Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Bachelorarbeit

Steve Herve Feutat Tamou

Subpixelgenaue Kantendetektion für ein Bildmesssystem

Fakultät Technik und Informatik
Department Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Steve Herve Feutat Tamou

Subpixelgenaue Kantendetektion für ein Bildmesssystem

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung
im Studiengang Technische Informatik
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer : Prof. Dr. Ing. Andreas Meisel
Zweitgutachter : Prof. Dr. Reinhard Baran

Abgegeben am 19.02.2016

Steve Herve Feutat Tamou

Subpixelgenaue Kantendetektion für ein Bildmesssystem

Stichworte

Konturensammlung, Subpixelgenaue Verfeinerung, Testroutinen

Kurzzusammenfassung

Ein Verfahren, welches Objektkonturen von 2-dimensionalen Objekten subpixelgenau identifizieren kann, wird in dieser Arbeit konstruiert. Es werden ein Gradient basiertes und ein 2. Ableitung Verfahren verwendet. Hierfür wird mithilfe einer Datenbank ein Vergleich zwischen realen und ermittelten Konturpositionen gezogen und die Abweichung zwischen diesen berechnet. Durch geringe Abweichung kann das verwendete Verfahren getestet werden. Formabweichungen sind im Verfahren berücksichtigt.

Steve Herve Feutat Tamou

Title of the paper

Subpixel edge detection for image measurement system

Keywords

Edge collection, subpixel refinement, test routine

Abstract

In this work is a method constructed which can identify object contours of 2-dimensional objects with subpixel accuracy. There is a gradient-based and a 2nd derivative method in use. To this end, actual and determined contour positions are compared by using a database drawn and calculate the deviation between them. The method can be tested by slight deviation. Shape variations are included in the process.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

1.1 Motivation

Ein Subpixel ist eine Untereinheit bei der Unterteilung eines Pixels. Heutzutage werden Subpixel-Techniken in vielen Bereich der Bildverarbeitung eingesetzt. Beispiele sind etwa:

1. geometrische Bildtransformationen (Bildrotation, Skalierung)
2. Bildmesstechnik (Abstände zwischen Kanten, Rauschen)

So erscheinen vergrößerte Bilder viel glatter und bei der Bestimmung von Kantenorten ist die Genauigkeit viel höher. Durch Interpolation ist es möglich, Information über die Lage eines Bildinhalts zu erhalten, wobei die durch die Pixel vorgegebene Auflösung übertroffen wird. Mit Kameras erzeugte Bilder enthalten je nach Qualität und Lichtverhältnisse mehr oder weniger Grauwertrauschen. Konturenfilter haben die Tendenz (aufgrund von Ableitungen), dieses Rauschen zu verstärken. Folgende Kriterien sind also für eine optimale Kantendetektion zu beachten: die Verminderung des Rauscheinflusses, sowie der fehlenden echten Kanten. Die pixelgenauen detektierten Kanten müssen so nah wie möglich an den wahren subpixelgenauen Kanten sein.

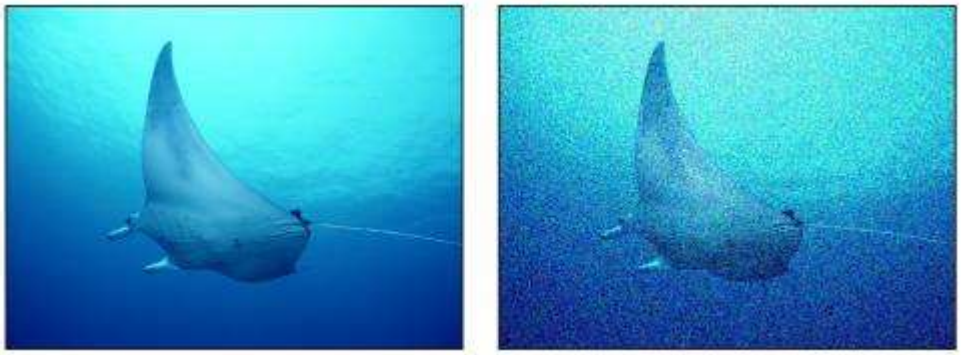


Abbildung 1.1: (Links) Originalbild (rechts) Verrauschtes Bild durch die Anwendung eines Filters

1.2 Aufgabenstellung

In dieser Arbeit soll die Position von Kanten mit möglichst hoher Genauigkeit bestimmt werden. Diese Genauigkeit soll hier nicht pixel-, sondern subpixelgenau sein. Im ersten Schritt sollen die Kantenpunkte pixelgenau erfasst und in eine Konturpunktliste eingetragen werden.

Im zweiten Schritt werden die Konturpunkte subpixelgenau verfeinert. Darauf aufbauend sollen Testroutinen durchgeführt werden, um die Genauigkeit und die Richtigkeit des Verfahrens zu ermitteln. Dazu werden die neu gemessenen Konturen des Werkstückes mit Ausgleichmodellen verglichen und jeweils ein Abweichungswert berechnet.

Ziel ist ein Verfahren mit möglichst geringer Abweichung.

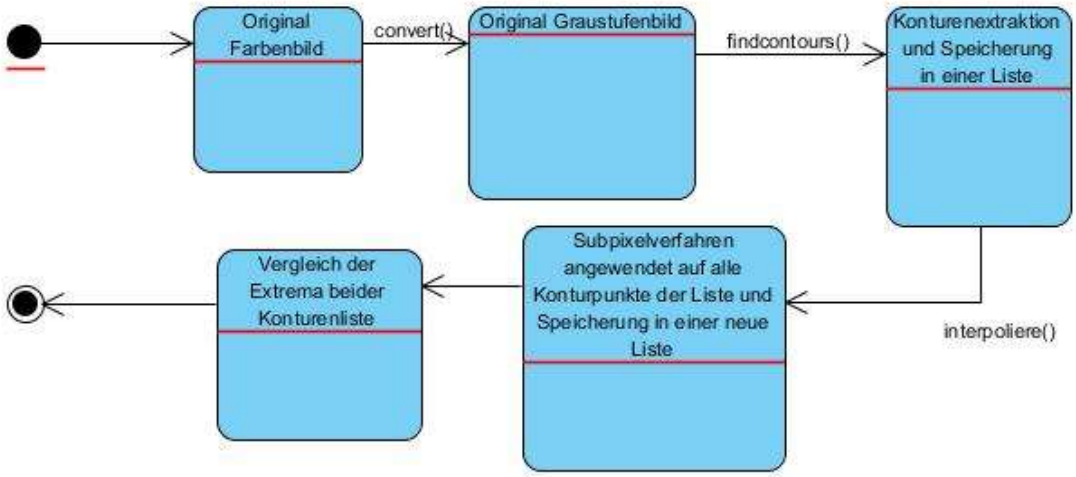


Abbildung 1.2: Sequenz der Verarbeitungsschritte

1.3 Kapitelüberblick

Im Kapitel 2 dieser Arbeit wird näher auf der Konturenerfassung eingegangen. Dabei werden die verschiedene Funktionen erläutert, sowie die Funktionsweise beschrieben.

Kapitel 3 erläutert die verwendeten Subpixelverfahren, die in dieser Arbeit verwendet werden. Es werden die Verfahren, sowie deren Funktionsweise beschrieben.

Kapitel 4 beschreibt die Testroutinen, dokumentiert sowohl die Ergebnisse und wertet diese aus. Hierfür werden die unterschiedlichen Verfahren auf Genauigkeit getestet und die Ergebnisse erläutert.

Kapitel 5 stellt eine Zusammenfassung der Arbeit dar und zeigt auf, welche Erweiterungen auf dieser Arbeit aufbauen können.

1.4 Projektumgebung

Für diese Arbeit wurde ein Projekt auf Microsoft Visual Studio angelegt. OpenCV[1] ist eine Open Source Computer Vision Bibliothek. Sie besteht aus einer Reihe von C-Funktionen, C++ Klassen und vielen üblichen Algorithmen zur Bildverarbeitung und Computer Vision. Die neueste Update-Version ist 3.0. Die Release-Versionen für Unix, iOS, Windows oder Android können heruntergeladen werden. Es hat C++, C, Python und Java-Schnittstellen. OpenCV ist für nicht-kommerzielle und kommerzielle Anwendungen kostenlos. Verschiedene Applikationen sind dort vorhanden, wie zum Beispiel 2D- und 3D-Feature-Toolkits, Schätzung der Eigenbewegung, Gesichtserkennungssystem und Gestenerkennung. OpenCV bietet auch eine Machine Learning Library, darunter den nearest neighbor Algorithmus, Decision trees und Gradient boosting trees.

Im Internet findet man einige Tutorials für Anfänger über OpenCV. Weitere Möglichkeiten und die Anbindung an OpenCV sind unter folgendem Link zu finden: [2]

Nach der Einbindung von #include <opencv/cv.h> und #include <opencv/highgui.h> ist es dann möglich, mit OpenCV zu arbeiten (OpenCV sollte übrigens in den Windows-Suchpfad aufgenommen werden).

[1] http://opencv.org
[2] www.csie.ntu.edu.tw

2 Konturenerfassung

Das Ziel hier ist, die pixelgenauen Koordinaten der Intensitätswerte zu erfassen und zu visualisieren. Durch die pixelgenaue Konturenerfassung soll festgestellt werden, welche der Randpixel als Rauschen verworfen und welche beibehalten werden sollen. Im einfachsten Fall wird ein Schwellenkriterium für die Kantenerkennung verwendet. Der Intensitätswert des Pixels wird in der Mitte des Pixels angenommen. Der Prozess der Konturenidentifikation besteht aus den folgenden Einzelschritten:

1. Farbbild laden.
2. Farbbild in Graustufenbild konvertieren.
3. Graustufenbild binarisieren: Dies dient der Aufbereitung des Bildes für die Konturenextraktion. Bei der Binarisierung wird der Grauwert eines jeden Bildpunktes mit einem Schwellwert verglichen. So wird der Bildpunkt auf schwarz gesetzt, wenn sein Grauwert unterhalb des Schwellwertes bzw. auf weiß gesetzt, wenn sein Grauwert oberhalb des Schwellwertes liegt.
4. Sollte das Bild nach der Binarisierung verrauscht sein, so kann nach der Binarisierung ein Gauss-Filter für das Entfernen des Rauschens angewendet werden. In dieser Arbeit wird dieses Verfahren in Verbindung mit dem Laplace Filter angewendet (LoG-Filter).
5. Konturen erfassen.

OpenCV bietet zahlreiche Funktionen für die Berechnung der Konturen. In dieser Arbeit erfolgt die Konturenerfassung mittels drei Verfahren: *Findcontours(),* ein Gradient-basiertes Verfahren (Sobel-Operator) und ein Zerocrossing-basiertes Verfahren (Laplace of Gaussian). In den folgenden Abschnitten werden diese Verfahren erläutert.

2.1 Pixelgenaue Konturenerfassung mit Hilfe von *Findcontours()*

Zur Konturenerfassung wird häufig die Funktion *Findcontours ()* von OpenCV verwendet. Es handelt sich um einen einfachen Algorithmus, der systematisch das gesamte Bild abtastet. Gemäß der Dokumentation der Funktion erzeugt er Konturen aus dem binären Bild mit dem Algorithmus von Suzuki, S. and Abe[3]. Dabei wird in jedem binären Bild ein Quadrat in der Region um den Bezugspunkt mit einer festen Größe in Betracht gezogen.

Die Funktion identifiziert miteinander verbundene Bereiche (connected components) im vorgesehenen binären Bild und gibt eine Liste aller identifizierten Konturen zurück.

Konturen im Bereich unter dem minimalen Schwellenwert werden entfernt, um das Debugging und den Test-Prozess schneller zu machen.

Als Ergebnis werden alle pixelgenauen Konturen in einer Baumdatenstruktur gespeichert, die einen Übergang von schwarz zu weiß oder umgekehrt vorweisen. Bei den gespeicherten Konturen kann man zwischen internen Konturen (schwarze Pixel im weißen Bereich) und externen Konturen (weiße Pixel im schwarzen Bereich) unterscheiden.

[3] Suzuki, S. and Abe, K., Topological Structural Analysis of Digitized Binary Images by Border Following. CVGIP 30 1, pp 32-46 (1985)

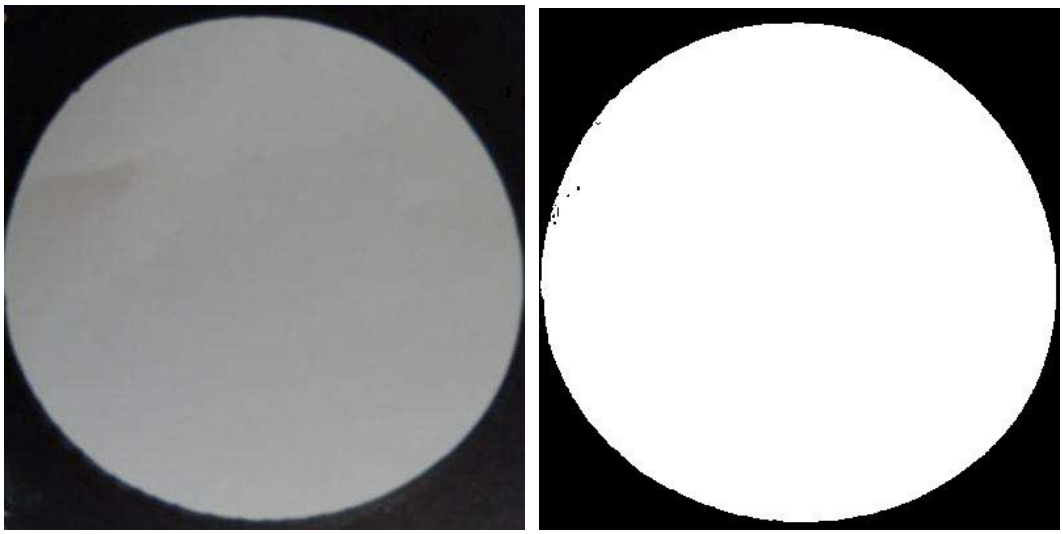


Abbildung 2.1: (Links) Originalbild eines Kreises (rechts) Binarisiertes Bild des Kreises

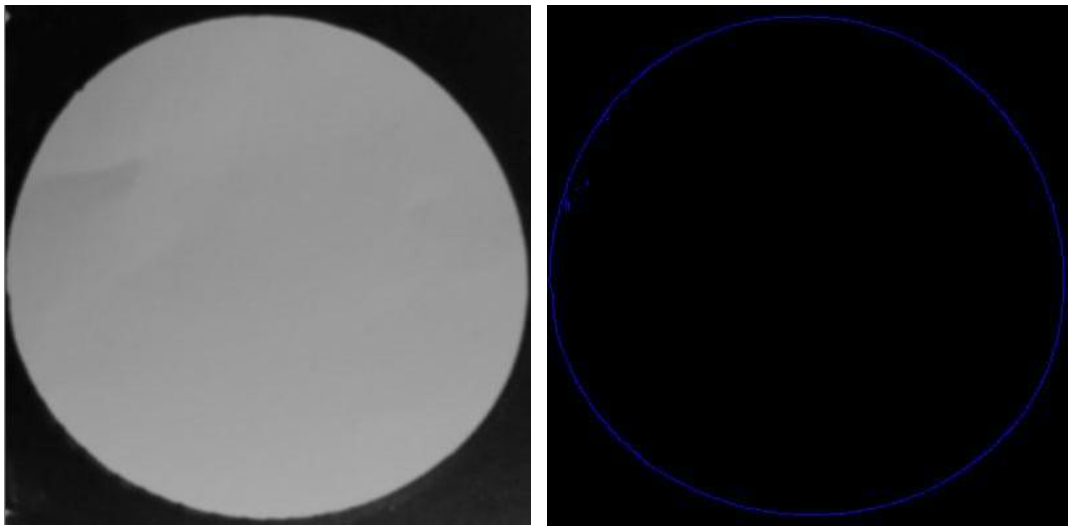


Abbildung 2.2: (Links) Graustufenbild eines Kreises (rechts) Konturen Extraktion mit der Funktion *Findcontours()* von OpenCV

2.2 Gradient-basiertes Verfahren

Der Gradient ist ein Vektorfeld. Jedem Bildpunkt werden ein bestimmter Betrag und eine Richtung zugewiesen.

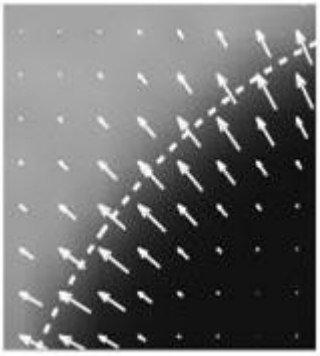


Abbildung2.3: Bild zur Gradientenrichtung senkrecht zur Kantenrichtung

Der Betrag des Gradienten liefert Informationen über die Stärke der Kante. Die Richtung des Gradienten ist immer senkrecht zur Richtung der Kante.

Entsprechendes ergibt sich für den Gradienten: $\nabla f(x,y) = \begin{bmatrix} \frac{\partial f(x,y)}{\partial x} \\ \frac{\partial f(x,y)}{\partial y} \end{bmatrix}$

Um aufwändige Berechnungen bei der Laufzeit zu sparen, wird der Betrag des Gradienten angenähert.

Der Sobel-Operator gehört zu den am häufigsten verwendeten Filtern in Gradient-basierten Verfahren, um Kanten in Bildern hervorzuheben. Dafür verwendet er zwei Faltungsmasken:

$$Hc = \frac{1}{4}\begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix}, \; Hr = \frac{1}{4}\begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

Diese berechnen die erste Ableitung der Bildpunkt-Helligkeitswerte.

Der Algorithmus nutzt eine Faltung mittels einer 3×3-Matrix (Faltungsmatrix), die über das Eingabebild geschoben wird.

Dies führt zu folgender Darstellung mit Hilfe der Faltung:

$$\nabla f(x,y) = \begin{bmatrix} Hc * f(x,y) \\ Hr * f(x,y) \end{bmatrix}$$

Anschließend kann dadurch die Umgebung des Pixels, die sich unter dem Zentrum der Faltungsmaske befindet, betrachtet werden. Um den Grauwert des Pixels in der Mitte der Matrix zu errechnen, wird das Pixel des Eingabebildes mit dem Wert in der entsprechenden Matrix-Zelle multipliziert und die Produkte aufaddiert. „Die Bereiche der größten Intensität sind dort, wo sich die Helligkeit des Originalbildes am stärksten ändert und somit die größten Kanten darstellt“.[4] Der Prozess der Konturenidentifikation besteht aus den folgenden Einzelschritten:

1. Das Eingangsbild mit dem Gauss-Filter glätten.
2. Horizontalableitung durch Berechnung der Faltung mit einem Kernel *Hc* bestimmen.
3. Vertikalableitung durch Berechnung der Faltung mit einem Kernel *Hr* bestimmen.
4. Den Betrag des Gradienten pro Bildpunkt berechnen.

Der Betrag des Gradienten wird mit folgender Formel berechnet:

$$|\nabla H| = \sqrt{Hc^2 + Hr^2}$$

$$Hc = \frac{\partial H}{\partial x}, Hr = \frac{\partial H}{\partial y}$$

Die Phase wird folgendermaßen berechnet:

$$\varphi = tan^{-1} \frac{Hr}{Hc}$$

[4] http://www.linkfang.de/wiki/Sobel-Operator; 02.01.2016

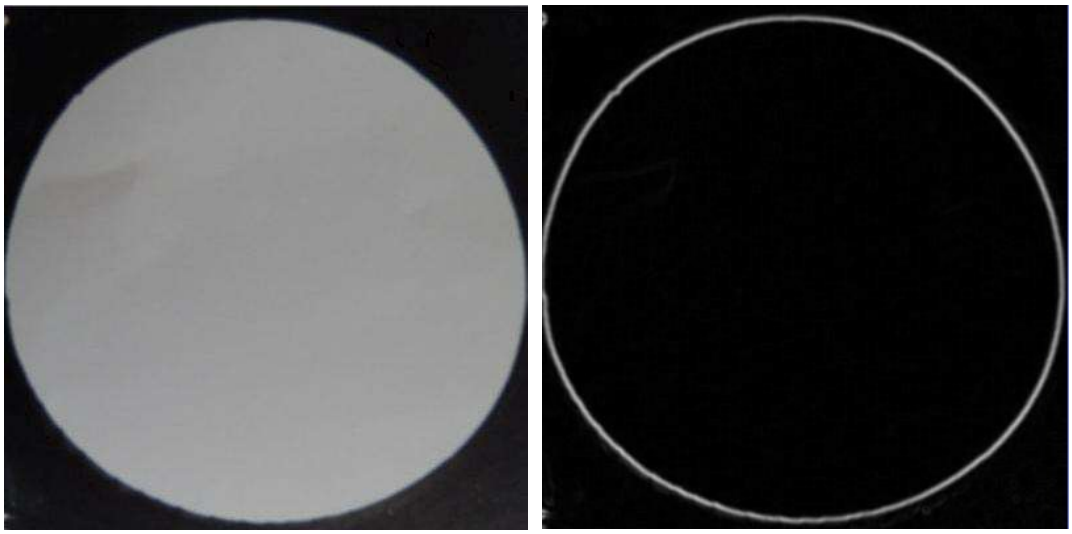


Abbildung2.4: (Links) Originalbild (rechts) Ergebnis nach Anwendung des Sobel-Operators

2.3 Zerocrossing-basiertes Verfahren

Kantenpunkte können durch Auffinden der Nulldurchgänge der zweiten Ableitung erkannt werden. Ein besserer Ansatz wäre, nur die Punkte zu finden, die lokale Maxima sind und diese dann als Kantenpunkte zu betrachten. Dies bedeutet, dass diese Kantenpunkte einen Nulldurchgang in der zweiten Ableitung haben. Es gibt einen Operator, der hier in Betracht genommen wird: der Laplace-Operator. Der Laplace –Operator

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

hebt Nulldurchgänge hervor und kann über einer 3x3-Matrix angenähert werden durch: $\nabla^2 f \approx L1 \cdot f$, wobei $L1 = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$

$$L1 = L1_{xx} + L1_{yy}$$

Dieser Operator zeigt sich leider sehr empfindlich gegenüber Rauschen. Um die Wirkung von Rauschen zu vermeiden, kann zuvor eine Bildglättung vorgenommen werden.

Im Rahmen dieser Arbeit wird der LoG-Operator verwendet, der die Gauss-Filterung mit der Laplace für die Kantenerkennung verbindet, um das Rauschen zu vermindern. Mit der Gauss-Funktion:

$$h(x,y) = e^{\left(-\frac{1x^2+y^2}{2\ \ \sigma^2}\right)}$$

ergibt sich der LoG-Filter:

$$\nabla^2 h(x,y) = \frac{x^2+y^2-2\sigma^2}{\sigma^4} e^{\left(\frac{-1x^2+y^2}{2\ \ \sigma^2}\right)}$$

Der Prozess der Konturenidentifikation auf einem Grauwertbild:

1. Mit der Gauss-Filterung das Bild glätten.
2. Den Laplace-Operator anwenden.

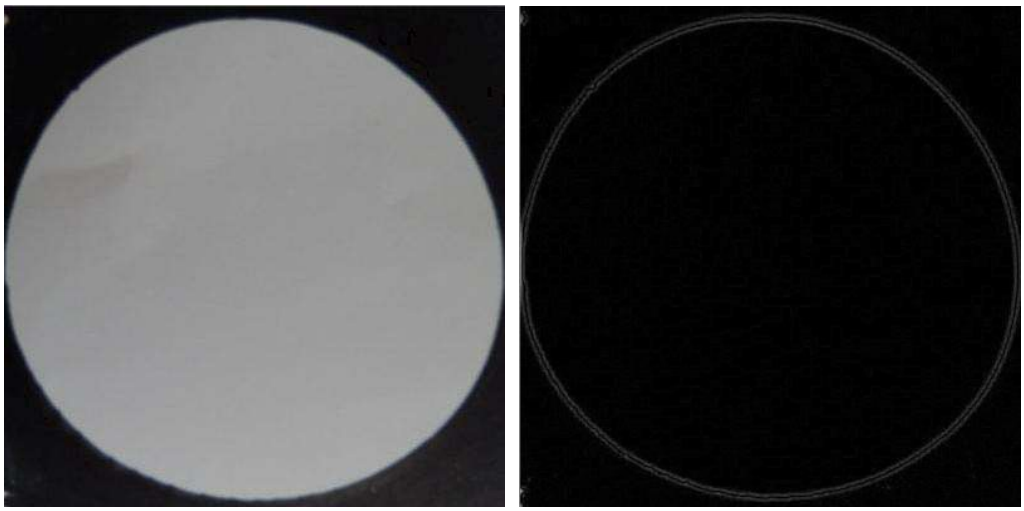


Abbildung2.5: Ausgabe der Anwendung des LoG-Detektors am Originalbild

3 Subpixelgenaue Verfeinerung

Ziel der subpixelgenauen Verfeinerung ist es, die Koordinaten der Intensitätswerte auf ein neues, höher aufgelöstes, diskretes Raster anzuordnen und dabei die Zwischenpositionen zu interpolieren und eine höhere Genauigkeit der Kantenposition zu bekommen. In diesem Schritt ist es erforderlich, vor Anwendung des Verfahrens (zur Verfeinerung) so nah wie möglich am wahren Maximum zu sein, da dieser sich in der Nachbarschaft des lokalen Maximums befindet. Dies erfolgt, indem das Konturprofil (quer zum Kantenverlauf) mit einer mathematischen Funktion angenähert und dann der Wendepunkt dieser Funktion mit hoher Genauigkeit berechnet wird. Dort liegt die genaue Kante. Interpolations- und/oder Approximationstechniken können hier angewendet werden, um kontinuierliche Funktionen aus den Bildableitungswerten herzustellen.

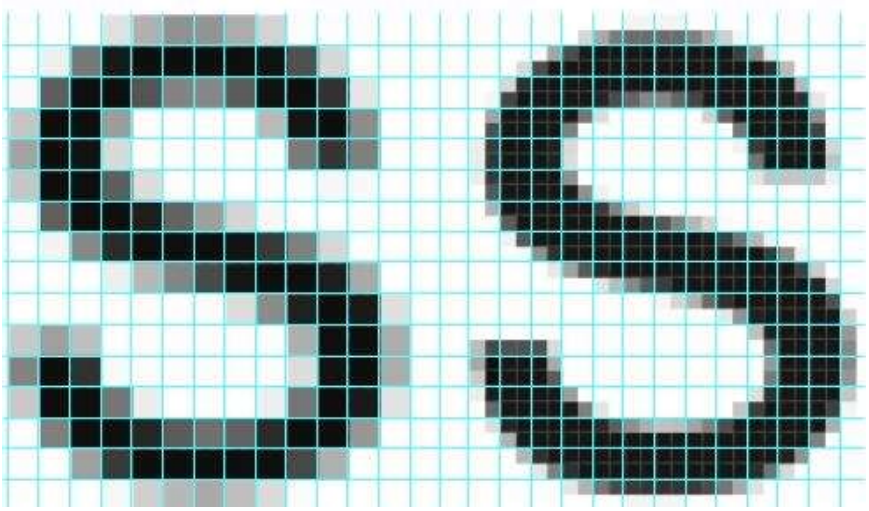
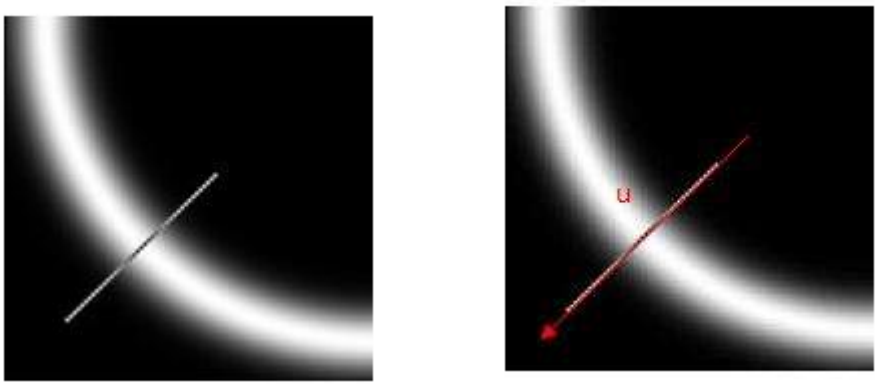


Abbildung 3.1: Simulation einer Vervierfachung der Auflösung mittels Graustufenpixel

Eine Interpolation bestimmt Funktionen, die alle Bildableitungswerte genauer anpasst. Eine Approximation sucht nach Funktionen, deren Datenpunkte sich den Bildgrauwerten annähern, aber notwendigerweise genau durch sie hindurchgehen.



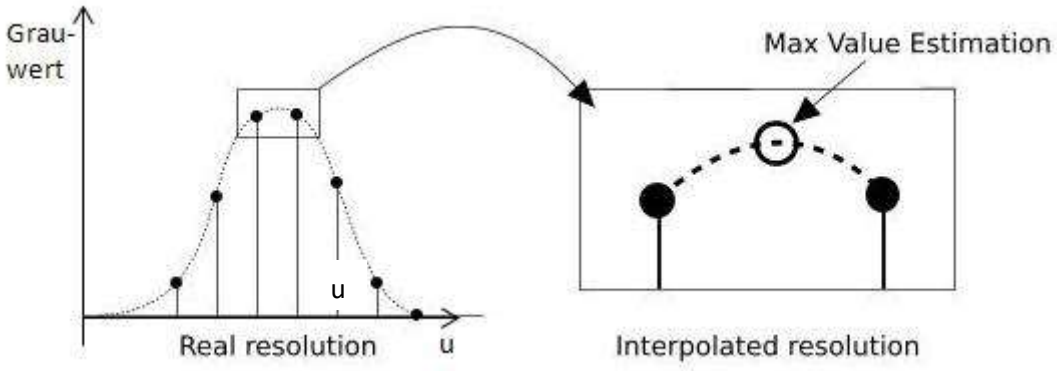


Abbildung 3.2: Erfassung der Maximalwertposition mit Subpixelauflösung mittels Interpolation (oben links) Originalbild (oben) rechts Gradientenbild

In den folgenden Abschnitten werden diese Methoden diskutiert. Sie wurden angewendet, um Kanten sowohl von der Gradient-basierten Suche, als auch von der Nulldurchgang-basierten Suche zu verfeinern.

3.1 Kantenverfeinerung durch Interpolation

Interpolation ist eine Bildverarbeitungstechnik, um die scheinbare Auflösung eines Bildes zu erhöhen. Es nutzt die Tatsache, dass jedes Pixel aus einzelnen Subpixeln mit mehr Details besteht. Durch die Anwendung der Interpolation werden die Koordinaten des Punktes mit den größten Grauwerten ermittelt.

Hierfür berechnet man also in jedem bekannten pixelgenauen Konturpunkt die Kantenrichtung (Tangente). Senkrecht zur Kantenrichtung ist die Gradientenrichtung. Vom Konturpunkt wird in der Gradientenrichtung ein kleines Stück (z.B. 0,5 Pixel) weitergegangen und dann in dieser Richtung gesucht, wo sich das Maximum der Grauwerte befindet. Dies wird auf jedem der zuvor in einer Konturpunktliste eingetragenen pixelgenauen Punkte angewendet. Alle Werte in der Richtung des Gradienten, die keine Spitzenwerte sind, werden hier unterdrückt.

Es gibt unterschiedliche Interpolationsmethoden z.B.

1. Bilineare Interpolation
2. Cubic Hermite Interpolation
3. Cubic spline Interpolation
4. Nearest neighbor Interpolation

In dieser Arbeit wurde die bilineare Interpolation verwendet. In Abbildung 3.1 werden der Grauwert von *C* zwischen den Grauwerten von *A7* und *A8* und der Grauwert von *B* zwischen den Grauwerten von *A3* und *A4* interpoliert. Nimmt man beispielsweise an, *f(x)* wäre der Grauwert an der aktuellen Position x und δ der Abstand zu der wahren Kante, so kann der Wert von *f(x)* in Bezug auf die Werte bei den Nachbarn mit der Formel berechnet werden:

$$f(x) = \left(\frac{1}{2} + \delta\right) \cdot f(x-1) + \left(\frac{1}{2} - \delta\right) \cdot f(x+1)$$

Die Subpixel-Kantenposition *x+δ* kann durch die folgende Formel gelöst werden:

$$\delta = \frac{2f(x) - f(x-1) - f(x+1)}{2(f(x-1) - f(x+1))}$$

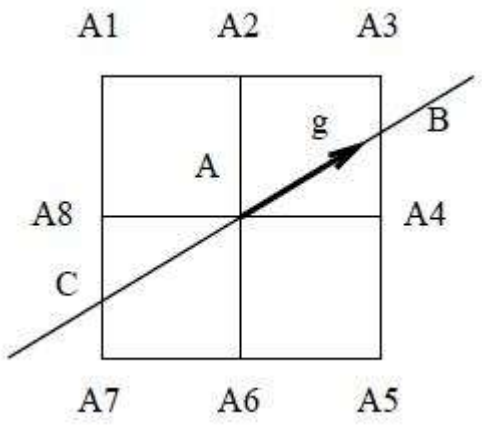


Abbildung 3.3: Prüfung durch Grauwertinterpolation an den Punkten *B* und *C*, ob das Pixel *A* ein lokales Maximum in der Gradientenrichtung ist

Wie aus vier benachbarten Pixeln der Grauwert eines Subpixels bestimmt werden kann, wird in Abbildung 3.4 verdeutlicht. Das Ziel ist, den Grauwert des Punktes mit den Koordinaten 20,5/14,5 (in der Abbildung als *Z* markiert) zu berechnen. Hierfür werden lediglich vier benachbarte Punkte (in der Abbildung benannt als Punkt *A1*, *A2*, *A3* und *A4*) benötigt. Die Subpixelkoordinaten sind in diesem Fall nicht mehr ganzzahlig und damit verbunden auch der Grauwert.

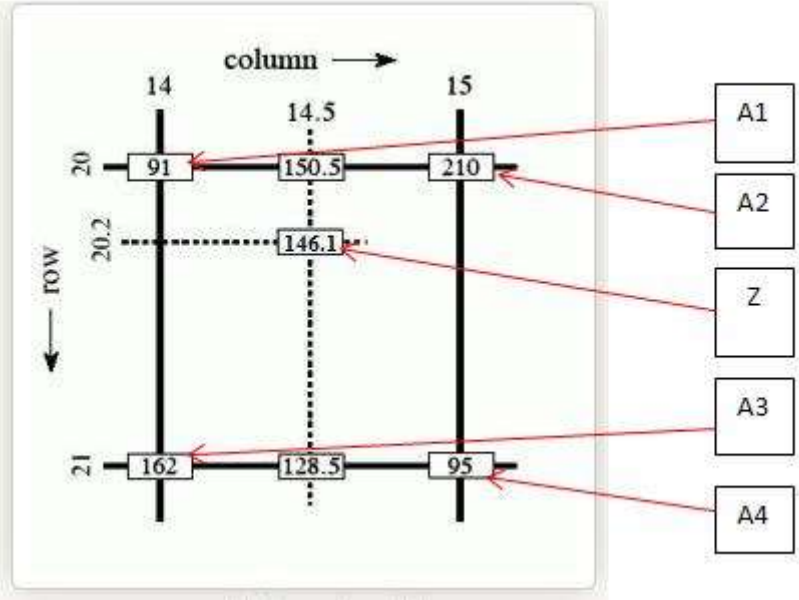


Abbildung3.4: Beispiel der Grauwertbestimmung durch Interpolation

Die folgenden Gleichungen verdeutlichen die Berechnung des Grauwertes an dieser Stelle:

$$G_{20;14,5} = \frac{15 - 14,5}{15 - 14} \cdot 91 + \frac{14,5 - 14}{15 - 14} \cdot 210 = 150,5$$

$$G_{21;14,5} = \frac{15 - 14,5}{15 - 14} \cdot 162 + \frac{14,5 - 14}{15 - 14} \cdot 95 = 128,5$$

$$G_{20,2;14,5} = \frac{21 - 20,2}{21 - 20} \cdot 150,5 + \frac{20,2 - 20}{21 - 20} \cdot 128,5 = 146,1$$

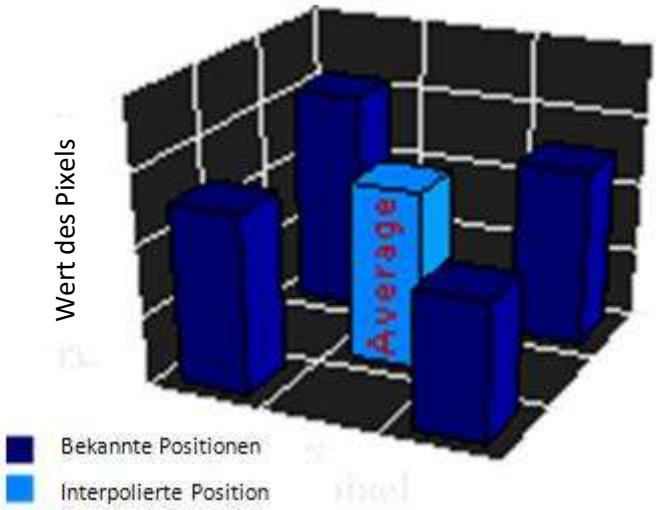


Abbildung 3.5: Bilineare Interpolation betrachtet die bekannten Pixelwerte in der 2x2 Nachbarschaft rund um das unbekannte Pixel

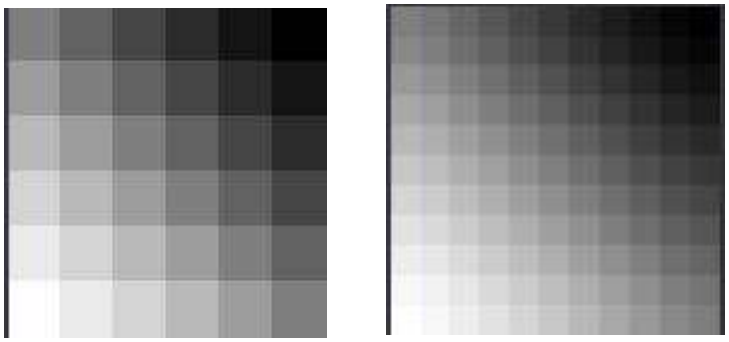


Abbildung 3.6: (Links) Originalbild (rechts) Ergebnis nach der Interpolation

```
float get_sub_pix(cv::Mat const &img, cv::Point2f const &pt)
{
    int x = static_cast<int>(pt.x);
    int y = static_cast<int>(pt.y);

    int x0 = cv::borderInterpolate(x,   img.cols, cv::BORDER_REPLICATE);
    int x1 = cv::borderInterpolate(x+1, img.cols, cv::BORDER_REPLICATE);
    int y0 = cv::borderInterpolate(y,   img.rows, cv::BORDER_REPLICATE);
    int y1 = cv::borderInterpolate(y+1, img.rows, cv::BORDER_REPLICATE);

    float a = pt.x - x;
    float c = pt.y - y;

    float x1_interpolate = (img.at<uchar>(y0, x0) * (1.0 - a)
                          + img.at<uchar>(y0, x1) * a);
    float x2_interpolate = (img.at<uchar>(y1, x0) * (1.0 - a)
                          + img.at<uchar>(y1, x1) * a);
    float target = x1_interpolate * (1.0 - c) + x2_interpolate * c;

    return target;
}
```

3.2 Non-Maxima Suppression

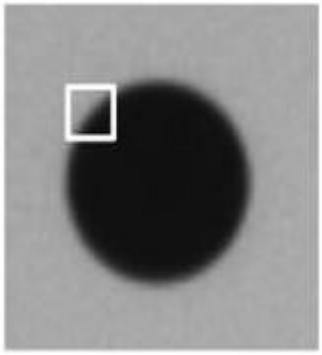
Dieses Verfahren basiert üblicherweise auf einem der zwei zur Kantenerkennung verwendeten Verfahren und dient der Unterdrückung der lokalen Nicht-Maxima in der Gradientenrichtung. Die Non-Maxima Suppression besteht aus folgenden Schritten:

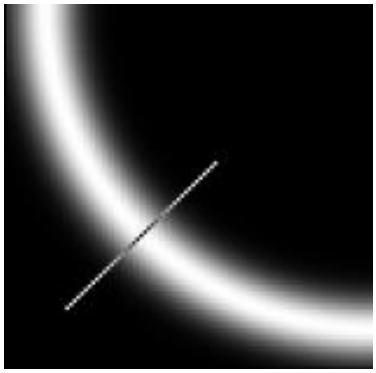
1. Ein Punkt (x, y) sei gegeben, wobei *x* und *y* ganze Zahlen und *I* (x, y) den Grauwert des Pixels (x, y) darstellen.
2. Den Betrag des Gradienten in (x, y) berechnen.
3. Den Betrag des Gradienten in der Nachbarschaft von (x, y) senkrecht zur Kantenrichtung bestimmen.
4. Wenn (x, y) kein lokales Maximum ist (Betrag des Gradienten an der Stelle (x, y) ist kleiner als der Betrag des Gradienten des Nachbarpunktes in

Gradientenrichtung), dann ist es kein Kantenpunkt. Der Punkt mit dem höchsten Gradientenbetrag wird als Kantenpunkt erkannt.

Im einfachsten Fall werden Beträge des Gradienten der in der Nachbarschaft liegenden Punkte linear interpoliert, bis ein lokales Maximum gefunden ist.

Danach wendet man eine Hysterese-Schwellenwertbildung an den Gradientenbeträgen an. Der Standardmodus sieht, dass im Originalbild alle Bildpunkte mit Grauwert unter den Schwellenwert als schwarz, und alle Werte darüber als weiß angenommen werden. Wenn (x, y) ein lokales Maximum ist, dann wird die Lage des Kantenpunktes in der Gradientenrichtung als Maximum einer Interpolation auf die Grauwerte bei (x, y) und den benachbarten Punkten geschätzt.





5	5	5	5	6	7	8
5	5	5	6	7	8	8
5	5	6	7	8	8	7
5	6	7	8	8	7	6
6	7	8	8	7	6	5
7	8	8	7	6	5	5
8	8	7	6	5	5	5

Abbildung 3.7: Non -Maxima Suppression senkrecht zur Kantenrichtung; die Maxima sind in grau dargestellt

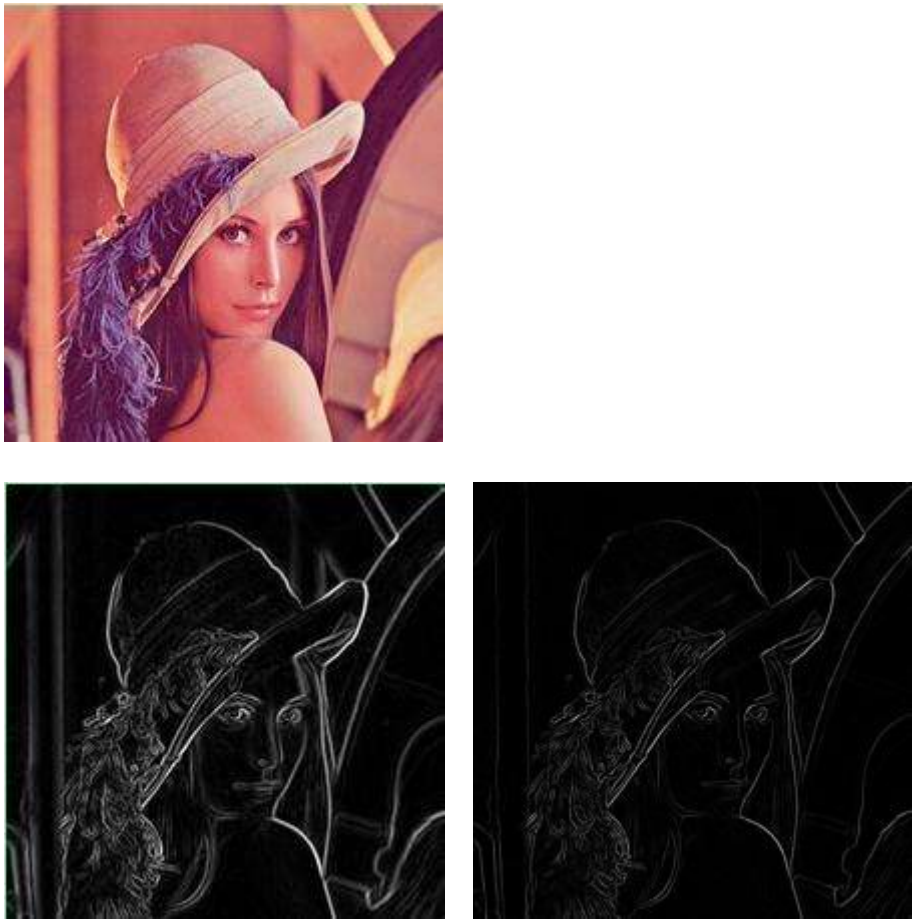


Abbildung 3.8: (Oben) Originalbild (links) Gradientenbild (rechts) Non-Maxima Supression Bild

Algorithmus
Foreach Pixel (x, y) auf der Kante:

lokales Maxima ist Punkt(i,j)

If Gradientenbetrag(i,j)< Gradientenbetrag(i+1,j+1)

Then lokales Maxima ist Punkt(i+1,j+1) ;

Grauwert(i+1,j+1)= Gradientenbetrag(i+1,j+1);

```
        Grauwert(i,j)= 0;
Else    Grauwert(i,j)= Gradientenbetrag(i,j);

Endforeach
```

3.3 Kantenverfeinerung durch Approximation

Die Position einer Kante kann schätzungsweise mit Subpixel-Auflösung durch Mittelung entlang des Profils in der Ausgabe des Zerocrossing Verfahrens ermittelt werden.

Um die Kantenlage in Subpixelauflösung zu berechnen, nimmt man Proben der Grauwerte des Gausschen Kantendetektorausgangs (ohne Non-Maxima Suppression) entlang der Gradientenrichtung zu beiden Seiten der Kante. Die Subpixel-Korrektur der Position der Kante entlang der Gradientenrichtung ist durch die folgende Formel gegeben:

$$\delta d = \frac{\sum_{i=1}^{n} gi \cdot di}{\sum_{i=1}^{n} gi}$$

i: Punktindex (aller Konturpunkte)
di: Abstand eines Subpixels entlang der Gradientenrichtung zum pixelgenauen Kantenpunkt
gi: Grauwert an dieser Stelle

Die Korrektur kann danach zu den Koordinaten des Pixels addiert werden, um eine genauere Schätzung der Kantenposition bereitzustellen.

Der Algorithmus zur Berechnung der Approximation entlang der Gradientenrichtung hat den Vorteil, dass die resultierenden subpixelgenauen Kanten mit den pixelgenauen Kanten unter Verwendung statistischer Techniken verglichen und die Ergebnisse dieses Vergleiches als Kriterium für die Kantenerkennung verwendet werden können.

Ist das subpixelgenaue Profil nicht nahe (d.h. nicht im Bereich zwischen [-0,5; 0,5]) an dem pixelgenauen Profil, wird die Kante nicht dem idealen Modell entsprechen. In diesem Fall ist eine genaue Schätzung der Kantenposition nicht möglich.

3.4 Subpixel-Lokalisierung mit Hilfe des Canny-Algorithmus

Die Interpolation der Randpositionen auf Subpixelgenauigkeit ist in einer Reihe von Anwendungen der Computer Vision von Bedeutung. Wahrscheinlich ist der Canny-Algorithmus der am weitesten verbreitete Kantendetektionsalgorithmus für den Erhalt von Subpixel-Schätzungen.

Der Canny-Algorithmus wurde von John F. Canny im Jahr 1986 entwickelt und erfüllt drei Hauptkriterien:

1. Niedrige Fehlerrate: Deutet auf eine gute Erkennung von Kanten hin.
2. Gute Lokalisierung: Der Abstand zwischen erkannten Kantenpixeln und realen Randpixeln müssen minimiert werden.
3. Minimale Antwort: Nur eine Detektorantwort pro Kante.

Der in dieser Arbeit benutzte Algorithmus besteht aus folgenden Schritten:

1. Den bereits erläuterten Gradient-basierten Prozess durchführen.
2. Die Non-Maxima Suppression anwenden. Dies beseitigt Pixel, die nicht als Kante gelten. Daher werden nur dünne Linien (Kandidatenkanten) bleiben.
3. Hysteresis thresholding auf den Gradientenwerten anwenden. Canny verwendet zwei Schwellen (obere und untere):
    - Wenn der Grauwert eines Pixels größer ist als der obere Schwellenwert, wird der Pixel als ein Kantenpunkt akzeptiert.
    - Wenn der Grauwert eines Pixels unterhalb der unteren Schwelle ist, dann wird er zurückgewiesen.
    - Wenn der Grauwert zwischen den beiden Schwellwerten ist, wird es nur dann akzeptiert, wenn dieser mit einem Pixel, der über dem

oberen Schwellenwert angeschlossen ist, verbunden ist. Der Canny-Algorithmus empfiehlt ein Oben/Unten-Verhältnis von 2:1 und 3:1.

4. Schließlich Subpixelinterpolation in Gradientenrichtung

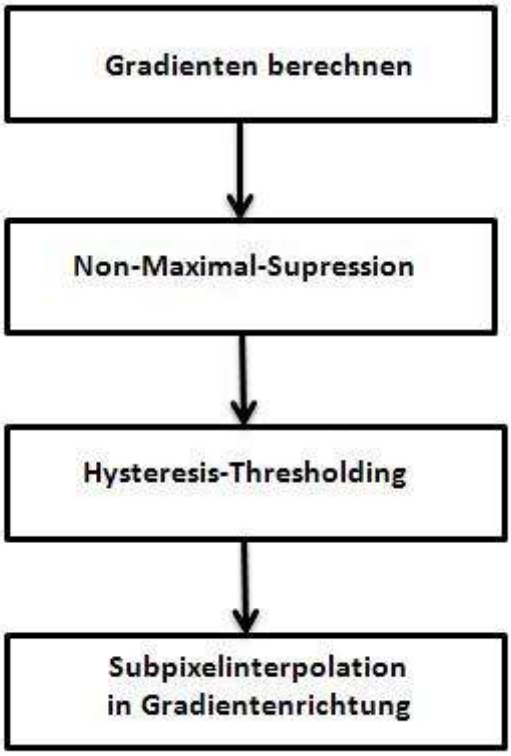


Abbildung 3.9: Schritte für die Canny-Kantendetektion mit Subpixelgenauigkeit

Ein Problem, das mit einem Subpixel-Kantendetektor entstehen kann, ist, dass es nicht möglich ist, allgemeine Techniken wie beispielsweise Hysteresis thresholding anzuwenden, da die Subpixel-Punktkoordinaten keine Integer mehr sind.

Dieses Problem kann gelöst werden, indem Kantenpunkte mit nichtganzzahligen Koordinaten mit den Kantenpunkten von ganzzahligen Koordinaten verbunden werden, aus denen sie berechnet wurden.

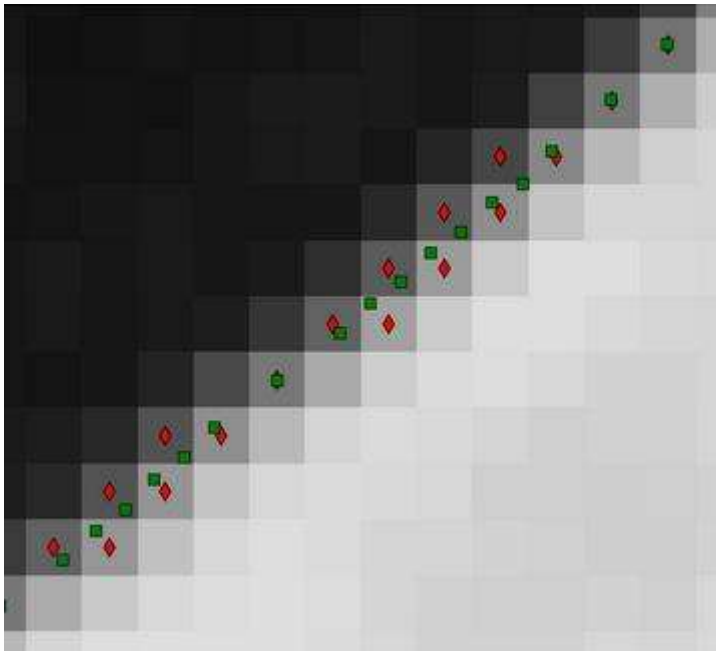


Abbildung 3.10: Subpixelgenauer Canny-Algorithmus. Das Grauwertbild zeigt eine lineare Kante. Die roten Diamanten zeigen an, wo sich ein Nulldurchgang befindet und die grünen Quadrate markieren die subpixelgenauen Stellen (Nulldurchgänge der zweiten Ableitung in der Gradientenrichtung).

3.5 Vorgeschlagene Ansätze

Bei der Entwicklung wurde zuerst die lineare Interpolation evaluiert. Eine reine lineare Interpolation benötigt die wenigsten Stützpunkte (maximal vier für ein zweidimensionales Bild). Sie hat allerdings den Nachteil, dass sie sich schlecht der konvexen Morphologie anpasst.

Die folgenden Verfahren benutzen die pixelgenauen Punkte und deren unmittelbar benachbarten Punkte (wenn der Index des pixelgenauen Punktes 0 ist, dann sind die Punkte mit dem Index 1 und -1 die gewählten Nachbarn), um die Subpixelgenauigkeit zu erreichen. Wenn diese drei Punkte in Betracht gezogen werden, erhält man für jeden der folgenden Verfahren eine Verschiebung zum wahren Maximum dessen Wert sich zwischen -0,5 und +0,5 befindet. Dieser Wert ist für die Auswertung des Verfahrens hilfreich, da er den Fehler repräsentiert.

3.5.1 Parabelinterpolation

Die Parabelinterpolation wurde als adäquate Methode identifiziert, um konvexe Verläufe abzubilden. Es ist eine einfache quadratische Interpolation, die einmal für jede Achse ausgeführt wird. Für eine solche Interpolation werden pro pixelgenauen Punkt drei Stützpunkte benötigt.

Für jede pixelgenaue Position werden das Grauwertmaximum und die Werte der zwei unmittelbar benachbarten Punkte (drei Punkte mit gleichem Abstand zueinander; in Abbildung 3.11 die Punkte *U1*, *U2* und *U3*) gespeichert, so dass auf sie zugegriffen werden kann.

Die drei Werte werden auf einer Parabel abgebildet und die Werte der unmittelbar benachbarten Punkte als Grenzen des Grauwertbereiches angenommen. Aus dem Grauwertmaximum und den zwei oben beschriebenen Grenzpunkten lässt sich durch Fitting einer Parabel Subpixelgenauigkeit erzielen.

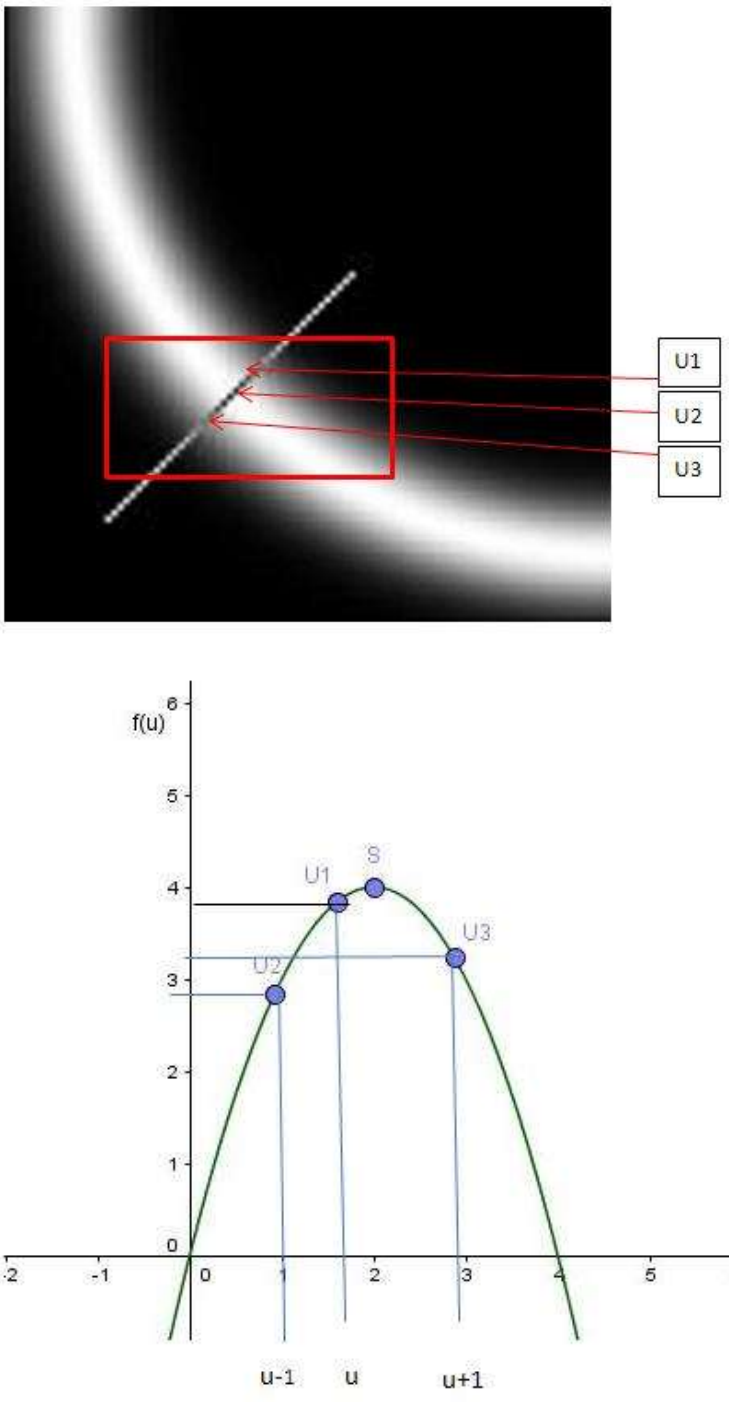


Abbildung 3.11: Quadratische Interpolation mit drei Stützpunkten(U1,U2,U3)

S stellt hier den geschätzten Scheitelpunkt der Parabel dar.

Die Koordinaten des Maximums werden als Koordinaten der Verschiebung genommen. Da es sich um einen pixelgenauen Punkt handelt, sind die Grauwerte damit immer ganzzahlig. Häufig wird in der Praxis an die Punkte in der Nähe des Maximums eine Funktion gefittet (hier eine Parabel). Anschließend wird das Maximum dieser Funktion bestimmt. Seine Koordinaten sind hierbei nicht mehr ganzzahlig.
Angenommen die Gleichung der Parabel laute: $f(U) = aU^2 + bU + c$.

Aus dieser Gleichung und den Stützpunkten lassen sich die Koeffizienten *a, b* und *c*

mit dem Gleichungssystem: $\begin{Bmatrix} f(U1) = aU1^2 + bU1 + c \\ f(U2) = aU2^2 + bU2 + c \\ f(U3) = aU3^2 + bU3 + c \end{Bmatrix}$ bestimmen.

Nach der Berechnung der Koeffizienten lassen sich diese in die oben genannte Ausgangsgleichung der Parabel einsetzen und die erste Ableitung der Funktion errechnen.

Die erste Ableitung lautet wie folgt: $f'(U) = 2aU + b$

Das Maximum befindet sich dort, wo die erste Ableitung gegen Null geht: $U_{subpixel} = -\frac{b}{2a}$

Der Ausschluss von Regionen, in denen keine plausiblen Messergebnisse erzeugt werden können, ist für die praktische Anwendbarkeit des Systems sehr wichtig. Die Differenz zwischen dem lokalen Maximum und dem wahren Maximum beträgt zwischen -0,5 und +0,5 und lässt sich durch folgende Formel berechnen:

$$x = 0{,}5 \cdot \left(\frac{f(U2) - f(U3)}{f(U2) - 2f(U1) + f(U3)}\right)$$

Dieses Verfahren ist mathematisch genau und liefert oft gute Ergebnisse, da die allgemeine Form der Interpolationsfunktion auch ohne Ableiten darstellbar ist (mit drei bekannten Punkten sind die Parameter berechenbar).

Das ist aber nur eine heuristisch begründete Herangehensweise. Signaltheoretisch ist es möglich, ein anderes, aber theoretisch begründetes Vorgehen anzugeben.

Im Buch „A class of Algorithms for Real-Time Subpixel Registration“[5] wurden verschiedene Interpolationsmethoden zur subpixelgenauen Registrierung verglichen. Die fourierbasierte trigonometrische Interpolation lieferte die besten Ergebnisse, allerdings steht dieser der hohe Rechenaufwand entgegen. Möchte man unterabgetastete Bilder mit Subpixelgenauigkeit registrieren, ist es notwendig, vorher ideale Tiefpässe anzuwenden.

3.5.2 Pyramideninterpolation

Ähnlich wie bei der Parabelinterpolation benötigt man für eine solche Interpolation pro pixelgenauen Punkt drei Stützpunkte. Neben deren pixelgenauen Positionen werden deren Grauwerte (die drei Punkte mit gleichem Abstand zueinander so wie bei der Parabelinterpolation) gespeichert, so dass auf sie zugegriffen werden kann.
Angenommen…

- die drei Punkte wären auf einer Pyramide abgebildet, so dass das wahre Maximum die Spitze der Pyramide ist
- dass der pixelgenaue Punkt ein lokales Maximum ist und
- die Steigung der Gerade zwischen den lokalen Maxima und einem der zwei benachbarten Punkte konstant wäre,

dann müsste man, um die Spitze der Pyramide zu lokalisieren, den Punkt finden, an dem sich die beiden Geraden, die zur Spitze der Pyramide führen, schneiden.

[5] R. W. Frischholz, K. P. Spinnler, “A class of Algorithms for Real-Time Subpixel Registration”, Europto Conference, München, Juni 1993

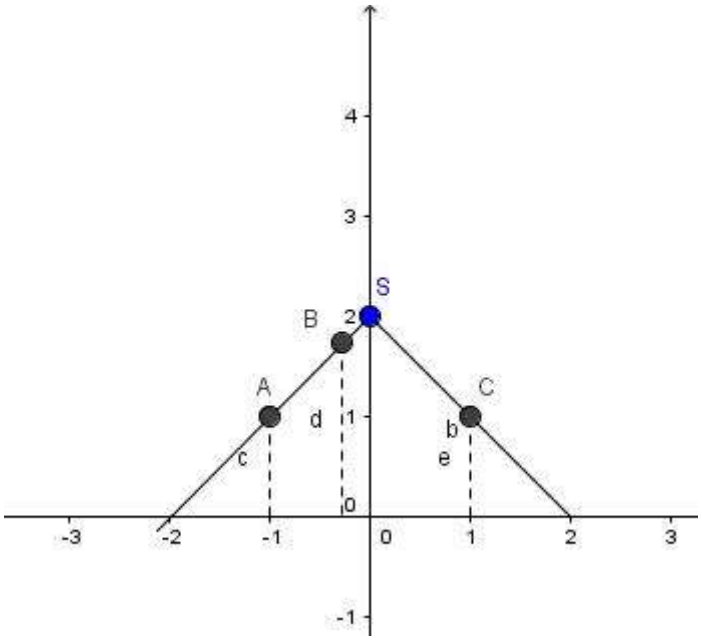


Abbildung 3.12: Pyramideninterpolation des lokalen Maximums

Wenn der Grauwert an der Stelle *C* größer ist, als der an der Stelle *A*, dann wird die Steigung der Geraden mit den Punkten *B* und *A* wie folgt berechnet:

$$Steigung = G_B - G_A \qquad G_n = Grauwert$$

Es ergibt sich daraus folgende Geradengleichung:

$$y = Steigung \cdot x + G_B$$

Die Gerade im ersten Quadranten des Koordinatensystems hat eine negative Steigung und seine Gleichung lässt sich aus der Steigung der ersten Formel und dem Grauwert des Punktes *C* ableiten. Die Formel lautet:

$$y = -Steigung \cdot x + (G_C + G_B - G_A)$$

Die zwei Geraden schneiden sich genau am wahren Maximum und die Verschiebung kann sich in diesem Fall durch die Gleichung $x = \frac{G_C - G_A}{2 \cdot (G_B - G_A)}$ berechnen lassen.

Für den Fall, dass der Grauwert an der Stelle *C* kleiner ist, als der an Stelle *A*, ändert sich die letzte Gleichung auf diese Weise: $x = \frac{G_{C-G_A}}{2\cdot(G_B-G_C)}$

Aus den gerade bekannten Formeln ergibt sich die allgemeine Gleichung:

$$x = \frac{G_C - G_A}{2 \cdot (G_B - Min(G_A, G_C))}$$

3.5.3 Sobel gefilterte Kante

Für die Lokalisierung der Kante in Subpixelgenauigkeit ist es wichtig, dass der Kantenerfassungsfilter keine Verschiebungen einführt. Der Sobel-Filter ist einer von diesen. Das folgende Bild stellt den Effekt der Anwendung des Sobels-Filters dar, um das wahre Maximum zu finden. Die Lokalisierung ist dann subpixelgenau.

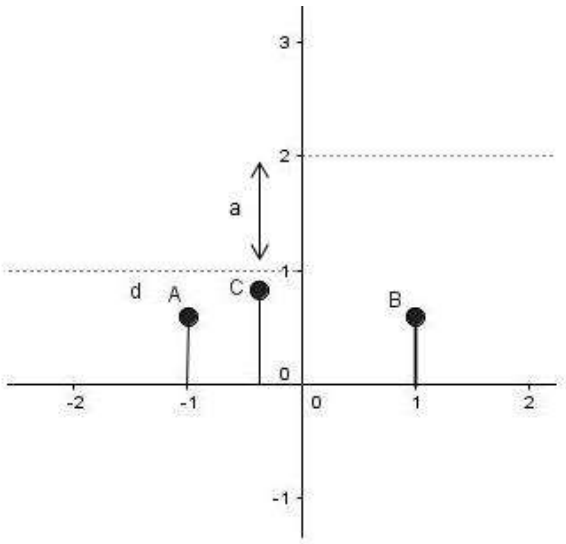


Abbildung 3.13: Anwendung des Sobels-Filters an der Kante

Mit Hilfe der Amplitude *a* werden die Pixelgrauwerte durch folgende Formel berechnet:

$$G_A = a(\frac{1}{2} - x) \qquad G_A; G_{B;} G_C = Grauwerte$$

$G_C = a$ *a=Amplitude*

$G_B = a(\frac{1}{2} + x)$

Nach *x* aufgelöst hat dies folgendes zur Folge:

$x = \frac{G_B - G_A}{2G_C}$ *x=Verschiebung zur wahren Kante*

4. Testroutinen und Auswertung

Im vorausgegangenen Kapitel wurden verschiedene Verfahren vorgeschlagen, um eine Subpixelgenauigkeit zu erreichen. In diesem Kapitel steht das Verfahren, das in Absatz 3.5 erklärt worden ist, im Fokus. Welche Software die an sie gestellten Anforderungen erfüllt, soll durch das Testen an verschiedenen Profilen nachgewiesen werden. Die Tests sollen unter bestimmten Testbedingungen eine Aussage darüber treffen, wie gut die subpixelgenauen Konturen erkannt werden können. Es soll einerseits nachgewiesen werden, dass eine Subpixelgenauigkeit bei der Konturidentifikation mit den verschiedenen Verfahren erzielt werden kann und andererseits soll durch die Testergebnisse das bessere Verfahren ermittelt werden.

4.1 Testumgebung

Die für den Test ausgesuchten Konturen bestehen zum größten Teil aus Aufnahmen von per Hand hergestellten Kreisen und Geraden. Mit einer Nikon COOLPIX S9400 wurden die Bilder aufgenommen und danach zu Testzwecken importiert.

4.2 Ergebnisse

In diesem Abschnitt werden die bei der Anwendung der verschiedenen Subpixelverfahren entstehenden Fehler ermittelt. Hierfür werden folgende Schritte durchgeführt:

1. Die Liste der pixelgenauen Punkte laden.
2. Die Grauwerte der Nachbarschaft in der Gradientenrichtung mit Hilfe der bilinearen Interpolation ermitteln.
3. Die lokalen Maxima mit Hilfe der Non-Maxima Suppression verbessern.
4. Die Fehler der Subpixelverfahren werden mit folgenden Formeln berechnet:

- Für die Parabelinterpolation Formel $x = 0{,}5 \cdot (\frac{f(U2)-f(U3)}{f(U2)-2f(U1)+f(U3)})$ anwenden (s.S.33).

- Für die Pyramideninterpolation Formel $x = \frac{G_C-G_A}{2\cdot(G_B-Min(G_A,G_C))}$ anwenden (s.S.36).

- Für die Sobel-gefilterte-Kante Formel $x = \frac{G_B-G_A}{2G_C}$ anwenden (s.S.37).
- Für die Verfeinerung durch Approximation Formel $\delta d = \frac{\sum_{i=1}^{n} gi\cdot di}{\sum_{i=1}^{n} gi} = \frac{G_B-G_A}{G_A+G_B+G_C}$ anwenden (s.S.28).

4.2.1 Ergebnisse beim Kreis

Die folgende Tabelle ist ein Ausschnitt aus der Konturliste des Kreises nach der Non-Maxima Suppression. Die unmittelbaren Nachbarn der Konturen wurden als rechte und linke Nachbarn erfasst. Mit Hilfe der bilinearen Interpolation wurden die Grauwerte ermittelt.

Die Verschiebung zum wahren Maximum nach der Parabelinterpolation wurde unter Δx_para, die der Pyramideninterpolation unter Δx_pyra und die nach dem Verfahren der Sobel-gefilterten-Kante Δx_sobel dokumentiert.

Konturpunkt	Grauwert	Linker Nachbar	Grauwert	Rechter Nachbar	Grauwert
393; 239	155	392,563; 238,937	153,626	393,437; 239,063	153,992
393; 193	146	392,52; 193,02	145,992	393,48; 192,98	145,98
391; 179	148	390,629; 179,129	147,871	391,371; 178,871	147,998
389; 169	149	388,617; 169,117	148,991	389,383; 168,883	148,999
387; 161	148	386,614; 161,114	147,995	387,386; 160,886	147,991
385; 153	148	384,162; 153,112	147,998	385,388; 152,888	147,997
383; 148	148	382,654; 148,154	147,997	383,346; 147,846	147,308
379; 137	148	378,618; 137,118	147,235	379,382; 136,882	147,998
377; 132	147	376,631; 132,131	146,9901	377,369; 131,869	146,262
371; 120	148	370,671; 120,171	147,98	371,329; 119,829	147,829
369; 116	147	368,667; 116,167	146,968	369,333; 115,833	146,998

Tabelle 4.1: Konturpunkte des Kreises mit den dazugehörigen Nachbarn und den Grauwerten

f1	f2	f3	Δx_para	Δx_pyra	Δx_sobel
155	153,626	153,992	0,0768262	0,13318777	0,004461271
146	145,992	145,98	-0,21428571	-0,3	0,000027
148	147,871	147,998	0,48473282	0,49224806	0,000435817
149	148,991	148,999	0,4	0,44444444	0,00003
148	147,995	147,991	-0,14285714	-0,22222222	0,000017
148	147,998	147,997	-0,1	-0,16666667	0,0000068
148	147,997	147,308	-0,49568345	-0,49783237	0,00001
148	147,235	147,998	0,49739244	0,49869281	0,002584494
147	146,9901	146,262	-0,48676294	-0,49329268	0,000034
148	147,98	147,829	-0,39528796	-0,44152047	0,000068
147	146,968	146,998	0,44117647	0,46875	0,000108845

Tabelle 4.2: Grauwerte des pixelgenauen Konturpunktes und dessen Nachbarn im Zusammenhang mit den Ergebnissen der verschiedenen Verfahren

f1 stellt den Grauwert des Konturpunktes, *f2* den Grauwert des linken Nachbarn und *f3* den Grauwert des rechten Nachbarn dar.

Um die Gültigkeit der Lösung zu überprüfen, wird nach einem Kreis gesucht, der möglichst genau durch die Punkte verläuft. Die Gleichung für den Ausgleichkreis mit Mittelpunkt (x_m, y_m) und Radius *r* lautet:

$$(x - x_m)^2 + (y - y_m)^2 = r^2$$

Für jeden pixelgenauen Messpunkt lässt sich eine lineare Gleichung für die Unbekannten (x_m, y_m) und den Radius *r* aufstellen. Da im Allgemeinen mehr als drei Messungen durchgeführt werden, sind mehr Gleichungen als Unbekannte vorhanden; die lineare Gleichung ist überbestimmt. Wenn die Koordinaten des jeweiligen Subpixels in der Gleichung eingesetzt werden bekommt man ein Radius $r_{subpixel}$.

Der Abstand des Subpixels zum Kreis beträgt: $\Delta d = r_{subpixel} - r_{Ausgleichgerade}$

Es ist erwünscht, dass die Abstände der gemessenen Punkte zum ausgeglichenen Kreis minimiert werden. Es wird anschließend geprüft, ob sich die Abstände der Subpixelpunkte zum Kreisbogen im Bereich [-0,5; 0,5] befinden. Wenn dies der Fall ist, lässt sich darauf schließen, dass sich die Auflösung verbessert hat. Das Vorzeichen den Abstandsberechnungen zeigt, wo die jeweiligen Subpixel liegen (innerhalb oder außerhalb des Kreises). In der folgenden Tabelle werden die Abstände dokumentiert:

Abstände nach Parabelinterpolation	Abstände nach Pyramideninterpolation	Abstände nach Sobel
0,090471959	0,153329977	0,009783421
-0,578265787	-0,654323117	-0,387905133
-0,010064696	-0,003956077	-0,402925174
-0,064067389	-0,0306044	-0,364580016
-0,448477927	-0,503945844	-0,34850479
-0,060439538	-0,103672458	0,004479415
-0,617310493	-0,618625308	-0,313025697
0,133549859	0,134258834	-0,135188718
-0,243628861	-0,24690512	0,001657591
-0,309110124	-0,328222031	-0,144884207
0,176024569	0,186814846	0,004367638

Tabelle 4.3: Abstände zum Ausgleichskreis nach den verschiedenen Subpixelverfahren

Aus der obigen Tabelle lässt sich entnehmen, dass die Mehrheit der Abstände zum Ausgleichskreis im Bereich [-0,5; 0,5] liegt und sich somit die Auflösung verbessert hat. Wenige Abweichungen lassen sich vermutlich mit Rauschen erklären. Alle drei verwendeten Verfahren bewiesen sich als zielführend. Das Sobel-Verfahren setzt sich jedoch gegenüber den übrigen Verfahren durch, da es die kleinsten Abweichungen aufweist und damit am genausten ist.
Zum Vergleich liegen folgende Ergebnisse bei der Approximation vor:

f1	f2	f3	Δx_approx	Abstand nach der Approximation
155	153,626	153,992	0,00079115	0,00648566
146	145,992	145,98	-2,7399E-05	-0,387157955
148	147,871	147,998	0,00028612	-0,402258709
149	148,991	148,999	1,7897E-05	-0,363809522
148	147,995	147,991	-9,0093E-06	-0,347751429
148	147,998	147,997	-2,2523E-06	0,00523584
148	147,997	147,308	-0,00155423	-0,313233454
148	147,235	147,998	0,00172144	-0,134916587
147	146,9901	146,262	-0,00165383	0,001534223
148	147,98	147,829	-0,00034024	-0,14434713
147	146,968	146,998	6,8032E-05	0,005050778

Tabelle 4.4: Verschiebung zum wahren Maximum und Abstände zum Ausgleichskreis nach der Verfeinerung durch Approximation

4.2.2 Ergebnisse bei einer Geraden

Die folgende Tabelle ist ein Ausschnitt der Konturpunkte einer Geraden nach der Non-Maxima Suppression. Wie bereits im vorangegangen Kapitel 4.2.1 über die Ergebnisse des Kreises, sind auch hier die unmittelbaren Nachbarn der Konturen als rechte und linke Nachbarn erfasst und die Grauwerte mit Hilfe der bilinearen Interpolation ermittelt.

Die Verschiebung zum wahren Maximum nach der Parabelinterpolation wurde unter Δx_para, die der Pyramideninterpolation unter Δx_pyra und die nach dem Verfahren der Sobel-gefilterten-Kante Δx_sobel dokumentiert.

Konturpunkt (x;y)	Grauwert	Linker Nachbar (x;y)	Grauwert	Rechter Nachbar (x;y)	Grauwert
375; 180	30	374,5; 181,5	28,75	375,5; 178,5	30
372; 179	31	371,337; 180,337	28,6747	372,663; 177,663	30,012
369; 177	32	368,403; 178,403	29,805	369,597; 175,597	31,2078
366; 176	32	365,415; 177,415	30,6582	366,585; 174,585	31,8307
351; 169	34	350,136; 170,136	31,3905	351,864; 167,864	32,4091
345; 165	35	344,418; 166,418	33,4179	345,582; 163,582	34,8358
318; 152	37	317,36; 153,36	33,9515	318,64; 150,64	35,7209
312; 149	37	311,194; 150,194	35,2317	312,806; 147,806	36,3881
300; 142	38	299,412; 143,412	36,8235	300,588; 140,588	35,8235

Tabelle 4.5: Konturpunkte der Gerade mit den dazugehörigen Nachbarn und den Grauwerten

f1	f2	f3	Δx_para	Δx_pyra	Δx_sobel
30	28,75	30	0,5	0,5	0,020833333
31	28,6747	30,012	0,201807865	0,287554294	0,038739504
32	29,805	31,2078	0,234801821	0,319544419	0,03516749
32	30,6582	31,8307	0,387962411	0,436913102	0,021077136
34	31,3905	32,4091	0,121250357	0,195171489	0,040258754
35	33,4179	34,8358	0,405972628	0,448106946	0,022707961
37	33,9515	35,7209	0,204432018	0,290208299	0,042671097
37	35,2317	36,3881	0,242920763	0,326980716	0,024297779
38	36,8235	35,8235	-0,14912019	0,35380058	0,016420785

Tabelle 4.6: Grauwerte der pixelgenauen Punkten und deren Nachbarn im Zusammenhang mit den Ergebnissen der verschiedenen Verfahren; f1=Grauwert des pixelgenauen Punktes; f2= Grauwert des linken Nachbarn; f3= Grauwert des rechten Nachbarn; Δx_para= Verschiebung nach der Parabelinterpolation; Δx_pyra=Verschiebung nach der Pyramideninterpolation; Δx_sobel= Verschiebung nach Sobel

Wie schon beim Kreis wird die Gültigkeit der Lösung überprüft, indem eine Ausgleichgerade gefunden wird, die durch die Punkte geht. Die Gerade lässt sich mathematisch durch die Funktionsgleichung $y = a \cdot x + b$ beschreiben (*a* die Steigung, *b* der y-Abschnitt). Man findet *a* und *b*, indem man die Koordinaten der Punkte in die allgemeine Funktionsgleichung einsetzt. Dadurch erhält man ein Gleichungssystem mit zwei Unbekannten, mit dem man *a* und *b* ermitteln kann. Der Abstand des jeweiligen Subpixels zur Geraden beträgt:

$$\Delta d = a * x_{subpixel} + b - y_{subpixel}$$

Ziel ist, die Abstände des jeweiligen Subpixels zur Ausgleichsgerade zu minimieren. Sind die Abstände nach dem Subpixelverfahren im Bereich [-0,5; 0,5], dann ergibt sich die Schlussfolgerung, dass sich die Auflösung verbessert hat. Die Ergebnisse werden in der folgenden Tabelle dokumentiert.

Abstände nach Parabelinterpolation	Abstände nach Pyramideninterpolation	Abstände nach Sobel
0,298611111	0,298611111	0,531539352
-0,098101046	-0,139783337	-0,018831703
0,344193559	0,302999241	0,441238026
-0,271926172	-0,295721647	-0,093579163
-0,850607812	-0,886541696	-0,811236895
-0,072347805	-0,092829766	0,113961408
-0,849376675	-0,891073479	-0,770742895
-0,951419815	-0,992282292	-0,845144754
0,072488982	-0,171986393	-0,007982326

Tabelle 4.7: Abstände zur Ausgleichsgerade nach den verschiedenen Subpixelverfahren

Wie auch beim Kreis zeigen die Ergebnisse, dass sich die Auflösung verbessert hat, da ein Großteil der Abstände zur Ausgleichgerade sich im Bereich [-0,5; 0,5] befindet. Lediglich einige Werte zeigen eine höhere Abweichung auf. Auch hier kann sich dies vermutlich auf Rauschen zurückführen lassen.
Alle drei angewendeten Verfahren haben sich als effektiv herausgestellt, ohne dass sich ein Verfahren von den anderen entscheidend durchsetzen konnte.
Zum Vergleich liegen folgende Ergebnisse bei der Approximation vor:

f1	f2	f3	Δx_approx	Abstand zur gerade nach Approximation
30	28,75	30	0,01408451	0,534820031
31	28,6747	30,012	0,0149108	-0,007248303
32	29,805	31,2078	0,0150818	0,451001905
32	30,6582	31,8307	0,01240886	-0,08936542
34	31,3905	32,4091	0,01041518	-0,796729599
35	33,4179	34,8358	0,0137322	0,118324627
37	33,9515	35,7209	0,01658723	-0,758063238
37	35,2317	36,3881	0,01064631	-0,838508623
38	36,8235	35,8235	-0,00903775	0,004393351

Tabelle 4.8: Verschiebung zum wahren Maximum und Abstände zur Ausgleichsgerade nach der Verfeinerung durch Approximation

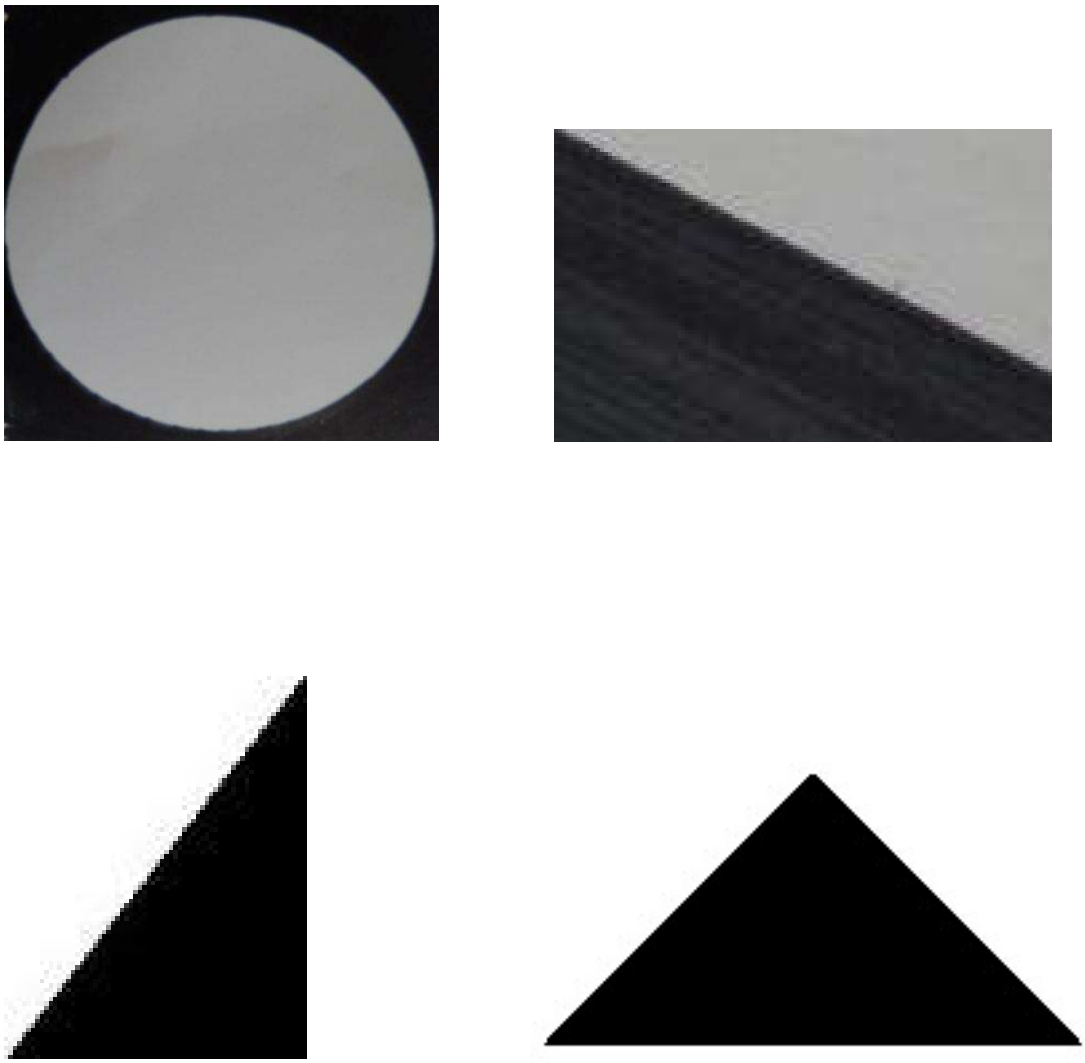


Abbildung 4.1: Testdatenbank: Kreis und Gerade, die für den Test benutzt worden sind

4.3 Auswertung

Nach der Anwendung der Subpixelverfahren war eine Abweichung des subpixelgenauen Punktes zum Ausgleichskreis bzw. zur Ausgleichsgerade im Bereich [-0,5; 0,5] erwartet. Dennoch wichen einige Kantenpunkte von diesem Bereich ab. Dadurch war es nicht möglich, eine sichere Bewertung der einzelnen Verfahren gegenüber ihrer Robustheit abzugeben. Mögliche Ursachen für die Abweichungen könnten Fehler bei der Aufnahme des Bildes oder Fehler in den Algorithmen sein. Im Rahmen dieser Arbeit war es nicht möglich, zu ermitteln, an welcher Stelle sich der Fehler befindet.
Ein Lösungsansatz wäre die Bilder für die Testzwecke unter industriellen Verhältnissen aufzunehmen (Beleuchtung, industrielle Kamera, hohe Bildqualität, Kameraposition,…). Dies würde garantieren, dass keine Fehler bei der Bildaufnahme auftreten. Sollten bei Testroutinen mit Bildern von dieser Qualität weiterhin Abweichungen auftreten, würde man dann darauf schließen, dass sich der Fehler im Algorithmus befindet.

Für den Fall, dass die Ergebnisse zufriedenstellend wären, wäre es möglich, die Subpixelverfahren auszuwerten.

5. Zusammenfassung

Eine Anzahl von Parametern, wie beispielsweise Lichtverhältnisse bei der Aufnahme des Bildes, Verschmutzungen oder die Position der Kamera, beeinflusst das gesamte Verfahren (von der Importierung des Bildes bis hin zum Vergleich der verschiedenen Konturenliste). Diese Parameter können nur bedingt erkannt werden. Dadurch können beim Messen Störkonturen und größere Ungenauigkeiten entstehen und damit die Qualität, mit der Konturen identifiziert werden, als auch die Robustheit der Subpixelverfahren beeinflusst werden. Ein ernstes praktisches Problem mit einem Kantendetektor ist die Frage nach der Wahl des Maßstabes der Glättung. Für viele Anwendungen ist es wünschenswert, in der Lage zu sein, ein Bild mit mehreren Skalen zu verarbeiten. Es lässt sich damit bestimmen, welche Kanten in Bezug auf den Bereich am wichtigsten sind.

Bei der zu Beginn durchzuführenden pixelgenauen Suche mit dem Gradienten-basierten Verfahren ist es hilfreich, wenn das Bild scharfe Intensitätsübergänge und niedriges Rauschen enthält. Bei der pixelgenauen Suche mit Wendepunkt-basierten Verfahren entgegen ist die Lokalisierung besser, wenn die Kanten nicht sehr scharf sind.

Die Non-Maxima Suppression erwies sich als ein wichtiger Zwischenschritt in den Subpixelverfahren, da man sonst zu große Differenzen in der Berechnung der Verschiebung zu den wahren Maxima bekommen hätte.

In dieser Arbeit wurden einige Methoden für die Subpixelgenaue Kantenerkennung vorgestellt. Diese Methoden verfeinern die zuvor pixelgenauen Kanten von Bildern auf Subpixelebene mit Hilfe der bilinearen Interpolation. Aber da der Grauwert eines Subpixels aus den Grauwerten seiner pixelgenauen Nachbarn bestimmt wird, kommt es manchmal zu einem Mangel an Details oder Nuancen, die dem Bild einen Eindruck von Unklarheiten geben.

Literaturverzeichnis

[1]BAILEY, Donald G.; Sub-pixel estimation of local extrema. In: Proceeding of Image and Vision Computing New Zealand. 2003. S. 414-419

[2]BERENSTEIN, Carlos A. et al.; A geometric approach to subpixel registration accuracy. Computer Vision, Graphics, and Image Processing, 1987, 40. Jg., Nr. 3, S. 334-360

[3]CANNY, John Francis; A Computational Approach to Edge Detection, IEEE Transactions on Pattern Analysis and Machine Intelligence, 1986

[4]Dawson-Howe, Kenneth; A Practical Introduction to Computer Vision with OpenCV, 2014.

[5]DEVERNAY, Frédéric; A fast and efficient subpixelic edge detector. In: Quatriemes Journees Orasis. CNRS, 1993

[6]DEVERNAY, Frédéric; A non-maxima suppression method for edge detection with sub-pixel accuracy. 1995

[7]FRISCHHOLZ, Robert W.; SPINNLER, Klaus P. ; Class of algorithms for real-time subpixel registration. In: Electronic Imaging Device Engineering. International Society for Optics and Photonics, 1993. S. 50-59

[8]GORMAN, Lawrence O.; Subpixel precision of straight-edged shapes for registration and measurement. Pattern Analysis and Machine Intelligence, IEEE Transactions on, 1996, 18. Jg., Nr. 7, S. 746-751

[9] HUERTAS, Andres; MEDIONI, Gerard; Detection of intensity changes with subpixel accuracy using Laplacian-Gaussian masks. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 1986, Nr. 5, S. 651-664

[10]HYDE, Peter D.; DAVIS, Larry S. ; Subpixel edge estimation. Pattern Recognition, 1983, 16. Jg., Nr. 4, S. 413-420

[11] JENSEN, Kris; ANASTASSIOU, Dimitris; Subpixel edge localization and the interpolation of still images. *Image Processing, IEEE Transactions on*, 1995, 4. Jg., Nr. 3, S. 285-295

[12]KOPLOWITZ, Jack; GRECO, Vito ; On the edge location error for local maximum and zero-crossing edge detectors. Pattern Analysis and Machine Intelligence, IEEE Transactions on, 1994, 16. Jg., Nr. 12, S. 1207-1212

[13] MARR, David; HILDRETH, Ellen; Theory of edge detection. Proceedings of the Royal Society of London B: Biological Sciences, 1980, 207. Jg., Nr. 1167, S. 187-217

[14]NALWA, Vishvjit S.; BINFORD, Thomas O; On detecting edges. Pattern Analysis and Machine Intelligence, IEEE Transactions on, 1986, Nr. 6, S. 699-714

[15]NOMURA, Yoshihiko et al. ; Edge location to subpixel precision and analysis. Systems and computers in Japan, 1991, 22. Jg., Nr. 9, S. 70-81

[16]PEDERSINI, Federico; SARTI, Augusto; TUBARO, Stefano; Estimation and compensation of subpixel edge localization error. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 1997, Nr. 11, S. 1278-1284.

[17] ROCKETT, Peter; The Accuracy of Sub-Pixel Localisation in the Canny Edge Detector. In: BMVC. 1999. S. 1-10

[18]STEGER, Carsten; Evaluation of subpixel line and edge detection precision and accuracy. International Archives of Photogrammetry and Remote Sensing, 1998, 32. Jg., S. 256-264

[19]STEGER, Carsten; Subpixel-precise extraction of lines and edges. International Archives of Photogrammetry and Remote Sensing, 2000, 33. Jg., Nr. 3, S. 141-156

[20]SUN, Qiucheng et al.; A robust edge detection method with sub-pixel accuracy. Optik-International Journal for Light and Electron Optics, 2014, 125. Jg., Nr. 14, S. 3449-3453

[21]SUZUKI, Satoshi et al.; Topological structural analysis of digitized binary images by border following. Computer Vision, Graphics, and Image Processing, 1985, 30. Jg., Nr. 1, S. 32-46

[22]ULUPINAR, Faith; MEDIONI, Gérard; Refining edges detected by a LoG operator. In: Computer Vision and Pattern Recognition, 1988. Proceedings CVPR'88., Computer Society Conference on. IEEE, 1988. S. 202-207

[23]YAO, Yuqin; JU, Hui; A sub-pixel edge detection method based on canny operator. In: Fuzzy Systems and Knowledge Discovery, 2009. FSKD'09. Sixth International Conference on. IEEE, 2009. S. 97-100

[24]YE, Jian; FU, Gongkang; POUDEL, Upendra P. ; High-accuracy edge detection with blurred edge model. Image and Vision Computing, 2005, 23. Jg., Nr. 5, S. 453-467

[25]YU, Zhi-jing et al.; Bilinear interpolation centroid algorithm used for circular optical target location. In: Second International Conference on Image and Graphics. International Society for Optics and Photonics, 2002. S. 333-339

Versicherung über Selbstständigkeit

Hiermit versichere ich, dass ich die vorliegende Arbeit ohne fremde Hilfe selbstständig verfasst und nur die angegebenen Hilfsmittel benutzt habe.

Hamburg, den _____ _____