Formelsammlung Echtzeit Bildverarbeitung

Mario Felder

3. Juli 2014

Inhaltsverzeichnis

Kapitel 1

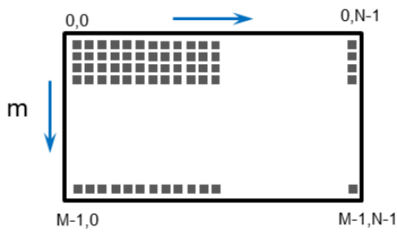
Einführung

Die Bilddaten können mathematisch als Matrix beschrieben werden:

$$I = [I_{m,n}]$$

mit $0 \leq m \leq M-1$ und $0 \leq n \leq N-1$

Achtung: MATLAB verwendet das Intervall $[1, N]$ bzw. $[1, M]$



1.1 Rasterung

Die Rasterung ist ein Mass für die Detailtreue eines Bildes. Bei gegebener CCD- bzw. CMOS-Sensorgrösse (M x N Pixel) wird die Auflösung durch die geometrische Abbildung bestimmt.

Dabei gelten folgende Zusammenhänge:

$$\frac{1}{g} + \frac{1}{b} = \frac{1}{f} \qquad \Leftrightarrow \qquad b = \frac{f \cdot g}{g - f}$$

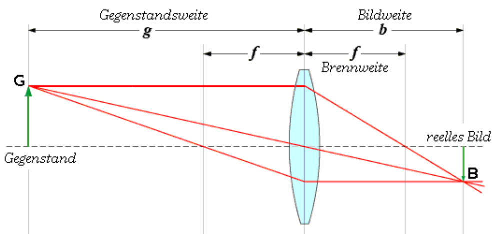
f: Brennweite
b: Bildweite
g: Gegenstandweite

Häufig ist $g \gg b$ und es kann somit $b = f$ gesetzt werden.

Weiter ergibt sich daraus:

$$\frac{B}{G} = \frac{b}{g}$$

B: Bildgrösse
G: Gegenstandsgrösse



Auflösung bei einem Sensor mit NxM Pixel Auflösung:

$$G_b = \frac{g \cdot W}{b \cdot N}$$
$$G_h = \frac{g \cdot H}{b \cdot M}$$

G_b : Auflösung in $\frac{mm}{Pixel}$ (Breite)
G_h : Auflösung in $\frac{mm}{Pixel}$ (Höhe)

W : Breite des Sensors $[mm]$
H : Höhe des Sensors $[mm]$
N : Horizontale Pixel
M : Vertikale Pixel

1.2 Nachbarschaftsrelationen und Abstand

Vierer-Nachbarschaft:

	$m-1, n$	
$m, n-1$	m, n	$m, n+1$
	$m+1, n$	

Achter-Nachbarschaft:

$m-1, n-1$	$m-1, n$	$m-1, n+1$
$m, n-1$	m, n	$m, n+1$
$m+1, n-1$	$m+1, n$	$m+1, n+1$

Euklidische Abstandsnorm:

$$d(I_{m,n}, I_{p,q}) = \sqrt{(m-p)^2 + (n-q)^2}$$

Maximum Abstandsorm:

$$d(I_{m,n}, I_{p,q}) = max\left(|m-p|, |n-q|\right)$$

1.3 Globale Charakterisierung von Bildern

1.3.1 Histogramm

Das Histogramm gibt die absolute oder relative $p_I(g)$ Häufigkeit aller Grauwerte $g \in [0, 255]$ eines Bildes an.

Relative Häufigkeit:

$$0 \leq p_I(g) \leq 1 \qquad , \forall g$$
$$\sum_g p_I(g) = 1$$

Kumulative Häufigkeit:

$$h_I(g) = \sum_{g' \leq g} p_I(g') \qquad , h_I(255) = 1$$

1.3.2 Mittelwert

$$\mu_I = \frac{1}{M \cdot N} \sum_{m,n} I_{m,n} = \sum_g g \cdot p_I(g)$$

1.3.3 Varianz

$$\sigma_I^2 = \frac{1}{M \cdot N} \sum_{m,n} (I_{m,n} - \mu_I)^2 = \sum_g (g - \mu_I)^2 \cdot p_I(g)$$

Lösung in MATLAB:

```
%read image
Image = imread('sample.png');

%using formula
[M, N] = size(Image);
mu = sum(Image(:))/(M*N)

sd = sum((double(Image(:)) - mu).^2)/(M*N);
sd = sqrt(sd)

```

```
%with MATLAB function
mu = mean2(Image)
sd = std2(Image)

%using the histogram
[Count, Value] = imhist(Image);
RelCount = Count/sum(Count);
mu = sum(Value .* RelCount)

sd = sum((Value - mu).^2 .* RelCount);
sd = sqrt(sd)
```

Lösung in C:

```
for(int m = 0; m < M; m++){
  for(int n = 0; n < N; n++){
    sum += Image[m][n];
  }
}
mu_I = sum/(M*N);

/**
*
* TODO
*
**/
```

Kapitel 2

Punktoperationen und Bildverknüpfungen

2.1 Transformationstabellen (LUT)

Jedem Grauwert $g \in G$ wird einen Grauwert $g' \in G'$ über eine Abbildung f zugeordnet:

$$f : G \rightarrow G' \qquad , f(g) = g'$$

Lösung in MATLAB:

```
%read image
Image = imread('sample.png');

%LUT for inverse image
LUT_Inv = uint8([255:-1:0]);

%apply LUT
ImageInv = intlut(Image, LUT_Inv);

%LUT for screener
LUT_Scr = uint8(1.5*[0:255]);

%apply LUT
```

```
ImageScr = intlut(Image, LUT_Scr);
```

2.2 Lineare und nichtlineare Grauwerttransformationen

Eine allgemeine lineare Grauwerttransformation lässt sich in folgender Notation schreiben:

$$f : [0, 255] \to [0, 255] \in \mathbb{R}$$

$$f(g) := \begin{cases} 0 & \text{,falls } (g + c_1) \cdot c_2 < 0 \\ 255 & \text{,falls } (g + c_1) \cdot c_2 > 255 \\ (g + c_1) \cdot c_2 & \text{,sonst}(c_1, c_2 \in \mathbb{R}) \end{cases}$$

2.2.1 Spreizung

Sind die Grauwerte eines Bildes auf das Intervall $[g_1, g_2]$ beschränkt, so kann durch Anwendung der linearen Grauwerttransformation als Spreizung die werte auf das gesamte Intervall $[0, 255]$ verteilt werden:

$$f : [0, 255] \to [0, 255] \in \mathbb{R}$$

$$f(g) := \begin{cases} 0 & \text{,falls } g < g_1 \\ 255 \cdot \frac{g - g_1}{g_2 - g_1} & \text{,falls } g \in [g_1, g_2]. \\ 255 & \text{,falls } g > g_2 \end{cases}$$

Lösung in MATLAB:

```
%read image
Image = imread('sample.png');

%LUT for spreading gray values
LUT_Spread = uint8(([0:255]-50)*2);

%apply LUT
ImageSpread = intlut(Image, LUT_Spread);
```

2.2.2 Gammakorrektur

Ein wichtiges Beispiel der nichtlinearen Grauwerttransfomration ist die Gammakorrektur. Der Ursprung dieser nitlinearen Korrektur der Grauwert liegt in der Nichtlinearität von Aufnahme- und Darstellungssystemen.:

$$f : [0, 255] \to [0, 255] \in \mathbb{R}$$

$$f(g) := 255 \cdot \left(\frac{g}{255}\right)^{\gamma}$$

Lösung in MATLAB:

```
%read image
Image = imread('sample.png');

%LUT for gamma correcture
LUT_Gamma = uint8(255*([0:255]/255).^0.5);

%apply LUT
ImageGammad = intlut(Image, LUT_Gamma);
```

2.3 Histogrammausgleich

Die Idee des Histogrammausgleichs ist, die Grauwerte so zu verteilen, dass jeder gleich häufig vorkommt. Dies kann allerdings nicht ganz erreicht werden, da die Grauwerte diskret sind. Näherungsweise kann die kumulierte Häufigkeit $h_I(g)$ herangezogen werden. Bei einer konstanten absoluten Häufigkeit, würde die kumulierte Häufigketi linear anwachsen.
Die entsprechende Transformation:

$$f : [0, 255] \to [0, 255] \in \mathbb{R}$$

$$f(g) := g_{max} \cdot \sum_{g'=0}^{g} p_I(g')$$

Lösung in MATLAB:

```
%read image
Image = imread('sample.png');

[Hist, Val] = imhist(Image);
CumHist = cumsum(Hist)/sum(Hist);

%LUT for histogram equalization
LUT_Equ = uint8(CumHist*255);
%apply LUT
ImageEqu = intlut(Image, LUT_Equ);
```

2.4 Arithemtische und logische Bildverknüpfungen

Während die Punktoperationen auf Einzelbildern vor allem der besseren optischen Darstellung von Bildern dienen, eröffnen Punktoperationen auf mehreren Bildern ein grosses Repertoire an Methoden, die schon erste einfache Computer Vision Anwendungen erlauben.

2.4.1 Differenzbildung zur Motiondetektion

Eine Methode zur Change Detektion basiert auf der Berechnung von Differenzbildern. Dabei wird vorausgegangen, dass eine Serie von Bildern als Funktion der Zeitstempel aufgnommen werden:

$$I(t) = I(t_k) = I(k \cdot \Delta t) = I_k$$

Differenzbild Berechnung:

$$\Delta I_{k+n} = \frac{1}{2} \cdot (255 + I_{k+n} - I_k) = \frac{1}{2} \cdot (255 + I((k+n) \cdot \Delta t) - I(k \cdot \Delta t)$$

Auf das Differenzbild kann noch eine Schwellwertioperation agewandt werden:

$$f : [0, 255] \to \{0, 255\}$$

$$f(g) := \begin{cases} 0 & \text{, falls } g < g_1 \vee g > g_2 \\ 255 & \text{, falls } g \in [g_1, g_2] \end{cases}$$

$$\text{mit} \quad g_1 = 128 - threshold, \quad g_2 = 128 + threshold$$

Lösung in MATLAB:

```
function [ThreshImage, DiffImage] =
    MotionDetektionFunct(ImageAct, ImageOld)

global Threshold

%this is the threshold value (chosen manually)
Threshold = 15;

%cast
ImageAct = double(ImageAct);
ImageOld = double(ImageOld);

%calcuate difference image
DiffImage = (255 + ImageAct - ImageOld)/2;

%calculate the threshold image
ThreshImage = (abs(DiffImage-128) > Threshold);

DiffImage = uint8(DiffImage);
```

2.4.2 Hintergrundschätzung durch gleitenden Mittelwert

Ziel ist es, eine möglichst exakte Hypothese des unbeweglichen Hintergrundes $B(t_k) = B(k \cdot \Delta t) = B_k$ der Bilder I_k zu ermitteln. Angenommen, dass der Hintergrund jedes Pixels mehrheitlich sichtbar ist, kann durch ein einfaches gleitendes Mittel eine brauchbare Hypothese B_k bestimmt werden:

$$B_k = \alpha \cdot B_{k-1} + (1 - \alpha) \cdot I_k \qquad , a \in]0, 1[$$

Eine plötzlich auftretende stationäre Änderung in der Bildfolge I_k ist nach $n \cdot \Delta t$ Zeitshcritten zu folgendem Anteil p in die Hintergrundhypothese B_k integriert:

$$p = 1 - \alpha^n$$

Lösung in MATLAB:

```
function [ThreshImage, DiffImage, BackGround] =
    GleitendesMittelFunct(ImageAct, BackGround, Params)

%make everything double
BackGround = double(BackGround);
ImageAct = double(ImageAct);

%calcuate forground estimate
DiffImage = abs(BackGround-ImageAct);
%estimate new background as sliding average
BackGround = Params.AvgFactor*BackGround+...
    (1-Params.AvgFactor)*ImageAct;

%calculate the threshold image
ThreshImage = DiffImage > Params.Threshold;
```

2.4.3 Hintergrundschätzung durch statistische Analyse

Die grundlegende Idee ist, für jedes Pixel individuell die Helligkeitsschwankungen zu messen und durch ein einfaches statistisches Modell zu approximieren. Letzteres besteht in einer Gauss'schen Approximation der Grauwertfluktuationen des Pixels durch das Paar aus Mittelwert und Varianz (μ, σ). Hierbei wird für jedes Pixel individuell die Schätzung von Mittelwert und Varianz über die Zeitschritte $k \cdot \Delta t$ folgendermassen durchgeführt:

$$\begin{aligned} \mu_k &= \alpha \cdot \mu_{k-1} + (1-\alpha) \cdot I_k \\ \sigma_k &= \alpha \cdot \sigma_{k-1} + (1-\alpha) \cdot (\mu_k - I_k)^T \cdot (\mu_k - I_k) \qquad , \alpha \in]0,1[\end{aligned}$$

Das Aufdatieren erfolgt nur, wenn sich der Mittelwert μ_k und aktueller Pixelwert I_k nur um weniger als $thr \cdot \sigma_k$ unterscheiden.

Kapitel 3

Filteroperatoren im Ortsraum

Bei diesen Filteroperationen werden für die Transformation enes Pixels auch die Werte der Nachbarpixel in Betracht gezogen.

$$f : G \to G'$$
$$f(g) = f_{I_{p,q}}(I_{m,n}) = g' \qquad , (p,q) \neq (m,n)$$

Klassifikationen der Filteroperatoren:

homogene Filter:
Die Berechnung der Transformation $f(g)$ wird für jedes Pixel unabhängig von dessen Position im Bild vorgenommen.

inhomogene Filter:
Die Berechnung der Transformation $f(g)$ hängt explizit von der Position des Pixels im Bild ab.

lineare Filter:
Die Berechnung der Transfomration $f(g)$ hängt für jedes Pixel linear ovn $g = I_{m,n}$ und den Werten der Nachbarpixel $I_{p,q}$ ab.

nicht lineare Filter:
Für die Berechnung der Transformation $f(g)$ eines Pixels werden die Werte von $g = I_{m,n}$ und der Nachbarpixel $I_{p,q}$ in nicht linearer Weise verknüpft.

3.1 Lineare Filter - Faltung

Mathematische Beschreibung der Faltung $I \otimes h$ eines Bildes $I_{m,n}$ mit einer Maske $h_{p,q}$:

$$I \otimes h : I_{m,n} \rightarrow \sum_{p=-u}^{u} \sum_{q=-v}^{v} I_{m-p,n-q} \cdot h_{p,q}$$

Rechengesetzte:

Kommutativität: $I \otimes J = J \otimes I$

Assoziativität: $(I \otimes J) \otimes K = I \otimes (J \otimes K)$

Distributivität: $I \otimes (J + K) = I \otimes J + I \otimes K$

skalare Assoziativität: $a \cdot (I \otimes J) = (a \cdot I) \otimes J = I \otimes (a \cdot J)$

Durch Anwendung der Rechenregeln kann eine Maske effizient angewendet werden, indem sie aus zwei eindimensionalen Masken zusammengesetzt wird:

$$I \otimes h = (I \otimes h_x) \otimes h_y$$

3.1.1 Glätten von Bildern (Tiefpass)

Bei der Glättung wird ein Pixel durch die Mittelung des Pixels mit den Nachbarpixeln ersetzt. Dies dient zur Rauschunterdrückung oder als Vorstufe zum Dezimieruren der räumlichen Auflösung (eng. down sampling).
Verwendete Standardfilter sind Rechteck- oder Gaussmasken:

$$R = \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \qquad G = \frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

Damit der Wertebereich der Abbildung $G' = [0\ 255]$ bleibt, muss die Summe über alle Maskenkoeffizienten 1 ergeben.

Lösung in MATLAB:

```
clear 'all'; close 'all'; format compact;

%read image
Image = imread('sample.png');

%convert to double
Image = double(Image);

%apply a filter of size 2x2
ImageFiltered = imfilter(Image,ones(2)/4);
imshow(Image1,[0 255]);
```

3.1.2 Kantenhervorhebung (Hochpass)

Harte Grauwertübergänge eines Bildes werden verstärkt, während weiche Übergänge abgeschwächt werden. Hierbei ist der Gradienten Filter eine wichtige Filterklasse. Diese berechnet den Gradienten, d.h. die

Ableitung der Grauwerte $I_{m,n}$ eines Bildes.

$$\frac{\partial I(x,y)}{\partial x} = \lim_{\Delta x \to 0} \frac{I(x+\Delta x, y) - I(x-\Delta x, y)}{2 \cdot \Delta x} = \frac{I_{m,n+1} - I_{m,n-1}}{2}$$
$$\frac{\partial I(x,y)}{\partial y} = \lim_{\Delta y \to 0} \frac{I(x, y+\Delta y) - I(x, y-\Delta y)}{2 \cdot \Delta y} = \frac{I_{m+1,n} - I_{m-1,n}}{2}$$

Mit folgenden Filtern kann diese Berechnung des Gradienten durchgeführt werden:

$$h_x = \frac{1}{2} \cdot \begin{bmatrix} -1 & 0 & 1 \end{bmatrix} \qquad h_y = \frac{1}{2} \cdot \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix}$$

Da diese fehleranfällig auf Rausche sind, wird jeweils mit einem Glättungsfilter kombiniert.

Prewitt-Filter:

$$h_x = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \otimes \begin{bmatrix} -1 & 0 & 1 \end{bmatrix} = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \qquad h_y = \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix} \otimes \begin{bmatrix} 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

Sobel-Filter:

$$h_x = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} \otimes \begin{bmatrix} -1 & 0 & 1 \end{bmatrix} = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \qquad h_y = \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix} \otimes \begin{bmatrix} 1 & 2 & 1 \end{bmatrix} = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

Lösung in MATLAB:

```
clear 'all'; close 'all'; format compact;

%read image
Image = imread('sample.png');

%convert to double
Image = double(Image);
```

```
%choose the filters
Sobel = 1;
if Sobel == 1
    DX = [1;2;1] * [-1 0 1];
    DY = DX';
else
    DX = ones(3,1) * [-1 0 1];
    DY = DX';
end

%apply the DX and DY filter
ImageDx = uint8(128+imfilter(Image, DX));
ImageDy = uint8(128+imfilter(Image, DY));
```

Der Gradient kann auch als Vektor dargestellt werden mit dem Betrag $\frac{\mathrm{d}I}{\mathrm{d}r}$ und Winkel α:

$$\frac{\mathrm{d}I}{\mathrm{d}r} = \sqrt{\left(\frac{\partial I}{\partial x}\right)^2 + \left(\frac{\partial I}{\partial y}\right)^2} \qquad \alpha = \arctan \frac{\frac{\partial I}{\partial y}}{\frac{\partial I}{\partial x}}$$

Lösung in MATLAB:

```
clear 'all'; close 'all'; format compact;

%read image
Image = imread('sample.png');

%convert to double
Image = double(Image);

%choose the filters
Sobel = 1;
if Sobel == 1
    DX = fspecial('sobel')';
  DY = fspecial('sobel');
else
    DX = fspecial('prewitt')';
  DY = fspecial('prewitt');
end

%apply the DX and DY filter
ImageDx = imfilter(Image, DX);
```

```
ImageDy = imfilter(Image, DY);

%calculate the norm of the derviative
ImageDr = sqrt(ImageDx.^2 + ImageDy.^2);

%determine the angle (atan2 gives back the whole interval ]-pi , pi[ )
Angle = pi+atan2(ImageDy, ImageDx);
%use only those values that are above a given threshold
Angle(ImageDr < threshold) = 0;

%plot
imshow(ImageDr, [0 255]);
imshow(Angle, [0 255]);
map=colormap(jet);
map(1,:) = 0;
colormap(map)
colorbar;
```

3.1.3 Bildschärfung

Für die Bildschärfung wird der Laplace Operator verwendet, welcher ein linearer richtungsunabhängiger Operator ist. Er ist definiert als die Summe der zweiten Ableitung:

$$\Delta I = L = \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2}$$

Die Approximation für diskrete Werte $I_{m,n}$ des Bildes (nur für x, y analog dazu):

$$\begin{aligned}
I_{xx} &\approx \frac{I_x(x+1,y) - I_x(x,y}{1} \\
&\approx \frac{I(x+1,y) - I(x,y}{1} - \frac{I(x,y) - I(x-1,y}{1} \\
\frac{\partial^2 I}{\partial x^2} &\approx I_{x+1} - 2 \cdot I_x + I_{x-1}
\end{aligned}$$

Die dazugehörige Maske:

$$L = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

Für die Bildschärfung wird der Laplace Operator in Kombination mit dem Identischen Operator I verwendet:

$$I + \beta \cdot L$$

β: Grad der Bildschärfung

Lösung in MATLAB:

```
clear 'all'; close 'all'; format compact;

%read image
Image = imread('sample.png');

%convert to double
Image = double(Image);

%define the filter
Beta = 0.5;
Mask = [0 0 0; 0 1 0; 0 0 0] + Beta * [0 -1 0; -1 4 -1; 0
    -1 0];

%apply the filter
ImageEnh = uint8(imfilter(Image, Mask));
```

Kapitel 4

Fourier-Transformation

Kapitel 5

Segmentierung und Merkmalsextraktion

Kapitel 6

Linien-Segmentierung und Merkmalsextraktion

Kapitel 7

Farbe

Kapitel 8

Anwendungen