und nicht zusätzlich über die Ordinate .

Das Eingabebild hierzu ist 557x537 Punkte groß und aus einem 512x512 Punkte Bild (Lena) folgendermaßen entstanden:

- Der Grünauszug ist gegenüber dem Rotauszug um (20,20) verschoben,
- Der Grünauszug ist gegenüber dem Rotauszug um 3^o Grad gedreht,
- Der Blauauszug ist gegenüber dem Rotauszug um (-25,-5) verschoben.

Für die Winkelbestimmung wird das Bild allerdings immer in X- und Y-Richtung auf eine Zweierpotenz als Größe skaliert, in diesem Fall 256x256. Eine kleinere Größe reduziert natürlich auch die Rechenzeit. Dafür wird aber die Ungenauigkeit größer.

Abbildung 52: Das Lenabild mit verdrehten und verschobenen Farbauszügen

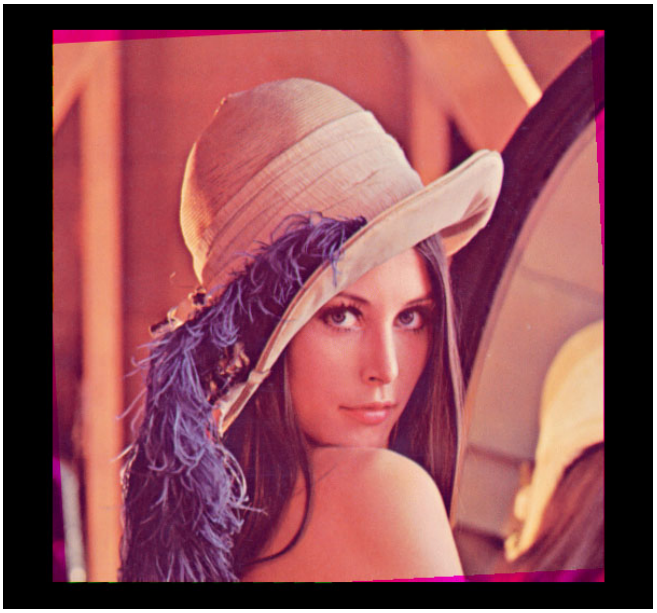


Abbildung 53: Das Lenabild nach Restaurierung der Offsets und Winkel

4.6.4 Offset und Drehwinkel: Pepper-Bild

In diesem Bild sind besonders große und intensive Grün- und Rot-Flächen vorhanden. Daher sollte es sich gut zum Testen eignen.

Abbildung 54: Das Eingabebild (pepper) im Original

Abbildung 55: Das Eingabebild (pepper) mit verschobenen und gedrehten Farbauszügen

Die Drehung wurde mit einem kleinen Programm realisiert, das den Farbauszug gedreht in ein ebensogroßes Bild kopiert. Daher sind am Rand des Grünauszuges Teile abgeschnitten. Da dieses aber eher die Robustheit des Verfahrens demonstriert, wurde auf eine Verfeinerung der Erzeugung der Testdatei verzichtet.

Abbildung 56: Das Eingabebild (pepper) nach der Ausrichtung. Der schwarze Rand, der im ausgerichteten Bild auftritt, wurde hier weggelassen.

Für die Winkelbestimmung des Grünauszuges brauchte der Brent-Algorithmus 8 Schritte (0.23^o Fehlanpassung) und für den Blauauszug 16 Schritte (0.17^o Fehlanpassung). Insgesamt dauerte diese Ausrichtung auf einem Pentium 133 etwa 50 Sekunden.

4.6.5 Offset und Drehwinkel: Transporter-Bild, Vergleich der Rechenzeiten mit verschiedenen Optimierungen

Abbildung 57: Das Eingabebild (Transporter) vor der Ausrichtung. Das Bild hat die Abmessungen 426x438 Punkte. Die Drehwinkel sind rot-grün= 2^o, rot-blau= -4^o. Die Offsets sind grün= $(20, 0)$ und blau= $(0, 30)$.

Abbildung 58: Das Eingabebild (Transporter) nach der Ausrichtung

Auf Abbildung 57 und Abbildung 58 sieht man einen Bildausschnitt von Abbildung 21 vor und nach der Ausrichtung. Um den starken Einfluß der Bildkanten zu mildern, ist in der aus verdrehten und verschobenen Farbauszügen zusammengesetzten Abbildung 57 der Randbereich abgeschnitten.

	512x512	256x256	128x128	64x64
Winkelbestimmung[sec]	98.39	22.02	5.66	1.22
Fehler rot-grün[°]	0.002	0.028	0.088	0.850
Fehler rot-blau[°]	0.009	0.001	0.208	0.828
Offsetbestimmung[sec]	8.61	8.88	8.39	9.61

Tabelle 10: Vergleich verschiedener FFT-Auflösungen bei der Rejustierung. Nicht exakt passende Bildabmessungen werden mittels eines einfachen Algorithmus skaliert. Alle Zeiten sind auf einem P133 ermittelt worden.

In Tabelle 10 sehen wir in der Zeile „Winkelbestimmung“ die Rechenzeit, die der Algorithmus für zwei Durchläufe für den Blauauszug und Grünauszug zusammen benötigt. Bei dem angegebenen Beispiel ist in der Darstellung von Abbildung 58 kein Unterschied zu entdecken bei der hier möglichen Wiedergabequalität bis zu einer FFT-Auflösung von 64x64. Bei 64x64 tritt bei der Offsetbestimmung des Blauauszuges durch den beträchtlichen Fehler der Winkelbestimmung ein Fehler von etwa 15 Pixeln auf. Die Offsetbestimmung läuft also in ein Nebenminimum.

Schrittweite	Grün 2x[sec]	Grün 1x[sec]	Blau 2x[sec]	Blau 1x[sec]
1x	0.65	1.13	0.82	1.41
2x	0.20	0.57	0.38	0.36
3x	0.06	0.13	0.14	0.19
4x	0.08	0.12	0.05	0.10

Tabelle 11: Vergleich verschiedener Schrittweiten bis 4x des Auslassungs-Verfahrens in diesem Beispiel. Damit sich die oben angegebenen 8.88 Sekunden ergeben, muß man zur Summe der Zeiten einer Zeile noch die Summe der Zeiten für die Initialisierung und Schwerpunktbestimmung hinzuaddieren. Das sind etwa 4.5 Sekunden in diesem Fall, die unabhängig von der Schrittweite des Auslassungsverfahren sind.

Die in Tabelle 11 gezeigten Schrittweiten führen alle zur Sollposition. Die Offsetbestimmung wurde in allen Fällen mit einer zweistufigen Pyramide gemacht (x2,x1). Das Minimierungsverfahren war „Powell“ und die Kennzifferfunktion die Differenzformel. Bei zusätzlicher Benutzung des Auslassungsverfahrens reduziert sich die Rechenzeit in der vorliegenden Implementation nur etwa auf die Hälfte, weil der Overhead der Initialisierungsfunktionen so hoch ist. Dieser Overhead kann leicht reduziert werden, allerdings wurde unter dem Gesichtspunkt

der Flexibilität für diese Arbeit darauf verzichtet. Trotzdem benötigt ein kompletter Lauf inklusive Einlesen des Eingabebildes und Ausgabe des Bildes nur etwa 15 Sekunden. Das umfaßt die Ausrichtung von zwei Farbauszugspaaren mit Drehwinkel- und Offsetbestimmung.

4.7 Fazit

Im vorangegangenen Kapitel wurden verschiedene Aspekte der angeführten Methoden untersucht. Es wurde gezeigt, daß bei Drehung und Offsetbestimmung bei natürlichen Bildern fast immer Nebenminima auftreten. Diese Nebenminima werden aber nur dann zum Problem, wenn ihr Abstand zur Sollposition so gering ist, daß der Fehler der Schwerpunktbestimmung in den Einzugsbereich des Nebenminimums führt.

Außerdem wurde gezeigt, daß ein Gaußfilter den Abstand der Nebenminima von der Sollposition vergrößern kann.

Untersuchungen zum Einfluß auf die Rechenzeit von verschiedenen Parametern deuten darauf hin, daß bestimmte rechenzeitverringernde Maßnahmen bis zu einem gewissen Grad mit wenig oder gar keinem negativen Effekt auf die Genauigkeit der Bestimmung des Drehwinkels und des Offsets benutzt werden können.

5 Ausblick

Die vorliegende Offsetbestimmung erreicht eine pixelgenaue Ausrichtung. Mit gewissen Erweiterungen wäre es denkbar, auch Subpixelauflösungen zu erreichen [6]. Bei der Drehwinkelbestimmung ist ebenfalls noch ein erheblicher Spielraum für Verbesserungen hinsichtlich Subpixelauflösungen vorhanden. Zwar ist der Drehalgorithmus in der Lage einzelne Bildpunkte um weniger als eine Pixelbreite zu verschieben. Dennoch ist eine Erhöhung der Auflösung durch eine Fourierinterpolation von Bildausschnitten möglich. Damit kann man wiederum iterativ zu höheren Auflösungen kommen und die Genauigkeit schrittweise verbessern.

Die Rechenzeit kann gegenüber der vorliegenden Implementation durch Verbesserung der Abbruchkriterien in den verwendeten Funktionen zur Minimierung und schnellere Verfahren zur Drehung von Bildern (wie zB. in dem „texture mapping“ Bereich benutzt) weiter verringert werden. Die inhärente Parallelisierbarkeit vieler Teilprobleme in dem vorgestellten Algorithmus läßt es sogar möglich erscheinen, ihn auf entsprechender Hardware potentiell echtzeitfähig zu implementieren.

Denkbar wäre auch eine Strategie, bei der Offsetberechnung und Drehung in einer konstanten Bildmatrix (zB. 128x128) in einem Unterprogramm „Work“ erfolgt. Dieses Unterprogramm „Work“ wird initialisiert mit der schlechtesten Auflösung und liefert beim 1. Durchgang verbesserte Startwerte für den nächsten Lauf. In einer äußeren Schleife wird in der konstanten Bildmatrix die Auflösung bis zur höchsten Stufe verbessert.

5.1 Mustererkennung

In der allgemeinen Mustererkennung ist dies Verfahren dann gut geeignet, wenn Kenntnisse von Bildinhalten nicht vorliegen. Wenn man dies Verfahren z.B. in der Schrifterkennung einsetzen würde, wäre es konventionellen OCR Verfahren sicherlich unterlegen. Zwar könnte man auch dann alle möglichen Muster miteinander vergleichen, aber dazu muß man alle auftretenden Schriften erst einmal kennen.

Ohne vorherige Kenntnis des Bildinhaltes ermöglicht es die Bestimmung eines numerischen Wertes, der im Fall der Übereinstimmung ein Minimum annimmt bei einer Verschiebung und Drehung. Problemverwandte Lösungen sind unter anderem in [8] diskutiert.

In einem weiteren Schritt könnte man suksessiv die zwei zu vergleichenden Bilder in immer kleinere Bildbereiche aufteilen und diese Bildbereiche nach dem oben beschriebenen Minimierungsverfahren positionieren. Dann erhält man ein 2 dimensionales Vektorfeld mit Verschiebungsvektoren, die die Verschiebung korrespondierender Bildbereiche dokumentieren. Damit können z.B. Geschwindigkeiten von Objekten ortsaufgelöst bestimmt werden. Außerdem führt das auf eine weitere Anwendung:

5.2 3d Rekonstruktion anhand von Stereobildern

Bei Stereobildern gibt die Länge dieser Vektoren im ortsaufgelösten 2 dimensionalen Vektorfeld der Verschiebungsvektoren direkt die Entfernung der korrespondierenden Objekte an. Man kann damit die 3 dimensionalen Geometriedaten von Objekten anhand eines Paares aus zwei Blickwinkeln aufgenommener Stereobilder bekommen.

5.3 Positionierung von Kunststoffformen

Es gibt Fälle in denen einfarbige, schwarze Kunstoffteile für die weitere Bearbeitung optisch erfaßt werden müssen. Dabei ist es wichtig, Ort und Lage eines Objektes nach gewissen Kriterien auszurichten. Ein Vergleichsbild liegt dabei vor. Ein Beispiel sind Kunststoffformen, deren Ränder nach dem Spritzguß geschnitten werden müssen. In so einem Fall ist das vorliegende Verfahren ebenfalls durchaus geeignet.

A Beschreibung des Sourcecodes

A.1 Bildbibliothek

Datei	Beschreibung
Makefile	Abhängigkeiten und Compileroptionen
Makefile.in	Abhängigkeiten und Compileroptionen (Vorlage)
amoebacall.c	Aufruf des Amoeba-Algorithmus
anaproio.c	Lade- und Speicherfunktionen für Berechnungsdaten
bild.h	Definitionen und Prototypen von Bildfunktionen und Datentypen
bildfehler.c	Funktion zur Ausgabe von Fehlermeldungen
computing.c	Berechnungen auf dem 2d Raster
debug.c	Fürs Debugging nützliche Funktionen
doall.c	Funktion, die der Reihe nach Winkel und Offsets bestimmt
config.h	Präprozessordefines zur Konfigurationsbestimmung
config.h.in	Präprozessordefines zur Konfigurationsbestimmung (Vorlage)
extremumfind.c	Gemeinsame Minimierungsinitialisierung
filters.c	Filter zur Bildvorverarbeitung (Laplace,Sobel,..)
fourierall.c	Fouriertransformation auf Bilddaten
handlenr.c	Postprocessing zur Fouriertransformation
libbild.a	Archivedatei für Bildfunktionen
init.c	Initialisierungen für temporäre Daten
kennziffer.c	Kennzifferberechnung
magic.c	Feststellung des Fileformats
magic.h	Definetabelle Fileformat
mathematica.c	Ausgabe in Mathematicakompatible Skriptdatei
powellsmethod.c	Aufruf des Powell-Algorithmus
readin.c	Funktion zum Datei Einlesen
schwerpunkt.c	Schwerpunktbestimmung
winkelfind.c	Winkelbestimmung für zwei Farbauszüge
writebild.c	Funktion zum Datei schreiben
writergb.c	Funktion zum Exportieren des Eingabebildes als Farbauszüge

Tabelle 12: Bestandteile der Bildbibliotheksfunktionen

Alle angegebenen Algorithmen, die nicht in [17] enthalten sind, sind in einer Linkbibliothek zusammengefaßt. Zusätzlich finden sich hier noch einige Funktionen zur Ein- und Ausgabe von Bildern und Zwischenergebnissen in magic.c, magic.h, mathematica.c, readin.c, writebild.c, writergb.c und anaproio.c.

A.2 NRbibliothek

Datei	Beschreibung
Makefile	Abhängigkeiten und Compileroptionen
Makefile.in	Abhängigkeiten und Compileroptionen (Vorlage)
amoeba.c	Amoeba-Algorithmus
amotry.c	Amotry-Algorithmus
brent.c	Brent-Algorithmus
f1dim.c	f1dim-Algorithmus
fourn.c	fourn-Algorithmus
libnr.a	Archivedatei für NR Algorithmen
linmin.c	linmin-Algorithmus
mnbrak.c	mnbrak-Algorithmus
nr.h	Prototypen der NR-Algorithmen
nrutil.c	Tools zu den NR-Algorithmen
nrutil.h	Prototypen zu den Tools
powell.c	powell-Algorithmus
rlft3.c	rlft3-Algorithmus

Tabelle 13: Bestandteile der NRbibliotheksfunktionen

Die aus [17] benutzten Funktionen sind ebenfalls in einer Linkbibliothek zusammengefaßt. Sie unterscheiden sich bis auf die kleinen Optimierungen für Ganzzahlgitter nicht von den veröffentlichten Funktionen.

A.3 Benutzerinterface und Beispielprogramme

Datei	Beschreibung
Makefile	Abhängigkeiten und Compileroptionen
Makefile.in	Abhängigkeiten und Compileroptionen (Vorlage)
ChangeLog	Protokoll der Änderungen im Sourcecode
README	kurze Beschreibung des Programms
alles	Executable zum Ausrichten gedrehter und verschobener Farbauszüge
alles.c	Programm zum Ausrichten gedrehter und verschobener Farbauszüge
allfunc.h	Prototypen der benutzten Funktionen
callbacks.c	Alle Motifcallbacks
callbacks.h	Prototypen der Callbacks
config.h	Präprozessordefines zur Konfigurationsbestimmung
config.h.in	Präprozessordefines zur Konfigurationsbestimmung (Vorlage)
configure	Automatisches Konfigurationsscript
configure.in	Automatisches Konfigurationsscript(Vorlage)
doc	Verzeichnis mit TEX-Dateien
drawfield.c	Funktion zum Zeichnen einer Kennzifferzeile
drehfilter	Executable zum Erzeugen gedrehter und verschobener Farbauszüge
drehfilter.c	Programm zum Erzeugen gedrehter und verschobener Farbauszüge
fehler.c	Funktionen zur Ausgabe von Fehlermeldungen
global.h	Präprozessordefines
gnuplotout.c	Ausgabe des Kennzifferfeldes in einem Gnuplotkompatiblen Format
initbaum.c	Initialisierung des (Motif-) Widgettrees
initbaum.h	Globale Variablen
install-sh	Fallback install script (benötigt von configure)
libbild	Verzeichnis der Bildbibliothek
libjpeg	Verzeichnis der Jpegbibliothek
libnr	Verzeichnis der NRbibliothek
main.c	Hauptprogramm mit Optionsparsing
movie.param	Parameterdatei für mpegencode
offset	Executable zum Ausrichten verschobener Farbauszüge
offset.c	Programm zum Ausrichten verschobener Farbauszüge
pixmap.h	Iconpixmap für Hauptprogramm
printing.c	Funktion zum Ausdrucken des Kennzifferbildes
schepper	Executable des Hauptprogramms
splitfind.ad	(Motif-) Resourcedatei des Hauptprogramms
testpic.c	Funktion zum erzeugen eines Testbildes
winkel	Executable zum Ausrichten gedrehter Farbauszüge
winkel.c	Programm zum Ausrichten gedrehter Farbauszüge
writeout.c	Funktion zum Berechnen der Kennziffer für den ganzen Bildausschnitt
xrender.c	Bildschirmausgabe des Kennzifferfeldes

Tabelle 14: Bestandteile des Programms

Das Benutzerinterface zu den Programmen alles, offset, winkel und drehfilter befindet sich in den entsprechenden C-Sourcecode-Dateien. Es beschränkt sich aber auf einige Kommandozeilenoptionen. Das grafische Benutzerinterface zum Testprogramm verteilt sich auf die übrigen Dateien. Zur Vereinfachung der Installation befindet sich hier auch ein configure Skript, das nach Aufruf eine Reihe von plattformabhängigen Parametern selbständig setzt.

Abbildungsverzeichnis

Tabellenverzeichnis

Literatur

[1] Milton Abramowitz and Irene A. Stegun. *Handbook of Mathematical Functions.* Dover Publications,Inc., New York, 1968.

[2] Dana H. Ballard and Christopher M. Brown. *Computer Vision.* Prentice Hall, 1982.

[3] Hans-Joachim Brede, Nicolai Josuttis, Sabine Lemberg, and Achim Lörke. *Programmieren mit OSF/Motif Version 2.* Addison-Wesley, Reading, Massachusetts, 1995.

[4] J. W. Cooley, P. A. W. Lewis, and P. D. Welsh. The fast Fourier transform and its application. *IEEE Trans. Education*, E-12:27–34, 1969.

[5] LeRoy E. DeMarsh and Edward J. Giorgianni. Color Science for Imaging Systems. *Physics Today*, 9:44–52, 1989.

[6] R.W. Frischholz and K.P. Spinnler. Bericht des Fraunhofer Instituts für Integrierte Schaltkreise IIS: Class of algorithms for realtime subpixel registration. 1997.

[7] Rafael C. Gonzalez. *Digital image processing.* Addison-Wesley, 1987.

[8] G.Sagerer, S.Posch, and F.Kummert. *Mustererkennung 1995.* Springer, 1995.

[9] Carsten Hellmich. Mustererkennungsprobleme bei der Ausrichtung zweidimensionaler Rasterpunktmuster. July 1993.

[10] Bernd Jähne. *Digitale Bildverarbeitung.* Springer, 1993.

[11] Stephan Klünker. Die Berechnung des optischen Flusses über eine Auflösungspyramide. November 1994.

[12] Franz Locher. *Numerische Mathematik für Informatiker.* Springer Verlag, 1992.

[13] Marshall Kir McKusick, Keith Bostic, Michael J. Karels, and John S. Quarterman. *The design and implementation of the 4.4BSD operating system.* Addison-Wesley, Reading, Massachusetts, 1996.

[14] J.A. Nelder and R. Mead. *Computer Journal*, 7:308–313, 1965.

[15] H. Niemann. *Klassifikation von Mustern.* Springer Verlag, 1983.

[16] Ioannis Pitas. *Digital Image Processing Algorithmens.* Prentice Hall, 1993.

[17] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes in C.* Cambridge University Press, 1994.

[18] Ajit Singh. *Optic Flow Computation: A Unified Perspective.* IEEE Computer Society Press, 1991.

[19] Mehrdad Soumekh. *Fourier Array Imaging.* P T R Prentice Hall, 1994.

[20] Rainer Steinbrecher. *Bildverarbeitung in der Praxis*. Oldenbourg, 1993.

[21] W. Richard Stevens. *Programmierung in der UNIX-Umgebung: die Referenz für Fortgeschrittene*. Addison-Wesley, Reading, Massachusetts, 1995.

[22] P. N. Swarztrauber. Symmetric FFTs. *Mathematics of Computation*, 47:323–346, 1986.

[23] Christoph Überhuber. *Computer Numerik 2*. Springer Verlag, 1995.