Bildverarbeitung und Bildkorrektur mithilfe von Matlab

Assignment zum Modul:

2D-Bildverarbeitung (GDV40)

22.07.2019

Vladimir Zhelezarov

.....

Studiengang: Digital Engineering und angewandte Informatik - Bachelor of Engineering

AKAD University

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Einleitung

Die Bilder, die Kameras, Handys u.a., die digitale Geräte speichern, sind auf dem Speichermedium nichts anderes als eine Menge Zahlen, geordnet nach einem bestimmten Format. Diese Zahlen lassen sich auf dem Rechner elektronisch verarbeiten. Damit können Akzente im Bild hervorgehoben werden oder unerwünschte Artefakte unterdrückt werden. Mit diesen zwei typischen Problemen der Bildverarbeitung beschäftigt sich die vorliegende Arbeit. Dabei benutzen wir die Software Matlab zur Bearbeitung und Darstellung von diesen numerischen Aufgaben.

Nach der Klärung der wichtigsten Begriffe, die benutzt werden, ist die Arbeit in drei Hauptteile unterteilt:

Im ersten Teil werden die vorhandenen Bilder optisch und mathematisch analysiert mit dem Ziel benötigte Informationen für die Problemlösungen nachher bereitzustellen. Zusammenhänge müssen gefunden werden, die auch die Richtung der weiteren Verarbeitung bestimmen oder gar als sinnvoll beweisen.

Im zweiten Teil wird ein Bild so verarbeitet, dass die interessanten Informationen (in diesem Fall Text und Zeichen) besser sichtbar werden, wobei der Hintergrund unterdrückt wird. Bei dieser Segmentierung ist zuerst zu definieren, wie die Akzente numerisch festgestellt werden und wie diese im Bild zu finden sind. Danach versuchen wir die Unterschiede zwischen dem Text und dem Rest des Bildes zu finden, und nutzen diese, um eine Aufteilung zu erreichen.

Das Ziel des dritten Teils der Arbeit ist die Entfernung von unerwünschten Bildartefakten (in diesem Fall ein Salt & Pepper-Rauschen) von dem zweiten verfügbarem Bild. Dabei werden unterschiedliche Filter angewendet und die Ergebnisse verglichen. Wenn das beste Ergebnis gefunden ist, versuchen wir die dabei neu-aufgetauchten unerwünschten Artefakten, wie Verschmierung, aufzuheben, durch Anwendung von weiteren Filtern mit Kantenerkennung.

2 Grundlagen

Eine digitale Kamera fasst Licht und Farbe als elektronische Daten auf. Die Reihenfolge und Ordnung dieser Daten auf dem Datenträger wird als Format bezeichnet.

- TIFF-Format:

Zu den oft-benutzten Formaten zählt unter anderem das TIFF-Format - „ein *tag*-basiertes Dateiformat zum Speichern und Austausch von Rasterbildern“[1] entwickelt ursprünglich von Aldus Corporation und heute im Besitz von Adobe Inc.

- Farbendarstellung:

Jedes Pixel, oder anders gesagt - darstellbarer Punkt des Bildes, wird durch seine Luminanz und bei farbiger Darstellung - auch durch seine Chrominanz beschrieben. Bei der farbigen Darstellung wird zusätzlich zwischen Farbkodierung, wie z.B. RGB oder CMYK unterschieden. Bei RGB Kodierung wird die Farbe in drei Kanäle zerlegt und so durch drei Zahlen dargestellt: Red, Green und Blue; bei CMYK - in vier Kanäle: Cyan, Magenta, Yellow, Black [2].

- Histogramm:

Das so durch Zahlen dargestellte Bild lässt sich analysieren und dabei können Zusammenhänge gefunden werden.

Ein typisches Werkzeug der Analyse ist das Histogramm - eine Darstellung der statistischen Verteilung von Pixelwerten. Es bieten sich drei Arten von Histogrammen an[3]:

1. Das normalisierte Histogramm: *imhist*[4] gibt die Werte des Histogramms an. Um zu normalisieren, teilen wir durch die Anzahl der Pixel:

   ```
   hist_norm = imhist(img)/(img_width * img_height)
   ```

2. Das logarithmische Histogramm:

   ```
   hist_log = log(imhist(img)) / log(max(imhist(img))
   ```

3. Das kumulative Histogramm: Dafür kommt die Matlab Funktion *cumsum*[5] in Frage.

[1] Adobe Systems Inc. (1992), S.4
[2] Vgl. Schwanecke (o.J. a), S.57ff
[3] Vgl. Schwanecke (o.J. b), S.22f
[4] https://de.mathworks.com/help/images/ref/imhist.html
[5] https://de.mathworks.com/help/matlab/ref/cumsum.html

Um eine bessere Übersicht zu verschaffen, können wir die drei Arten von Histogrammen in einem Diagramm kombinieren. Für eine 8 Bit Farbtiefe ist die Abszisse durch den Werten 0..255[1] bestimmt. Bei der Ordinate skalieren wir die Ergebnisse von Null bis zur höchsten relativen Häufigkeit vom normalisierten Histogramm:

```
hist_log = hist_log * max(hist_norm)
hist_cum = hist_cum * max(hist_norm)
```

- Segmentierung und Rauschentfernung:

Wenn durch die optische und digitale Analyse Zusammenhänge gefunden sind (z.B. was alle interessanten Objekte im Bild gemeinsam haben), kann dies genutzt werden, um diese ausgewählten Objekte zu selektieren und zum Vordergrund zu bringen. Ähnlich, wenn unerwünschte Artefakten erkannt sind, kann mithilfe von verschiedenen mathematischen Algorithmen versucht werden, das Bild zu korrigieren. Dabei wird meistens die im Bild fehlende oder verfälschte Information aus den benachbarten Pixel erratet[2].

- Gammawert und Korrektur:

Oftmals bildet eine Bearbeitung, wie z.B. die *imadjust*[3] Funktion, die Eingangswerte in Ausgangswerte linear ab. Das heißt, dass wenn am Eingang ein Wert verdoppelt wird, sich auch der entsprechende Ausgangswert verdoppelt. Diese Beziehung ist aber nicht immer erwünscht und lässt sich durch die s.g. Gammakorrektur ändern[4]. Wenn z.B. $0 < gamma < 1$ tendiert die Abbildung zu den helleren Werten, sonst bei $gamma > 1$ - zu den dunkleren.

- Glätten und Schärfen:

Wenn der Wert eines Pixels aus den Werten der Umgebung approximiert wird, führt das zu einer Reduzierung der Spitzenwerte im Bild, also zur Glättung. Andersherum, durch die Anwendung von geeigneten Filtern, die Kanten erkennen und sie hervorbringen, kann das Bild verschärft werden. In dieser Arbeit wird als Kantenerkennung der Laplace Filter benutzt[5].

[1] Die Zahl 255 ergibt sich aus den 8 Bits Farbtiefe pro Pixel: $2^8 - 1 = 255$, also 255 mögliche Werte zzgl. die Null.
[2] Vgl. https://de.mathworks.com/help/images/what-is-image-filtering-in-the-spatial-domain.html
[3] https://de.mathworks.com/help/images/ref/imadjust.html
[4] Vgl. https://de.mathworks.com/help/images/gamma-correction.html
[5] Schwanecke (o.J. b), S.44

3 Bildanalyse

3.1 Vorhandene Bilder

Es liegen zwei Bild-Dateien im TIFF-Format vor:

Abbildung 1: Bild zum Segmentieren

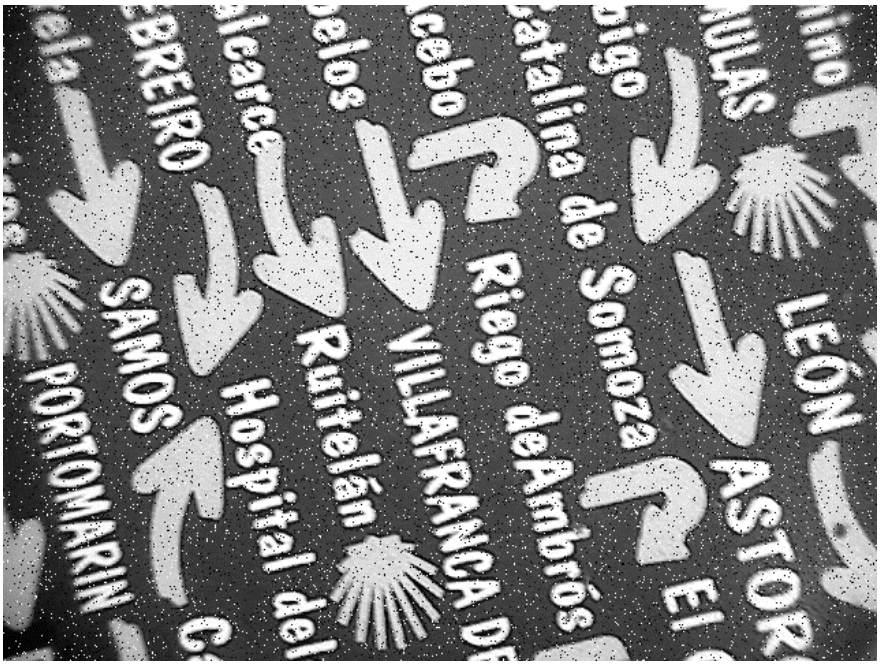

Abbildung 2: Bild zum Entrauschen

3.2 Bildparameter und Importieren in Matlab

Im Matlab zeigt *imfinfo*[1] alle Tiff-Parameter vom jeweiligen Bild. Die für die spätere Bildverarbeitung interessanten Informationen sind:

- Abmessungen (in Pixel): *Width*: 640, *Height* : 480. Gleich für die beiden Bilder;
- *PhotometricInterpretation RGB* : Die beiden Bilder sind im RGB Farbmodell gespeichert;
- *SamplesPerPixel 4* und *BitsPerSample 8,8,8,8* kombiniert mit *ExtraSamples 2* bedeuten, dass ein vierter Byte pro Pixel vorhanden ist, der Information über dem Alpha-Kanal (Transparenz) beinhaltet[2].

Es lässt sich in Matlab leicht erkennen, dass bei den beiden Bildern dieser extra Byte immer 255 beträgt, also keine nützliche Information erhält - z.B. durch ein Import über *readRGBAImage*[3]:

```
t = Tiff('bild1.tiff','r');
[img1,img1_alpha] = readRGBAImage(t);
sum(sum(img1_alpha~=255))

ans =
     0
```

Daher ist es um die weitere Bearbeitung zu vereinfachen sinnvoll, diesen extra Byte einfach zu ignorieren. Eine Option ist wie schon gezeigt mit *readRGBAImage*, eine andere ist mit *imread*:

```
img1 = imread('bild1.tiff');
img1 = img1(:,:,1:3);          % Den vierten Kanal ignorieren
```

Weiterhin sieht das zweite Bild optisch wie Graustufen aus. Das lässt sich leicht mit *isequal*[4] beweisen:

```
isequal(img2(:,:,1), img2(:,:,2), img2(:,:,3))

ans =
     logical
     1
```

[1] https://de.mathworks.com/help/matlab/ref/imfinfo.html, (Zugriff am 22.07.2019)
[2] Adobe Systems Inc. (1992), S.32
[3] https://de.mathworks.com/help/matlab/ref/tiff.readrgbaimage.html, (Zugriff am 22.07.2019)
[4] https://de.mathworks.com/help/matlab/ref/isequal.html, (Zugriff am 22.07.2019)

Für die weitere Bildverarbeitung wandeln wir die beiden Bilder in Graustufen um. Dabei wird nur bei Bild 1 Information verloren. Dabei kommt die Matlab Funktion *rgb2gray*[1] in Frage.

Für manche Operationen, vor allem bei der Division, müssen die Werte in eine andere, präzisere Darstellung umgewandelt werden, um Rundungsfehler zu vermeiden. Der Typ *double* ist eine Option. Dafür bietet Matlab den Befehl *im2double*[2] an.

3.3 Histogramme

Aus den gezeigten Histogrammen an der Verteilung der Werte lässt sich erkennen, dass bei beiden Bildern ein guter Kontrast und Helligkeit vorliegt, allerdings sind bei Bild 2 die Einzelwerte der Farben wesentlich weniger. Um sie zu berechnen, können wir die Zahl von vorhandenen Werten (*unique*[3]) zusammenrechnen (*numel*[4]):

```
numel(unique(img1_g))
numel(unique(img2_g))
```

So erkennen wir genau wieviele verschiedene Farbwerte vorhanden sind: jeweils 256 für Bild 1 und nur 16 für Bild 2.

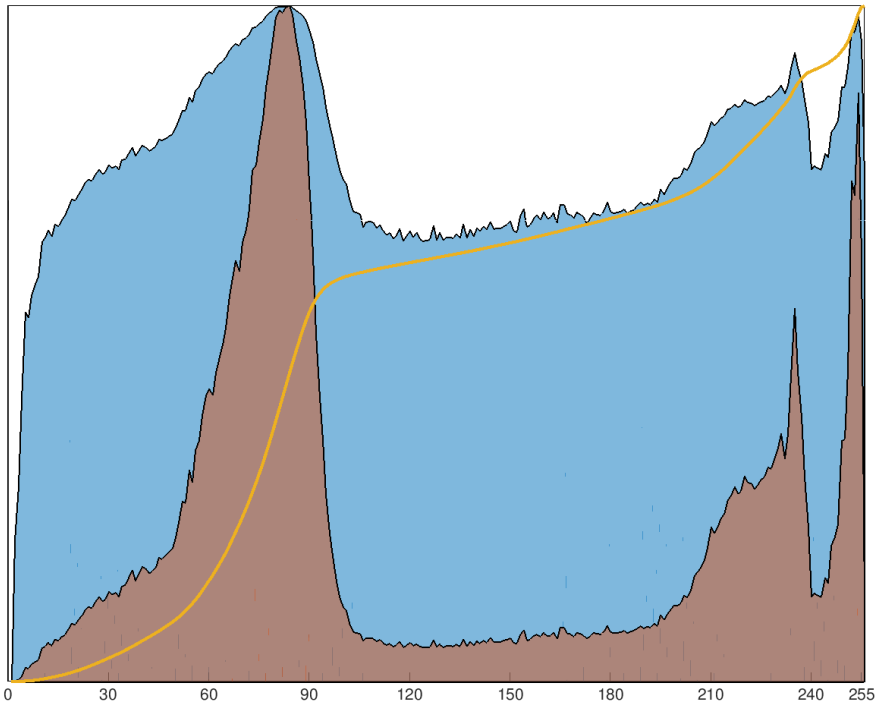

Abbildung 3: Kombiniertes Histogramm Bild 1

[1] https://de.mathworks.com/help/matlab/ref/rgb2gray.html
[2] https://de.mathworks.com/help/matlab/ref/im2double.html
[3] https://de.mathworks.com/help/matlab/ref/unique.html
[4] https://de.mathworks.com/help/matlab/ref/numel.html

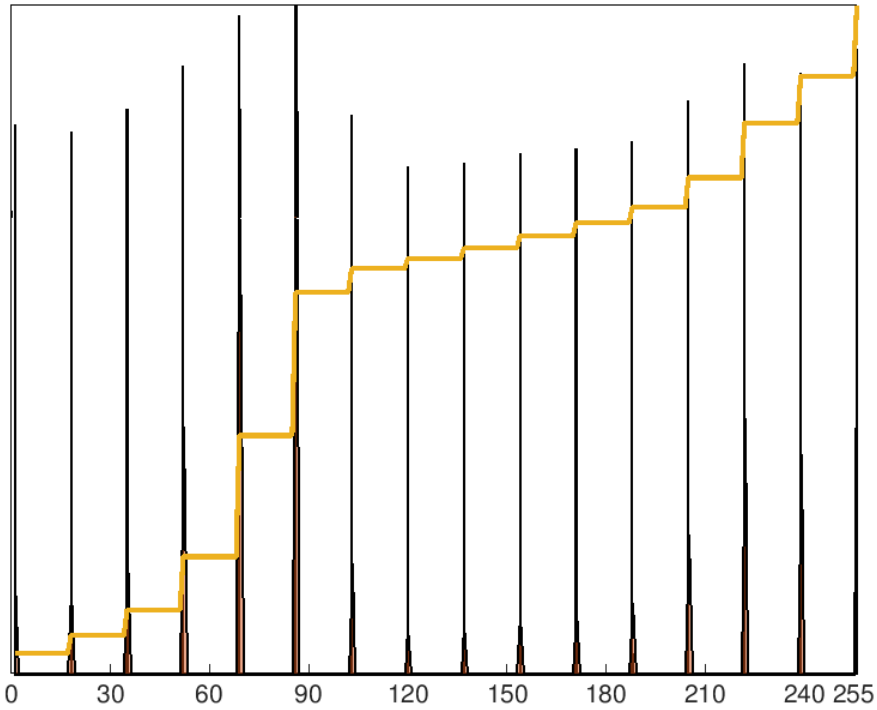

Abbildung 4: Kombiniertes Histogramm Bild 2

4 Segmentierung

Durch die optische Analyse des ersten Bildes nach der Umwandlung in Graustufen kann man schon gut die zwei Objekte erkennen:

- Objekt 1: heller Text, inklusive Zeichen wie Pfeile und
- Objekt 2: dunklerer Hintergrund

Mithilfe vom Histogramm können wir den Objekten Bereiche von der Farbverteilung zuweisen. So gehört offensichtlich der erste Bereich in der Gegend von 60-90 zum Hintergrund, und alles im rechten Teil des Histogramms - zum Text.

Durch *imadjust* kann ein Bereich des Bildes übernommen werden und als ein neues Bild abgebildet werden. In unserem Fall bietet sich auch an, das neue Bild gleich zu invertieren, um eine gewöhnliche Schwarz-auf-Weiß-Kombination zu erreichen. Das Ergebnis kann sofort mit *imshow* visualisiert werden:

```
start_color = 140;
imshow(imadjust(img1_g, [start_color/255 255/255], [255/255 0/255]));
```

Der Parameter *start_color* spielt dabei die entscheidende Rolle, da er bestimmt, wieviel vom ursprünglichen Bild übernommen wird. Wenn diese Zahl zu groß ist, fehlen teilweise farbige Pixel vom Text oder von den Zeichen. Wenn aber die Zahl zu klein ist, werden

auch Teile vom Hintergrund übernommen. Beides ist unerwünscht. Durch Experimentieren erkennen wir den Wert 140 als optimal.

Nach dieser Transformation erscheint das Schwarze an den Ecken schwach und ausgewaschen. Dieser Defekt kann durch geeignete Auswahl von einem Gammawert-Korrektur-Parameter im *imadjust* behoben werden. Wenn der Parameter klein genug ist - in unserem Fall etwa bei 0,0001 - wird die *imadjust* Transformation so durchgeführt, dass die Eingangswerte auf genau zwei Ausgangswerten abgebildet werden - 0 und 255, also Schwarz und Weiß. So wird der größte Kontrast erreicht.

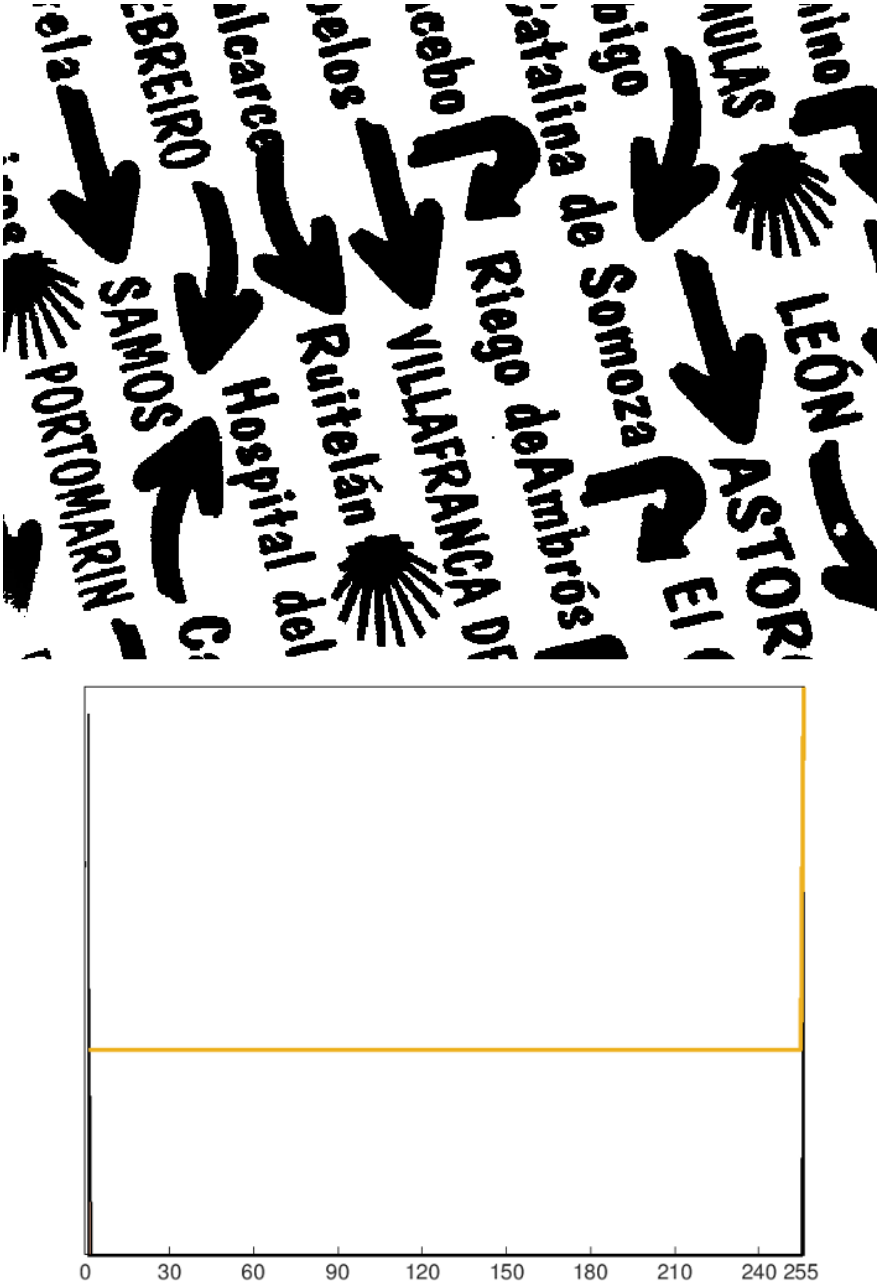

Abbildung 5: Bild 1 und Histogramme nach Segmentierung und Gammakorrektur

5 Rauschen-Entfernung

Das Rauschen auf dem zweiten Bild weist folgende Merkmale auf:

- Sporadisch verteilte kleine weiße Pixel beim dunkleren Hintergrund und
- Sporadisch verteilte kleine schwarze Pixel beim helleren Hintergrund.

Es sind praktisch Spitzwerte, die gleichmäßig verteilt sind und einen riesigen Unterschied von ihrer Umgebung haben. In diesem Fall bietet sich an, zu versuchen diese Pixel durch ihre Nachbarn zu approximieren. Das würde aber zu dem unerwünschten Nebeneffekt führen, dass die Kanten auch verschmiert werden, da eine Kante im Grunde auch ein Spitzwert ist, umgeben von weitaus verschiedenen Werten (wenigstens von der einen Seite). Diese negative Wirkung können wir nachher etwas minimieren, durch die Anwendung vom Laplace-Filter.

Als geeignete Filter für die Aufgabe können Mittelwert-, Gauß-, Minimum-, Maximum- und Medianfilter benutzt werden. Alle berechnen den Wert des Ausgangspixel aus der Umgebung vom Eingangspixel. Die Umgebung kann definiert sein als z.B. eine 3x3 oder 5x5 Matrix.

- Bei Mittelwert, wie schon der Name sagt, ist der Wert der algebraische Mittelwert von allen Pixeln in der Umgebung. Alle Pixel haben das gleiche Gewicht in der Berechnung;
- Beim Gaußfilter ist es ähnlich wie beim Mittelwert, nur dass die Gewichte einer Gaußverteilung folgen;
- Die drei Filter: Minimum, Maximum und Median arbeiten wie folgt: Die Pixel aus der Umgebung werden nach ihrem Wert sortiert. Der berechnete Wert für den Ausgangspixel ist dann: bei Minimum - der Mindestwert aus der Umgebung, bei Maximum - der Maximalwert und bei Median - der Wert in der Mitte der sortierten Liste.

Alle diese Filter unterdrücken das Rauschen mehr oder weniger, und das beste Rauschen-Unterdrückung/Kantenbewahrung-Verhältnis liefert der Medianfilter.

Jetzt kann die Schärfe des Bildes etwas verbessert werden. Dafür eignet sich der Laplace-Filter. Eine isotrope Matrix kann den iso-Kernel[1] anbieten:

```
lap_iso = [1 2 1; 2 -12 2; 1 2 1] /4;
img2_lap = imfilter(img2_med, lap_iso);
```

[1] Vgl. Schwanecke (o.J. b), S.45

Abbildung 6: Bild 2 nach dem Medianfilter

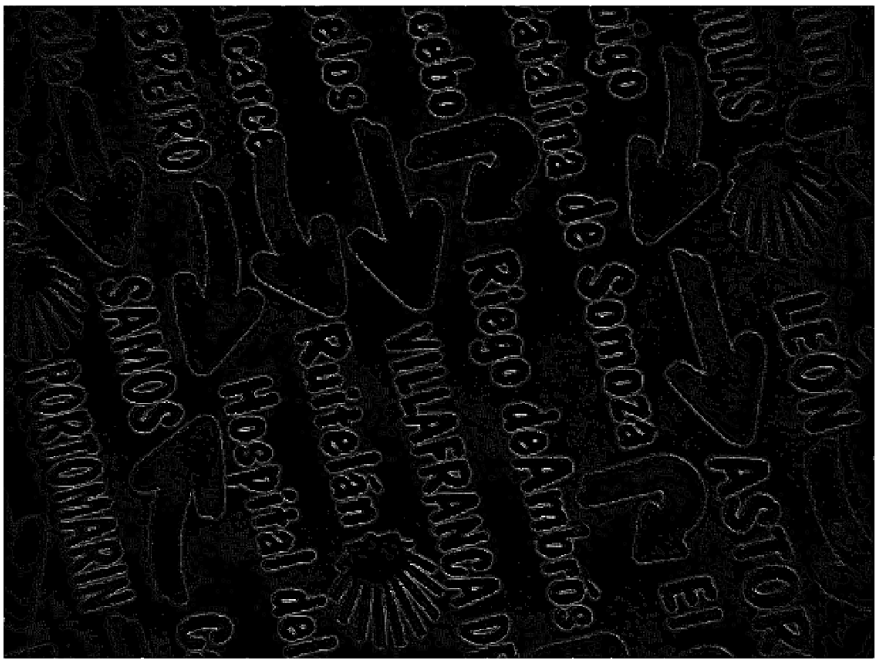

Abbildung 7: Bild 2 mit dem Laplace Kantenfilter

So hat der Laplace-Filter die Kanten hervorgerufen. Dieses Ergebnis können wir mit dem gefilterten Bild kombinieren. Die Kombination kann durch eine Subtraktion ablaufen:

```
img2_sub = imsubtract(img2_med, img2_lap);
```

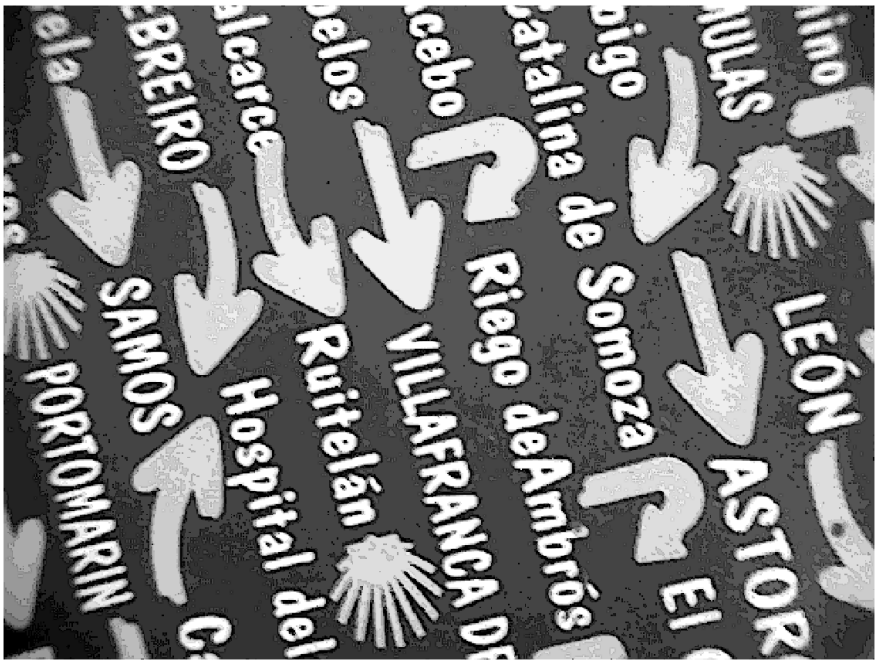

Abbildung 8: Bild 2 nach Filtern und Schärfen durch Subtraktion

Eine andere Möglichkeit, die Kanten-Korrektur anzuwenden, ist durch Division. Dabei ist allerdings zu beachten, dass die Bilder in *double* umgewandelt werden müssen, damit keine Rundungsfehler das Ergebnis verfälschen. Außerdem ist das Resultat von der Division durch einen geeigneten Parameter zu skalieren, damit es wieder im darstellbaren Bereich ist[1]. Zusätzlich muss der Divisor etwas geschoben werden, um Division durch Null zu vermeiden:

```
shift = 1; scale = 150;
img2_div = uint8(imdivide(im2double(img2_med), im2double(img2_lap) + 1) * scale);
```

[1]Vgl. Schwanecke (o.J. b), S.35

6 Zusammenfassung

Die digitale Bildverarbeitung bietet ein breites Spektrum an Möglichkeiten ein Bild nach der Aufnahme zu analysieren oder zu verbessern.

Als zwei typische Vertreter der Bildverarbeitung haben wir in dieser Arbeit das Hervorheben von Text und die Rauschunterdrückung gesehen.

Eine wichtige Voraussetzung dafür ist zuerst die Analyse. Die optische Analyse kann schon Ideen und Richtung für die weiteren Prozesse geben. Numerisch lässt sich das Bild auch untersuchen um für das Ziel wichtige Zusammenhänge zu finden. Wenn diese schon vorliegen, kann die gewünschte Bearbeitung erfolgen. Da die Bilder im Computer nur eine Menge von Zahlen sind, lassen sie sich gut bearbeiten.

Das Hervorheben von Text und Zeichen basiert auf den Unterschieden zwischen der interessanten Information und dem Hintergrund, der unterdrückt wird. Bei der Bearbeitung kann auch der Kontrast und die Helligkeit geändert werden. In unserem Fall haben wir die besten Ergebnisse mit dem klassischem Schwarz-auf-Weiss erreicht.

Wenn das Rauschen analysiert ist, im unseren Fall ergab die Analyse ein Salt-and-Pepper Rauschen, können verschiedene digitale Filter angesetzt werden um es möglichst vollständig zu entfernen. Bei den verschiedenen Operationen muss ein Gleichgewicht gefunden werden, zwischen Rauschunterdrückung und Bildverschmierung - zwei praktisch entgegengesetzte Effekte. Der Medianfilter zeigt bei der gegebenen Problemstellung die besten Ergebnisse. Um das Bild nach dem Filter etwas zu schärfen kann ein Laplace Kantenfilter benutzt werden und sein Effekt durch Subtraktion oder Division auf dem verschmierten Bild angewendet werden.

Bei beiden Operationen haben wir zuerst die Bilder in Graustufen umgewandelt. Dass das erste Bild farbig ist, hat zu einem Informationsverlust geführt. Das originale farbige Bild würde mehr Optionen anbieten um Objekte zu segmentieren. Zum Beispiel ist es möglich den Text und die Zeichen (Pfeile u.a.) zu separieren, da sie unterschiedliche Farben haben. Der Nachteil ist, dass dann mit drei Dimensionen (mit den drei Farben) gleichzeitig gearbeitet werden muss.

Bei der Rauschunterdrückung sind auch andere Filter oder Reihenfolgen von Filtern denkbar. Außerdem sind die Kernel für die Transformationen weitaus umfangreicher als die hier gezeigten 3x3 Einsen oder der 3x3 Iso Laplace Kernel.

Weil die Ergebnisse für unsere Ziele schon gut genug sind und um den Umfang der Arbeit in Grenzen zu halten, haben wir auf weitere Analyse oder weiteres Experimentieren mit einem Filter und Kernel verzichtet.

Anhang

Anhang 1: Matlab Code

```
clc; clear; close all;
cd '/MATLAB␣Drive'/GDV40/;

img1 = imread('bild1.tiff');
img2 = imread('bild2.tiff');
img1 = img1(:,:,1:3);
img2 = img2(:,:,1:3);
img1_g = rgb2gray(img1);
img2_g = rgb2gray(img2);
img1_gd = im2double(img1_g);
img2_gd = im2double(img2_g);

img_width = 640;
img_height = 480;
all_pixels = img_width * img_height;

%histograms
figure;
all_hist(img1_gd, all_pixels);
title('img1');

figure;
all_hist(img2_gd, all_pixels);
title('img2');

figure;
start_color = 140;
img1_reworked = imadjust(img1_g, [start_color/255 255/255], [255/255 0/255], 0.0001);
imshow(img1_reworked, []);
title('img1␣reworked');

figure;
all_hist(img1_reworked, all_pixels);
title('img1␣reworked');

%noise removal
h = ones(3,3)/9;
lap_iso = [1 2 1; 2 -12 2; 1 2 1] / 4;

img2_mean = imfilter(img2_g, h);
```

```
img2_gauss = imgaussfilt(img2_g, 1.5);
img2_min = ordfilt2(img2_g, 1, ones(3,3));
img2_max = ordfilt2(img2_g, 9, ones(3,3));
img2_minmax = ordfilt2(img2_min, 9, ones(3,3));
img2_med = medfilt2(img2_g); % same as: ordfilt2(img2_g, 5, ones(3,3));
img2_lap = imfilter(img2_med, lap_iso);

%enhance
img2_sub = imsubtract(img2_med, img2_lap);
shift = 1; scale = 150;
img2_div = uint8(imdivide(im2double(img2_med), im2double(img2_lap) + 1) * scale);

%show all
figure;
imshow(img2_gd, []);
title('Original');

figure;
imshow(img2_mean, []);
title('Mean value');

figure;
imshow(img2_gauss, []);
title('Gauss filter with sigma = 1.5');

figure;
imshow(img2_min, []);
title('Minimum filter');

figure;
imshow(img2_max, []);
title('Maximum filter');

figure;
imshow(img2_minmax, []);
title('Minimum + Maximum filter');

figure;
imshow(img2_med, []);
title('Median filter');

figure;
imshow(img2_lap, []);
title('Laplace over Median filter');
```

```
figure;
imshow(img2_sub, []);
title('Enhanced through subtraction');

figure;
imshow(img2_div, []);
title('Enhanced through division');

function all_hist(myimg, all_pixels)
    imh = imhist(myimg);
    cdf1 = cumsum(imh);
    hold on
    h_max = max(imh/all_pixels);
    axis([0 256 0 h_max]);
    rectangle('Position', [0 0 256 h_max]);
    xticks([0:30:240 255]);
    yticks([]);
    %plot logarithmic histogram:
    area(log(imh) * h_max/ log(max(imh)), 'FaceAlpha', 0.5, 'LineWidth', 1);
    %plot normalised histogram:
    area(imh/all_pixels, 'FaceAlpha', 0.5, 'LineWidth', 1);
    %plot cumulative histogram:
    plot(cdf1 * h_max/all_pixels, 'LineWidth', 2);
    %p3.Color(4) = 0.8;
    hold off;
    end

function m_img = img_norm(img)
    m_img = (uint8(mat2gray(img) * 255));
end
```

Anhang 2: Bilder Segmentierung

Unterschiedliche Werte für *start_color* und Gammakorrektur:

Abbildung 9: Bild 1 durch *imadjust* mit *start_wert = 100* vor der Gammakorrektur

Abbildung 10: Bild 1 durch *imadjust* mit *start_wert = 140* vor der Gammakorrektur

Abbildung 11: Bild 1 durch *imadjust* mit *start_wert = 180* vor der Gammakorrektur

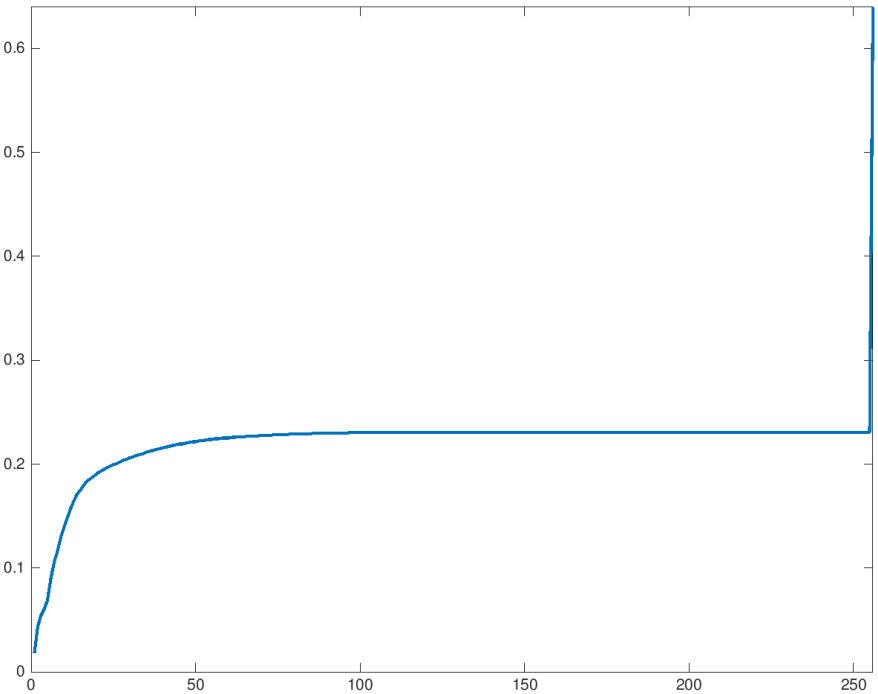

Abbildung 12: Log. Histogramm vom Bild 1 mit *start_wert = 140* und Gammakorrektur *0.1*

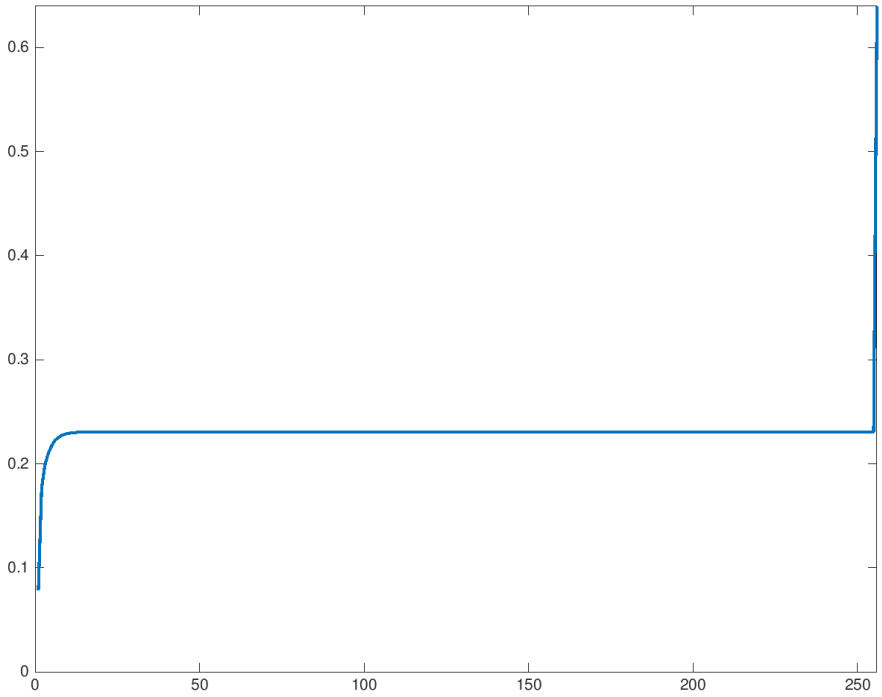

Abbildung 13: Log. Histogramm vom Bild 1 mit *start_wert = 140* und Gammakorrektur *0.01*

Anhang 3: Bilder Entrauschen

Ergebnis beim Bild 2 von unterschiedlichen Filter:

Abbildung 14: Bild 2 mit dem Mittelwert-Filter

Abbildung 15: Bild 2 mit dem Minimum-Filter

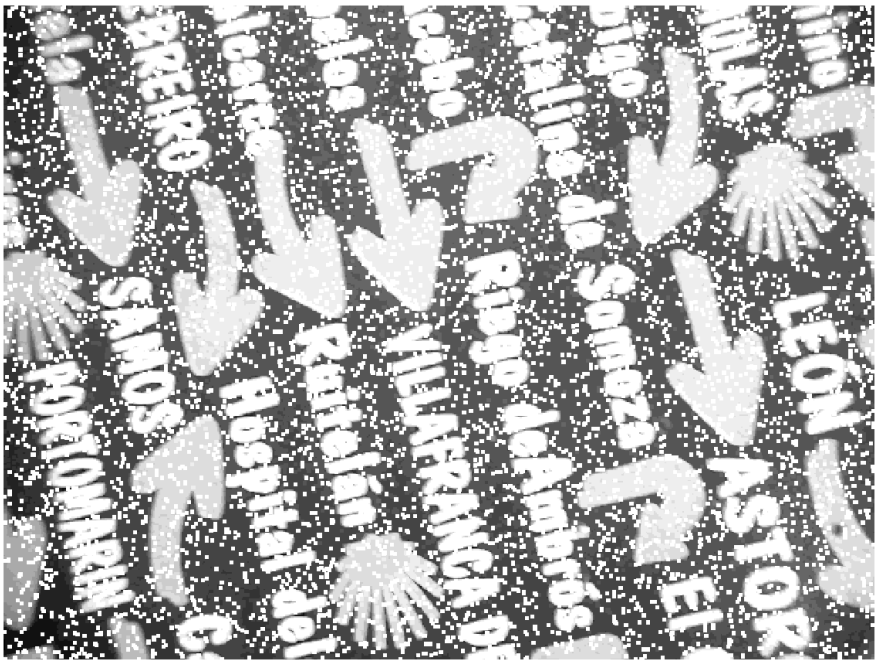

Abbildung 16: Bild 2 mit dem Maximum-Filter

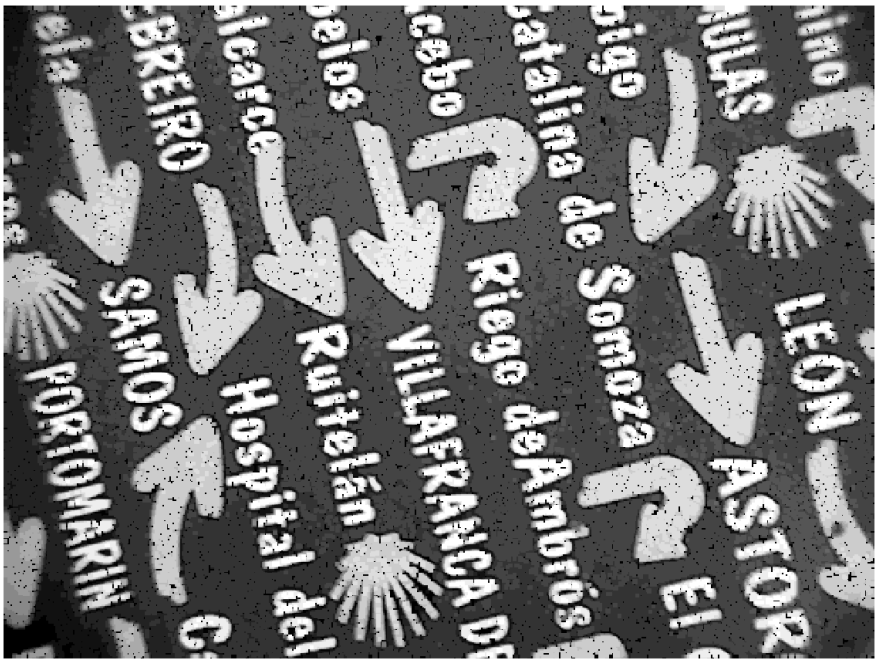

Abbildung 17: Bild 2 mit dem Minimum- gefolgt vom Maximum-Filter

Abbildung 18: Bild 2 mit dem Gauß-Filter und *sigma = 1.5*

Ergebnis von der Kantenschärfung mittels Division:

Abbildung 19: Bild 2 Kantenschärfung mittels Division

Literaturverzeichnis

Adobe Systems Inc. (1992)
TIFF Revision 6.0,
https://www.adobe.io/content/dam/udp/en/open/standards/tiff/TIFF6.pdf (Zugriff am 22.07.2019)

The MathWorks, Inc. (2019)
MATLAB Documentation,
https://de.mathworks.com/help/index.html (Zugriff am 22.07.2019)

Schwanecke, U. (o.J. a)
Industrielle Bildverarbeitung, AKAD Studienbrief ROB201, o.O.

Schwanecke, U. (o.J. b)
Methoden und Algorithmen der 2D-Bildverarbeitung, AKAD Studienbrief ROB202, o.O.