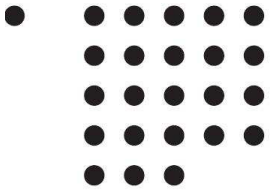


Fachhochschule Köln
Cologne University of Applied Sciences

Institut für Medien- und Phototechnik

Bachelorarbeit Medientechnik

Bestimmung der Kameraverzerrung mit Hilfe der Hough-Transformation

vorgelegt von

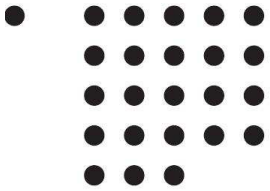
Gina Maria Wagner

Mat.-Nr. 11062084

Erstgutachter: Prof. Dr. rer. nat. Dietmar Kunz (Fachhochschule Köln)

Zweitgutachter: Prof. Dr.-Ing. Gregor Fischer (Fachhochschule Köln)

April 2011



Fachhochschule Köln
Cologne University of Applied Sciences

Institut für Medien- und Phototechnik

Bachelor Thesis

Determination of camera distortion by using the Hough Transform

submitted by

Gina Maria Wagner

Mat.-Nr. 11062084

First Reviewer: Prof. Dr. rer. nat. Dietmar Kunz
(Cologne University of Applied Sciences)

Second Reviewer: Prof. Dr.-Ing. Gregor Fischer
(Cologne University of Applied Sciences)

April 2011

Bachelorarbeit

Titel: Bestimmung der Kameraverzerrung mit Hilfe der Hough-Transformation

Gutachter: Prof. Dr. rer. nat. Dietmar Kunz (Fachhochschule Köln)
Prof. Dr.-Ing. Gregor Fischer (Fachhochschule Köln)

Zusammenfassung: Die Hough-Transformation liefert die Parameter von in einem Bild vorkommenden Geraden. Durch die Kameraverzerrung werden die in der realen Welt vorkommenden Geraden jedoch nicht mehr auf Geraden abgebildet. Dies führt dazu, dass im Hough-Raum die entsprechenden Geradenpunkte nicht mehr auf den gleichen Punkt abgebildet werden. Durch Variation der Verzerrungsparameter mit Hilfe eines bereits vorliegenden Optimierungsprogrammes ist eine Korrektur zu ermitteln. Dabei sollen drei unterschiedliche Ansätze miteinander verglichen werden. Bildmaterial sollte Aufnahmen von Testobjekten und Gebäuden sein.

Stichwörter: Verzeichnung, Hough-Transformation, Nelder-Mead-Verfahren, Bildverarbeitung, ImageJ

Sperrvermerk: kein Sperrvermerk

Datum: 04. April 2011

Bachelor Thesis

Titel: Determination of camera distortion by using the Hough Transform

Reviewers: Prof. Dr. rer. nat. Dietmar Kunz
(Cologne University of Applied Sciences)
Prof. Dr.-Ing. Gregor Fischer
(Cologne University of Applied Sciences)

Abstract: The Hough Transform provides the parameters of lines occuring in an image. Due to the camera distortion lines from the real world are not mapped to straight lines. This leads to the fact, that corresponding line points are not mapped to one single point in the Hough domain. By using a given optimization algorithm, a correction is supposed to be determined through varying the distortion parameters. For this purpose, three different approaches are compared. Graphical material should consist of pictures of test objects and buildings.

Keywords: Distortion, Hough Transform, Nelder-Mead Method, Image Processing, ImageJ

Restriction notice: none

Date: 04. April 2011

Inhaltsverzeichnis

1 Einleitung 1

2 Theoretischer Hintergrund 1

2.1 Verzeichnung 2

2.1.1 Radiale Verzeichnung 2

3 Vorgeschlagene Methode 4

4 Testmaterial 6

4.1 Künstlich erzeugte Testbilder 6

4.2 Testkameras und Testobjektive 8

4.3 Testaufnahmen von Objekten 9

4.4 Aufnahmen von Gebäuden 9

5 Verwendete Methoden 11

5.1 Die Hough-Transformation 11

5.2 Nelder-Mead-Verfahren 14

6 Implementierung 16

6.1 Software 16

6.2 Aufbau des Programms 16

6.2.1 Vorverarbeitung der Bilder 17

6.2.2 Benutzerdefinierte Parametereinstellungen 18

6.2.3 Die Methode optimize() 18

6.2.4 Hough-Transformation 20

6.2.5 Die Qualitätskriterien 24
6.2.5.1 Die Methode entropy() 25
6.2.5.2 Die Methode variance() 27
6.2.5.3 Die Methode countMaxima() 28
6.2.6 Abbruchkriterium 29
6.2.7 Entzerrung 30

7 Ergebnisse 31
7.1 Künstlich erzeugte Testbilder 32
7.1.1 Änderung der Einstellungen 32
7.1.2 Änderung der Zielfunktion 35
7.2 Reale Testaufnahmen 39

8 Diskussion 48

9 Anhang 50
9.1 Quellenverzeichnis 50
9.2 Abbildungsverzeichnis 52
9.3 Tabellenverzeichnis 54

Eidesstattliche Erklärung 55

Sperrvermerk 56

Weitergabeerklärung 57

1 Einleitung

Die vorliegende Arbeit beschäftigt sich mit dem Problem der Kameraverzerrung verursacht durch ein optisches System im Allgemeinen und die dazu gehörende radiale Verzeichnung im Speziellen.
Nach dem heutigen Stand der Technik bestehen bereits diverse Ansätze zur Korrektur derartiger Abbildungsfehler, einerseits durch integrierte optische Verfahren, andererseits durch nachträgliche elektronische Bildbearbeitung.
Im Folgenden wird ein Verfahren zur nachträglichen Entzerrung von Bildmaterial vorgestellt werden. Es basiert auf der Hough-Transformation, die es ermöglicht, Geraden im Bild zu detektieren. Über ein Optimierungsverfahren werden mittels einer geeigneten Zielfunktion die besten Parameter zur Entzerrung gefunden. Hierbei werden drei unterschiedliche Ansätze zur Implementierung dieser Zielfunktion vorgestellt und miteinander verglichen.
Ziel der Arbeit ist es, durch das vorgestellte Verfahren und der am besten geeigneten Zielfunktion sowohl künstlich erzeugte als auch reale Bilddaten erfolgreich entzerren zu können.

2 Theoretischer Hintergrund

Theoretischer Hintergrund dieser Arbeit sind Abbildungsfehler (Aberrationen).
In der Optik versteht man unter Abbildungsfehlern Abweichungen von „den idealisierten Bedingungen der gaußschen Optik“ [1], welche sich generell in zwei Gruppen einteilen lassen: Neben den chromatischen Aberrationen, den Farbfehlern, gibt es die monochromatischen Aberrationen, die geometrische Verzerrungen zur Folge haben.
Zu letzteren gehört neben diversen anderen auch die Verzeichnung [1], die die Grundlage dieser Arbeit bildet und die im Folgenden genauer betrachtet werden soll.

2.1 Verzeichnung

Grundsätzlich gibt es zwei Arten von Verzeichnung, die tangentiale und die radiale Verzeichnung. Aufgrund der Tatsache, dass die radiale Verzeichnung einen weitaus größeren Einfluss auf eine Abbildung hat, insbesondere in Hinblick auf die betrachteten Weitwinkelaufnahmen, wird auf diese der Schwerpunkt gelegt und die tangentiale Verzeichnung vernachlässigt werden.

2.1.1 Radiale Verzeichnung

Da die radiale Verzeichnung wie der Name schon sagt abhängig vom Radius ist (siehe Abbildung 1), ist sie in den einzelnen Bildregionen unterschiedlich stark ausgeprägt. In der Nähe des Bild- bzw. Verzerrungszentrums ist die Verzeichnung am geringsten, mit steigendem Radius nimmt sie zu.

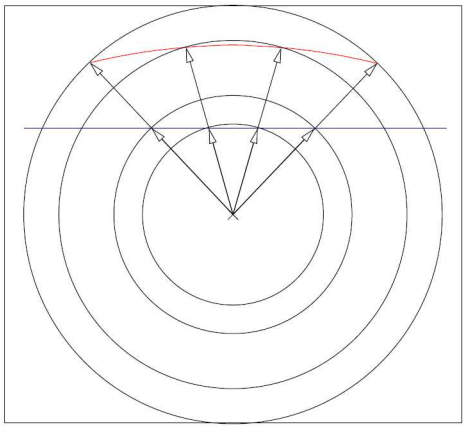


Abb. 1: radiale Verzerrung einer Geraden [2]

Radiale Verzeichnung entsteht durch die Begrenzung des Strahlenbündels durch beispielsweise Aperturen oder Linseneinfassungen, die wie eine Blende wirken können [3]. Je nach Anordnung von Linse und Blende, kommt es zu einer Vergrößerung bzw. Verkleinerung der Objektweite und somit zu einer Änderung des Abbildungsmaßstabes β.

Wird die Blende vor einer Positivlinse positioniert, entsteht eine negative, tonnenförmige (barrel) Verzeichnung. Die Objektweite nimmt zu und der Abbildungsmaßstab folglich ab. Befindet sich die Blende hinter der Linse, verkleinert sich die Objektweite, was zu einem größeren Abbildungsmaßstab und einer positiven, auch kissenförmig (pincusion) genannten Verzeichnung führt [3]:

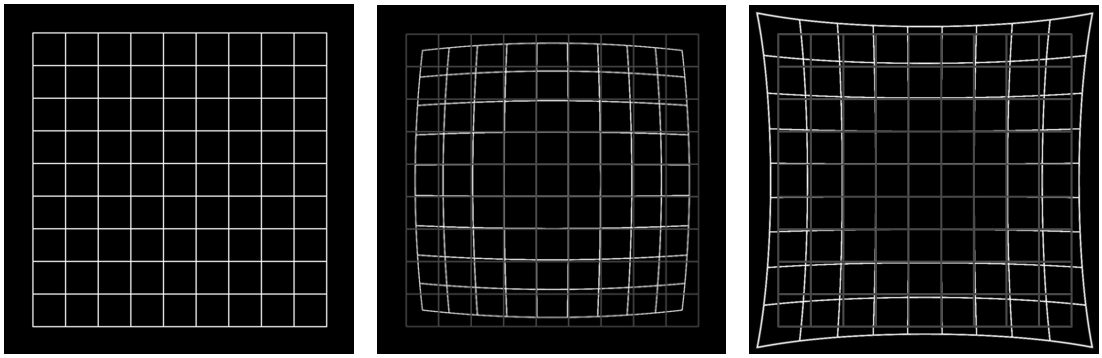


Abb. 2: Original, tonnenförmige Verzeichnung (0.5; 0.3; 0.1; 0.05) und kissenförmige Verzeichnung (-0.3; -0.1; -0.05; -0.01) von Gitter.tif

Beide Verzeichnungsarten können wie folgt nachgebildet werden:
Der neue verzerrte Bildpunkt wird berechnet, indem der entsprechende alte mit einem bestimmten Verzeichnungsfaktor $L(r)$ multipliziert wird. Dieser Faktor setzt sich aus einem Polynom zusammen und ist abhängig von dem Radius r zwischen dem betrachteten Bildpunkt und dem Verzerrungszentrum. In unserem Falle besitzt $L(r)$ insgesamt vier Parameter r_1, r_2, r_3 und r_4, die die Stärke und Art der Verzeichnung bestimmen. Des Weiteren wird in den folgenden Betrachtungen angenommen, dass das Verzerrungszentrum dem Bild- bzw. optischen Zentrum entspricht.

$$x_{neu} = x_{alt} \cdot L(r) = x_{alt} \cdot \left(1 + r_1 \cdot r^2 + r_2 \cdot r^4 + r_3 \cdot r^4 + r_4 \cdot r^6\right)$$

$$y_{neu} = y_{alt} \cdot L(r) = y_{alt} \cdot \left(1 + r_1 \cdot r^2 + r_2 \cdot r^4 + r_3 \cdot r^4 + r_4 \cdot r^6\right)$$

3 Vorgeschlagene Methode

Es stehen selbst angefertigte reale sowie künstlich erzeugt Testbilder zur Verfügung, die zahlreiche Geraden enthalten, über die die Verzeichnung des jeweiligen Testobjektives bestimmt werden kann.

Zur Bestimmung dieser Verzeichnung werden die einzelnen Testbilder zunächst der Hough-Transformation (HT) unterzogen [p1]. Das Ergebnis dieser Transformation ist ein zweidimensionales Histogramm, der Hough-Raum, in dem zu erkennen ist, wo im Bild sich Geraden befinden und wie stark diese sind. Dabei wird jede Gerade auf einen Punkt im Hough-Raum abgebildet. Je stärker eine Gerade im Originalbild, desto höher ist der Wert im Hough-Raum, der über ein Akkumulator-Array gespeichert wird. Des Weiteren sagt die Verteilung im Hough-Raum etwas über die Beschaffenheit der gefundenen Gerade aus. Je mehr sich die Pixelwerte im Hough-Raum auf einen Punkt konzentrieren, desto unverzeichneter ist die entsprechende Gerade im Originalbild, was umgekehrt bedeutet, dass eine verzerrte Gerade nicht auf einen einzigen Punkt sondern um diesen herum gestreut abgebildet wird. Diese Eigenschaft kann nun genutzt werden, um die Verzeichnung der Aufnahme zu bestimmen. Dazu muss zunächst ein Qualitätskriterium gefunden werden, welches ein Maß für die Konzentration auf einen bestimmten Punkt darstellen kann.

In dieser Arbeit werden drei unterschiedliche Ansätze für diese Messziffer vorgestellt.
Zum einen wird die Entropie um das jeweilige Geradenmaximum berechnet. Je kleiner der Wert der Entropie, desto konzentrierter ist die Verteilung auf einen Punkt. Der zweite Ansatz berechnet die Varianz um das jeweilige Maximum, die maximal werden muss und im dritten Ansatz wird die jeweilige Höhe des Maximums als Maßzahl genommen.

Alle drei Qualitätskriterien bilden jeweils die zu optimierende Zielfunktion eines Optimierungsalgorithmus. Das angewendete Optimierungsverfahren ist das Downhill-Simplex-Verfahren nach Nelder und Mead [7],[p3]. Dieses Verfahren bekommt durch eine Initialisierungsmethode geeignete Startparameterwerte sowie den zuvor berechneten Wert der Zielfunktion des Eingangsbildes mitgeteilt. Die Startwerte für die Verzerrungsparameter werden mithilfe des Optimierungsalgorithmus so lange verändert, bis die besten Werte gefunden sind, die Zielfunktion also entsprechend der unterschiedlichen Ansätze minimal bzw. maximal wird. Des Weiteren muss ein Abbruchkriterium für den Algorithmus gegeben sein.

Sind die besten Werte für die Verzerrungsparameter gefunden, kann das Bild über die diese Einstellungen optimiert, entzerrt werden.

4 Testmaterial

Als Testmaterial zur Bestimmung der Verzeichnung und anschließender Entzerrung wurden sowohl künstlich erzeugte Bilddaten als auch reale Objekt- und Architekturfotografien verwendet und ausgewertet.

4.1 Künstlich erzeugte Testbilder

Der Vorteil von künstlich erzeugten Testbildern ist, dass die Verzerrungsparameter des Bildes bekannt sind. Somit ist eine so genannte Grundwahrheit vorhanden, die bei späteren Auswertungen als Referenz zum berechneten Ergebnis verwendet werden kann.
Inhalt dieser Testbilder sind einzelne und mehrere Geraden unterschiedlicher Ausrichtung, Dicke und Länge, wie beispielsweise die folgenden:

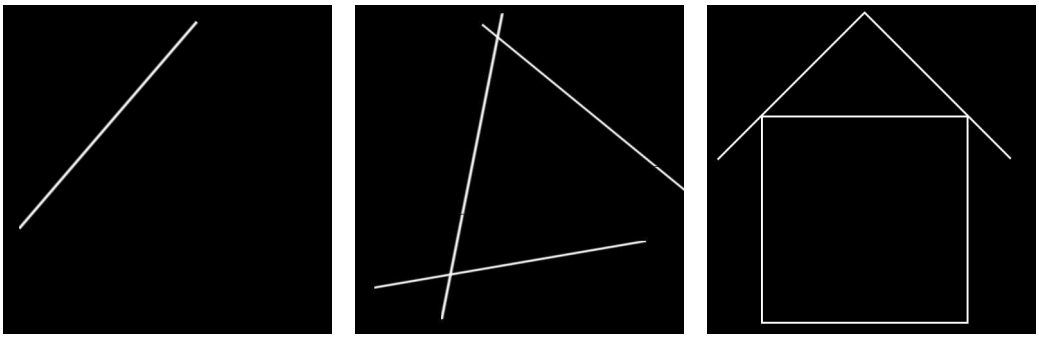


Abb. 3: künstlich erzeugte Testbilder: Gerade01.tif, Test01.tif und Haus.tif

Mit Hilfe des ImageJ-Plugins RadialDistortion_ wurden diese Testdateien mit verschiedenen Parametern radial verzerrt. Die verzeichnende Koordinatentransformation findet hierbei über das Target-to-Source-Verfahren (Rückwärtsabbildung) statt. Dies bedeutet, dass „man zu jedem Pixel [...] des neuen Koordinatensystems die Koordinaten [...] im alten Koordinatensystem bestimmen“ [4] muss.

Da dieser Punkt jedoch zwischen den alten Gitterpunkten liegt, muss er über Interpolation dieser bekannten umliegenden Gitterpunkte berechnet werden. Die spätere Entzerrung des Bildes wird allerdings genau über den umgekehrten Weg des Source-to-Target-Verfahrens vorgenommen, damit die gleiche Verzeichnungsfunktion

$$x_{neu} = x_{alt} \cdot L(r) = x_{alt} \cdot \left(1 + r_1 \cdot r^2 + r_2 \cdot r^4 + r_3 \cdot r^4 + r_4 \cdot r^6\right)$$

auch für die Entzerrung übernommen werden kann, und nicht die weitaus schwierigere zu bildende Umkehrfunktion dazu.
Ein sichtbarer Nachteil der letzteren Vorwärtsabbildung ist allerdings, dass im entzerrten Bild Lücken entstehen können, da „einzelne Pixel des neuen Bildes [...] keinen Wert zugewiesen“ [4] bekommen. Dies wird jedoch aufgrund der Tatsache, dass es auf das eigentliche Ergebnis keinerlei negative Auswirkungen hat in Kauf genommen. Das nachfolgende Code-Beispiel zeigt die Berechnung der Verzerrung:

```
for(int x = 0; x < width; x++){
  for(int y = 0; y < height; y++){

    double r = ((x-xCtr)*(x-xCtr)+(y-yCtr)*(y-yCtr))/maxR;
    double fac = (((r4 * r + r3)* r + r2)* r + r1)* r + 1;

    int xDis = (int)(xCtr + (x-xCtr) * fac);
    int yDis = (int)(yCtr + (y-yCtr) * fac);

    if((xDis>0 && xDis<width) && (yDis>0 && Dis<height)){

      a[y * width + x] = ip.getPixel(xDis, yDis);

}    }    }
```

Die verwendeten Verzerrungswerte finden sich jeweils im Dateinamen des Bildes wieder, beispielsweise entsteht das verzeichnete Bild Gerade01_09050501.tif in Abbildung 4 über eine Verzerrung des Originalbildes Gerade01.tif mit $r_1 = 0.9$, $r_1 = 0.5$, $r_1 = 0.5$ und $r_1 = 0.1$.

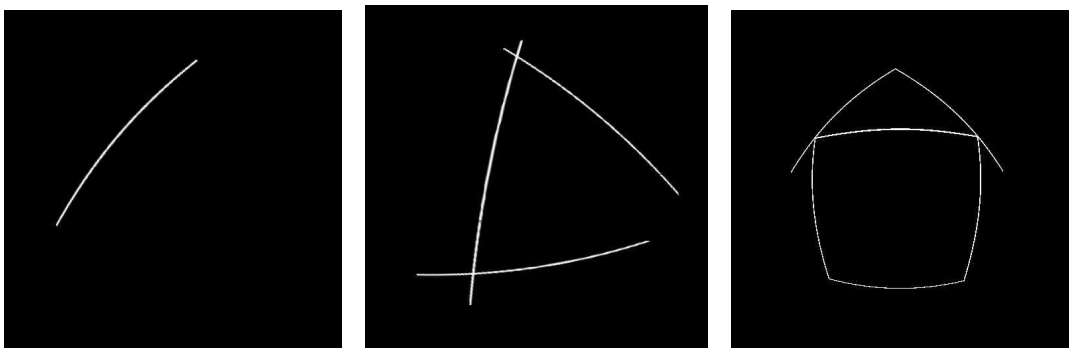


Abb. 4: verzeichnete Testbilder:
Gerade01_09050501.tif, Test01_0503005001.tif und Haus_19090909.tif

Insgesamt wurden für die hier vorzustellenden repräsentativen Testbilder vier unterschiedliche Parametereinstellungen vorgenommen, wobei der Schwerpunkt deutlich auf der tonnenförmigen Verzeichnung liegt, da ausschließlich diese bei Weitwinkelaufnahmen auftritt. Der Vollständigkeit halber ruft der vierte Test-Parametersatz eine kissenförmige Verzerrung hervor. Somit kann getestet werden, inwiefern sich die vorgestellten Methoden auch für diese Art von radialer Verzeichnung eignen.

4.2 Testkameras und Testobjektive

Zunächst wurde für die Aufnahmen von Testobjekten eine Canon EOS 350D mit einer Brennweite von 18mm (bzw. 28,8mm aufgrund des Crop-Faktors) verwendet.
Für die Architekturaufnahmen wurde unter anderem ein 8-16mm Testobjektiv von Canon verwendet. Kombiniert wurde dieses Weitwinkel-Objektiv zunächst mit der Canon EOS-1Ds Mark, die einen Vollformat-CMOS-Sensor mit ca. 16,7 Megapixeln besitzt. Zum Vergleich wurde zusätzlich eine Canon EOS 40D (10,1 Megapixel CMOS-Sensor) im APS-C-Format (Cropfaktor 1,6) eingesetzt, sodass insgesamt Brennweiten von 8mm bis 16mm (bzw. 25,6mm) getestet werden konnten.
Des Weiteren wurden Aufnahmen mit einem 10-22mm Canon- sowie einem 10-20mm Sigma-Objektiv mit der Nikon D300 (DX-Format) gemacht, die jedoch aufgrund des Crop-Faktors und der daraus resultierenden höheren Brennweite keine sichtbaren Verzeichnungen aufwiesen.

4.3 Testaufnahmen von Objekten

Um einen Übergang von den künstlich erzeugten Testbildern zu den Architekturaufnahmen herzustellen, wurden Fotos von Testobjekten verwendet, die eine Vielzahl an Geraden aufweisen (Bilderrahmen, Kommode). Diese ermöglichen es, erste Auswertungen mit realen Daten zu testen, bevor die eigentlichen Architekturaufnahmen verwendet werden.

Canon 18mm (Canon EOS 350D):

Abb. 5: Rahmen02.jpg
(f4; 1/60; 18mm)

Abb. 6: IMG_0077.jpg
(f8; 1/200; 18mm)

4.4 Aufnahmen von Gebäuden

Für den zweiten Satz an realen Bilddaten wurden Architekturaufnahmen angefertigt, da diese im Besonderen dem Problem der Kameraverzeichnung unterliegen. Architekturfotografie hat sowohl den Anspruch möglichst das gesamte Gebäude oder Bauwerk aufzunehmen, was zu einer möglichst geringen Brennweite führt, als auch die Linien und Geraden im Bild naturgetreu abzubilden. Abgesehen von eventuellen

gestalterischen Aspekten ist dies beispielsweise wichtig für die photogrammetrische Arbeit (zur Vermessung von Bauwerken). Somit sind die Aufnahmen von Gebäuden mit weitwinkligen Objektiven ideal für die Auswertung von realem Testmaterial. Untenstehend werden beispielhafte Testaufnahmen vorgestellt, die auch später in den Ergebnissen ausgewertet werden. In der Bildunterschrift sind neben der jeweiligen Blende auch die Belichtungszeit und die Brennweite aufgeführt. Im Anhang auf der beiliegenden CD befinden sich zusätzliche Aufnahmen und die dazugehörigen Auswertungen.

Sigma 8-16mm (Canon EOS-1Ds Mark) :

Abb. 7: YV6P0015.jpg
(f8; 1/160; 8mm)

Abb. 8: YV6P0023.jpg
(f8; 1/250; 16mm)

Sigma 8-16mm (Canon EOS 40D):

Abb. 9: _MG_1305.jpg
(f5,6; 1/320; 8mm)

Abb. 10: _MG_1314.jpg
(f5,6; 1/200; 8mm)

5 Verwendete Methoden

5.1 Die Hough-Transformation

Die Hough-Transformation dient im Allgemeinen dazu, beliebige geometrische Formen, die sich eindeutig parametrisieren lassen, in „Punktverteilungen“ [5] lokalisieren zu können.
Gemäß dem Thema dieser Arbeit wird sich die folgende Erläuterung der Hough-Transformation speziell auf die Detektion von Geraden in binären Kantenbildern beschränken, zunächst allgemein, später in Kapitel 6 anhand des verwendeten ImageJ-Plugins.
Aufgrund der Tatsache, dass man Geraden in einem Bild auffinden möchte, wird zunächst die allgemeine Beschreibung linearer Funktionen $y = k \cdot x + d$ betrachtet, wobei die Konstante k die Steigung, die Konstante d den y-Achsenabschnitt beschreibt. Die Hough-Transformation bedient sich allerdings nicht der Möglichkeit, über diese Geradengleichung alle möglichen Geraden ins Bild einzuzeichnen und dann ihre Punkte zu zählen, sondern genau dem umgekehrten Wege. Es werden vielmehr alle Geraden ermittelt, die sich in einem Punkt schneiden, ein so genanntes „Geradenbündel“ [5] wird bestimmt. So kann man Geraden dort detektieren, wo sich relativ viele Geraden in einem Punkt schneiden. Realisiert wird dies dadurch, dass der originale Bildraum, der x und y als Variablen besitzt sowie k und d als Konstanten, in einen Parameter- bzw. Hough-Raum transformiert wird, was folgendermaßen funktioniert:
Da nun im Grunde Punkte betrachtet werden sollen, durch die bestimmte Geraden verlaufen, sieht die Geradengleichung für einen bestimmten Punkt $p(x_0 / y_0)$ wie folgt aus: $y_0 = k \cdot x_0 + d$. Alle Geraden, die durch p laufen, müssen geeignete Werte für k und d aufweisen, diese sind nun also unsere Variablen, und x und y die Konstanten.

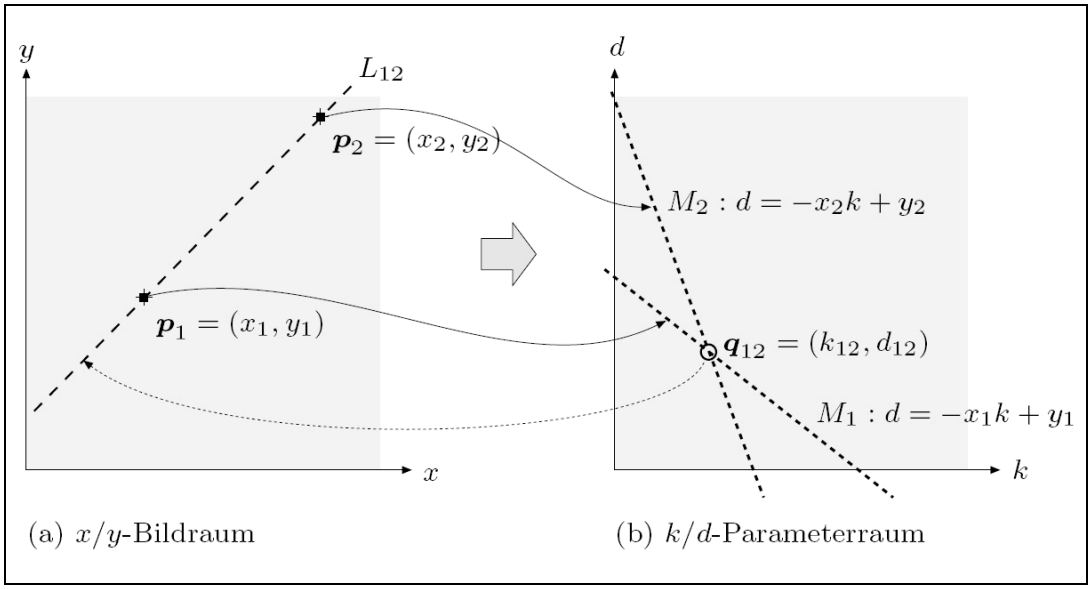


Abb. 11: Geraden im Bildraum (a) werden auf Punkte im Parameter-/Hough-Raum (b) abgebildet [5]

Formt man die Gleichung nach diesen um, erhält man: $d = -x_0 \cdot k + y_0$, was wiederum eine lineare Funktion ist. Freilich wird diese nun im k/d - Koordinatensystem eingetragen, was den Parameter- bzw. Hough-Raum darstellt.

Betrachtet man nun im Bildraum einen Punkt mit zwei festen Koordinaten, erhält man im Parameterraum eine Gerade, die diese Koordinaten als Steigung und y-Achsenabschnitt besitzt. Ein Punkt im Parameterraum, in dem sich viele verschiedene Geraden schneiden, besitzt ein bestimmtes Wertepaar (k/d), welches im Bildraum wieder eine Funktion der Steigung sowie des Achsenabschnittes übernehmen, sodass eine Gerade gefunden wurde (siehe Abbildung 11). Allgemein lässt sich sagen, dass wenn sich „N Geraden an einer Position ($k´$, $d´$) im Parameterraum schneiden, […] auf der entsprechenden Geraden $y = k \cdot x + d$ im Bildraum insgesamt N Bildpunkte [liegen]“ [5].

Damit könnte man sicherlich begrenzt arbeiten, allerdings tritt spätestens bei einer Steigung von unendlich, also bei Geraden, die vertikal verlaufen, ein Problem auf, da diese dann nicht bestimmbar sind. Somit wird eine günstigere Parametrisierung vorgenommen, nach Radius und Winkel (Polarkoordinaten) anstatt nach Steigung und Achsenabschnitt.

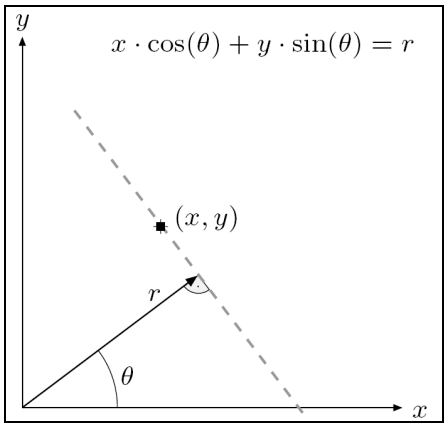


Abb. 12: Polarkoordinaten [5]

Dies entspricht der Hesseschen Normalform $r = x \cdot \cos(\varphi) + y \cdot \sin(\varphi)$ (HNF), die ein Wertepaar liefert, das dem Radius („senkrechter Abstand des Nullpunktes [...] von der Geraden“ [6]) und dem Winkel („Winkel zwischen dem Lot [...] auf die Gerade und der positiven x-Achse“ [6]) entspricht und durch welches im Bildraum die entsprechende Gerade definiert wird (siehe Abbildung 12).

Der Winkelbereich geht hierbei von 0 bis π, der Radius von dem negativen größten Radius bis zum positiven größten Radius. Betrachtet man das Bildzentrum als Ursprung, ist der größte Radius die halbe Bilddiagonale, also $0{,}5 \cdot \sqrt{M^2 + N^2}$, mit M als Bildbreite und N als Bildhöhe.

In Abbildung 13 kann man gut erkennen, welche Gerade im Bildraum zu welchem Punkt im Hough-Raum gehört. Zudem erkennt man, dass sich durch die Parametrisierung nach Winkel und Radius keine Geraden in einem Punkt schneiden sondern vielmehr Sinus- bzw. Kosinuswellen, entsprechend der Hesseschen Normalform.

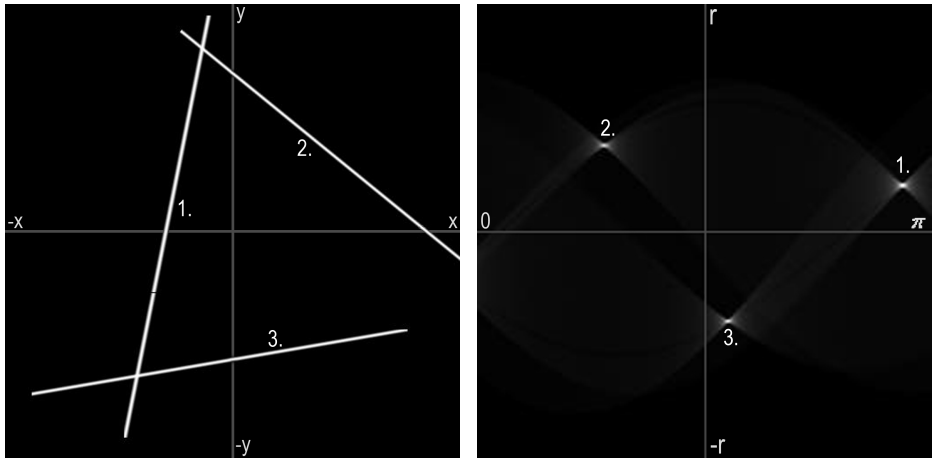


Abb. 13: Geraden im Bildraum (x/y) und ihre entsprechenden Punkte im Hough-Raum (Winkel/Radius)

5.2 Nelder-Mead-Verfahren

Das Nelder-Mead-Verfahren, welches auch Downhill-Simplex-Verfahren genannt wird, ist eine Methode zur Optimierung einer Funktion von n Parametern. Der Algorithmus basiert auf dem Vergleich der Funktionswerten von $n+1$ Eckpunkten eines allgemeinen Simplex (n-dimensionaler Körper) [7].
Im eindimensionalen Fall beispielsweise entspräche der Simplex einer Strecke, im zweidimensionalen Fall einem Dreieck. Jeder dieser Punkte bildet einen Parametersatz, der zu der benutzerdefinierten Zielfunktion gehört und der durch das Verfahren optimiert werden soll. Die Optimierung kann hierbei in zwei Richtungen erfolgen: entweder werden die Parameter dahingehend untersucht, dass die Zielfunktion unter ihnen minimal oder maximal wird. Dies muss der Benutzer selber festlegen ebenso wie die zu optimierende Zielfunktion und einen Startparametersatz beziehungsweise einen Startpunkt (siehe auch Kapitel 6.2.4).
Mit diesem Startparametersatz kann der Algorithmus nun wie folgt arbeiten:
Zunächst werden n weitere Punkte festgelegt, sodass der Start-Simplex aus den so erhaltenen $n+1$ Punkten aufgespannt werden kann. Nun werden diese $n+1$ Punkte und ihr entsprechender Wert der Zielfunktion ausgewertet. Der Punkt mit dem besten Ergebnis wird als bisher beste Lösung beibehalten, der mit dem schlechtesten Ergebnis wird durch einen neuen Punkt ersetzt.
Dieser neue Punkt wird bestimmt, indem durch verschiedene Operationen eine Koordinatenänderung vorgenommen wird: Operationen wie die Reflexion „am Mittelpunkt des restlichen Simplex“ [9] über den Faktor Rho, die Expansion in eine bestimmte Richtung über den Faktor Khi (Bezeichnung der Faktoren gemäß des verwendeten NelderMead-Alorithmus), die Kontraktion zum Mittelpunkt der übrigen Punkte (Faktor Gamma) oder die Komprimierung um den besten Punkt herum (Faktor Sigma), verändern die Punkte des Simplex so, dass dieser sich um das Optimum zusammenzieht und dieses im besten Fall gefunden wird [9]. Abbildung 14 zeigt diese verschiedenen Möglichkeiten der Simplex-Änderung.

Die oben genannten Operationen werden bei jedem Iterationsschritt mit dem neu gewonnenen Simplex wiederholt, solange bis das Optimum, d.h. die Parameter, die die Zielfunktion minimal bzw. maximal werden lassen, gefunden wurde.
Die Anzahl an Iterationen sowie ein Abbruchkriterium werden durch den Benutzer festgelegt.

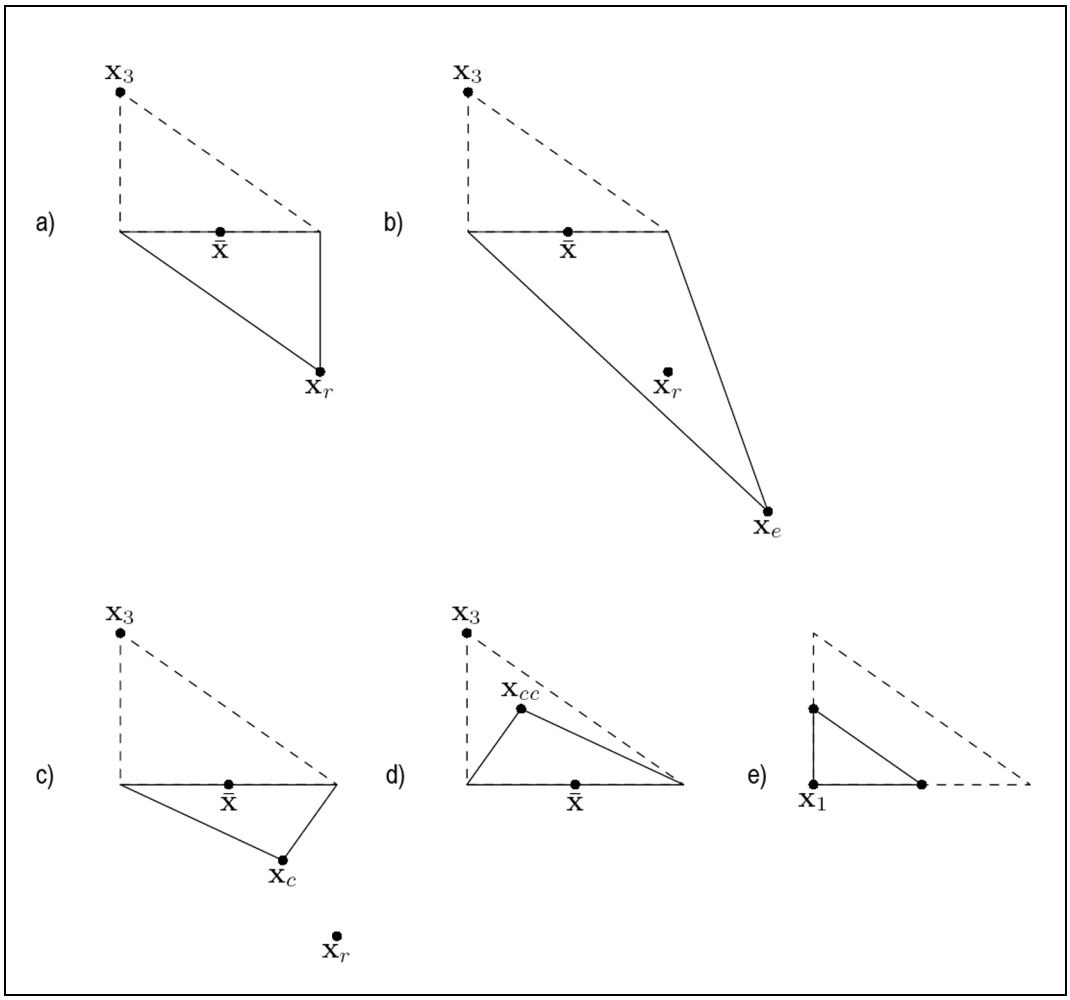


Abb. 14: Nelder-Mead-Simplex nach einer Reflexion (a), Expansion (b), Kontraktion nach außen (c), Kontraktion nach innen (d) und einer Komprimierung (e) [8]

6 Implementierung

6.1 Software

Das in der Programmiersprache Java implementierte Programm zur Optimierung der Kameraverzeichnung wurde mit Hilfe der Open-Source-Entwicklungsplattform Eclipse [p2] erstellt.
Das verwendete Plugin zur Berechnung der Hough-Transformation basiert auf dem frei verfügbaren Java-Code „Hough-Transformation für Geraden“ [p1], der für die Programmbeispiele aus [5] entnommen und an wenigen Stellen leicht modifiziert wurde.
Für die Implementierung des zuvor beschriebenen Optimierungsalgorithmus nach Nelder und Mead (Downhill-Simplex-Verfahren) wurde die Apache Commons Mathematics Library [p3] verwendet, die zu einem Projekt der Apache Software Foundation gehört.
Des Weiteren wurde das Programm mit der Bildbearbeitungssoftware ImageJ [p4] verknüpft, um Plugins zur direkten Visualisierung der Bild-Optimierung erstellen zu können.

6.2 Aufbau des Programms

Das implementierte Programm wird via ImageJ bzw. Eclipse je nach verwendeter Zielfunktion über das Plugin OptimizeDisEntropy_/ OptimizeDisVariance_/ OptimizeDisCount_ aufgerufen. In diesem Plugin wird die Optimierung der Verzerrungsparameter vorgenommen, es beinhaltet demnach neben dem Objekt des Nelder-Mead-Optimierers verschiedene Methoden, die für das Optimierungsverfahren notwendig sind und die im Folgenden näher beschrieben werden sollen. Die für das Auffinden der Geraden im Bild zuständige Hough-Transformation bildet ebenso wie der Nelder-Mead-Optimierer eine eigene Klasse, die im Plugin aufgerufen wird.

6.2.1 Vorverarbeitung der Bilder

Da die hier verwendete Hough-Transformation binäre Kantenbilder voraussetzt, muss aus dem Originalbild zunächst ein solches Kantenbild erzeugt werden. Dieses kann nach der Umwandlung des Originals in ein 8-Bit Graustufenbild über den Aufruf „Find Edges“ berechnet werden, was manuell im Menü Process in ImageJ oder aber über ein Macro ausgeführt werden kann. Macros sind in ImageJ einfache Programme, die es erlauben, eine Reihe von ImageJ-Befehlen und Plugins hintereinander auszuführen, wie das unten stehende Beispiel zeigt.

```
open("C:\\Dokumente und Einstellungen\\wagner\\Desktop\\BA\\Auswertung\\Architektur-Testbilder\\_MG_1305.jpg");
run("8-bit");
run("Find Edges");
run("Brightness/Contrast...");
setMinAndMax(50, 255);
```

Der Schritt der Kantenbild-Erzeugung bezieht sich selbstverständlich nur auf reale Testbilder, da es sich bei den künstlich erzeugten Bildern generell schon um solche handelt. Um den Effekt des Kantenbildes zu verstärken bzw. die Kanten insbesondere bei den realen Daten weiter hervorzuheben, wurde in diesen zusätzlich der untere Wert des Histogramms über den Brightness/Contrast-Filter auf 50 gesetzt.
Des Weiteren kommt bei den realen Testaufnahmen hinzu, dass diese oftmals zu groß sind, um über das Plugin in einer angemessenen Zeit verarbeitet werden zu können. Daher werden Versionen von den Originalaufnahmen in einer niedrigeren Auflösung verwendet. Bei Testbildern sehr geringer Brennweite wurde zudem der schwarze Rand um das eigentliche Motiv herum abgeschnitten. Dieser entsteht dadurch, dass durch die sehr weitwinklige Aufnahme Teile der Objektiv-Abschirmung mit aufgenommen werden.

6.2.2 Benutzerdefinierte Parametereinstellungen

Die Plugins bieten die Möglichkeit, verschiedene Parameter vor dem Start einzustellen, einerseits über eine entsprechende ImageJ-Gui, über die die Parameter-Werte bei jedem Aufruf per Hand eingegeben werden müssen, andererseits über den Aufruf des Plugins in einem Macro. Dies ermöglicht eine schnelle, automatisierte Bedienung des Programms insbesondere für Testdurchläufe und Auswertungen mit verschiedenen Voreinstellungen.
Zunächst kann über den Parameter „count“ (Default: 1) eingestellt werden, wie viele Geraden im Bild gefunden und ihrer Verzeichnung nach ausgewertet werden sollen. Dies ist insbesondere bei künstlichen Testbildern nützlich, da hier nur eine begrenzte Anzahl an Geraden vorhanden ist und demnach theoretisch auch nur diese Anzahl an Geraden ausgewertet werden kann. Der nachfolgende einstellbare Wert epsilon ist derjenige, der das Abbruchkriterium definiert.
Die beiden letzten Parameter deltaX und deltaY definieren den Umkreis in x- und y-Richtung um das jeweilige Maximum einer Geraden, in dem die Pixelwerte in die Berechnung der jeweiligen Zielfunktion mit einfließen. Default-Werte hierfür sind jeweils 5 Pixel in beide Richtungen.

6.2.3 Die Methode optimize()

In dieser Methode, die über run() aufgerufen wird, wird der Optimierer anhand eines Nelder-Mead-Objektes erstellt. Dieses bekommt einige Werte zugewiesen wie beispielsweise die Anzahl der maximalen Optimierungsschritte maxIterations und über die Methode setConvergenceChecker() wird das individuell implementierte Abbruchkriterium angegeben. Im Konstruktor des Optimierers wird neben den Startwerten des Simplex zum einen die Zielfunktion mit „this“ angegeben, was bedeutet, dass sie in einer entsprechenden Methode des Plugins definiert wurde, zum anderen über GoalType.MINIMIZE/MAXIMIZE die Richtung der Optimierung ins Minimale

oder Maximale, je nach Zielfunktion.
Nachdem die Optimierung vollendet ist, können mit Hilfe der Methoden getPoint() und getValue() die optimierten Parameterwerte und ihr zugehöriger Funktionswert aus dem RealPointValuePair-Objekt (Objekt, das sowohl die Parameterwerte als auch ihren zugehörigen Funktionswert beinhaltet) ausgelesen werden.

```
NelderMead optimizer = new NelderMead();
optimizer.setMaxIterations(maxIterations);
optimizer.setConvergenceChecker(this);
optimizer.setStartConfiguration(steps);
RealPointValuePair res = null;
double[] start = {0, 0, 0, 0};
try {
 res = optimizer.optimize(this, GoalType.MINIMIZE, start);
 …
}
params = res.getPoint();
result = entropy(params);
```

6.2.4 Hough-Transformation

In jedem Schritt des Optimierungsalgorithmus wird zunächst das Originalbild mit den aktuell detektierten Verzeichnungsparametern wie folgt verzerrt

```
for(int x = 0; x < width; x++){
 for(int y = 0; y < height; y++){

   double r = (((x-xCtr)*(x-xCtr))+
               ((y-yCtr)*(y-yCtr)))/maxRadius;

   double fac = (((r4 * r + r3)* r + r2)* r + r1)* r + 1;

   int xDis = (int)((xCtr + (x-xCtr)) * fac);
   int yDis = (int)((yCtr + (y-yCtr)) * fac);

   if((xDis>0 && xDis<width) && (yDis>0 && yDis<height)){
      pixels[yDis * width + xDis] = ip.getPixel(x, y);
}}}

FloatProcessor fpDis = new FloatProcessor(width, height,
pixels);
ImagePlus impDis = new ImagePlus("verzerrtes Bild", fpDis);
```

und anschließend die Hough-Transformation von diesem berechnet. Wie diese Berechnung aufgerufen wird, zeigt das folgende Code-Beispiel.

```
LinearHT2 HT2Dis = new LinearHT2(fpDis, res, res);
FloatProcessor hipDis = HT2Dis.createFloatProcessor();
hipDis.flipHorizontal(); //because angle runs reverse
                            (negative y)
```

```
ImagePlus himDis = new ImagePlus("verzerrte HT of " +
imp.getTitle(), hipDis);

FloatProcessor hmaxDis = HT2Dis.localMax(hipDis);
HoughSet hsDis = HT2Dis.getMaxList(hmaxDis, count);
HT2Dis.printHoughSet(hsDis);
```

Im weiteren Verlauf wird die detaillierte Ermittlung der Hough-Transformation anhand des Programmcodes, welcher aus [p1] entnommen wurde, erläutert werden.
Das Hauptprogramm zur Durchführung der Hough-Transformation zum Auffinden von Geraden ist das Plugin PluginLinearHoughTransform_. Dieses setzt ein binäres Kantenbild voraus, wobei Pixelwerte größer Null zu einer Kante gehören.
In der zugehörigen run()-Methode wird die Klasse LinearHT2 aufgerufen, in welcher explizit die Berechnung der Hough-Transformation stattfindet. Nachdem die Hough-Transformation und auch die lokalen Maxima dieser berechnet wurden – was in einem nächsten Schritt erklärt werden wird – werden drei neue Datensätze generiert: Zum einen das Hough-Bild mit dem Namen „HT of ´Bildtitel´“ (hipDis), zum anderen das Maximumbild „Maxima of ´Bildtitel´“ (hmaxDis). In der dritten Datei ist eine Tabelle (hsDis) zu finden, die sowohl die errechneten Winkel und Radien als auch die entsprechenden Zähler der stärksten Geraden enthält.
Wie schon angedeutet, erfolgt die eigentliche Berechnung der Hough-Transformation in der Klasse namens LinearHT2. In dem Konstruktor dieser Klasse werden zunächst einige Variablen definiert und initialisiert. So möchte man beispielsweise die Koordinaten des Bildzentrums kennen, $xCtr$ und $yCtr$, die sich aus der halben Bildbreite bzw. halben Bildhöhe ergeben. Des Weiteren sind die Anzahl der Schritte auf der x-Achse, also des Winkels und die der Schritte auf der y-Achse, also die des Radius interessant, genau wie ihre entsprechende Schrittweite. Die Anzahl der jeweiligen Schritte entspricht in unserem Falle der Auflösung des Originalbildes, also beispielsweise 256. Die Schrittweite der Winkel-Achse ergibt sich entsprechend durch $\pi / 256$, die der Radius-Achse durch $(2 \cdot \max Radius) / 256$, also dem zweifachen

maximalen Radius (Diagonale) geteilt durch die horizontale Auflösung. Der maximale Radius entspricht bei einem in der Bildmitte gelegenem Ursprung der halben Bilddiagonalen: $\max Radius = \sqrt{xCtr^2 + yCtr^2}$.

Zu guter Letzt wird ein 2-dimensionales Hough-Array deklariert mit den Dimensionen des Originalbildes, das im nächsten Schritt durch die Methode fillAccumulator() mit Zählwerten gefüllt wird.

Dies geschieht dadurch, dass das Originalbild Pixel für Pixel durchlaufen wird und festgestellt wird, ob diese ungleich Null, also nicht schwarz sind. Ist das Pixel ein Kantenpixel, wird die Methode doPixel() an dieser Stelle aufgerufen [p1]:

```
void doPixel(int u, int v) {
   int x = u-xCtr,  y = v-yCtr;
   for (int a = 0;   a < nAng; a++) {
     double theta = dAng * a;
     int r = (int) Math.round((x*Math.cos(theta)+
                  y*Math.sin(theta))/dRad)+nRad/2;

     if (r >= 0 && r < nRad) {
       houghArray[a][r]++;
}}}
```

Diese Methode wiederum durchläuft in einer for-Schleife die gesamte x-Achse, also alle Winkel und berechnet den dazu passenden Radius. Dies bedeutet, dass alle Geraden bzw. Winkel-Radius-Paare berechnet werden, die gerade durch den aktuellen Pixelwert gehen. Im Akkumulator-Array wird an der so berechneten Stelle [Winkel][Radius] der Wert um eins erhöht. Dies bedeutet, dass ein anderes Pixel, das zu dem gleichen Winkel-Radius-Paar führt, ebenfalls auf dieser Gerade liegt. Es schneiden sich somit die beiden Geraden der beiden Pixel im Parameterraum in einem Punkt. Nun ist das Zähler-Array gefüllt und dort, wo die höchsten Einträge stehen, befinden sich die Parameter zu den am häufigsten im Bild vorkommenden Geraden. Diese werden in die erste ausgegebene Datei „HT of ...“ gezeichnet.

Das zweite Bild zeigt im Grunde das gleiche, allerdings reduziert auf die stärksten Geraden. Im Parameterraum bedeutet dies, dass nur noch die Schnittpunkte zu sehen sind, also die Punkte mit den höchsten Akkumulator-Einträgen.
In der Methode localMax() werden diese Punkte herausgefunden, indem die Einträge des Akkumulator-Arrays geeignet verglichen werden und die stärksten herausgefiltert werden. In der Methode getMaxList() schließlich wird über ein Hough-Set (Array bestehend aus Wertetripeln: Winkel, Radius und Zähler) die Tabelle mit den Parametern der gefundenen Geraden generiert.
Ein Beispiel für das Bild der Hough-Transformation sowie das der herausgefilterten Geradenmaxima ist in Abbildung 15 veranschaulicht.
Die darunter stehende Abbildung 16 zeigt die Auswirkung einer Verzerrung des Originalbildes. Man kann deutlich erkennen, dass die verzeichneten Geraden nicht mehr auf einen Punkt abgebildet werden sondern über mehrere Pixel verteilt werden.

Abb. 15: Originalbild "Test01.tif", Hough-Transformation "HT of Test01.tif" und dazugehöriges Maximumbild "Maxima of Test01.tif"

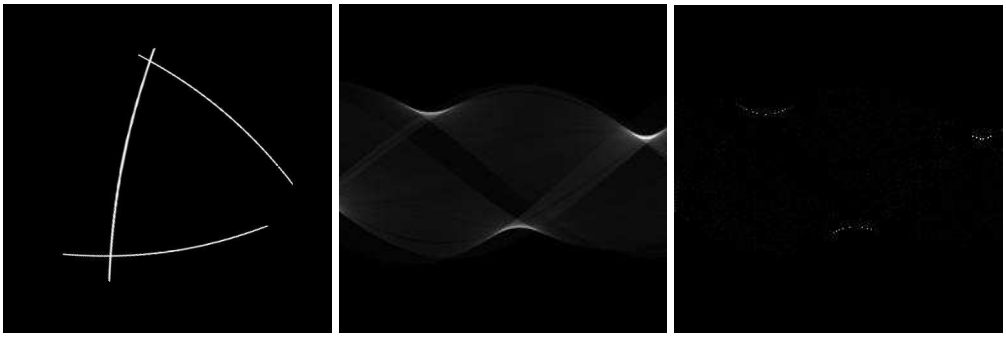


Abb. 16: Originalbild "Test01_09050501.tif", Hough-Transformation "HT of Test01_09050501.tif" und dazugehöriges Maximumbild "Maxima of Test01_09050501.tif"

6.2.5 Die Qualitätskriterien

Insgesamt werden drei unterschiedliche Funktionen zur Bestimmung der Qualität der Optimierung betrachtet.
Um im Konstruktor des Nelder-Mead-Objektes die Zielfunktion bezeichnen zu können, muss das Plugin eine Methode mit dem Namen value(double[] point) besitzen. Diese bekommt ein eindimensionales double-Array übergeben, in dem die jeweiligen Werte für die Verzerrungsparameter gespeichert sind. In jedem Iterationsschritt der Optimierung wird diese Funktion mit dem neuen Array aufgerufen und mit diesem die Zielfunktion gespeist.

Die ersten beiden vorgestellten Qualitätskriterien (Entropie und Varianz) werden in einer separaten Methode berechnet, die jeweils das aktuelle, verzeichnete Bild, in einem FloatProcessor gespeichert, sowie die Koordinaten der Geradenmaxima übergeben bekommt. Diese wurden im Vorhinein aus Winkel und Radius umgerechnet über die Methoden calculateX(double a) und calculateY(radius r). Ist die Pixelpositionen bekannt, kann um diese herum nun die Zielfunktion berechnet werden.

Diese Einzelergebnisse der jeweiligen Geradenmaxima werden anschließend in der übergeordneten Methode aufaddiert und können nun mit den jeweils vorangegangenen Ergebnissen verglichen werden.
Die Auswertung der Pixelwerte der Geradenmaxima (countMaxima) erfolgt direkt, ohne untergeordnete Berechnungsmethode.

6.2.5.1 Die Methode entropy()

Die Berechnung der Entropie erfolgt für jede Gerade einzeln und in einer separaten Methode calculateEntropy(FloatProcessor fp, int x, int y). Aufgrund der Tatsache, dass die Berechnung über mehrere Pixel in sowohl x- als auch y-Richtung erfolgt, muss man im Besonderen die Fälle betrachten, in denen das Geradenmaximum sehr nahe am Rand liegt. Liegt das Maximum in y-Richtung nahe am Rand, wird diese Gerade aus der Berechnung ausgenommen, da dieser Fall ohnehin nicht häufig auftreten wird. Geraden mit einem solch großen Radius liegen am äußersten Bildrand, haben somit weniger Geradenpunkte und werden nicht als stärkste Geraden über die Hough-Transformation detektiert werden. Da auf der x-Achse jedoch die Winkel aufgetragen sind, werden die x-Werte über den Rand hinaus berechnet, was die zweite Zeile des Programmcodes zeigt.

```
for(int i = -dx2; i <= dx2; i++){

  int x = ((x0 + i + res) % res);
  int y;

  for(int j = -dy2; j <= dy2; j++){
    if(x == (x0+i))
      y = y0+j;
    else
      y = (res - y0)+j;

    pix = Float.intBitsToFloat(fp.getPixel(x, y));

    if(pix != 0.0){
      sum += pix;
      sumH += pix * Math.log(pix);
}}}

 h = (-1) * (sumH/sum - Math.log(sum));
```

Zum Schluss werden alle Einzelentropien in der übergeordneten Methode entropy() aufsummiert. Diese Summe bildet nun das Ergebnis der Zielfunktion der aktuellen Verzeichnungs-Parameterwerten.
Dieses Ergebnis gilt es nun zu optimieren bzw. zu minimieren.

```
double[] maxA = HT2Dis.getMaxA();
double[] maxR = HT2Dis.getMaxR();

for(int i = 0; i < count; i++){

  /** convert from angle/radius to x/y (pixel) */
  int x = calculateX(maxA[i]);
  int y = calculateY(maxR[i]);

  /** if y-coordinate is near border of ip, ignore it */
  if((y < (dy2)) || (y > (res - (dy2)))){
    System.out.println("y out of border");
  }

  else{
    h = calculateEntropy(hipDis, x, y);
    totalH += h;
 }}

 return totalH;
```

6.2.5.2 Die Methode variance()

Der zweite Ansatz in calculateVariance(FloatProcessor fp, int x, int y) besteht darin, die Zielfunktion bzw. die Qualität der gefundenen Verzerrungsparameter nicht über die Entropie zu berechnen sondern über die Varianz, bzw. die Abweichung der einzelnen Pixelwerte zu ihrem Mittelwert.

Die Berechnung erfolgt über folgende Formel [10] $\mathrm{var}(x) = \frac{1}{n}\sum_{i=1}^{n}\left(x_i - \bar{x}\right)^2$, die durch geeignete Umformungen in folgende umgeschrieben werden kann :

$$\mathrm{var}(x) = \frac{1}{n}\sum_{i=1}^{n} x_i^{\,2} - \bar{x}^2$$

("Mittelwert der Quadrate minus Quadrat des Mittelwerts" [10])
Diese Funktion muss maximiert werden, um ein Maß für die Güte der Konzentration auf einen Punkt darzustellen. Denn wenn diese Konzentration sehr hoch ist und die umliegenden Pixel verhältnismäßig geringe Werte aufweisen, wird sowohl der Mittelwert als auch die Abweichung der einzelnen Pixelwerte zu diesem, die Varianz im hiesigen Fall, sehr hoch werden.

```
for(int j = -dy2; j <= dy2; j++){

  pix = Float.intBitsToFloat(fp.getPixel(x0, y0+j));
  sum += pix;
  pixSum2 += (pix*pix);

}

var = (pixSum2/n) - ((sum/n) * (sum/n));

return var;
```

Nachdem alle Ergebnisse der einzelnen Maxima berechnet wurden, werden diese zu der Gesamtvarianz aufsummiert, die nun den zu maximierenden Funktionswert bildet.

```
double[] maxA = HT2Dis.getMaxA();
double[] maxR = HT2Dis.getMaxR();

for(int i = 0; i < count; i++){

  /** convert from angle/radius to x/y (pixel) */
  int x = calculateX(maxA[i]);
  int y = calculateY(maxR[i]);
  totalVar += calculateVariance(hipDis, x, y);

}
return totalVar;
```

6.2.5.3 Die Methode countMaxima()

Der dritte Ansatz betrachtet als Qualitätskriterium den Zähler der jeweiligen Maxima im Akkumulatorarray der Hough-Transformation. Dazu werden in jedem Schritt die Koordinaten der neuen Maxima des neuen verzerrten Zwischen-Bildes bestimmt und deren Pixelwert, der ja der Anzahl an gefundenen Geradenpunkte entspricht, als Zielfunktionswert an den Optimierer zurückgegeben. Da ein höherer Akkumulatorzähler auf eine stärkere Konzentration auf einen Punkt und somit auf eine gerade Linie im Bild schließen lässt, muss bei diesem Ansatz der Optimierungsalgorithmus eine maximierende Funktion einnehmen.
Falls mehrere Geraden betrachtet werden (einstellbar über Parameter count), werden alle Zähler der einzelnen Geradenmaxima aufaddiert und der Gesamtzähler maximiert.

```
double[] maxA = HT2Dis.getMaxA();
double[] maxR = HT2Dis.getMaxR();

for(int i = 0; i < count; i++){

  /** convert from angle/radius to x/y (pixel) */
  int x = calculateX(maxA[i]);
  int y = calculateY(maxR[i]);

  c += Float.intBitsToFloat(hipDis.getPixel(x, y));
}

return c;
```

6.2.6 Abbruchkriterium

Über die Methode converged(int iteration, RealPointValuePair previous, RealPointValuePair current) wird das Abbruchkriterium definiert. Dieses greift in Zusammenhang mit der vorgegebenen Anzahl an Iterationsschritten und ist wie folgt definiert: wenn die Differenz des Funktionswertes des letzten und des aktuellen Parametersatzes kleiner ist als ein vordefiniertes epsilon, ist das Ergebnis gut genug und die Optimierung kann abgebrochen werden. Andernfalls wird solange weiter gesucht, bis das Kriterium erfüllt ist.

```
return current.getValue()-previous.getValue() < epsilon;
```

6.2.7 Entzerrung

Nach erfolgreicher Optimierung der Verzeichnungsparameter kann das Originalbild letztendlich noch einmal separat wie unter 6.2.4 beschrieben entzerrt und auch angezeigt werden. Dies passiert in der zu Anfang erwähnten run()-Methode. Zu Kontrollzwecken und zur Veranschaulichung, dass sich die Punkte im Hough-Raum nach der Optimierung wirklich um bestimmte, im besten Falle um die anfänglichen Maxima herum zusammengezogen haben, wird auch das Hough-Bild des entzerrten Bildes ausgegeben. Des Weiteren werden (für die Auswertung) interessante Daten wie die anfangs eingestellten Werte für die Plugin-Parameter, die Polar-Koordinaten der einzelnen Geraden sowie selbstverständlich die besten Ergebnisse für die Verzerrungsparameter in einem ImageJ-Textdokument ausgegeben:

Haus_09050501.tif
count: 3 deltaX: 5 deltaY: 5 epsilon: 1.0

angle: 1.607611865704152 radius: -79.19595949289332 count: 340
angle: 1.509437095279471 radius: -79.19595949289332 count: 334
angle: 1.5585244804918115 radius: -79.19595949289332 count: 313
...
angle: 1.5585244804918115 radius: -82.02438661763952 count: 664
angle: 3.129320807286708 radius: 147.0782104868019 count: 473
angle: 2.3439226438892597 radius: -178.19090885901 count: 441

r1: 0.7138525112842098 r2: 0.9899655546662138 r3: 0.6881204255346243
r4: 0.433172768255328
variance: 1578.0

radiusOriginal: -79.19595949289332 angleOriginal: 1.607611865704152
radiusBest: -82.02438661763952 angleBest: 1.5585244804918115

Die oben genannten Schritte laufen automatisch hintereinander ab, beziehen allerdings nicht das Originalbild mit ein. Lediglich das 8-Bit-Kantenbild wird in einem letzten Schritt entzerrt zur schnellen und direkten visuellen Kontrolle. Ist man mit dem Ergebnis der Optimierung auf den ersten Blick zufrieden, kann über ein externes ImageJ-Plugin (siehe nachfolgendes Kapitel) mit den erhaltenen Parameterwerten die finale Entzerrung des Ausgangsbildes erfolgen.

7 Ergebnisse

Im Folgenden werden die Ergebnisse sowohl der künstlich erzeugten Testbilder als auch der Objekt- und Gebäudeaufnahmen dargestellt. Hierzu werden alle drei Ansätze zur Detektion der Kameraverzerrung auf jedes einzelne Testbild angewandt und miteinander verglichen.

Die Auswertung der Ergebnisse erfolgt auf rein visueller Ebene. Eine numerische Auswertung wäre nur im Falle der künstlich erzeugten Testbilder möglich, da hier die bekannten Verzerrungsparameter (Grundwahrheit) mit den über den Optimierungsalgorithmus ermittelten Parametern verglichen werden können. Allerdings stellt man schnell fest, dass hier relativ große Abweichungen unabhängig des verwendeten Ansatzes und der jeweiligen Einstellungen entstehen. Da insgesamt vier Verzeichnungsparameter betrachtet werden, gibt es verschiedene Kombinationsmöglichkeiten dieser innerhalb der Verzeichnungsfunktion. Aufgrund dessen wird eine diese Werte vergleichende Auswertung außer Acht gelassen.

Vielmehr liegt die Konzentration auf der visuellen Bewertung der entzerrten Bilder als entscheidendes Kriterium für die Qualifizierung der einzelnen Methoden.

Die Auswertungen finden sich in den untenstehenden Tabellen wieder, wobei sich ändernde oder interessante Werte fett hervorgehoben werden. Generell sind diese Tabellen nach der verwendeten Zielfunktion aufgebaut, in den ersten vier Zeilen werden die Voreinstellungen aufgeführt, danach werden im Falle der künstlichen Testbilder die bekannten, originalen sowie die besten, detektierten Verzeichnungsparameter angegeben. Die beiden letzten Zeilen dienen dem Funktionswert des zu minimierenden oder maximierenden Qualitätskriteriums.

Unter den Tabellen werden zusätzlich ihre entsprechenden Bilder angezeigt, links das verzerrte Original, rechts die mit den oben stehenden Werten entzerrten Ergebnisse.

Je nach gewünschter Visualisierung werden in das entzerrte Bild zusätzlich zwei (oder mehrere) Geraden eingezeichnet, um einen direkten visuellen Vergleich zu erhalten. Die erste - im Folgenden rot gekennzeichnet - ist eine Gerade, die aus dem originalen, verzerrten Testbild stammt, die zweite - im Folgenden grün markiert - die entsprechende entzerrte Gerade.
Zurückgerechnet werden diese Geraden über die Polarkoordinaten der Geradenmaxima im Hough-Raum.

7.1 Künstlich erzeugte Testbilder

Die vier ausgewählten künstlich erzeugten Testbilder sind wie bereits in Abschnitt 4 erwähnt die folgenden: Gerade01.tif, Test01.tif und Haus.tif.
Auf diese wurden jeweils vier Sätze à vier Verzerrungsparametern angewendet, die eine unterschiedlich starke Verzerrung des Bildes bzw. der in diesem vorkommenden Geraden mit sich bringen.

7.1.1 Änderung der Einstellungen

Zunächst wird der Parameter „steps“ betrachtet, der nicht über eine Benutzeroberfläche eingestellt werden kann, da er im Plugin selbst festgelegt wurde. Er beschreibt ein Array, in welchem die Anfangswerte der Verzeichnungsparameter für den Optimierungsalgorithmus übergeben werden. Aus diesen Startwerten baut der Optimierer sein weiteres Suchverfahren auf. Aufgrund von diversen Auswertungen und Testdurchläufen hat sich gezeigt, dass ein Wert von 0.5 für jeden Startparameter zu guten Ergebnissen führt. Wählt man die Werte zu niedrig (bspw. 0.1), kann es insbesondere bei starken Verzeichnungen dazu kommen, dass das Suchverfahren keine dementsprechend hohen Ergebnisparameter liefert, wie man in Tabelle 1 erkennen kann:

target function	countMax	countMax
steps	**0,1**	**0,5**
dx, dy	5	5
count	1	1
epsilon	1	1
original r1	1,9	1,9
original r2	0,9	0,9
original r3	0,9	0,9
original r4	0,9	0,9
best r1	**0,0341**	**1,41328**
best r2	**0,09685**	**1,28357**
best r3	**0,15887**	**1,11865**
best r4	**0,07063**	**0,63269**
original result	359	359
best result	373	586

a) b) c)

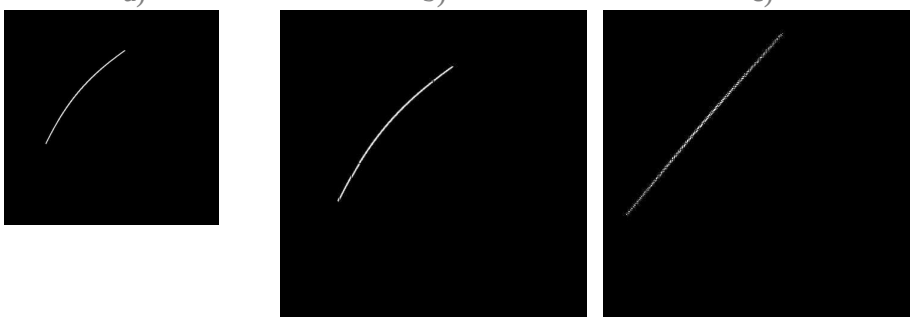


Tabelle 1: Gerade01_19090909.tif : Änderung des Parameters steps:
a) Originalbild; b) entzerrt mit steps = 0.1; c) entzerrt mit steps = 0.5

Alle drei Ansätze liefern beste Ergebnisse, wenn der Umkreis, über den die jeweilige Zielfunktion berechnet wird nicht zu groß wird. Das bedeutet je einen Wert von 5 Pixeln in x- und y-Richtung (deltaX und deltaY).

Der für das Abbruchkriterium wichtige Parameter epsilon liegt für die Entropie- und Varianzberechnung bei 0.01, für die Auswertung des Zählers bei 1.

Ein weiterer Parameter, der eingestellt werden kann, ist count. Dieser gibt an, wie viele Geraden im Bild gesucht werden und mit in die Optimierung einfließen sollen. Selbstverständlich ist es sinnvoll, bei mehreren Geraden im Bild auch diverse zu

betrachten. Allerdings werden auch dann gute Ergebnisse erzielt, wenn standardmäßig nur eine Gerade angegeben ist. Betrachtet man mehr Geraden als vorhanden, kann dies zu einem schlechten Ergebnis führen.

Einzig bei der Berechnung der Entropie scheint es eine Rolle zu spielen, wie viele Geraden betrachtet werden. Dies verläuft allerdings nicht nach einem bestimmten Muster, es werden von Testbild zu Testbild unterschiedliche Resultate erzielt. Ein Beispiel ist in Tabelle 2 aufgeführt.

target function	entropy	entropy	entropy
steps	0,5	0,5	0,5
dx, dy	5	5	5
count	**1**	**3**	**6**
epsilon	0,01	0,01	0,01
original r1	0,9	0,9	0,9
original r2	0,5	0,5	0,5
original r3	0,5	0,5	0,5
original r4	0,1	0,1	0,1
best r1	0,5000	0,4377	0,6684
best r2	0,5000	0,8777	0,1502
best r3	0,5000	0,8364	-0,6640
best r4	0,5000	0,4469	-0,9728
original result	4,42	13,23	26,07
best result	4,09	12,48	24,95

a) b) c) d)

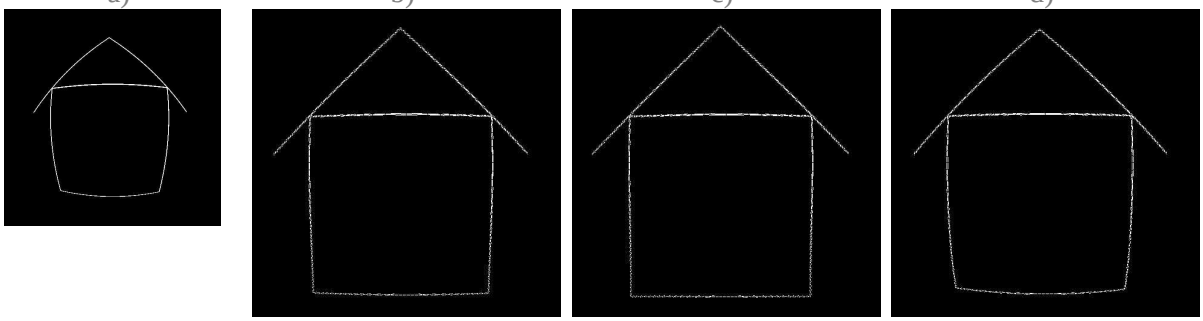


Tabelle 2: Haus_19090909.tif : Änderung des Parameters count:
a) Originalbild; b) entzerrt mit count = 1; c) entzerrt mit count = 3;
d) entzerrt mit count = 6

Hier erkennt man, dass ein count von drei sowohl rechnerisch als auch visuell das beste Ergebnis erzielt.

7.1.2 Änderung der Zielfunktion

Sowohl an Tabelle 3 als auch an Tabelle 4 wird deutlich, dass die Wahl der Entropie als Zielfunktion deutlich schlechtere Resultate erzielt als die anderen beiden.

target function	**entropy**	**variance**	**countMax**
steps	0,5	0,5	0,5
dx, dy	5	5	5
count	**1**	**1**	**1**
epsilon	0,01	0,01	1
original r1	0,9	0,9	0,9
original r2	0,5	0,5	0,5
original r3	0,5	0,5	0,5
original r4	0,1	0,1	0,1
best r1	0,5000	0,7500	0,6885
best r2	0,5000	0,3750	1,1390
best r3	0,5000	0,7500	0,8245
best r4	0,5000	0,3750	0,3077
original result	4,42	8094,15	340
best result	4,09	28097,82	683

a) b) c) d)

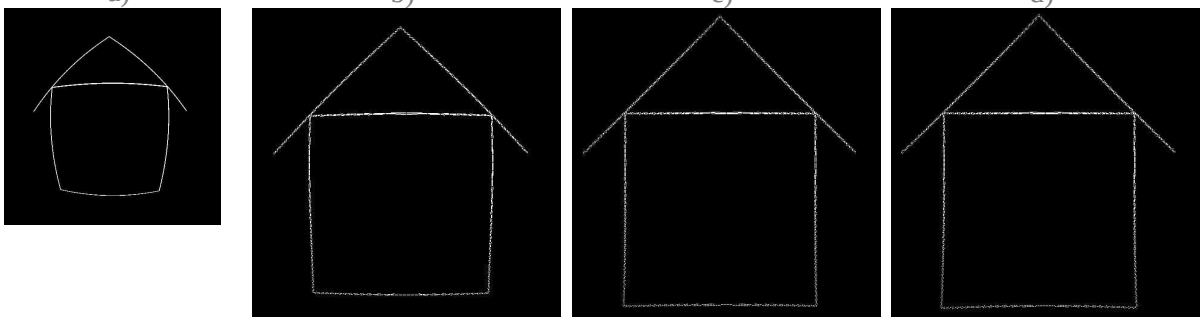


Tabelle 3: Haus_09050501.tif: Vergleich zwischen den Zielfunktionen mit count = 1:
a) Originalbild; b) entzerrt über entropy; c) entzerrt über variance;
d) entzerrt über countMax

target function	**entropy**	**variance**	**countMax**
steps	0,5	0,5	0,5
dx, dy	5	5	5
count	**6**	**6**	**6**
epsilon	0,01	0,01	1
original r1	0,9	0,9	0,9
original r2	0,5	0,5	0,5
original r3	0,5	0,5	0,5
original r4	0,1	0,1	0,1
best r1	0,6684	0,7419	0,6947
best r2	0,1502	0,8363	0,7622
best r3	-0,6640	0,6595	0,7119
best r4	-0,9728	0,3091	0,3604
original result	26,07	44533,31	1850
best result	24,95	106069,72	2659

a) b) c) d)

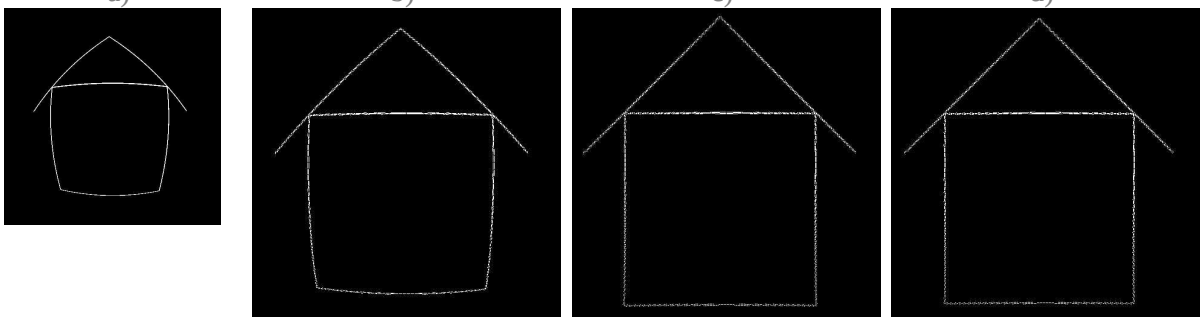


Tabelle 4: Haus_09050501.tif: Vergleich zwischen den Zielfunktionen mit count = 6:
a) Originalbild; b) entzerrt über entropy; c) entzerrt über variance;
d) entzerrt über countMax

An den folgenden Auswertungen (Abbildung 17 bis 19) kann man die Schwäche der Entropie-Zielfunktion gegenüber der Varianz- und Zählerberechnung klar erkennen. Hier wird zunächst die Hough-Transformation des verzeichneten Originalbildes dargestellt, daneben die des best entzerrten. Die letzte Darstellung zeigt das entsprechend entzerrte Bild inklusive mehrerer Linien. Die roten Geraden werden aus den Maxima der originalen Hough-Transformation berechnet, die grünen aus den Maxima der Hough-Transformation des best entzerrten Bildes. Es wurden jeweils sechs Geraden betrachtet.

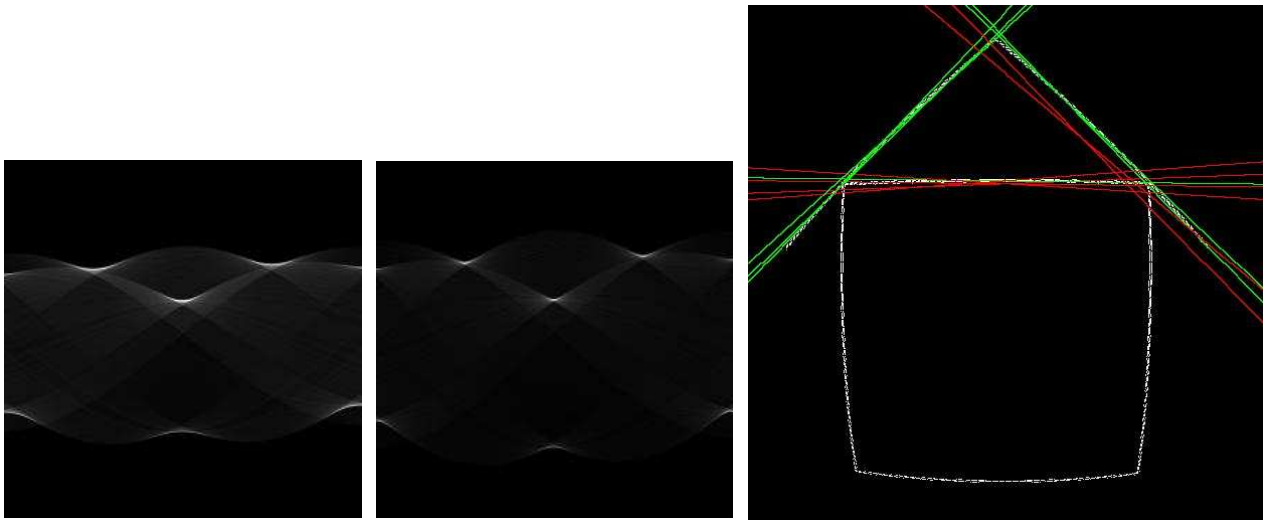


Abb. 17: Entropie-Methode:
HT des Originals (Haus_09050501_HT.tif), HT des besten Ergebnisses (Haus_09050501_best HT.tif) und entzerrtes Original mit eingezeichneten Geraden; die roten Geraden ergeben sich aus den Maxima der originalen HT, die grünen aus den Maxima der besten HT

Abbildung 17 veranschaulicht die Ergebnisse der Optimierung mittels der Entropieberechnung. Schon an der Hough-Transformation kann man erkennen, dass die Geraden nicht vollständig entzerrt wurden, da immer noch eine leichte Streuung um die eigentlichen Geradenmaxima vorhanden ist. Dies spiegelt sich im entzerrten Bild wider: mehrere grüne Geraden repräsentieren eine Bildgerade, was bedeutet, dass die höchsten Maxima im Hough-Raum nicht jeweils eine Gerade beschreiben sondern aufgrund ihrer Streuung diverse leicht gegeneinander versetzte.

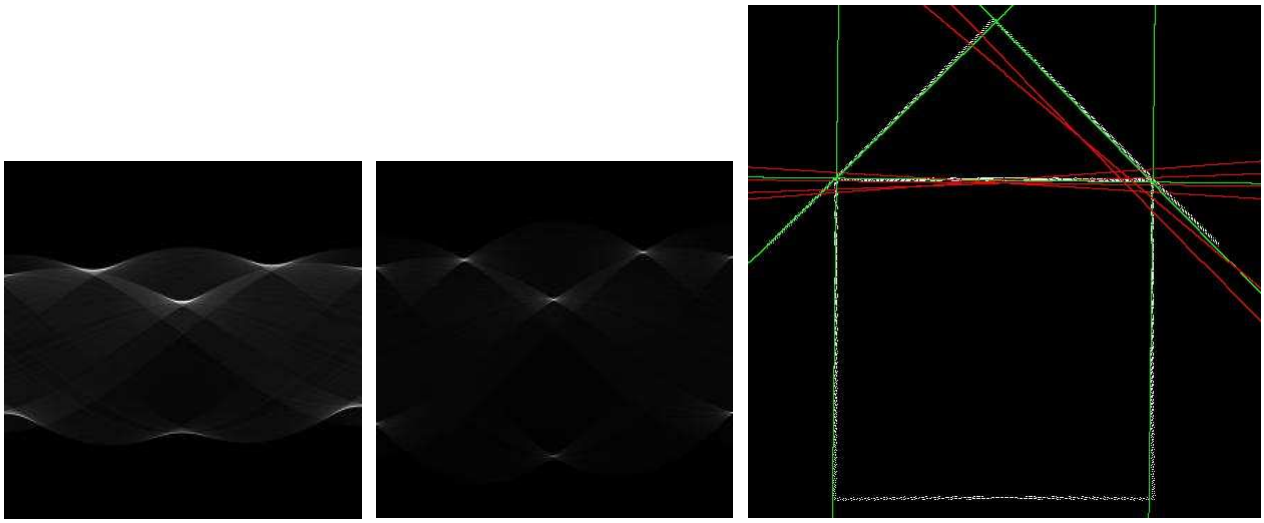


Abb. 18: Varianz-Methode:
HT des Originals (rote Geraden), HT des besten Ergebnisses (grüne Geraden)
und entzerrtes Original mit eingezeichneten Geraden

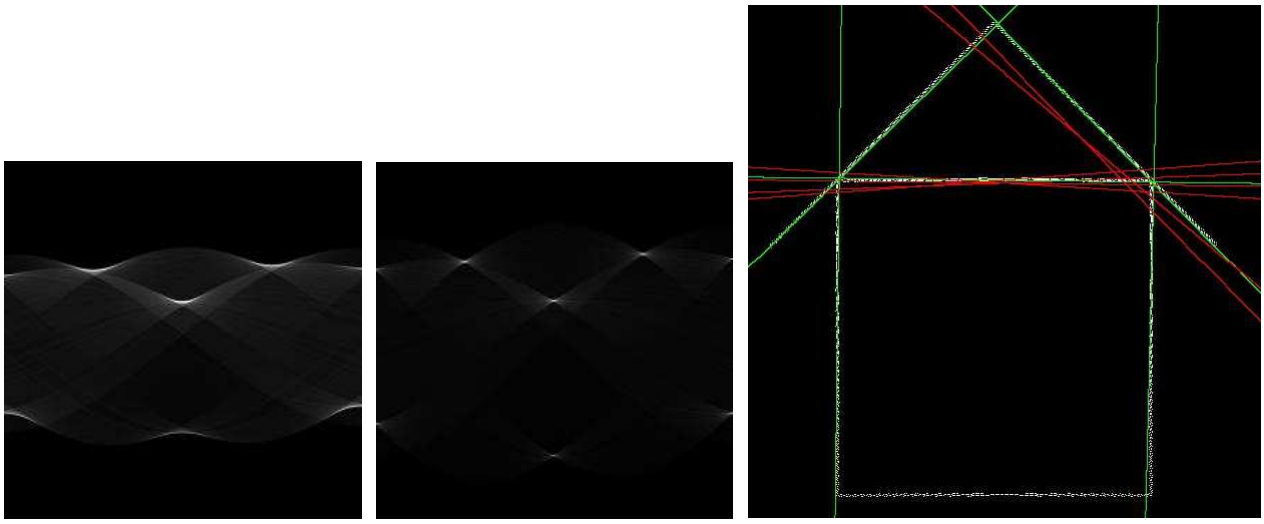


Abb. 19: CountMax-Methode:
HT des Originals (rote Geraden), HT des besten Ergebnisses (grüne Geraden)
und entzerrtes Original mit eingezeichneten Geraden

Im Gegensatz dazu finden sich nach der Optimierung und Entzerrung über sowohl die Varianz- als auch die CountMax-Zielfunktion (Abbildung 18 und 19) im neuen Hough-Raum Maxima wieder, die auf einen Punkt konzentriert sind und somit jeweils nur eine (grüne) Gerade darstellen.

7.2 Reale Testaufnahmen

Aufgrund der Tatsache, dass es im Falle der realen Testbilder keine bekannten Verzerrungsparameter und somit keinerlei zahlenmäßige Referenz gibt, ist hier die visuelle Beurteilung der Ergebnisse von noch größerer Wichtigkeit.
Auch hier wurden die drei unterschiedlichen Methoden angewandt und verglichen, allerdings erst nach entsprechender Vorverarbeitung der Originalbilder (siehe Kapitel 4.5). Die Originalfotos wurden abschließend ebenfalls entzerrt. Die dort zu erkennenden Artefakte (schwarze Linien) sind die schon zuvor in Kapitel 4.1 beschrieben Auswirkungen der Entzerrung über das Source-to-Target-Verfahren.

target function	**entropy**	**variance**	**count**
steps	0,5	0,5	0,5
dx, dy	5	5	5
count	**3**	**3**	**3**
epsilon	1	0,01	1
best r1	-0,1563	0,0876	0,0393
best r2	-0,8125	-0,0162	-0,0022
best r3	-0,8438	-0,0316	-0,0267
best r4	-0,7500	0,0096	0,0842
original result	13,70	107444,71	1729
best result	13,15	143144,35	1986

a) b) c) d)

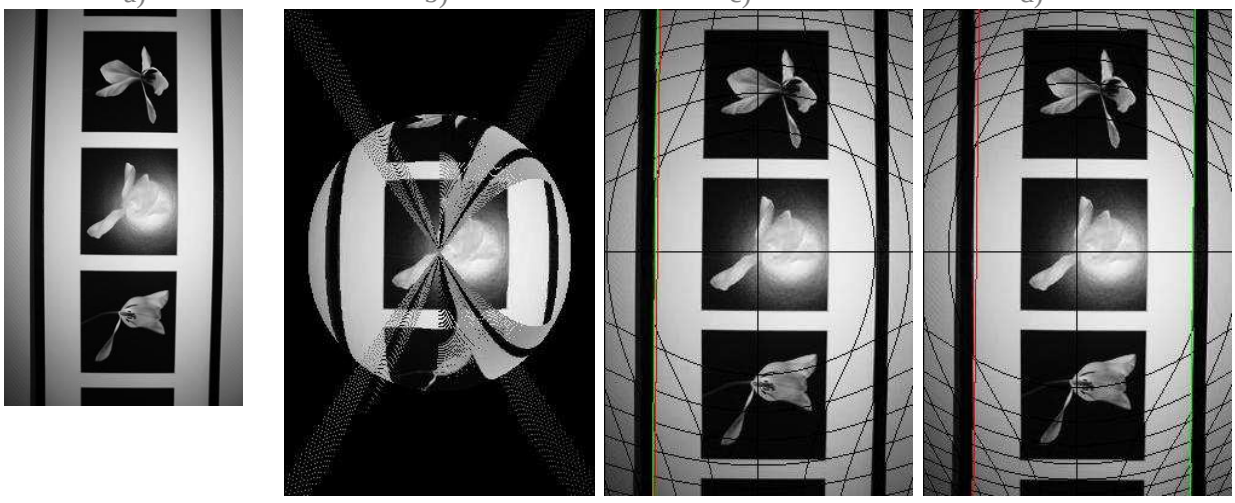


Tabelle 5: Rahmen02.tif: Vergleich zwischen den Zielfunktionen mit count = 3:
a) Originalbild; b) entzerrt über variance; c) entzerrt über countMax

In Tabelle 5 wird die Auswertung der Rahmen-Testaufnahme dargestellt. Zunächst kann man hier deutlich feststellen, dass die Verwendung der Entropie als Messziffer für die Güte der Entzerrung im Gegensatz zu den beiden anderen Ansätzen zu einem sehr schlechten Ergebnis führt. Bei realen Testaufnahmen scheint sie noch ungünstiger als bei den zuvor betrachteten künstlich erzeugten Bildern zu sein. Eine mögliche Erklärung für das Fehlschlagen der Entropie als Maß für die Konzentration in einem Punkt ist, dass die Testbilder teilweise derart verzerrt werden, dass die Zielfunktion zwar minimal wird, dies allerdings nur deshalb geschieht, da die in die Berechnung einfließenden Werte immer geringer werden. Das bedeutet, dass die Geraden so stark verzeichnet werden, dass immer weniger Geradenpunkte im Hough-Raum auf ein Maximum abgebildet werden. Dies kann man nachvollziehen, wenn man sich die Entwicklung der Geradenmaxima und ihre jeweiligen Zähler während des Optimierungsvorganges anschaut:

Rahmen02.tif

count: 3 deltaX: 5 deltaY: 5 epsilon: 0.01

angle: 3.129320807286708 radius: -84.73045497340374 count: 602
angle: 3.129320807286708 radius: -95.54710879979571 count: 594
angle: 3.129320807286708 radius: 95.54710879979571 count: 535
...
angle: 3.129320807286708 radius: -46.872166581031856 count: 232
angle: 0.0 radius: 45.069390943299865 count: 208
angle: 3.104777114680538 radius: -46.872166581031856 count: 194

r1: -0.15625 r2: -0.8125 r3: -0.84375 r4: -0.75
variance: 13.149656072077155

radiusOriginal: -84.73045497340374 angleOriginal: 3.129320807286708
radiusBest: -46.872166581031856 angleBest:
3.129320807286708

Eigentlich müsste der Zähler (count) größer werden, was bedeutet, dass viele Geradenpunkte einer (entzerrten) Geraden gefunden wurden. Bei der Varianzberechnung sowie der reinen Zählerauswertung ist dies auch der Fall. Nur im Falle der Entropie werden die auszuwertenden Pixelwerte immer kleiner und somit auch die Zielfunktion, die Geraden jedoch nicht entzerrt.

Dieses Problem tritt unabhängig von den Voreinstellungen auf. In Tabelle 6 und 7, die die Auswertung der Kommode darstellen, scheint das Ergebnis nicht ganz so schlecht zu sein, allerdings sind auch hier die über die Entropie gefundenen Verzerrungsparameter weitaus größer als die der anderen Methoden.
Eine Alternative in der Implementierung des Ansatzes wäre, die Entropie nicht um die immer neu entstehenden Geradenmaxima zu berechnen, sondern stets um die Maxima des Ausgangsbildes. Doch auch hier verbessert sich das Ergebnis nicht.

target function	**entropy**	**variance**	**count**
steps	0,5	0,5	0,5
dx, dy	5	5	5
count	**1**	**1**	**1**
epsilon	0,01	0,01	1
best r1	0,8750	0,1935	0,0923
best r2	1,6875	-0,0626	-0,0625
best r3	1,1250	0,0062	0,0912
best r4	0,5625	-0,0940	0,0035
original result	4,49	108793,70	1007
best result	4,41	126597,64	1167

a) b) c) d)

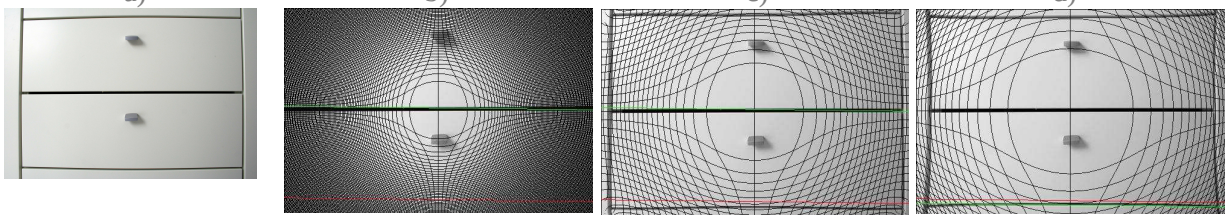


Tabelle 6: IMG_0077.tif: Vergleich zwischen den Zielfunktionen mit count = 1:
a) Originalbild; b) entzerrt über entropy; c) entzerrt über variance; d) entzerrt über countMax

target function	**entropy**	**variance**	**count**
steps	0,5	0,5	0,5
dx, dy	5	5	5
count	**8**	**8**	**8**
epsilon	0,01	0,01	1
best r1	0,3822	0,0948	0,0974
best r2	-0,0043	-0,0157	-0,0128
best r3	0,2205	-0,0415	-0,0083
best r4	0,1348	0,0317	-0,0341
original result	35,73	614256,83	6284
best result	32,63	740104,96	6719

a) b) c) d)

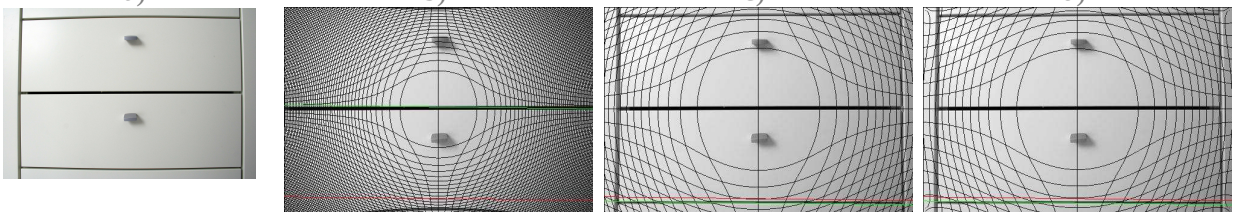


Tabelle 7: IMG_0077.tif: Vergleich zwischen den Zielfunktionen mit count = 8:
a) Originalbild; b) entzerrt über entropy;c) entzerrt über variance; d) entzerrt über countMax

Aufgrund dessen wird in den folgenden Auswertungen der interessanten Architekturaufnahmen auf den Entropie-Ansatz verzichtet. Der Vollständigkeit halber sind diese Auswertungen jedoch, ebenso wie alle hier präsentierten, auf der beiliegenden CD unter „Auswertung“ (geordnet nach Bilddateien) zu finden. Insbesondere die Original- und entzerrten Testbilder und ihre entsprechenden Hough-Transformationen können dort zum detaillierten Vergleich noch mal eingesehen werden.

Einige repräsentative Architekturbilder sollen nun als letzte Ergebnisse vorgestellt werden. Hierfür wurden Aufnahmen unterschiedlicher Brennweite (8mm und 16mm) und Kameratypen (Canon EOS-1D-Mark und Canon EOS 40D) ausgesucht. In den Tabellen 8 bis 11 werden ein letztes Mal die beiden Ansätze variance und countMax verglichen, die Abbildung unter jeder Tabelle (20 bis 23) zeigt das jeweils entzerrte Bild, welches dem besten visuellen Ergebnis des darüber stehenden Vergleichs entspricht und zur besseren Beurteilung vergrößert dargestellt ist.

target function	**variance**	**countMax**
steps	0,5	0,5
dx, dy	5	5
count	**3**	**3**
epsilon	0,0001	1
best r1	0,04501	0,01160
best r2	-0,12349	0,01391
best r3	0,01077	0,01356
best r4	0,16623	0,06566
original count	101536,17	2241
best count	137989,37	2512

a) b) c)

Tabelle 8: YV6P0015.jpg: Vergleich zwischen den Zielfunktionen mit count = 3:
a) Originalbild; b) entzerrt über countMax; c) entzerrt über variance

Abb. 20: YV6P0015_entzerrt.jpg (Sigma 8-16mm; Canon EOS-1D-Mark; f8; 1/160; 8mm)
entzerrt über countMax mit count = 3

target function	**variance**	**countMax**
steps	0,5	0,5
dx, dy	5	5
count	**3**	**3**
epsilon	0,0001	1
best r1	-0,07163	-0,03700
best r2	0,08502	0,04539
best r3	0,02688	-0,05111
best r4	-0,00464	0,06099
original count	397958,91	3372
best count	438998,74	3676

a) b) c)

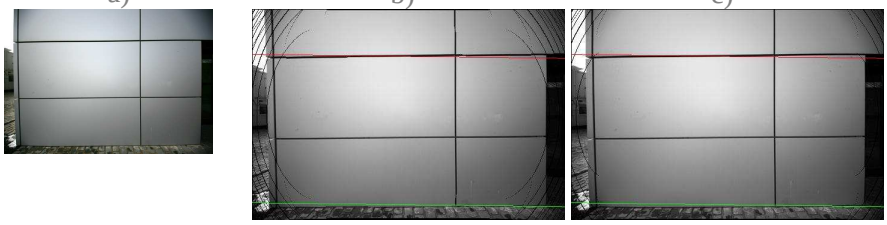


Tabelle 9: YV6P0023.jpg: Vergleich zwischen den Zielfunktionen mit count = 3:
a) Originalbild; b) entzerrt über countMax; c) entzerrt über variance

Abb. 21: YV6P0023_entzerrt.jpg (Sigma 8-16mm; Canon EOS-1D-Mark; f8; 1/250; 16mm)
entzerrt über countMax mit count = 3

target function	**variance**	**countMax**
steps	0,5	0,5
dx, dy	5	5
count	**3**	**3**
epsilon	0,0001	1
best r1	0,01230	0,00029
best r2	-0,00167	0,03289
best r3	-0,01295	0,00628
best r4	0,06576	-0,01782
original count	314942,99	3645
best count	341066,88	3747

a) b) c)

Tabelle 10: _MG_1305.jpg: Vergleich zwischen den Zielfunktionen mit count = 3:
a) Originalbild; b) entzerrt über countMax; c) entzerrt über variance

Abb. 22: _MG_1305_entzerrt.jpg (Sigma 8-16mm; Canon EOS 40D; f5.6; 1/320; 8mm)
entzerrt über variance mit count = 3

target function	**variance**	**countMax**
steps	0,5	0,5
dx, dy	5	5
count	**3**	**3**
epsilon	0,0001	1
best r1	-0,00402	-0,10746
best r2	0,00175	0,08193
best r3	0,00313	0,04369
best r4	0,02194	0,07038
original count	319656,30	3368
best count	330290,41	3698

a) b) c)

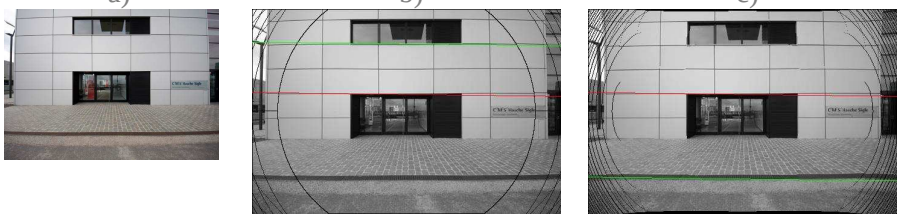


Tabelle 11: _MG_1314.jpg: Vergleich zwischen den Zielfunktionen mit count = 3: a) Originalbild; b) entzerrt über countMax; c) entzerrt über variance

Abb. 23: _MG_1314_entzerrt.jpg (Sigma 8-16mm; Canon EOS 40D; f5.6; 1/200; 8mm) entzerrt über variance mit count = 6

Abschließend soll die eigentliche Grundlage dieser Arbeit, die Hough-Transformation, in Hinblick auf die realen Testaufnahmen noch einmal aufgegriffen werden. In Abbildung 24 ist links der Hough-Raum des Originalbildes zu erkennen, daneben befindet sich die Hough-Transformation des besten Ergebnisses nach erfolgreicher Optimierung. Wie schon bei den künstlich erzeugten Testbildern, kann man im Ergebnis-Hough-Raum eine deutlich stärkere Konzentration auf einzelne Punkte (Maxima) feststellen. Insbesondere die Maxima um 90° herum kann man gut ausmachen, da diese den zahlreichen horizontalen Linien im Bild entsprechen.

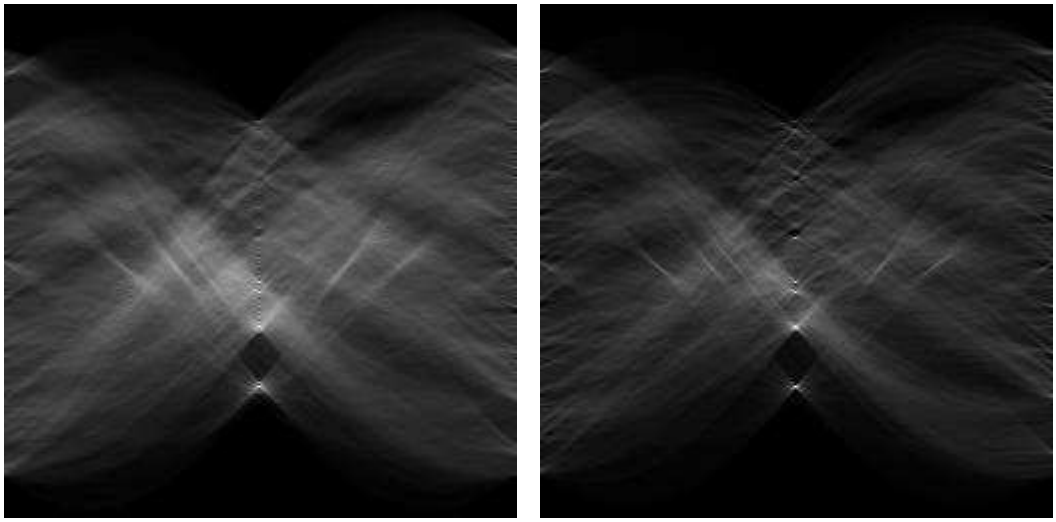


Abb. 24: Hough-Transformation des Originalbildes _MG_1305_HT und Hough-Transformation des entzerrten Bildes _MG_1305_HT_c3 (variance, count = 3)

8 Diskussion

Die Ergebnisse im vorangegangenen Kapitel lassen erkennen, dass das angestrebte Ziel der vorliegenden Arbeit erreicht wurde. Der grundlegende Gedanke, eine Entzerrung auf Basis der Hough-Transformation durchzuführen, wurde zu einem robust funktionierenden, weiterhin ausbaubaren Ansatz ausgearbeitet.

Neben der erfolgreichen Implementierung des ImageJ-Plugins und der drei unterschiedlichen Berechnungsfunktionen, konnten die angefertigten Testbilder- und Aufnahmen zufrieden stellend optimiert werden. Insbesondere die Auswertungen der künstlich erzeugten Testbilder führen zu sehr guten Ergebnissen. Am schwächsten kann sich hier die Optimierung mittels der Entropieberechnung behaupten, im Gegensatz dazu funktionieren die beiden anderen Methoden nach adäquaten Voreinstellungen robust und zuverlässig, unabhängig vom Inhalt der verwendeten Testbilder.

Schlägt ein Entzerrungsversuch fehl, kann das unter anderem die Ursache haben, dass der Optimierungsalgorithmus nicht das globale Minimum oder Maximum der Zielfunktion findet, sondern auf ein Nebenoptimum konvergiert. Des Weiteren kann es vorkommen, dass das definierte Abbruchkriterium schon vor einer sichtbaren Optimierung erreicht wird, wenn sich die Ergebnisse der einzelnen Iterationsschritte nur geringfügig ändern. Diese Schwachstellen des Optimierungsverfahrens sollten bekannt sein, treten allerdings in den seltensten Fällen ein.

Hinsichtlich der zu entzerrenden Objekt- und Gebäudeaufnahmen verliefen die Auswertungen über die Plugins ebenfalls zufrieden stellend, mit Ausnahme des Entropie-Ansatzes. Wie schon bei den künstlichen Testbildern, liefert dieser Ansatz die schlechtesten Resultate. Die Originalaufnahmen verschlechterten sich sogar teilweise oder wurden gar unbrauchbar.

Dies führt zu dem Ergebnis, dass sich die Entropie als Qualitätskriterium für die Konzentration auf einen Punkt im hiesigen Fall nicht eignet.

Die Varianzberechnung sowie die Auswertung des Zählers der Geradenpunkte eignen sich umso mehr und erzielen bei den ausgewählten Aufnahmen durchweg gute Ergebnisse.
Aufgrund der Tatsache, dass die bisherigen Auswertungen auf speziell ausgewählten Testaufnahmen aufgebaut sind, die leicht zu detektierende Kanten und einen ruhigen Hintergrund aufweisen, besteht die Möglichkeit, den vorgestellten Ansatz weiter auszubauen und zu verfeinern. Ein denkbares höheres Ziel, das zahlreiche weitere Auswertungen mit unterschiedlichsten Aufnahmen, Voreinstellungen und eventuellen leichten Änderungen des Programm-Codes voraussetzt, wäre ein vollkommen automatisiertes Anwenden des Plugins auf jedes erdenkliche Foto mit anschließender Entzerrung.

Das grundlegende Ziel, die Bestimmung der Kameraverzerrung inklusive anschließender Entzerrung mittels der Hough-Transformation, konnte jedoch im Rahmen dieser Arbeit vollends erreicht werden.

9 Anhang

9.1 Quellenverzeichnis

[1] E. Hecht.: *Optik*; Oldenbourg Verlag München Wien; 4. Auflage; 2005

[2] S. Graf: *Kamerakalibrierung mit radialer Verzeichnung – die radiale essentielle Matrix*; Dissertation Universität Passau; 2007
Internetquelle: http://www.opus-bayern.de/uni-passau/volltexte/2008/1271/pdf/Graf_Simone.pdf (Stand: 03.04.2011)

[3] F. Pedrotti, L. Pedrotti, W. Bausch, H. Schmidt: *Optik für Ingenieure - Grundlagen*; Springer Verlag Berlin Heidelberg; 4. Auflage; 2008

[4] Prof. Dr. rer. nat D. Kunz: *Algorithmen der Bildverarbeitung*; Kapitel 13; Vorlesungsskript 2010.

[5] W. Burger, M. Burge: *Digitale Bildverarbeitung , Eine Einführung mit Java und ImageJ*; Springer-Verlag Berlin Heidelberg; 2. Auflage; 2006

[6] L. Papula: *Mathematische Formelsammlung – Für Ingenieure und Naturwissenschaftler*; Viewegs Fachbücher der Technik; 9. Auflage; 2006

[7] J.A. Nelder, R. Mead: *A Simplex Method for Function Minimization*; Computer Journal; 1965
Internetquelle: http://comjnl.oxfordjournals.org/content/7/4/308.full.pdf (Stand: 03.04.2011)

[8] J.C. Lagarius, J.A. Reeds, M.H. Wright, P.E. Wright: *Convergence Properties of the Nelder-Mead Simplex Method in Low Dimensions*; SIAM J. Optim.; 1998
Internetquelle: http://www.aoe.vt.edu/~cliff/aoe5244/nelder_mead_2.pdf (Stand: 03.04.2011)

[9] Wikipedia - *Die freie Enzyklopädie*:
http://de.wikipedia.org/wiki/Downhill-Simplex-Verfahren (Stand: 03.04.2011)

[10] Grundbegriffe der Statistik:
Internetquelle: http://www.borg-krems.ac.at/~aschoendorfer/kurs_mathe /kurs/statistik1.htm (Stand: 03.04.2011)

[p1] Java/ImageJ Source Code zu [5]:
http://www.imagingbook.com/index.php?id=82 (Stand: 03.04.2011)

[p2] Eclipse: http://www.eclipse.org/

[p3] Commons-Math: http://commons.apache.org/math/

[p4] ImageJ: http://rsbweb.nih.gov/ij/

9.2 Abbildungsverzeichnis

Abb. 1: radiale Verzerrung einer Geraden [2] 2

Abb. 2: Original, tonnenförmige Verzeichnung (0.5; 0.3; 0.1; 0.05) und kissenförmige Verzeichnung (-0.3; -0.1; -0.05; -0.01) von Gitter.tif 3

Abb. 3: künstlich erzeugte Testbilder: Gerade01.tif, Test01.tif und Haus.tif 6

Abb. 4: verzeichnete Testbilder: Gerade01_09050501.tif, Test01_0503005001.tif und Haus_19090909.tif 8

Abb. 5: Rahmen02.jpg 9

Abb. 6: IMG_0077.jpg 9

Abb. 7: YV6P0015.jpg 10

Abb. 8: YV6P0023.jpg 10

Abb. 9: _MG_1305.jpg 10

Abb. 10: _MG_1314.jpg 10

Abb. 11: Geraden im Bildraum (a) werden auf Punkte im Parameter-/Hough-Raum (b) abgebildet [5] 12

Abb. 12: Polarkoordinaten [5] 13

Abb. 13: Geraden im Bildraum (x/y) und ihre entsprechenden Punkte im Hough-Raum (Winkel/Radius) 13

Abb. 14: Nelder-Mead-Simplex nach einer Reflexion (a), Expansion (b), Kontraktion nach außen (c), Kontraktion nach innen (d) und einer Komprimierung (e) [8] 15

Abb. 15: Originalbild “Test01.tif”, Hough-Transformation “HT of Test01.tif” und dazugehöriges Maximumbild “Maxima of Test01.tif” 23

Abb. 16: Originalbild “Test01_09050501.tif”, Hough-Transformation “HT of Test01_09050501.tif” und dazugehöriges Maximumbild “Maxima of Test01_09050501.tif” 23

Abb. 17: Entropie-Methode: HT des Originals (Haus_09050501_HT.tif), HT des besten Ergebnisses (Haus_09050501_best HT.tif) und entzerrtes Original mit eingezeichneten Geraden; die roten Geraden ergeben sich aus den Maxima der originalen HT, die grünen aus den Maxima der besten HT 37

Abb. 18: Varianz-Methode: HT des Originals (rote Geraden), HT des besten Ergebnisses (grüne Geraden) und entzerrtes Original mit eingezeichneten Geraden 38
Abb. 19: CountMax-Methode: HT des Originals (rote Geraden), HT des besten Ergebnisses (grüne Geraden) und entzerrtes Original mit eingezeichneten Geraden 38
Abb. 20: YV6P0015_entzerrt.jpg (Sigma 8-16mm; Canon EOS-1D-Mark; f8; 1/160; 8mm) entzerrt über countMax mit count = 3 43
Abb. 21: YV6P0023_entzerrt.jpg (Sigma 8-16mm; Canon EOS-1D-Mark; f8; 1/250; 16mm) entzerrt über countMax mit count = 3 44
Abb. 22: _MG_1305_entzerrt.jpg (Sigma 8-16mm; Canon EOS 40D; f5.6; 1/320; 8mm) entzerrt über variance mit count = 3 45
Abb. 23: _MG_1314_entzerrt.jpg (Sigma 8-16mm; Canon EOS 40D; f5.6; 1/200; 8mm) entzerrt über variance mit count = 6 46
Abb. 24: Hough-Transformation des Originalbildes _MG_1314_HT und Hough-Transformation des entzerrten Bildes _MG_1314_HT_c3 (variance, count = 3) 46

9.3 Tabellenverzeichnis

Tabelle 1: Gerade01_19090909.tif : Änderung des Parameters steps: a) Originalbild;
b) entzerrt mit steps = 0.1; c) entzerrt mit steps = 0.5..... 33
Tabelle 2: Haus_19090909.tif : Änderung des Parameters count: a) Originalbild;
b) entzerrt mit count = 1; c) entzerrt mit count = 3;
d) entzerrt mit count = 6..... 34
Tabelle 3: Haus_09050501.tif: Vergleich zwischen den Zielfunktionen mit count = 1:
a) Originalbild; b) entzerrt über entropy; c) entzerrt über variance;
d) entzerrt über countMax..... 35
Tabelle 4: Haus_09050501.tif: Vergleich zwischen den Zielfunktionen mit count = 6:
a) Originalbild; b) entzerrt über entropy; c) entzerrt über variance;
d) entzerrt über countMax..... 36
Tabelle 5: Rahmen02.tif: Vergleich zwischen den Zielfunktionen mit count = 3:
a) Originalbild; b) entzerrt über variance; c) entzerrt über countMax..... 39
Tabelle 6: IMG_0077.tif: Vergleich zwischen den Zielfunktionen mit count = 1:
a) Originalbild; b) entzerrt über entropy; c) entzerrt über variance;
d) entzerrt über countMax..... 41
Tabelle 7: IMG_0077.tif: Vergleich zwischen den Zielfunktionen mit count = 8:
a) Originalbild; b) entzerrt über entropy;c) entzerrt über variance;
d) entzerrt über countMax..... 42
Tabelle 8: YV6P0015.jpg: Vergleich zwischen den Zielfunktionen mit count = 3:
a) Originalbild; b) entzerrt über countMax; c) entzerrt über variance..... 43
Tabelle 9: YV6P0023.jpg: Vergleich zwischen den Zielfunktionen mit count = 3:
a) Originalbild; b) entzerrt über countMax; c) entzerrt über variance..... 44
Tabelle 10 : _MG_1305.jpg: Vergleich zwischen den Zielfunktionen mit count = 3:
a) Originalbild; b) entzerrt über countMax; c) entzerrt über variance..... 45
Tabelle 11: _MG_1314.jpg: Vergleich zwischen den Zielfunktionen mit count = 3:
a) Originalbild; b) entzerrt über countMax; c) entzerrt über variance..... 46

Eidesstattliche Erklärung

Ich versichere hiermit, die vorgelegte Arbeit in dem gemeldeten Zeitraum ohne fremde Hilfe verfasst und mich keiner anderen als der angegebenen Hilfsmittel und Quellen bedient zu haben.

Köln, den 04. April 2011

Unterschrift
(Gina Maria Wagner)

Sperrvermerk

Die vorgelegte Arbeit unterliegt keinem Sperrvermerk.

Weitergabeerklärung

Ich erkläre hiermit mein Einverständnis, dass das vorliegende Exemplar meiner Bachelorarbeit oder eine Kopie hiervon für wissenschaftliche Zwecke verwendet werden darf.

Köln, den 04. April 2011

Unterschrift
(Gina Maria Wagner)