Wilhelm Burger · Mark James Burge

Digitale Bildverarbeitung

Eine algorithmische Einführung mit Java

3., überarbeitete und ergänzte Auflage
Mit 375 Abbildungen, 25 Tabellen und 60 Algorithmen

ERRATA

Springer
Berlin Heidelberg New York
Hongkong London
Mailand Paris Tokio

Programm 2.1
ImageJ-Plugin zum Invertieren von 8-Bit-Grauwertbildern. Das Plugin implementiert das Interface `PlugInFilter` und enthält die dafür erforderlichen Methoden `setup()` und `run()`. Das eigentliche Bild wird der `run()`-Methode des Plugins mit dem Parameter `ip` als Objekt vom Typ `ImageProcessor` übergeben. ImageJ nimmt an, dass das Plugin das übergebene Bild verändert und aktualisiert nach Ausführung des Plugins die Bildanzeige automatisch. Programm Prog. 2.2 zeigt eine alternative Implementierung basierend auf dem `PlugIn`-Interface.

```
1  import ij.ImagePlus;
2  import ij.plugin.filter.PlugInFilter;
3  import ij.process.ImageProcessor;
4
5  public class My_Inverter_A implements PlugInFilter {
6
7    public int setup(String args, ImagePlus im) {
8      return DOES_8G; // this plugin accepts 8-bit grayscale images
9    }
10
11   public void run(ImageProcessor ip) {
12     int M = ip.getWidth();
13     int N = ip.getHeight();
14
15     // iterate over all image coordinates (u,v)
16     for (int u = 0; u < M; u++) {
17       for (int v = 0; v < N; v++) {
18         int p = ip.getPixel(u, v);
19         ip.putPixel(u, v, 255 - p);
20       }
21     }
22   }
23
24 }
```

`int getPixel (int u, int v)`
: Liefert den Wert des Bildelements an der Position (u, v) oder null, wenn (u, v) außerhalb der Bildgrenzen liegt.

`void putPixel (int u, int v, int a)`
: Setzt das Bildelement an der Position (u, v) auf den neuen Wert a. Die Anwendung hat keine Auswirkungen auf das Bild, wenn (u, v) außerhalb der Bildgrenzen liegt.

Beide Methoden überprüfen die übergebenen Koordinaten und Pixelwerte genau, um allfällige Fehler zu vermeiden, und sind dadurch zwar „idiotensicher“ aber naturgemäß auch relativ langsam. Wenn man sicher sein kann, dass alle besuchten Bildelemente innerhalb der Bildgrenzen liegen und die eingefügten Pixelwerte garantiert im zulässigen Wertebereich des verwendeten Bildtyps liegen (wie in Prog. 2.1), dann kann man sie durch schnellere Zugriffsmethoden ersetzen, wie `get()` und `set()` anstelle von `getPixel()` und `putPixel()` (siehe Prog. 2.2). Am effizientesten ist es, überhaupt keine Methoden zum Lesen bzw. Schreiben zu verwenden, sondern direkt auf die Elemente des Pixel-Arrays zuzugreifen. Details zu diesen und anderen Methoden finden sich online in der API-Dokumentation zu ImageJ.[12]

[12] http://rsbweb.nih.gov/ij/developer/api/index.html

nur *einer* gemeinsamen Iteration über die Elemente des Bilds bzw. des Histogramms ermittelt werden, nämlich in der Form

$$\mu_I = \frac{1}{MN} \cdot A \qquad \text{und} \tag{3.13}$$

$$\sigma_I^2 = \frac{1}{MN} \cdot \Big(B - \frac{1}{MN} \cdot A^2\Big), \tag{3.14}$$

mit den Größen

$$A = \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} I(u,v) \;= \sum_{i=0}^{K-1} i \cdot \mathsf{h}(i), \tag{3.15}$$

$$B = \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} I^2(u,v) = \sum_{i=0}^{K-1} i^2 \cdot \mathsf{h}(i). \tag{3.16}$$

Die Formulierung in Gl. 3.13–3.16 hat auch den technischen Vorteil, dass die Summen mit ganzzahligen Werten berechnet werden können (im Unterschied zu Gl. 3.12, wo Gleitkommawerte summiert werden müssen).

3.7.2 Median

Auch der *Median* eines Bilds kann leicht aus dem Histogramm berechnet werden. Dieser ist definiert als jener Pixelwert, der einerseits größer als eine Hälfte aller Bildwerte ist und gleichzeitig kleiner als die andere Hälfte, also genau „in der Mitte“ aller Werte liegt.[7]

Um den Median für ein Bild der Größe MN aus dem zugehörigen Histogramm zu bestimmen, genügt es daher jenen Index i zu finden, der das Histogramm so in zwei Hälften teilt, dass die Summen der Histogrammeinträge links und rechts von i annähernd gleich sind. Anders formuliert ist dies der kleinste Index i, wo die Summe aller Histogrammeinträge unterhalb von i annähernd der Hälfte aller Bildelemente entspricht, d. h.,

$$m_I = \min\Big\{i \mid \sum_{j=0}^{i} \mathsf{h}(j) \geq \frac{MN}{2}\Big\}. \tag{3.17}$$

Da $\sum_{j=0}^{i} \mathsf{h}(j) = \mathsf{H}(i)$ (s. Gl. 3.7), lässt sich der Median noch einfacher in der Form

$$m_I = \min\Big\{i \mid \mathsf{H}(i) \geq \frac{MN}{2}\Big\} \tag{3.18}$$

aus dem *kumulativen* Histogramm H bestimmen.

[7] Eine alternative Definition des Medians findet sich in Abschn. 5.4.2.

werden, wobei wir Bildelemente mit dem Wert 1 als *Vordergrund*-Pixel (*foreground*) bzw. mit dem Wert 0 als *Hintergrund*-Pixel (*background*) definieren. In den meisten der folgenden Beispiele ist, wie auch im Druck üblich, der Vordergrund schwarz dargestellt und der Hintergrund weiß.

Am Ende dieses Kapitels werden wir sehen, dass morphologische Filter nicht nur für Binärbilder anwendbar sind, sondern auch für Grauwertbilder und sogar Farbbilder, allerdings unterscheiden sich diese Filter deutlich von den binären Operationen.

9.1 Schrumpfen und wachsen lassen

Ausgangspunkt ist also die Beobachtung, dass, wenn wir ein gewöhnliches 3×3-Medianfilter auf ein Binärbild anwenden, sich größere Bildstrukturen abrunden und kleinere Bildstrukturen, wie Punkte und dünne Linien, vollständig verschwinden. Das könnte nützlich sein, um etwa Strukturen unterhalb einer bestimmten Größe aus dem Bild zu eliminieren. Wie kann man aber die Größe und möglicherweise auch die Form der von einer solchen Operation betroffenen Strukturen kontrollieren?

Die strukturellen Auswirkungen eines Medianfilters sind zwar interessant, aber beginnen wir mit dieser Aufgabe nochmals von vorne. Nehmen wir also an, wir möchten kleine Strukturen in einem Binärbild entfernen, ohne die größeren Strukturen dabei wesentlich zu verändern. Dazu könnte die Kernidee folgende sein (Abb. 9.2):

1. Zunächst werden alle Strukturen im Bild schrittweise „geschrumpft“, wobei jeweils außen eine Schicht von bestimmter Stärke abgelöst wird.
2. Durch das Schrumpfen verschwinden kleinere Strukturen nach und nach, und nur die größeren Strukturen bleiben übrig.
3. Schließlich lassen wir die verbliebenen Strukturen wieder im selben Umfang wachsen.

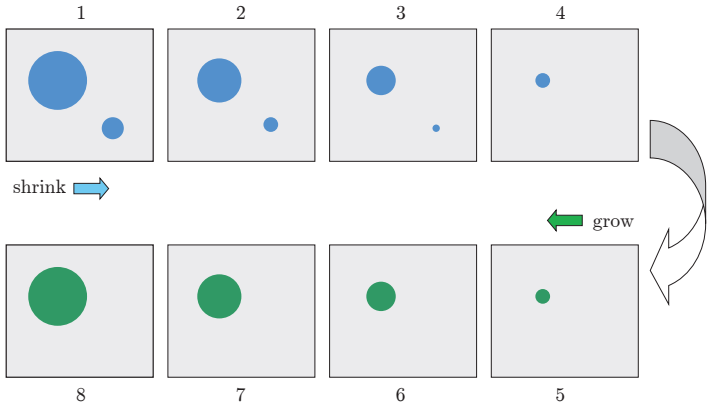

Abbildung 9.2
Idee der größenabhängigen Elimination. Durch schrittweises Schrumpfen und anschließendes Wachsen können kleinere Bildstrukturen entfernt werden, während die „überlebenden“ Strukturen weitgehend unverändert bleiben.

$$c_{pq}(\mathcal{R}) = \sum_{(u,v)\in\mathcal{R}} I(u,v) \cdot (u + \mathrm{i}\cdot v)^p \cdot (u - \mathrm{i}\cdot v)^q, \tag{10.43}$$

beziehungsweise für Binärbilder (mit $I(u,v) \in [0,1]$) durch

$$c_{pq}(\mathcal{R}) = \sum_{(u,v)\in\mathcal{R}} (u + \mathrm{i}\cdot v)^p \cdot (u - \mathrm{i}\cdot v)^q \tag{10.44}$$

(i bezeichnet die imaginäre Einheit). Die darauf aufbauenden Momente 2. bis 4. Ordnung sind in [64] folgendermaßen definiert:

$$\begin{aligned}
&\psi_1 = c_{11} = \phi_1, && \psi_2 = c_{21}c_{12} = \phi_4, && \psi_3 = \mathrm{Re}(c_{20}c_{12}^2) = \phi_6,\\
&\psi_4 = \mathrm{Im}(c_{20}c_{12}^2), && \psi_5 = \mathrm{Re}(c_{30}c_{12}^3) = \phi_5, && \psi_6 = \mathrm{Im}(c_{30}c_{12}^3) = \phi_7,\\
&\psi_7 = c_{22}, && \psi_8 = \mathrm{Re}(c_{31}c_{12}^2), && \psi_9 = \mathrm{Im}(c_{31}c_{12}^2),\\
&\psi_{10} = \mathrm{Re}(c_{40}c_{12}^4), && \psi_{11} = \mathrm{Im}(c_{40}c_{12}^4). &&
\end{aligned} \tag{10.45}$$

Wie man sieht, stimmen einige dieser Mermale mit den Merkmalen von Hu (ϕ_k) überein. Für die effiziente Berechnung von Momenten findet man in der Literatur zahlreiche Hinweise, z. B. [65, 121].

10.4.5 Projektionen

Projektionen von Bildern sind eindimensionale Abbildungen der Bilddaten, üblicherweise parallel zu den Koordinatenachsen. In diesem Fall ist die horizontale bzw. vertikale Projektion für ein Bild $I(u,v)$, mit $0 \le u < M$, $0 \le v < N$, definiert als

$$\mathsf{P}_{\mathrm{hor}}(v) = \sum_{u=0}^{M-1} I(u,v) \quad \text{für } 0 < v < N, \tag{10.46}$$

$$\mathsf{P}_{\mathrm{ver}}(u) = \sum_{v=0}^{N-1} I(u,v) \quad \text{für } 0 < u < M. \tag{10.47}$$

Die *horizontale* Projektion $\mathsf{P}_{\mathrm{hor}}(v)$ in Gl. 10.46 bildet also die Summe der Pixelwerte der Bild*zeile* v und hat die Länge N, die der Höhe des Bilds entspricht. Umgekehrt ist die *vertikale* Projektion $\mathsf{P}_{\mathrm{ver}}(u)$ die Summe aller Bildwerte in der *Spalte* u (Gl. 10.47). Im Fall eines Binärbilds mit $I(u,v) \in 0, 1$ enthält die Projektion die Anzahl der Vordergrundpixel in der zugehörigen Bildzeile bzw. -spalte.

Prog. 10.4 zeigt eine einfache Implementierung der Projektionsberechnung als `run()`-Methode eines ImageJ-Plugin,[16] wobei beide Projektionen in einem Bilddurchlauf berechnet werden.

Projektionen in Richtung der Koordinatenachsen sind beispielsweise zur schnellen Analyse von strukturierten Bildern nützlich, wie etwa zur

[16] Plugin `Make_Projections`

vektor $(\nabla I)(\boldsymbol{x}) = (I_x(\boldsymbol{x}), I_y(\boldsymbol{x}))^\mathsf{T}$ der Orientierung der Tangentialebene zur Bildfunktion I an der Position $\boldsymbol{x}$. Mit

$$A = I_x^2(\boldsymbol{x}), \qquad B = I_y^2(\boldsymbol{x}), \qquad C = I_x(\boldsymbol{x}) \cdot I_y(\boldsymbol{x}) \tag{16.33}$$

ist (analog zu Gl. 16.24)

$$S_\theta(I, \boldsymbol{x}) = \big(I_x(\boldsymbol{x}) \cdot \cos(\theta) + I_y(\boldsymbol{x}) \cdot \sin(\theta)\big)^2 \tag{16.34}$$

der quadratische lokale Kontrast an der Stelle $\boldsymbol{x}$ in Richtung θ (s. Gl. 16.19). Die Eigenwerte der lokalen Strukturmatrix $\mathbf{M} = \left(\begin{smallmatrix} A & C \\ C & B \end{smallmatrix}\right)$ an der Position $\boldsymbol{x}$ sind (nach Gl. 16.25)

$$\lambda_{1,2}(\boldsymbol{x}) = \frac{A + B \pm \sqrt{(A-B)^2 + 4C^2}}{2}. \tag{16.35}$$

aber in diesem Fall, da I_x, I_y nun skalare Werte und keine Vektoren sind, gilt $C^2 = (I_x \cdot I_y)^2 = I_x^2 \cdot I_y^2$ und daher auch $(A-B)^2 + 4\,C^2 = (A+B)^2$, sodass

$$\lambda_{1,2}(\boldsymbol{x}) = \frac{A + B \pm (A+B)}{2}\,. \tag{16.36}$$

Wir sehen also, dass bei einem skalarwertigen Bild der dominante Eigenwert von $\mathbf{M}$,

$$\lambda_1(\boldsymbol{x}) = A + B = I_x^2 + I_y^2 = \|\nabla I\|_2^2\,, \tag{16.37}$$

dem Quadrat der L_2-Norm des lokalen Gradienten entspricht, während der kleinere Eigenwert λ_2 immer Null ist. Bei einem Grauwertbild ist somit die maximale Kantenstärke $\sqrt{\lambda_1} = \|\nabla I\|_2$ gleich dem Betrag des lokalen Intensitätsgradienten.[15] Der Umstand, dass $\lambda_2 = 0$, macht deutlich, dass der lokale Anstieg der Bildfunktion in *orthgogonaler* Richtung (also entlang der Kantentangente) Null ist (siehe Abb. 16.3 (c)).

Um die lokale Kanten*richtung* im Punkt $\boldsymbol{x}$ zu ermitteln, erhalten wir zunächst mit Gl. 16.30

$$\tan(\theta_1(\boldsymbol{x})) = \frac{2C}{A - B + (A+B)} = \frac{2C}{2A} = \frac{I_x I_y}{I_x^2} = \frac{I_y}{I_x} \tag{16.38}$$

und daraus die Richtung des maximalen Kontrasts[16] als

$$\theta_1(\boldsymbol{x}) = \tan^{-1}\Big(\frac{I_y}{I_x}\Big) = \mathrm{Arctan}(I_x, I_y). \tag{16.39}$$

Für skalarwertige Bilder führt also das allgemeine (mehrdimensionale), auf den Eigenwerten der Strukturmatrix basierende Verfahren zu exakt dem selben Ergebnis wie der in Abschn. 6.3 beschriebene traditionelle Ansatz für einfache Kantenoperatoren.

[15] Siehe auch Abschn. 6.2, Gl. 6.5 und 6.13.
[16] Siehe auch Abschn. 6.3, Gl. 6.14.

25.5 Vergleich von SIFT-Merkmalen

Die häufigste Anwendung des SIFT-Verfahrens ist die Lokalisierung von zusammengehörigen Merkmalspunkten in zwei oder mehreren Bildern der selben Szene, beispielsweise für die Zuordnung von Referenzpunkten in Stereoaufnahmen, für den Zusammenbau von Panoramabildern oder zur Verfolgung von Referenzpunkten in Bildsequenzen. Bei andere Einsatzfällen, wie z. B. der Objekterkennung oder Selbstlokalisierung, ist es wiederum notwendig, in Einzelbildern oder Videosequenzen detektierte Merkmale einer großen Zahl von gespeicherten Modellmerkmalen zuzuordnen. In all diesen Fällen ist ein zuverlässiger und effizienter Vergleich zwischen Paaren von SIFT-Merkmalen erforderlich.

25.5.1 Bestimmung der Ähnlichkeit von Merkmalen

Eine typische Standardsituation ist, dass für zwei Ausgangsbilder I_a, I_b zunächst jeweils eine Folge von SIFT-Merkmalen,

$$S^{(a)} = (\boldsymbol{s}_1^{(a)}, \boldsymbol{s}_2^{(a)}, \ldots, \boldsymbol{s}_{N_a}^{(a)}) \quad \text{bzw.} \quad S^{(b)} = (\boldsymbol{s}_1^{(b)}, \boldsymbol{s}_2^{(b)}, \ldots, \boldsymbol{s}_{N_b}^{(b)}),$$

berechnet wird mit dem Ziel, darin Paare von zusammengehörigen Merkmalspunkten zu finden. Die Ähnlichkeit zwischen zwei Deskriptoren $\boldsymbol{s}_i = \langle x_i, y_i, \sigma_i, \theta_i, \boldsymbol{f}_i \rangle$ und $\boldsymbol{s}_j = \langle x_j, y_j, \sigma_j, \theta_j, \boldsymbol{f}_j \rangle$ wird bestimmt über die Distanz der zugehörigen Merkmalsvektoren, d. h.,

$$\text{dist}(\boldsymbol{s}_i, \boldsymbol{s}_j) := \left\| \boldsymbol{f}_i - \boldsymbol{f}_j \right\|, \tag{25.100}$$

wobei $\|\cdots\|$ eine geeignete (typischerweise die Euklidische) Norm bezeichnet.[24]

Dabei ist zu beachten, dass hier die Distanz zwischen einzelnen Punkten in einem Vektorraum berechnet wird, der aufgrund der hohen Dimensionalität (typ. 128) äußerst dünn besetzt ist. Da sich zu einem gegebenen Deskriptor naturgemäß *immer* ein nächstliegendes Gegenstück im Merkmalsraum findet, können leicht auch Merkmale als ähnlich interpretiert werden, die in keinem Zusammenhang zueinander stehen. Dies ist vor allem dann ein Problem, wenn durch den Vergleich von Merkmalen festgestellt werden soll, ob es in zwei Bildern überhaupt Übereinstimmungen gibt.

Naturgemäß sollen „gute“ Paarungen von Merkmalen mit einer geringen Distanz im Merkmalsraum einhergehen, in der Praxis erweist sich jedoch die bloße Vorgabe einer fixen Maximaldistanz als unzureichend. Die Grundlage der in [138] vorgeschlagenen Lösung ist, die Distanz für die beste Übereinstimmung mit der „zweitbesten“ Distanz in Relation zu setzen. Das zu einem aus $S^{(a)}$ gewählten Referenzdeskriptor $\boldsymbol{s}_\text{r}$ „beste“ Gegenstück ist jener Deskriptor $\boldsymbol{s}_1$ in $S^{(b)}$, der gegenüber $\boldsymbol{s}_\text{r}$ die geringste Distanz (Gl. 25.100) aufweist, d. h.,

[24] Siehe auch Abschn. B.1.2 im Anhang.

$\cap$	Durchschnitt (Schnittmenge) von zwei Mengen (z. B. $A \cap B$).
$\bigcup_{A_i}$	Vereinigung mehrerer Mengen A_i.
$\bigcap_{A_i}$	Durchschnitt (Schnittmenge) mehrerer Mengen A_i.
$\setminus$	Differenzmenge: wenn $x \in A \setminus B$, dann gilt $x \in A$ und $x \notin B$.

A.3 Komplexe Zahlen

Definitionen:

$$z = a + \mathrm{i}\, b, \qquad \text{mit } z, \mathrm{i} \in \mathbb{C},\ a, b \in \mathbb{R},\ i^2 = -1, \tag{A.1}$$

$$z^* = a - \mathrm{i}\, b \qquad \text{(konjugiert-komplexe Zahl)}, \tag{A.2}$$

$$sz = sa + \mathrm{i}\, sb, \qquad s \in \mathbb{R}, \tag{A.3}$$

$$|z| = \sqrt{a^2 + b^2}, \quad |sz| = s\,|z|, \tag{A.4}$$

$$\begin{aligned} z &= a + \mathrm{i}\, b = |z| \cdot (\cos\psi + \mathrm{i}\, \sin\psi) \\ &= |z| \cdot e^{\mathrm{i}\psi}, \quad \text{wobei } \psi = \mathrm{Arctan}(a, b) \end{aligned} \tag{A.5}$$

$$\mathrm{Re}\big(a + \mathrm{i}\, b\big) = a, \quad \mathrm{Re}\big(e^{\mathrm{i}\varphi}\big) = \cos\varphi, \tag{A.6}$$

$$\mathrm{Im}\big(a + \mathrm{i}\, b\big) = b, \quad \mathrm{Im}\big(e^{\mathrm{i}\varphi}\big) = \sin\varphi, \tag{A.7}$$

$$e^{\mathrm{i}\varphi} = \cos\varphi + \mathrm{i}\, \sin\varphi, \tag{A.8}$$

$$e^{-\mathrm{i}\varphi} = \cos\varphi - \mathrm{i}\, \sin\varphi, \tag{A.9}$$

$$\cos(\varphi) = \tfrac{1}{2} \cdot \big(e^{\mathrm{i}\varphi} + e^{-\mathrm{i}\varphi}\big), \tag{A.10}$$

$$\sin(\varphi) = \tfrac{1}{2i} \cdot \big(e^{\mathrm{i}\varphi} - e^{-\mathrm{i}\varphi}\big). \tag{A.11}$$

Rechenoperationen:

$$z_1 = (a_1 + \mathrm{i}b_1) = |z_1|\, e^{\mathrm{i}\varphi_1}, \tag{A.12}$$

$$z_2 = (a_2 + \mathrm{i}b_2) = |z_2|\, e^{\mathrm{i}\varphi_2}, \tag{A.13}$$

$$z_1 + z_2 = (a_1 + a_2) + \mathrm{i}\,(b_1 + b_2), \tag{A.14}$$

$$\begin{aligned} z_1 \cdot z_2 &= (a_1 a_2 - b_1 b_2) + \mathrm{i}\,(a_1 b_2 + b_1 a_2) \\ &= |z_1| \cdot |z_2| \cdot e^{\mathrm{i}\,(\varphi_1 + \varphi_2)}, \end{aligned} \tag{A.15}$$

$$\frac{z_1}{z_2} = \frac{a_1 a_2 + b_1 b_2}{a_2^2 + b_2^2} + \mathrm{i}\,\frac{a_2 b_1 - a_1 b_2}{a_2^2 + b_2^2} = \frac{|z_1|}{|z_2|} \cdot e^{\mathrm{i}\,(\varphi_1 - \varphi_2)}. \tag{A.16}$$

Burger, Burge: Digitale Bildverarbeitung – Eine algorithmische Einführung mit Java,
3. Auflage, © Springer-Verlag, 2015. www.imagingbook.com

Die inverse Hesse-Matrix $\mathbf{H}_f^{-1}$ ist übrigens ebenfalls symmetrisch.

Den zugehörigen Extrem*wert* der Annäherungsfunktion $\tilde{f}$ ergibt sich aus Gl. C.56, indem man $\boldsymbol{x}$ durch die in Gl. C.60 berechnete Position $\breve{\boldsymbol{x}}$ ersetzt, d. h.,

$$\begin{aligned}\tilde{f}_{\text{extreme}} = \tilde{f}(\breve{\boldsymbol{x}}) &= f + \nabla_f^\mathsf{T} \cdot \breve{\boldsymbol{x}} + \frac{1}{2} \cdot \breve{\boldsymbol{x}}^\mathsf{T} \cdot \mathbf{H}_f \cdot \breve{\boldsymbol{x}} \\ &= f + \nabla_f^\mathsf{T} \cdot \breve{\boldsymbol{x}} + \frac{1}{2} \cdot \breve{\boldsymbol{x}}^\mathsf{T} \cdot \mathbf{H}_f \cdot (-\mathbf{H}_f^{-1}) \cdot \nabla_f \\ &= f + \nabla_f^\mathsf{T} \cdot \breve{\boldsymbol{x}} - \frac{1}{2} \cdot \breve{\boldsymbol{x}}^\mathsf{T} \cdot \mathbf{I} \cdot \nabla_f \\ &= f + \nabla_f^\mathsf{T} \cdot \breve{\boldsymbol{x}} - \frac{1}{2} \cdot \nabla_f^\mathsf{T} \cdot \breve{\boldsymbol{x}} \\ &= f + \frac{1}{2} \cdot \nabla_f^\mathsf{T} \cdot \breve{\boldsymbol{x}}\,,\end{aligned} \tag{C.62}$$

wiederum für den Expansionspunkt $\boldsymbol{a} = \mathbf{0}$. Im Fall eines beliebigen Expansionspunkts $\boldsymbol{a}$ ist das Ergebnis

$$\tilde{f}_{\text{extreme}} = \tilde{f}(\breve{\boldsymbol{x}}) = f(\boldsymbol{a}) + \frac{1}{2} \cdot \nabla_f^\mathsf{T}(\boldsymbol{a}) \cdot (\breve{\boldsymbol{x}} - \boldsymbol{a})\,. \tag{C.63}$$

Man beachte, dass $\breve{\boldsymbol{x}}$ zwar typischerweise ein lokales Minimum oder Maximum ist, grundsätzlich aber auch ein *Sattelpunkt* sein kann, wo die ersten Ableitungen der Funktion ebenfalls Null sind.

Lokale Extrema in 2D

Das oben beschriebene Schema ist allgemein auf n-dimensionale Funktionen anwendbar. Im speziellen Fall einer zweidimensionalen Funktion $f\colon \mathbb{R}^2 \mapsto \mathbb{R}$ (z. B. ein Bild) ergibt sich der Gradientenvektor bzw. die Hesse-Matrix für einen gegebenen Expansionspunkt $\boldsymbol{a} = (x_a, y_a)^\mathsf{T}$ als

$$\nabla_f(\boldsymbol{a}) = \begin{pmatrix} d_x \\ d_y \end{pmatrix} \quad \text{und} \quad \mathbf{H}_f(\boldsymbol{a}) = \begin{pmatrix} h_{11} & h_{12} \\ h_{12} & h_{22} \end{pmatrix}. \tag{C.64}$$

In diesem Fall kann die Inverse der Hesse-Matrix durch

$$\mathbf{H}_f^{-1} = \frac{1}{h_{12}^2 - h_{11} \cdot h_{22}} \cdot \begin{pmatrix} -h_{22} & h_{12} \\ h_{12} & -h_{11} \end{pmatrix} \tag{C.65}$$

in geschlossener Form bestimmt werden und die resultierende Position des Extremwerts (s. Gl. C.61) ist folglich

$$\begin{aligned}\breve{\boldsymbol{x}} = \begin{pmatrix} \breve{x} \\ \breve{y} \end{pmatrix} &= \begin{pmatrix} x_a \\ y_a \end{pmatrix} - \frac{1}{h_{12}^2 - h_{11} \cdot h_{22}} \cdot \begin{pmatrix} -h_{22} & h_{12} \\ h_{12} & -h_{11} \end{pmatrix} \cdot \begin{pmatrix} d_x \\ d_y \end{pmatrix} \\ &= \begin{pmatrix} x_a \\ y_a \end{pmatrix} - \frac{1}{h_{12}^2 - h_{11} \cdot h_{22}} \cdot \begin{pmatrix} h_{12} \cdot d_y - h_{22} \cdot d_x \\ h_{12} \cdot d_x - h_{11} \cdot d_y \end{pmatrix}.\end{aligned} \tag{C.66}$$

Diese Position ist natürlich nur dann definiert, wenn der obige Nenner $h_{12}^2 - h_{11} \cdot h_{22}$ (die Determinante von $\mathbf{H}_f$) nicht Null ist, die Hesse-Matrix

definieren, wobei $\otimes$ das äußere (Vektor-)Produkt bezeichnet.

Die Spur (d. h., die Summe der Diagonalelemente) der Kovarianzmatrix, d. h.,

$$\sigma_{\text{total}}(X) = \text{trace}\,(\Sigma(X))\,, \tag{D.10}$$

wird als *totale Varianz* der multivariaten Stichprobe bezeichnet. Alternativ dazu kann auch die (*Frobenius-*)*Norm* der Kovarianzmatrix,

$$\|\Sigma(X)\|_2 = \Big(\sum_{i=1}^{m}\sum_{j=1}^{m}\sigma_{i,j}^2\Big)^{1/2}, \tag{D.11}$$

zur Bestimmung der Gesamtvariabilität (Streuung) in den Messdaten herangezogen werden.

Beispiel

Wir nehmen an, die Datenfolge X besteht aus den folgenden vier ($n=4$) dreidimensionalen ($m=3$) Vektoren

$$\boldsymbol{x}_1 = \begin{pmatrix}75\\37\\12\end{pmatrix}, \quad \boldsymbol{x}_2 = \begin{pmatrix}41\\27\\20\end{pmatrix}, \quad \boldsymbol{x}_3 = \begin{pmatrix}93\\81\\11\end{pmatrix}, \quad \boldsymbol{x}_4 = \begin{pmatrix}12\\48\\52\end{pmatrix},$$

wobei jeder Vektor $\boldsymbol{x}_i = (x_{i,\mathrm{R}}, x_{i,\mathrm{G}}, x_{i,\mathrm{B}})^\mathsf{T}$ beispielsweise ein RGB-Farbtupel repräsentiert. Der resultierende *Durchschnittsvektor* (s. Gl. D.2) ist

$$\boldsymbol{\mu}(X) = \begin{pmatrix}\mu_R\\\mu_G\\\mu_B\end{pmatrix} = \frac{1}{4}\cdot\begin{pmatrix}75+41+93+12\\37+27+81+48\\12+20+11+52\end{pmatrix} = \frac{1}{4}\cdot\begin{pmatrix}221\\193\\95\end{pmatrix} = \begin{pmatrix}55.25\\48.25\\23.75\end{pmatrix},$$

und die zugehörige Kovarianzmatrix (s. Gl. D.8) ist

$$\Sigma(X) = \begin{pmatrix} 972.188 & 331.938 & -470.438\\ 331.938 & 412.688 & -53.188\\ -470.438 & -53.188 & 278.188\end{pmatrix}.$$

Wie erwartet ist diese Matrix symmetrisch und die Werte ihrer Diagonalelemente sind positiv. Die *totale Varianz* (s. Gl. D.10) dieser Datenfolge ist

$$\sigma_{\text{total}}(X) = \text{trace}\,(\Sigma(X)) = 972.188+412.688+278.188 \approx 1663.06$$

und die *Norm* der Kovarianzmatrix (siehe Gl. D.11) beträgt $\|\Sigma(X)\|_2 \approx 1364.36$.[1]

[1] **Anmerkung:** Für die ursprünglichen Beispieldaten wurde versehentlich der Bias-korrigierende Skalierungsfaktor von $1/(n-1)$ anstelle von $1/n$ (wie in den Gleichungen in diesem Abschnitt angegeben) zur Berechnung der Kovarianzwerte verwendet. Diese Form der statistischen Bias-Kompensation wird hier nicht verwendet.