Hochschule für Technik und Wirtschaft Berlin
University of Applied Sciences

Internationaler Studiengang Medieninformatik (IMI) – Master

Masterarbeit

Computergestützte Identifizierung von Pflanzen anhand ihrer Blattmerkmale

Wintersemester 2016/2017

Student: Christoph Franke

549642

Betreuer: Prof. Dr. Klaus Jung

Nico Hezel

Inhaltsverzeichnis

1. Einleitung

Erkennung von Pflanzen herausforderndes Problem der computer vision, unregelmäßige Formen, hochvariable Texturen(?)

2. Grundlagen

2.1. Morphologie von Pflanzen

2.1.1. Bestimmungsmerkmale höherer Pflanzen

Die als Kormophyten bezeichneten höheren Pflanzen weisen einen Kormus genannten Pflanzenkörper auf, der sich aus den Grundorganen Laubblatt, Wurzel und Sprossachse zusammensetzt. Die Gestalt der Organe steht in enger Wechselbeziehung mit ihrer Funktion. (Welle, 2014, S. 16)

Abhängig von den Lebensumständen des Organismus, variiert die Beschaffenheit der Organe zwischen den Arten. Anhand des Erscheinungsbildes ihrer Organe, können Pflanzen bestimmt und zugeordnet werden. Mit dieser Thematik beschäftigt sich die Morphologie, die Lehre von der äußeren Gestalt der Pflanzen. (Welle, 2014, S. 13)

Die Wurzel hat die Befestigung der Pflanze im Boden und die Aufnahme von Wasser und Nährsalzen zur Aufgabe. Die Gestalt des Wurzelsystems wird durch Standortfaktoren wie die Bodenbeschaffenheit, bestimmt (Welle, 2014, S. 18). Wurzeln können neben den ursprünglichen Aufgaben weitere Funktionen übernehmen, wie die Speicherung von Nährstoffen als Rüben oder Wurzelknollen. Die Anpassung an die zusätzliche Aufgabenstellung hat eine Veränderung der Grundform, das heißt eine Metamorphose zur Folge (Welle, 2014, S. 42).

Hauptaufgabe der Sprossachse ist die Laubblätter zu tragen und sie mit der Wurzel zu verbinden, um Wasser und Nährsalze zwischen diesen beiden Organen zu transportieren (Welle, 2014, S. 22). Wichtige Unterscheidungsmerkmale von Sprossachsen sind ihre Stellung, ihre Beschaffenheit und die Form ihres Achsenquerschnitts (Rothmaler, 1967, S. X). Metamorphosen der Sprossachse können, unter anderen, Sprossdornen, Knollen und Zwiebeln sein (Welle, 2014, S. 48, 50).

Bei Samenpflanzen können zusätzlich Früchte und Samen als Bestimmungsmerkmal dienen. Diese bilden sich nach der Befruchtung aus dem Fruchtknoten mit der eingeschlossenen Samenanlage. (Welle, 2014, S. 140)

2.1.2. Morphologie des Blattes

2.1.2.1. Funktion und Aufbau des Blattes

Laubblätter haben in ihrer Grundform die Funktionen des Gasaustauschs, der Transpiration und der Photosynthese inne. Die Photosynthese dient dem Aufbau von energiereichen Kohlenhydraten aus dem energieärmeren Kohlendioxid und Wasser unter dem Einfluss des Sonnenlichtes und der Anwesenheit von Chlorophyll. Die Konzentration von Letzterem bestimmt die Intensität der Grünfärbung der Blätter. (Welle, 2014, S. 26)

In Anpassung an bestimmte Gegebenheiten können Blätter als Metamorphosen die Gestalt von Blattranken als Halte- und Kletterorgane, von Blattdornen zum Schutz vor Tierfraß und von Fangblättern bei fleischfressenden Pflanzen annehmen. (Welle, 2014, S. 54)

Die sich aus metaphorisierten Laubblättern zusammensetzenden Blüten sind ein weiteres wichtiges Unterscheidungsmerkmal von sich geschlechtlich fortpflanzenden Samenpflanzen. Die Blütenhülle wird aus den umschließenden schützenden Kelchblättern und den gefärbten Blumen- oder Kronblättern gebildet. Die Staubblätter stellen das männliche und die Fruchtblätter das weibliche Geschlechtsorgan der Pflanze dar. (Welle, 2014, S. 118)

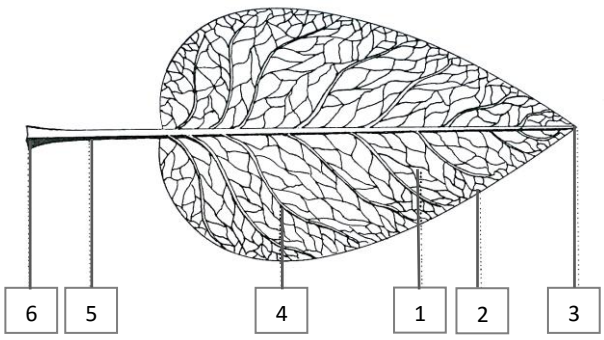


Abbildung 1 - Bau des Laubblattes: Blattspreite (1), Blattrand (2), Blattspitze (3), Blattnerv (4), Blattstiel (5), Blattgrund (6) (nach Welle, 2014, S. 27)

In Abbildung 1 ist der grundsätzliche Aufbau eines Blattes dargestellt. Die Fläche des Laubblattes wird als Blattspreite bezeichnet. Das Ende des Blattes bezeichnet man als Blattspitze. Abgegrenzt wird das Blatt vom Blattrand. Die Blattfläche wird von Blattnerven durchzogen.

Über den Blattstiel ist das Blatt mit der Sprossachse verbunden. Der Blattgrund bezeichnet den direkt an die Sprossachse angrenzenden Teil des Blattes. (Welle, 2014, S. 26)

Im Folgenden wird betrachtet, anhand welcher Merkmale man Laubblätter morphologisch unterscheiden kann.

2.1.2.2. Blattformen

Die Form der Blattspreite ist sehr variantenreich und laut Welle (2014, S. 32) ein wichtiges Merkmal zur Bestimmung der Pflanzen. Ein Auszug aus den möglichen Formen ist in den Abbildungen 2, 3 und 4 zu sehen.

Abbildung 2 - Beispiele möglicher Formen einfacher ungeteilter Blätter, von links nach rechts: linealisch, lanzettlich, elliptisch, eiförmig, verkehrt eiförmig, kreisrund, keilig, spatelig, rautenförmig, schildförmig, herzförmig, verkehrt herzförmig (Rothmaler, 1967, S. XV, XVI)

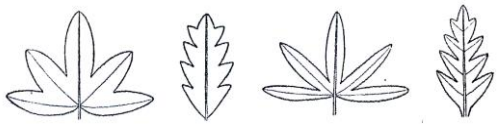


Abbildung 3 - Beispiele geteilter und gespaltener Laubblätter, von links nach rechts: handförmig gespalten, fiederförmig gespalten, handförmig geteilt, fiederteilig (Rothmaler, 1967, S. XIV)

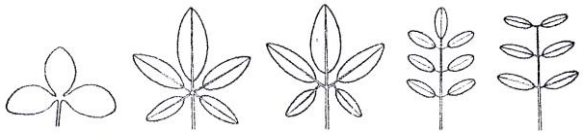


Abbildung 4 - Beispiele zusammengesetzter Blätter, von links nach rechts: 3-zählig, 5-zählig, fußförmig, gefiedert, unpaarig gefiedert (Rothmaler, 1967, S. XIV)

Grundsätzlich kann die Form unterschieden werden in einfache ungeteilte Blätter, bei denen der Rand nicht oder nur leicht eingeschnitten oder gelappt ist (Abbildung 2), und geteilte oder gespaltene Laubblätter (Abbildung 3), deren Blattspreite durch mehr oder weniger tiefe Einschnitte geteilt sind (Welle, 2014, S. 32, 34). Besteht die Blattspreite aus mehreren selbstständigen Blattteilen, spricht man von zusammengesetzten Blättern (Rothmaler, 1967, S. XIV). Die Anordnung der Blättchen genannten Teile ist eine weitere Möglichkeit der Differenzierung (Abbildung 4).

Zur Variation innerhalb der Blattformen tragen auch die Form des Spreitengrundes und der Spreitenspitze bei, welche in Abbildung 5 zu sehen sind.

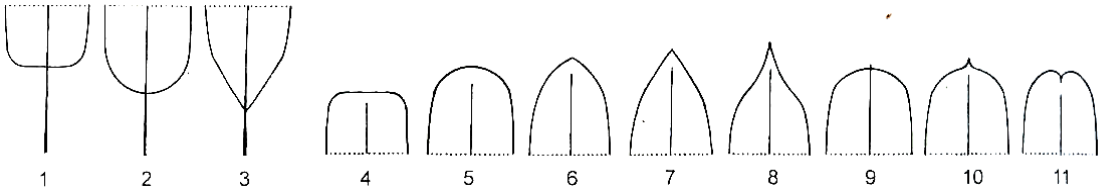


Abbildung 5 - Formen des Spreitengrundes und der Spreitenspitze, Spreitengrund (1 bis 3) von links nach rechts: gestutzt, abgerundet, keilig, | Spreitenspitze (4 bis 11) von links nach rechts: gestutzt, abgerundet, stumpf, spitz, zugespitzt, stachelspitz, bespitzt, ausgerandet (Rothmaler, 2011, S. 886)

Der Blattrand kann als weiteres Unterscheidungsmerkmal dienen. Die Gliederung nach Rothmaler (1967) ist in Abbildung 6 dargestellt.

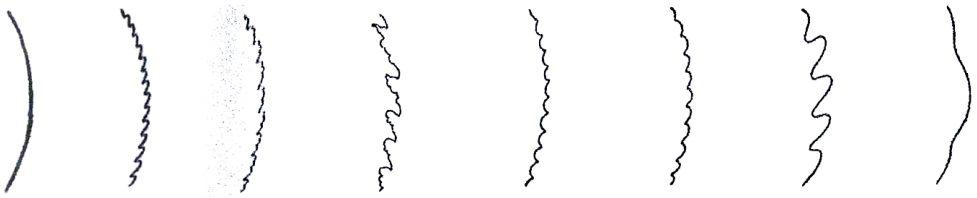


Abbildung 6 - Formen von Blatträndern, von links nach rechts: ganzrandig, gesägt, doppelt gesägt schrotsägeförmig, gezähnt, gekerbt, gebuchtet, geschweift (Rothmaler, 1967, S. XVI, XVII)

Nicht alle Blätter weisen, abhängig von der Art der Anheftung an die Sprossachse, einen Stiel auf. Bei gestielten Blättern ist der Blattstiel deutlich ausgebildet, während stengelumfassende, durchwachsene oder sitzende Blätter keinen Stiel ausbilden. (Rothmaler, 1967, S. XII)

Weiter kann man Blätter anhand ihrer Anordnung an der Sprossachse in wechselständige, gegenständige, quirlständige und bodenständige Blätter differenzieren (Rothmaler, 1967, S. XIII). Ferner dienen der Unterscheidung die Beschaffenheit des Blattes, die sich in ledrig, krautig, fleischig und häutig unterteilen lässt (Rothmaler, 1967, S. XVII) und der Spreitenquerschnitt, der die Art der Rollung oder Faltung des Blattes beschreibt (Rothmaler, 2011, S. 887).

2.1.2.3. Blattnervatur

Die Blattnerven, oder synonym Blattadern, sind das Blatt durchziehende Leitbündel. Ihre Aufgabe ist der Transport von Stoffen und die Wahrung der Festigkeit der Blattspreite (Welle, 2014, S. 86). Die Anordnung der Nerven kann zwischen den Arten variieren.

Nach Rothmaler (1967, S. XIII) unterscheidet man nach dem Verlauf der Blattnerven Blätter in:

- streifennervig: gleich starke Nerven verlaufen vom Blattgrund ohne Verzweigung nebeneinander zur Blattspitze
- netznervig: von einer oder mehreren Hauptnerven gehen kleinere Seitennerven ab, die in ein engmaschiges Adernetz übergehen. Weiter untergliedert man netznervige Blätter in:

- fiedernervig: die Seitennerven gehen zu beiden Seiten von einem Hauptnerv ab
- fingernervig: die größeren Nerven gehen strahlenförmig vom Blattstiel aus

Beispiele von Blättern, unterschieden in der Art des Verlaufs ihrer Blattnerven, sind in Abbildung 7 aufgeführt.

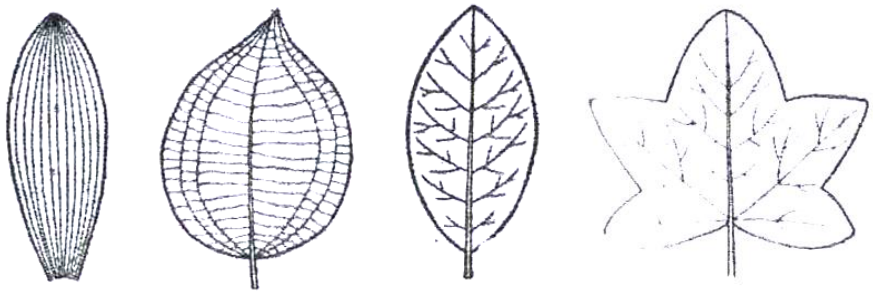


Abbildung 7 - mögliche Verläufe der Blattnerven in Laubblättern, von links nach rechts: streifennervig, netznervig, fiedernervig, fingernervig

2.2. Digitale Bildverarbeitung

2.2.1. Diskrete Bildrepräsentation

Da Computer keine kontinuierlichen analogen Bilder verarbeiten können, werden diese im einfachsten Fall in ein zweidimensionales Gitter überführt. Die meist mit der Matrixnotation angegebene Position innerhalb des Gitters deckt eine rechteckige Region des Bildes ab, die von einem Pixel repräsentiert wird. Der Wert des Pixels gibt die mittlere Intensität der Bestrahlung in dieser Bildregion wieder. (Jähne, 2012, S. 110) Ist das verständlich?

Die Auflösung des diskreten Bildes bestimmt die Anzahl der Pixel und somit den Detailgrad der Darstellung. Laut Jähne (2012, S. 111) „sollte die Pixelgröße kleiner sein als die kleinsten Objekte, die untersucht werden sollen“. Dagegen steht, dass der Aufwand der Bildverarbeitung mit steigender Pixelzahl zunimmt (Jähne, 2012, S. 111).

Ein weiterer Aspekt der Auflösung ist die Tiefe. Diese bezieht sich auf die Anzahl der Bits, die einem Pixel zur Verfügung stehen, um den Intensitätswert zu speichern. Bei 8-Bit Tiefe lassen sich beispielsweise $2^8 = 256$ verschiedene Intensitäts- oder Farbwerte abbilden. (Burdick, 1997, S. 17)

Mit einem Wert je Pixel können lediglich Graustufen dargestellt werden. Für die Repräsentation von Farben werden jedoch mehr Dimensionen benötigt. Bei den Farbräumen handelt es sich um dreidimensionale orthogonale Räume, deren Achsen senkrecht zueinander stehen. Im Falle des RGB-Farbraums bilden die Achsen die Intensitäten der Farben Rot, Grün und Blau.

Der RGB-Farbraum ist additiv, das heißt alle Farben werden durch Addition der einzelnen Farbwerte abgeleitet, beginnend bei Schwarz, wenn kein Farbwert gesetzt ist. In der Bildverarbeitung werden häufig RGB-Bilder verwendet, da Computermonitore ihre Darstellung nativ, also direkt unterstützen. (Burdick, 1997, S. 22)

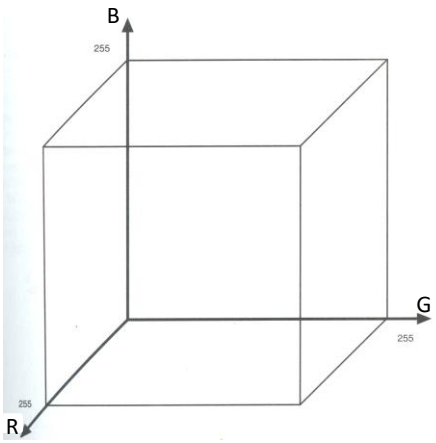


Abbildung 8 - dreidimensionaler RGB-Farbraum mit den Achsen R (Rot), G (Grün) und B (Blau) (Burdick, 1997, S. 23)

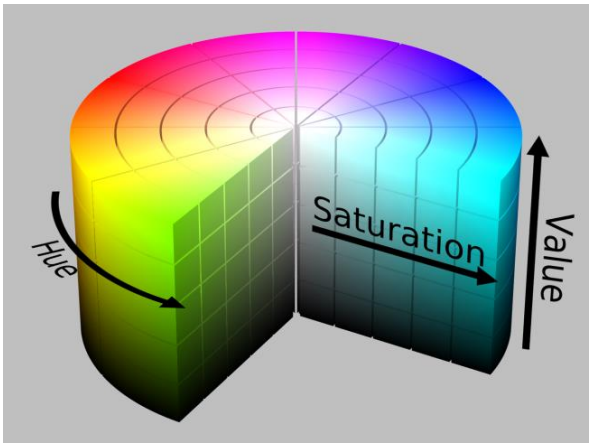


Abbildung 9 - dreidimensionaler HSV-Farbraum, dargestellt in zylindrischer Form (Wikipedia, 2017)

Das RGB-System entspricht allerdings nicht der menschlichen Farbwahrnehmung. Eine natürlichere Beschreibung von Farben ergibt sich durch ihre Unterteilung in Farbton (Hue), Sättigung (Saturation) und Intensität (Intensity) im sogenannten HSI-Raum, der ein Polarkoordinatensystem verwendet. (Jähne, 2012, S. 54)

Es gibt diverse Methoden, um Bilder vom RGB-Raum in den HSI-Raum und zurück zu transformieren. Eine spezielle Implementierung ist das HSV-Farbmodell, dessen Farbraum in Abbildung 9 zylindrisch dargestellt ist. Das Modell generiert den Farbton zwischen 0° und 360°, beginnend mit Rot. Die Sättigung liegt im Wertebereich von 0 bis 1, wobei 0 die Abwesenheit von Farbe codiert, also Grau, in Abbildung 9 entlang der zentralen Achse zu sehen. Der Value ist eine Variation der Intensität und liegt zwischen 0 (schwarz) und 1 (weiß). (Burdick, 1997, S. 29)

2.2.2. Binarisierung

Binärbilder sind Bilder mit nur einem Bit je Pixel. Jeder Pixel kann folglich nur einen von zwei Werten annehmen, schwarz oder weiß. In der Bildverarbeitung werden die beiden Werte auch häufig als Vorder- beziehungsweise Hintergrund bezeichnet. (Burger & Burge, 2015, S. 223)

Die Trennung von Vorder- und Hintergrund bezeichnet man als Binarisierung, oder auch Segmentierung, und kann zur Freistellung von Bildobjekten dienen (Jähne, 2012, S. 541). Anwendung finden die resultierenden Binärbilder bei morphologischen Filtern (Abschnitt

2.2.3.4), zur Extraktion von Konturen im Bild (Abschnitt??) und zur Bestimmung momentbasierter geometrischer Eigenschaften von segmentierten Objekten (Abschnitt **Fehler! Verweisquelle konnte nicht gefunden werden.**).

Um ein Bild, meist ein Graustufenbild, zu binarisieren, wendet man Schwellwertoperationen an. Die Pixel werden, anhand ihrer Intensität α, in einer Punktoperation in zwei Klassen unterteilt. Die Klasse dient der Zuweisung von einem von zwei festen Werten, α_0 und α_1. Das heißt, dass Pixel mit einer Intensität unterhalb eines gewählten Schwellwertes q den Wert von α_0 erhalten, darüber den Wert α_1 (Burger & Burge, 2015, S. 61):

$$f(\alpha) = \begin{cases} \alpha_0 & \text{für } \alpha < q \\ \alpha_1 & \text{für } \alpha \geq q \end{cases} \quad (1)$$

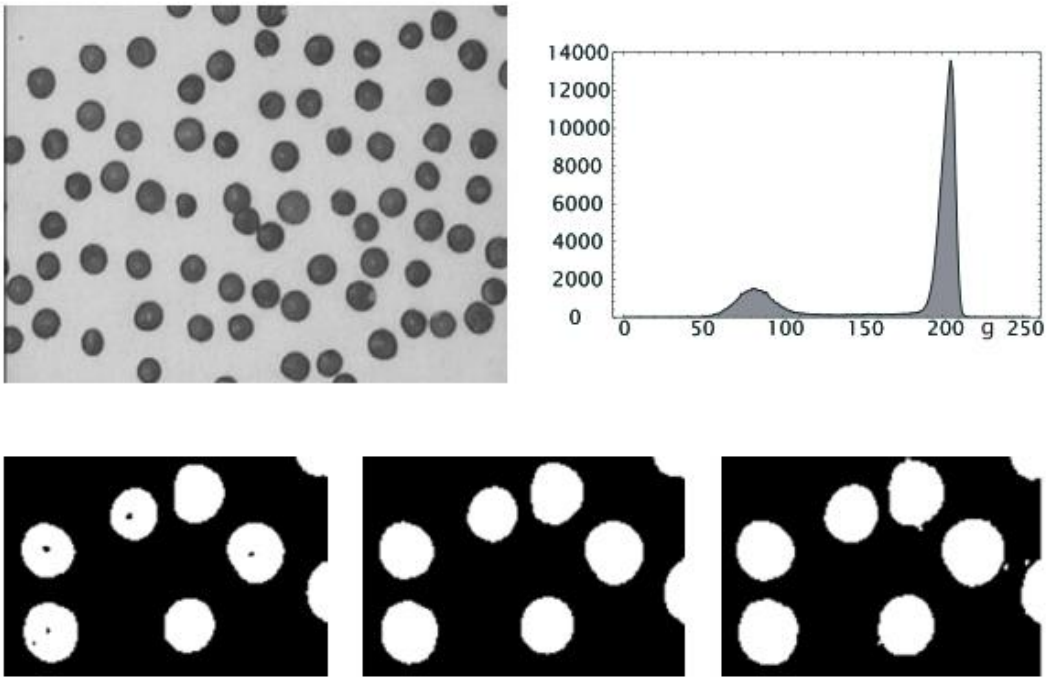


Abbildung 10 - Graustufenbild (links oben) mit seinem Histogramm (rechts oben), Binarisierung mit globalen Schwellwerten von 110, 147 und 185 (unten, von links nach rechts) (Jähne, 2012, S. 543)

Die Ermittlung eines geeigneten Schwellwertes ist die Aufgabe von automatischen Schwellwertoperationen. Viele Verfahren zur Schwellwertberechnung werten dazu die Histogramme des Ausgangsbildes aus. Histogramme sind Darstellungen von Häufigkeitsverteilungen bestimmter Merkmale. Im Falle von Bildern zeigen Histogramme die Häufigkeit der auftretenden Intensitätswerte an (Burger & Burge, 2015, S. 42).

Bei Verfahren zur automatischen Schwellwertbestimmung wird zwischen globalen und adaptiven Methoden unterschieden. In Abbildung 10 ist ein 8-Bit Graustufenbild (links oben) mit zugehörigem Histogramm (rechts oben) und die aus globalen Schwellwertoperationen resultierenden Binärbilder (unten) zu sehen. Globale Schwellwertmethoden bestimmen für ein

Graustufenbild einen einzigen optimalen Schwellwert, anhand dessen die Pixel dem Vorder- oder Hintergrund zugeordnet werden (Burger & Burge, 2015, S. 268). Globale Schwellwerte können bei ungleichmäßiger Beleuchtung im Bild unerwünschte Ergebnisse erzielen. Bei adaptiven Methoden wird daher für jeden einzelnen Bildpunkt ein angepasster Schwellwert bestimmt (Burger & Burge, 2015, S. 291).

Otsu ist ein Beispiel für eine globale histogrammbasierte Schwellwertbestimmung. Das Verfahren versucht die Varianz, also die Streuung der Grauwerte, innerhalb der beiden zu bildenden Klassen zu minimieren. Die Varianz kann direkt aus dem Histogramm des Bildes abgeleitet werden. Gleichzeitig wird sichergestellt, dass die Trennung zwischen den Klassen möglichst groß ist. Dazu müssen ihre Mittelwerte möglichst weit auseinander liegen. (Burger & Burge, 2015, S. 275)

2.2.3. Filter

2.2.3.1. Allgemeine Definition

Filter, auch Nachbarschaftsoperationen genannt, betrachten, im Gegensatz zu Punktoperationen, nicht nur den einzelnen Pixel, sondern auch eine kleine Region um ihn herum, die Nachbarschaft. Sie kombinieren die benachbarten Pixel in einer bestimmten Weise und liefern so ein geändertes Bild. Wichtige Aufgaben, die Filter übernehmen können, sind beispielsweise die Unterdrückung von Rauschen, die Korrektur von Störungen oder die Detektion und Unterscheidung lokaler Strukturen im Bild, wie Kanten oder Linien. (Jähne, 2012, S. 296, 297)

Die Größe der Filterregion, also die Größe der Nachbarschaft, ist ein wichtiger Parameter des Filters. Sie bestimmt, wie viele umgebende Pixel zur Berechnung des neuen Wertes für den betrachteten Pixel herangezogen werden. Mit unterschiedlichen Größen erzielt man unter Umständen auch unterschiedlich starke Effekte. (Burger & Burge, 2015, S. 94)

Die Anwendung des Filters erfolgt für jeden Bildpunkt, beispielsweise durch zeilenweises Ablaufen des Bildes, wie in Abbildung 11 zu sehen ist. Die grauen Bereiche in der Abbildung stellen bereits betrachtete und geänderte Pixel dar. Wichtig ist, dass für die Betrachtung der Umgebung nur die ursprünglichen Werte des Bildes I herangezogen werden. Die Änderungen werden deshalb separat in einem Ergebnisbild I' gespeichert werden. (Jähne, 2012, S. 303)

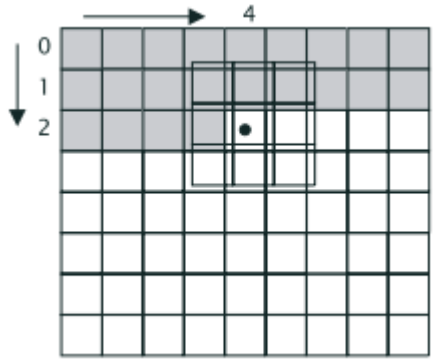


Abbildung 11 - Anwendung eines Filters durch zeilenweises Verschieben der Filtermaske in Pfeilrichtung, graue Zellen sind bereits bearbeitet (Jähne, 2012, S. 303)

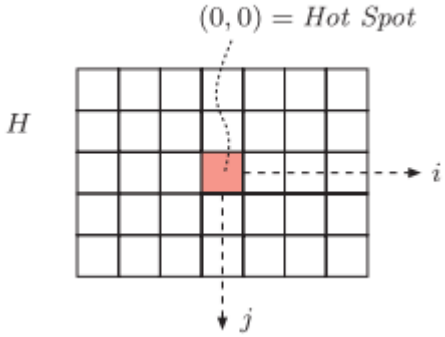


Abbildung 12 - Koordinatensystem der Filtermatrix H mit „Hot Spot" genanntem Ursprung in der Mitte (rot markiert) (Burger & Burge, 2015, S. 96)

2.2.3.2. Lineare Filter

Lineare Filter verknüpfen die Werte der Pixel in der Umgebung in linearer Form durch eine gewichtete Summation. Die Filterfunktion wird durch eine Matrix abgebildet, in der die Gewichtungen der Pixelwerte der Umgebung gespeichert sind. Die Dimension der Matrix gibt die Größe der betrachteten Umgebung an. (Burger & Burge, 2015, S. 95)

Die Filtermatrix, auch Filterkernel genannt, besitzt ein eigenes Koordinatensystem, dessen Ursprung üblicherweise in ihrer Mitte liegt (Abbildung 12). Bei Anwendung der Filtermatrix H, wird sie mit ihrem Ursprung über dem aktuell betrachteten Pixel $I(u, v)$ positioniert. Darauf folgt eine Multiplikation aller gewichteten Elemente $H(i, j)$ innerhalb der Matrixregion R mit den darunter liegenden Bildpunkten $I(u+i, v+j)$ und eine anschließende Summation aller Produkte. Das Ergebnis wird im Ergebnisbild $I'(u, v)$ gespeichert, an derselben Position wie im Ausgangsbild $I(u, v)$. (Burger & Burge, 2015, S. 96)

Auf alle Bildpunkte angewendet, lässt sich die Operation wie folgt formulieren:

$$I'(u,v) = \sum_{(i,j)\in R} I(u+i, v+j) \cdot H(i,j) \qquad (2)$$

Die durchgeführte Operation entspricht einer Faltung $I' = I * H$ mit dem Faltungsoperator (*). In der Mathematik beschreibt die Faltung einen Operator, der zwei Funktionen gleicher Dimension miteinander verknüpft. (Burger & Burge, 2015, S. 105)

Wichtige mathematische Eigenschaften von linearen Faltungen sind die Kommutativität $I * H = H * I$, also die Vertauschbarkeit der Argumente, die Linearität und die Assoziativität $(I * H_1) * H_2 = I * (H_1 * H_2)$. Auf Filter bezogen bedeutet die Assoziativität, dass man das gleiche Ergebnisbild erhält, auch wenn die Reihenfolge der Anwendung verschiedener Filter verändert

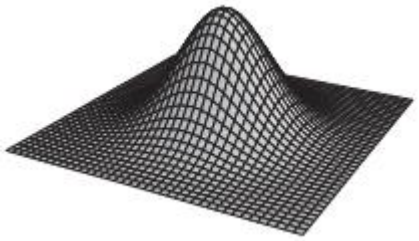
wird, oder sie zu durch paarweise Faltung zu neuen Filtern zusammengefasst werden. (Burger & Burge, 2015, S. 107)

Aus der Assoziativität folgt auch die Separierbarkeit eines einzelnen Filters in kleinere Filter, die nacheinander ausgeführt werden können. Voraussetzung ist, dass sich der zweidimensionale Filter als Produkt von zwei eindimensionalen Filtern darstellen lässt. Die Zerlegung ist dahingehend relevant, dass sich der Berechnungsaufwand mit aus einer großen Filtermatrix separierten einzelnen Filtern reduzieren lässt. (Burger & Burge, 2015, S. 108)

Ein Beispiel für einen solchen separierbaren Filter ist der Gaußfilter $G_\sigma(x, y)$, der in zwei einzelne Funktionen zerlegt werden kann (Burger & Burge, 2015, S. 109):

$$G_\sigma(x,y) = e^{-\frac{x^2+y^2}{2\sigma^2}} = e^{-\frac{x^2}{2\sigma^2}} \cdot e^{-\frac{y^2}{2\sigma^2}} = g_\sigma(x) \cdot g_\sigma(y) \qquad (3)$$

Die Gaußfunktion ist glockenförmig, dargestellt in Abbildung 13 links. Die Standardabweichung σ bestimmt den Radius und somit auch die Größe der Filtermatrix. Der Ursprung der Matrix erhält das maximale Gewicht, die Größe der übrigen Koeffizienten nimmt zu den Rändern hin kontinuierlich ab (Abbildung 13 rechts).



$$\frac{1}{57} \cdot \begin{bmatrix} 0 & 1 & 2 & 1 & 0 \\ 1 & 3 & 5 & 3 & 1 \\ 2 & 5 & 9 & 5 & 2 \\ 1 & 3 & 5 & 3 & 1 \\ 0 & 1 & 2 & 1 & 0 \end{bmatrix}$$

Abbildung 13 - Beispiel für einen Gaußfilter, Darstellung der Filterfunktion H als 3D-Plot (links) und als Filtermatrix (rechts) (Burger & Burge, 2015, S. 103)

Der Gaußfilter ist ein sogenannter Glättungsfilter, oder auch Tiefpassfilter. In der Bildverarbeitung werden durch Glättung die Intensitätsunterschiede benachbarter Bildelemente verringert. Die Unterdrückung hoher Signale führt zu einem Verwischen von Kanten, der Filter ist somit nicht kantenerhaltend. In der menschlichen Wahrnehmung wirken Bilder dadurch unschärfer. (Burger & Burge, 2015, S. 104)

zu Canny überleiten? Gaborfilter?

2.2.3.3. Kanten, Konturen, POI

2.2.3.4. Morphologische Filter

Mit morphologischen Filtern können gezielt Strukturen von Bildern verändert werden (Burger & Burge, 2015, S. 191). Mit ihnen lassen sich die Formen der durch Binarisierung vom Hintergrund getrennten Objekte ändern, indem diesen Bildpunkte hinzugefügt oder aus ihnen gelöscht werden (Jähne, 2012, S. 556).

Das Verhalten eines morphologischen Filters wird durch eine Matrix bestimmt, ähnlich der Filtermatrix bei linearen Filtern. Die Matrix wird hier als Strukturelement H bezeichnet (Abbildung 14, Mitte). (Burger & Burge, 2015, S. 194)

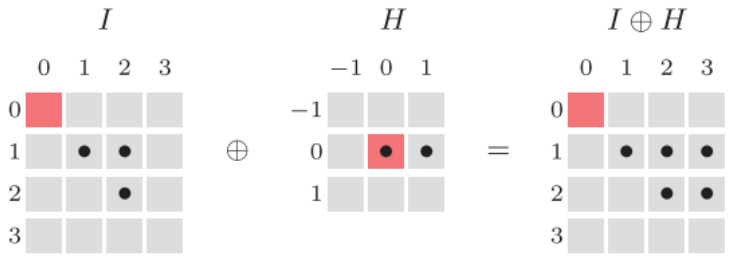


Abbildung 14 - Beispiel einer Dilation auf Binärbild I mit Strukturelement H (Mitte) und dem Ergebnisbild $I \oplus H$ (rechts), Elemente mit dem Wert 1 sind mit • markiert, die Ursprünge der jeweiligen Koordinatensysteme sind rot markiert. (Burger & Burge, 2015, S. 196)

Die grundlegenden Operationen der morphologischen Filter sind Dilation („Wachsen“) und Erosion („Schrumpfen“) (Burger & Burge, 2015, S. 193).

In der Mengennotation ausgedrückt, entspricht die Dilation der Summe aller Kombinationen von Koordinatenpaaren aus den Mengen der Punkte des Binärbildes Q_I und der Punkte des Strukturelementes Q_H (Burger & Burge, 2015, S. 196):

$$I \oplus H = \{(p+q) \mid \text{für alle } p \in I \text{ und } q \in H\} \qquad (4)$$

Laut Burger & Burge (2015, S. 196) kann man *„die Operation auch so interpretieren, dass das Strukturelement H an jedem Vordergrundpunkt des Bilds I repliziert wird“*. Eine Dilation auf einem Binärbild I mit dem Strukturelement H ist beispielhaft in Abbildung 14 illustriert.

Die Erosion ist in Mengennotation definiert als (Burger & Burge, 2015, S. 196):

$$I \ominus H = \{p \in \mathbb{Z}^2 \mid (p+q) \in I, \text{für alle } q \in H\} \qquad (5)$$

Nach Burger & Burge (2015, S. 196) lässt sich die Erosion folgendermaßen interpretieren: *„Eine Position p ist im Ergebnis $I \ominus H$ dann (und nur dann) enthalten, wenn das Strukturelement H –*

mit seinem Ursprung positioniert an der Position p – vollständig im ursprünglichen Bild eingebettet ist, d. h., wo sich für jedes Element in H auch ein entsprechendes Element in I findet".

Morphologische Filter teilen viele, aber nicht alle Eigenschaften mit den linearen Filtern. So sind morphologische Operationen, im Allgemeinen nicht kommutativ. Im Speziellen ist die Erosion, im Gegensatz zur Dilation, nicht kommutativ: $I \ominus H \neq H \ominus I$. Es gelten aber bei aufeinanderfolgender Ausführung von Erosion und Dilation auf das gleiche Bild folgende Regeln: $(I \ominus H_1) \ominus H_2 = I \ominus (H_1 \oplus H_2)$ und $(I \oplus H_1) \oplus H_2 = I \oplus (H_1 \oplus H_2)$. Wie auch lineare Filter, lassen sich morphologische Filter zu neuen Filtern zusammenfassen oder auch in einzelne Filter separieren. (Jähne, 2012, S. 559)

Die elementaren Filter Dilation und Erosion lassen sich zu verschiedenen Operationen kombinieren. Die Operation des Öffnens (Opening) führt eine Erosion zur Entfernung kleiner Strukturen durch, gefolgt von einer Dilation zum Ausgleich der durch die Erosion bewirkten Verkleinerung der Objekte: $I \circ H = (I \ominus H) \oplus H$ (Abbildung 15, Mitte). (Jähne, 2012, S. 561)

Bei der Operation des Schließens (Closing) folgt auf eine Dilation zur Schließung kleiner Löcher, eine Erosion, die die Vergrößerung der Objekte, als Folge der Dilation, durch Schrumpfung wieder ausgleicht (Abbildung 15, rechts). (Jähne, 2012, S. 562)

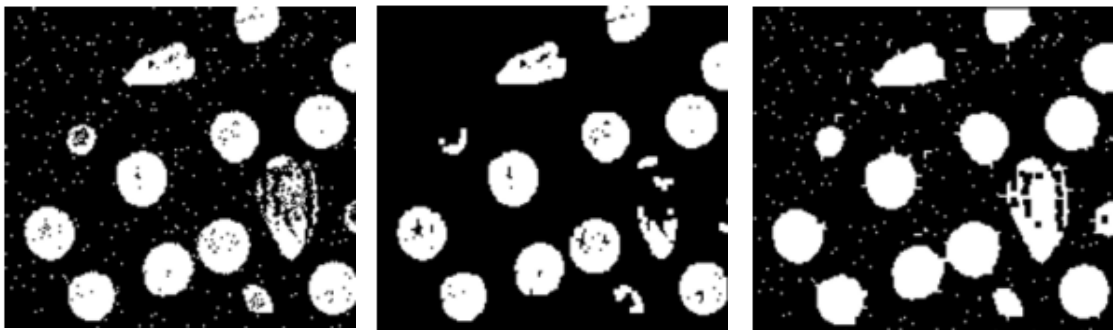


Abbildung 15 - Originalbild (links) nach Ausführung von Opening (Mitte) beziehungsweise Closing (rechts) mit Strukturelementen der Dimension 3 x 3 (nach Jähne, 2012, S. 561, 562)

2.3. Maschinelles Lernen

2.3.1. maschinelle Lernverfahren

Die Entwicklung maschineller Lernverfahren ist ein wichtiger Teilbereich der künstlichen Intelligenz (Ertel, 2016, S. 191). Es handelt sich beim maschinellen Lernen nicht um simples Auswendiglernen, sondern um die Generalisierung gelernter Beispiele, um das Gelernte auf theoretisch unendlich viele neue ungesehene Beispiele anwenden zu können. (Ertel, 2016, S. 192)

Beim maschinellen Lernen unterscheidet man zwischen überwachtem und unüberwachtem Lernen. Bei unüberwachten Lernverfahren sind zu den Trainingsdaten keine Zieldaten verfügbar. Ziel von unüberwachtem Lernen kann unter anderem sein, in den Daten Gruppen von ähnlichen Strukturen mittels Clustering (Kapitel 2.3.5) zu finden oder hochdimensionale Daten auf wenige Dimensionen zu projizieren. Dagegen ist bei überwachtem Lernen zu den zu trainierenden Beispielen als Eingabe die zugehörige Ausgabe bereits bekannt. Zum überwachten Lernen zählen Klassifizierungsprobleme (Kapitel 2.3.2). (Bishop, 2006, S. 3)

2.3.2. Systeme zur Klassifizierung

Bei der Klassifizierung wird aus bereits klassifizierten Trainingsdaten, eine Funktion generiert, die die Klasse eines neuen Objektes anhand seiner Merkmale berechnen kann (Ertel, 2016, S. 193). Die Aufgabe sowohl des Trainings, als auch der Abbildung eines Merkmalsvektors auf einen Klassenwert, übernimmt ein Klassifizierer, schematisch in Abbildung 16 dargestellt. Die Merkmalsvektoren werden im Folgenden auch Features genannt.

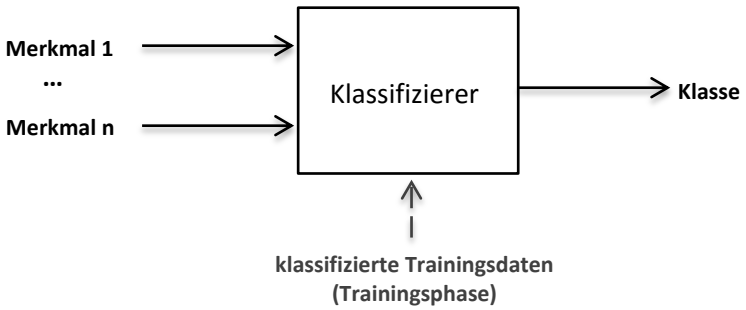


Abbildung 16 - funktionale Struktur eines Klassifikators (nach Ertel, 2016, S. 194)

Bei der Klassifizierung können Fehlentscheidungen auftreten, wenn ein Objekt der falschen Klasse zugeordnet wird. Die Summen der gewichteten Fehler bezeichnet man als Kosten. Ziel beim Training des Klassifizierers ist es, die aus den Fehlentscheidungen entstehenden Kosten zu minimieren. (Ertel, 2016, S. 20)

Zur Minimierung der Kosten lassen sich Entscheidungsregeln definieren und variieren (Duda, Hart, & Stork, 2001, S. 4). Sie lassen sich veranschaulichen, indem man die verschiedenen Eingangsmerkmale in Relation zueinander setzt. Es spannt sich ein mehrdimensionaler Featureraum auf, der anhand der Entscheidungsregel so partitioniert werden kann, dass ausreichend viele Punkte einer partitionierten Region nur einer einzelnen Klasse zugeordnet werden können.

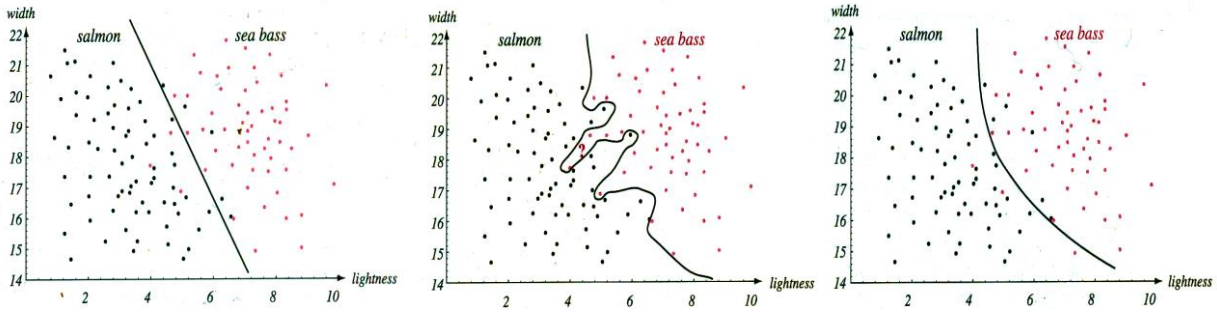


Abbildung 17 - verschieden komplexe Entscheidungsgrenzen in einem zweidimensionalen Featureraum zur Unterscheidung zwischen zwei Klassen - Lachs („salmon“, schwarze Punkte) und Wolfsbarsch („sea bass“, rote Punkte) - anhand zweier gegenübergestellter Merkmale - Breite („width“) und Helligkeit („lightness“) (nach Duda, Hart, & Stork, 2001, S. 5, 6)

Ein zweidimensionaler Featureraum ist in Abbildung 17 illustriert. Hier wird beispielhaft zwischen zwei Klassen anhand von zwei Merkmalen unterschieden. Als Entscheidungsregel dient eine Entscheidungsgrenze, die den Featureraum in die zwei zu unterscheiden Klassen teilt. Dargestellt ist auch die Problematik, eine geeignete Komplexität des Modells zu wählen. Die lineare Grenze in Abbildung 17 (links) lässt noch relativ viele Fehler zu, zu beiden Seiten finden sich auch Punkte der jeweils anderen Klasse. Während in der Mitte die Entscheidungsgrenze so gestaltet ist, dass die Grenze die Klassen perfekt trennt, erscheint diese Trennung zu stark an die Trainingsdaten angepasst. Es könnte sich das Problem ergeben, dass die Klassifizierung zwar auf den Trainingsdaten zu 100 Prozent korrekt ist, neu hinzukommende ungesehene Objekte aber falsch klassifiziert werden. Man spricht auch von „Overfitting“ oder Überanpassung. Das widerspricht der gewünschten Generalisierung des Erlernten. Die rechte Entscheidungsgrenze in Abbildung 17 bildet seinen Kompromiss aus den beiden vorangegangenen Grenzen.

Die Wahl eines geeigneten Modells ist nur ein Teil im Entwurf eines Systems, das anhand von Merkmalen Muster erkennen und Entscheidungen zur Klassifizierung treffen soll. Nach Duda, Hart, & Stork (2001, S. 14) lässt sich der Prozess des Entwerfens, in fünf Schritte unterteilen, dargestellt in Abbildung 18.

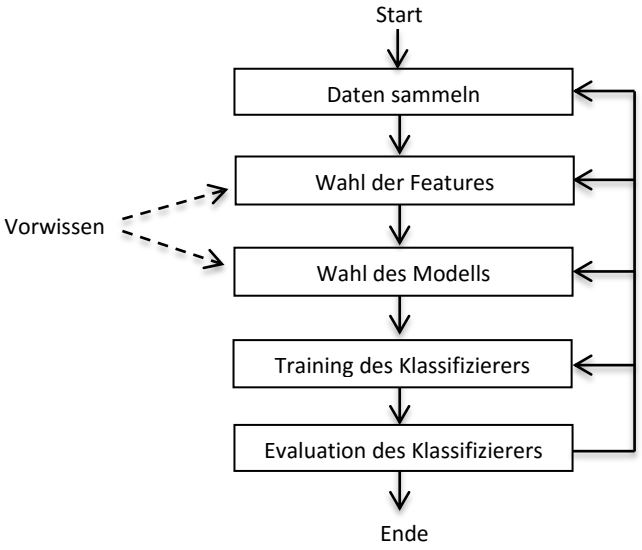


Abbildung 18 - Schema des Entwerfens eines Systems zur Mustererkennung und Klassifizierung
(nach Duda et al., 2001, S. 14)

Laut Duda, Hart, & Stork (2001, S. 14) ist die Wahl der Features ein kritischer Entwurfsschritt und hängt von der Charakteristik der zu untersuchenden Problematik ab. Vorwissen über Merkmale, mit denen sich die Klassen möglicherweise gut differenzieren lassen, kann bei der Auswahl helfen und fließt auch in die Wahl des Modells mit ein. Die Evaluation hat schließlich die Aufgabe, die Performanz des Systems zu überprüfen und Möglichkeiten der Verbesserungen des Systems aufzuzeigen, was wiederum Anpassungen der vorherigen Schritte erfordert. (Duda et al., 2001, S. 14, 15)

Es gibt eine Vielzahl an Modellen, die Entscheidungen im Sinne einer Klassifizierung treffen können. Anhand ihrer Komplexität kann man die zugrundeliegenden in Lazy-Learning (faules Lernen) und Eager-Learning (eifriges Lernen) unterteilen. Beim Lazy-Learning liegt der größte Aufwand nicht in der Lernphase, sondern in der Anwendung auf neue Beispiele. Eager-Learning-Verfahren besitzen deutlich aufwändigere Lernphasen, aber eine effiziente Anwendung. (Ertel, 2016, S. 214)

Beispiele für Lazy-Learning sind die k-Nearest-Neighbour-Methode, lokal gewichtete Regression oder fallbasiertes Lernen. Zu den Modellen mit Eager-Learning zählen Entscheidungsbäume, Bayes-Netze, neuronale Netze und Support-Vektor-Maschinen. (Ertel, 2016, S. 258)

Die beiden letztgenannten Modelle werden in den folgenden Kapiteln 2.3.3 beziehungsweise 2.3.4 näher betrachtet.

2.3.3. Support-Vektor-Maschinen

Support-Vektor-Maschinen basieren auf der Idee, dass sich Daten zweier Kategorien, nach einer geeigneten nicht-linearen Abbildung in einen Raum mit ausreichend großer Dimension, immer durch eine Hyperebene trennen lassen. Die Dimension ist dabei typischerweise deutlich höher als die des originalen Featureraumes. (Duda et al., 2001, S. 259)

Da es meist unendlich viele solcher Ebenen gibt, wird die Hyperebene gesucht, *„die zu beiden Klassen einen möglichst großen minimalen Abstand hat“* (Ertel, 2016, S. 298). Das bedeutet, dass die optimal trennende Hyperebene so gelegt wird, dass ihr parallel laufender Rand, oder auch Margin, möglichst groß ist, ohne dass ein Punkt innerhalb des Randes liegt (siehe Abbildung 19). Je größer der Rand ist, desto besser ist die Generalisierung des Klassifizierers (Duda et al., 2001, S. 262). Die Punkte auf dem Rand nennt man Support-Vektoren. Sie parametrisieren die Hyperebene und haben aufgrund der Definition des Margins alle den gleichen Abstand zu ihr (Ertel, 2016, S. 298).

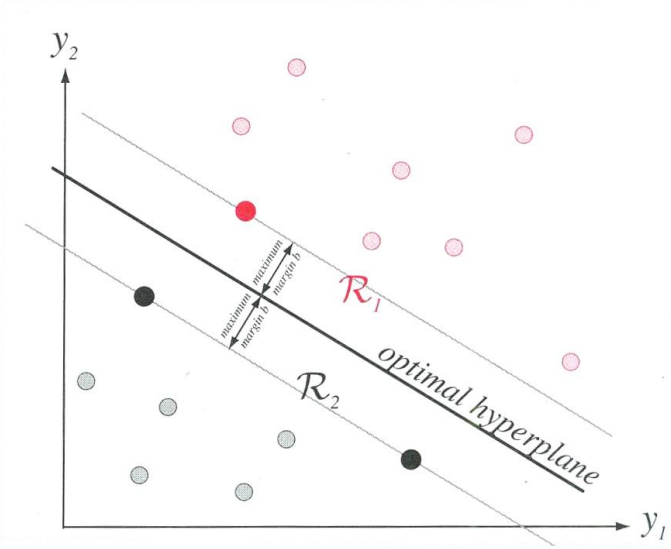


Abbildung 19 - die zwei Klassen R_1 und R_2 trennende optimale Hyperebene („hyperplane“) mit ihren Support-Vektoren (hervorgehobene Punkte) und dem Rand (zwischen Support-Vektoren und Hyperebene)

2.3.4. Künstliche Neuronale Netze

2.3.4.1. Grundstruktur eines neuronalen Netzes

Künstliche neuronale Netze haben ihr Vorbild meist in der Natur. Künstliche Neuronen (Abbildung 21) bilden biologische Neuronen (Abbildung 20) ab. Sie sind allerdings nur äußerst simple Abstraktionen, realisiert als Elemente von Soft- oder Hardware. (Keller, Liu, & Fogel, 2016, S. 8)

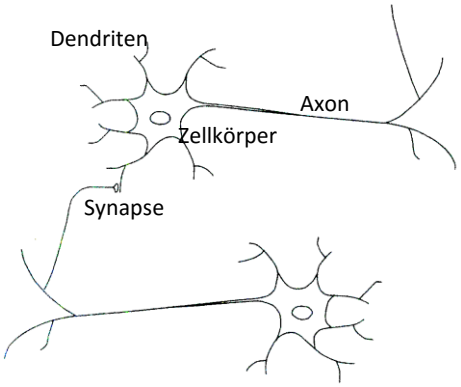


Abbildung 20 - schematische Darstellung von zwei biologischen Neuronen (nach Keller, Liu, & Fogel, 2016, S. 8)

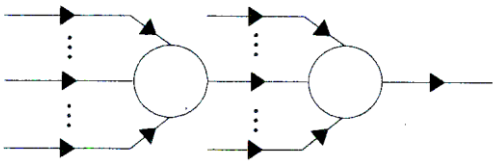


Abbildung 21 - schematische Darstellung von zwei künstlichen Neuronen mit Eingabe- und Ausgabesignalen (Keller et al., 2016, S. 8)

Die Zellkörper biologischer Neuronen können elektrische Spannung speichern. Die Spannung steigt, je mehr Impulse von anderen Neuronen ankommen. Ist ein bestimmter Schwellwert überschritten, feuert das Neuron und leitet einen Impuls über das Axon an andere verknüpfte Nervenzellen. (Ertel, 2016, S. 266)

Biologisch inspirierte künstliche neuronale Netze bilden diesen Vorgang mit einem Aktivierungspotential in einer diskreten Zeitskala ab. Das Aufladen des Potentials erfolgt durch gewichtete Summation aller eingehenden Verbindungen $x_1,\ldots,x_n$. Für ein Neuron i wird dies wie folgt formuliert (Ertel, 2016, S. 269):

$$x_i = f\left(\sum_{j=1}^{n} w_{ij}x_j\right) \tag{6}$$

wobei w_{ij} die Gewichtung zwischen Neuron i und dem am Eingang verknüpften Neuron j ist. Die Aktivierungsfunktion f wird auf die gewichtete Summe angewandt und bestimmt welches Ergebnis x_i als Ausgabe an die benachbarten Neuronen weitergeleitet wird. Ein einfaches Beispiel für eine Aktivierungsfunktion, das ist vor allem bei binären Neuronen sinnvoll ist, ist die Schwellwert- oder Stufenfunktion, die in Abbildung 22 links abgebildet ist (Ertel, 2016, S. 269):

$$H_\Theta = \begin{cases} 0 & \text{falls } x < \Theta \\ 1 & \text{sonst} \end{cases} \tag{7}$$

Sigmoid-Funktionen sind die am häufigsten eingesetzten Aktivierungsfunktionen in neuronalen Netzen. Sie halten *„eine exzellente Balance zwischen linearem und nicht-linearem Verhalten“* (nach Keller u. a., 2016, S. 29). Ein Beispiel für eine Sigmoid-Funktion, wie in Abbildung 22 in der Mitte zu sehen, ist die logistische Funktion:

$$f(x) = \frac{1}{1 + \mathrm{e}^{-\mathrm{ax}}} \tag{8}$$

wobei a die Steigung der Sigmoid-Funktion parametrisiert.

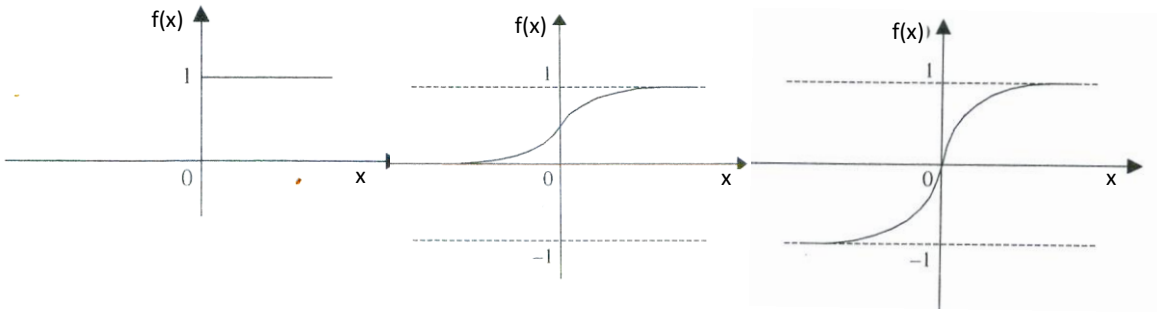


Abbildung 22 - Aktivierungsfunktionen von links nach rechts: Schwellwertfunktion, logistische Funktion, hyberbolischer Tangens (nach Keller u. a., 2016, S. 28, 29, 30)

2.3.4.2. Multilayer Perceptron und Backpropagation

Wie biologische Neuronen lassen sich auch künstliche Neuronen zu komplexen Netzwerken zusammenfügen, die einen hohen Grad der Verknüpfung vorweisen (Keller et al., 2016, S. 9). Netze mit mindestens einer versteckten Schicht sind in der Lage, nicht-lineare Abbildungen zu erlernen (Ertel, 2016, S. 293).

Dazu zählen die mehrlagigen Perzeptrons, die sogenannten Multilayer Perceptrons, schematisch in Abbildung 23 dargestellt. Ihre Merkmale sind nach Keller et al. (2016, S. 35):

- Jedes Neuronenmodell im Netzwerk beinhaltet eine nicht-lineare, differenzierbare Funktion
- Das Netzwerk enthält eine oder mehr Schichten, die versteckt sind vor den Eingangs- und Ausgangsmodi
- Das Netzwerk hat einen hohen Grad an Konnektivität, deren Umfang durch die synaptischen Gewichte des Netzwerks bestimmt wird

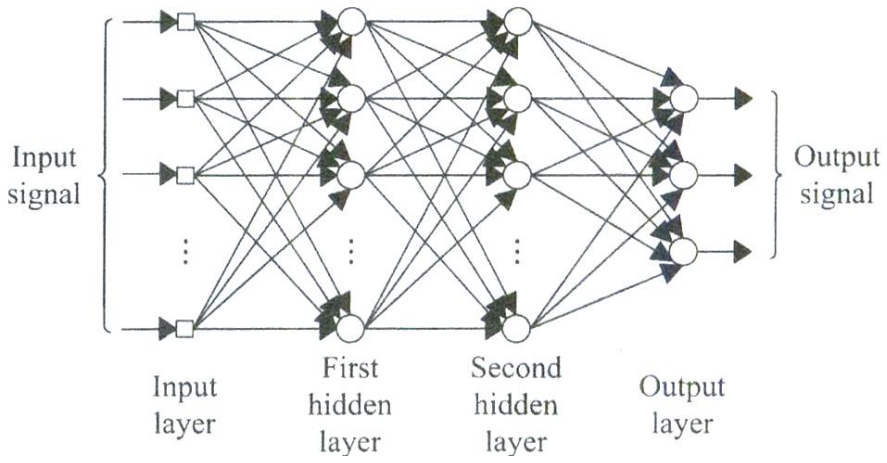


Abbildung 23 -Schema eines Multilayer-Perceptron mit zwei versteckten Schichten („hidden layer"), der Eingabeschicht („input layer") und der Ausgabeschicht („output layer"). Das Eingangssignal („input signal") wird vorwärts durch das Netz propagiert, mit dem Ausgangssignal („output signal") als Ergebnis (Keller et al., 2016, S. 36)

Üblicherweise erfolgt das Training von Multilayer Perceptrons mit der Backpropagation genannten Lernmethode. Lerneffekte werden während des Trainings durch Änderungen der Gewichtungen w zwischen den Neuronen erzielt, in Abhängigkeit von den auftretenden Fehlern. Die Fehlerrate wird anhand einer summierten quadratischen Fehlerfunktion ermittelt (Ertel, 2016, S. 292):

$$E_p(w) = \frac{1}{2}\left(\sum_{k \in Ausgabe} (t_k^p - x_k^p)^2\right) \tag{9}$$

wobei t^p die gewünschte Antwort und x^p die tatsächliche Antwort des Netzes für ein Trainingsmuster p darstellt. Ziel ist es den Wert der Fehlerfunktion zu minimieren.

Die Richtung des stärksten Anstiegs der Fehlerfunktion wird mit einem Fehlergradienten angezeigt. Zur Bestimmung eines Minimums wird deshalb der Richtung des negativen Gradienten gefolgt (Ertel, 2016, S. 289). Die Anpassung der Gewichte anhand des negativen Gradienten der Fehlerfunktion E_p aus (9) wird folgendermaßen formuliert (Ertel, 2016, S. 292):

$$\Delta_p w_{ji} = -\eta \frac{\delta E_p}{\delta w_{ji}} = \eta \delta_j^p x_i^p \tag{9}$$

Der Parameter η ist die Lernrate zur Festlegung der Stärke der Gewichtsänderung und bestimmt die Geschwindigkeit der Konvergenz Richtung Minimum der Fehlerfunktion (Ertel, 2016, S. 289). δ_j^p ist das Fehlersignal des Neurons j und abhängig davon, ob dieses ein Neuron der Ausgabe- oder der versteckten Schicht ist. Für ein verstecktes Neuron j errechnet sich δ_j^p aus den Änderungen aller δ_k^p der nächsthöheren Schicht. Bei einem Ausgabeneuron ist δ_j^p proportional zum Fehler des Netzes.

Der generelle Ablauf der Backpropagation beginnt mit einer Initialisierung aller Gewichte mit zufälligen Werten. Darauf folgen die Schritte:

- das Trainingsmuster wird an der Eingabeschicht angelegt
- das Muster wird vorwärts durch das Netz propagiert, das heißt die Aktivierungen werden ab der ersten versteckten Schicht für alle Neuronen berechnet
- der summierte quadratische Fehler E_p wird berechnet

- der Fehler wird durch alle Schichten rückwärts propagiert und die Gewichte angepasst

Die Schritte werden für jedes Trainingsmuster wiederholt, bis die Gewichte konvergiert sind oder eine festgelegte diskrete Zeitschranke erreicht ist.

2.3.5. k-Means Clustering

Ziel des Clusterings ist es, in einem multidimensionalen Raum Gruppen oder „Cluster“ von Daten zu identifizieren (Bishop, 2006, S. 424). Cluster sind Anhäufungen von benachbarten Punkten, deren Abstand untereinander typischerweise kleiner ist als der Abstand zwischen Punkten aus unterschiedlichen Clustern (Ertel, 2016, S. 245).

Ein sehr simples Verfahren zur Clusterbildung ist k-Means, bei dem die Anzahl k der zu bildenden Cluster von vornherein bekannt sein muss. Die k Clusterzentren werden dabei zunächst zufällig oder per Hand initialisiert. Darauf folgen zwei Schritte (Ertel, 2016, S. 246):

- Zuordnung der Datenpunkte zu einem Clusterzentrum
- Neuberechnung der Clusterzentren

Diese beiden Schritte werden wiederholt, bis sich die Position der Clusterzentren nicht mehr verändert. Beispielhaft mit $k = 2$ ist der Ablauf in Abbildung 24 skizziert, wobei die Konvergenz nach drei Iterationen erreicht ist.

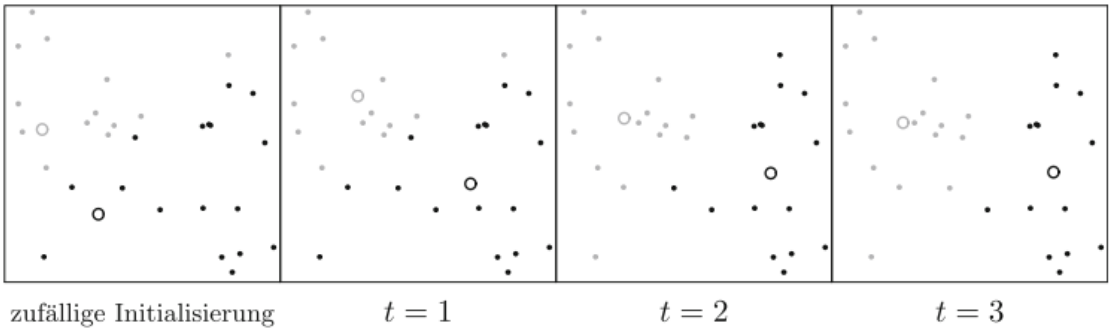


Abbildung 24 - k-Means mit zwei zu bildenden Clustern (hellgrau und schwarz) nach der Initialisierung (links) und nach t Iterationen (ab 2. von links). Die Zuordnung der Datenpunkte zu den Clusterzentren (Kreise) ist durch die Graustufe (hellgrau oder schwarz) codiert. (Ertel, 2016, S. 247)

Die Zuordnung der Punkte zum nächstgelegenen Zentrum erfolgt mit Abstandsfunktionen wie beispielsweise dem euklidischen Abstand (Ertel, 2016, S. 245). Das jeweilige Clusterzentrum μ für die $x_1,\ldots,x_l$ Punkte, die dem Clusterzentrum zugeordnet sind, wird in jeder Iteration folgendermaßen neu berechnet :

$$\mu = \frac{1}{l}\sum_{i=1}^{l} x_i \qquad (10)$$

3. Konzeption

3.1. Zielstellung

Ziel der Arbeit ist es, ein System zu entwerfen, dass bei Eingabe von spezifischen morphologischen Merkmalen, diese einer bestimmten Pflanzenart zuordnen kann. Der Fokus der Arbeit liegt dabei auf der Erkennung von Pflanzen anhand ihrer Blattmerkmale, weshalb auch die Eingangsdaten des Systems aus Bildern von Blättern extrahiert werden.

Das System muss Muster in den Bildern erkennen, die sich innerhalb der gleichen Art ähneln, sie aber gleichzeitig von anderen Arten abgrenzen. Untersucht werden soll, in welchem Ausmaß die verschiedenen Merkmale der Blätter der automatisierten Differenzierung zwischen den Arten beitragen können.

Die Entwicklung des Systems erfolgt nach dem Entwurfsmuster von Duda et al. (Kapitel 2.3.2), was viele Tests und eine ständige Anpassung der Entwicklungsschritte erfordert. Der erste Schritt ist die Sammlung von Daten. Es wird ein eigener Datensatz erstellt und wei

In den Schritt der Auswahl der Features soll möglichst viel Vorwissen über die Beschaffenheit des Blattes einfließen, und so abbilden, wie der Mensch Pflanzen anhand der Blätter unterscheidet. In Frage kommende Charakteristiken sind dabei die Blattform, die Blattränder und die Struktur der Blattnervatur.

Es wird angenommen, dass sich Blätter besonders gut über die Blattform unterscheiden lassen, da diese zwischen den Arten sehr variabel ist. Die Blattnervatur ist dagegen weniger divers. Hinsichtlich der Blattnervatur soll zudem überprüft werden, ob die Beschreibbarkeit von der betrachteten Blattseite abhängt. Von der menschlichen Wahrnehmung ausgehend, sind die Nerven auf der Unterseite des Blattes meist deutlich besser erkenn-und beschreibbar.

Weiter soll untersucht werden, wie sich die einzelnen ausgewählten Features miteinander kombinieren lassen, um ein Optimum an deskriptiver Leistung zu erzielen. Die Annahme ist, dass sich eine Kombination aus jeweils einem Feature der Form und aus einem Feature der Textur bilden lässt, die nahe dem Optimum liegt. Features der gleichen Kategorie, also Form oder Textur, liefern kombiniert vermutlich viele redundante Informationen.

Auch die Abhängigkeit der Performanz von der Anzahl an Trainingsdaten soll überprüft werden. Es wird angenommen, dass bei steigender Zahl der für das Training verfügbaren Daten auch die Erkennungsrate des trainierten Systems zunimmt. Ein Optimum wird vermutlich erst bei sehr hoher Anzahl an Daten erreicht, die außerhalb der verfügbaren Datenmenge liegt.

Die Klassifizierung soll durch ein künstliches neuronales Netz erfolgen. Es muss getestet werden, wie gut die Performanz des Netzes gegenüber einem Vergleichsmodell ist. Dazu zählen auch Tests mit verschiedenen Parametern und unterschiedlichen Graden der Komplexität des Netzes. Ziel dabei ist es, den Grad zu finden, an dem die Erkennungsleistung ausreichend ist und mit steigender Komplexität nicht mehr wesentlich verbessert werden kann.

Evaluation und Tests sollen aus einer zu implementierenden Oberfläche heraus erfolgen, die die einzelnen Komponenten des Systems miteinander verbindet und steuert.

3.2. Abgrenzung

Zur Verringerung der Komplexität sollen Blätter nur in ihrer Grundform als Laubblatt betrachtet werden. Analysen von Metamorphosen des Blattes, wie beispielsweise der Blütenblätter, könnten das geplante System erweitern, sind aber nicht Teil dieser Arbeit. Zudem werden Blätter nur einzeln betrachten, die Blattanordnung soll kein Kriterium sein.

Die geplante Oberfläche dient nur dem Test und der Evaluation des Systems. Somit wird auch nicht gesondert auf Ansprüche hinsichtlich der Usability der Oberfläche geachtet.

Im Rahmen dieser Arbeit wird keine über die Oberfläche hinausgehende Anwendung des Systems entwickelt. Das schließt eine eventuelle Client-Server-Anwendung mit dem Klassifizierungssytem als Backend und einer Smartphone-App als Frontend ebenfalls aus.

3.3. Einordnung der Arbeit in den Kontext

Das Themenfeld der Klassifizierung von Objekten durch Verfahren des maschinellen Lernens bringt stetig neue Ansätze und Techniken hervor. In jüngster Zeit sind auch zahlreiche Arbeiten erschienen, die sich speziell der Blatterkennung anhand von Form- und Texturmerkmalen widmen. Aufgrund der Fülle kann folgend nur ein Auszug der Veröffentlichungen aufgeführt werden, die entweder für den Systementwurf der vorliegenden Arbeit relevant sind oder interessante weiterführende Ansätze bieten.

Bereits 2007 entwickelten Wu et al. ein System zur Klassifizierung von Blättern. Dazu erstellten sie einen eigenen „Flavia“ genannten Bild-Datensatz von 32 Arten. Die Segmentierung der Bilder erfolgt mit einem einfachen Schwellwertverfahren mit festgelegtem Schwellwert. Für die Beschreibung der Merkmale verwenden sie 12 verschiedene geometrische Merkmale, wie Fläche, Länge, Breite, Seitenverhältnisse und Formfaktor. Zudem extrahieren sie die Blattnerven mittels morphologischem Opening auf den Graustufenbildern und beschrieben sie mit „vein features“, auf die sie in ihrer Ausarbeitung aber nicht näher eingehen. Die Menge der

Features wird mit einer Hauptkomponentenanalyse (PCA) vereinfacht und reduziert. Zur Klassifizierung nutzen sie ein Probabilistic neural network (PNN), da es nach ihrer Aussage schnell arbeitet und eine simple Struktur hat. Ihr System erzielt auf dem eigens erstellten Datensatz eine Genauigkeit von 90% in der Erkennung (Wu et al., 2007).

Kumar et al. veröffentlichten 2012 die Grundlagen zur Entwicklung des elektronischen Feldführers „Leafsnap“ für Smartphones in Zusammenarbeit mit der Columbia University, der University of Maryland und der Smithsonian Institution. Auch sie erstellten für ihr System einen eigenen Datensatz. Die Bilder werden mit einer Expectation-Maximization-Methode im HSV-Farbraum segmentiert. Als Features dienen über mehrere Skalen gebildete sogenannte Curvature-Histogramme. Die Klassifikation erfolgt mit einer Nearest-Neighbors-Methode. Das System liefert als Ergebnis einen Rang der Wahrscheinlichkeiten der Zuordnung zu einer Spezies. Sie geben als Messergebnis an, dass sie eine Quote von 96,8% erzielen, bei der die korrekte Spezies unter den Top-5 der Wahrscheinlichkeiten auftaucht (Kumar et al., 2012).

Einen allgemeinen Ansatz um Formen zu beschreiben, liefert der invariante Inner Distance Shape Context (IDSC) von Ling & Jacobs (2007). Er erweitert den von Belongie, Malik, & Puzicha (2002) beschriebenen Shape Context, welcher Punkte auf der Kontur über die Winkel und die euklidische Distanz zu allen anderen Konturpunkten in Beziehung setzt. Die Erweiterung erfolgt dahingehend, dass der IDSC die euklidische Distanz ersetzt mit einer Distanz in Bezug auf die kürzesten Wege entlang der Kontur. Dadurch wird eine Invarianz der Artikulation der Form erzielt. In derselben Veröffentlichung schlagen Ling & Jacobs auch eine Erweiterung des IDSC vor: den Shortest Path Texture Content (SPCT). Mit dem SPCT lassen sich Form und Textur gleichzeitig beschreiben, indem dem Context Informationen der Texturgradienten auf den Geraden zwischen den Konturpunkten hinzugefügt werden. Die Klassifizierung erfolgt mit Methoden des dynamischen Programmierens. Mit IDSC erreichen Ling & Jacobs (2007) im Swedish Leaf Dataset eine Erkennungsrate von 94,13% um Vergleich zur ursprünglichen Shape Context-Methode von 88,12%. Mit dem SPTC verbessern sie die Rate noch weiter auf 95,33%

Zhao et al. (2015) passen mit dem Independent Inner Distance Shape Context (I-IDSC) in ihrer Arbeit den IDSC speziell an die Erkennung von Blättern an. Dabei heben sie die Unterscheidung der globalen von den lokalen Strukturen von Blättern hervor, indem sie die Inner Distance Shape Contexts auf verschiedenen Skalen beschreiben und die Größe der betrachteten Nachbarschaften der Konturpunkte an die Skalierungsgröße anpassen. Statt dynamischer Programmierung zur Klassifizierung suchen sie Muster in den Shape Contexts, um sie als Eingangsdaten für ein neuronales Netz nutzen zu können. Zhao et al. erzielen in ihren Tests im

Swedish Leaf Dataset mit einer Erkennungsrate von 97,07% bessere Ergebnisse im Vergleich zu IDSC.

Ein weiterer interessanter Ansatz zur Beschreibung von Formen sind die von Cao, Wang & Brown (2016) vorgestellten rotations-, skalierungs- und translationsinvarianten R-Angles. Ein R-Angle beschreibt den zum Schwerpunkt hin geöffneten Winkel zwischen einem Konturpunkt und den Schnittpunkten eines um den Punkt gezogenen Kreises mit der Kontur, wobei der Radius des Kreises von der betrachteten Skalierung abhängt. Mit ihrer Methode erzielen Cao, Wang & Brown nach eigener Aussage bessere Ergebnisse als vergleichbare Methoden wie Shape Context und Inner Distance Shape Context.

Für die Beschreibung der Textur sind Verfahren wie die Local Binary Patterns, Co-Occurence Matrizen und Gabor-Features gebräuchlich. So nutzen Arun et al. (2013) beispielsweise Local Binary Patterns zusammen mit Grey Tone Spatial Dependency Matrizen zur Beschreibung der Blatttextur. Chaki et al. (2015) verwenden die Gray Level Co-occurrenceMatrix und Gaborfilter in Kombination mit den formbeschreibenden Hu-Momenten und Curvelet-Features.

Auffällig ist, dass sich viele Veröffentlichungen auf eine Beschreibung der Form konzentrieren. Die Beschreibung der Textur erfolgt oft nur zur Unterstützung der Form-Merkmale. Ein System, das die Blattnerven getrennt betrachtet, wurde von Larese et al. (2014) erarbeitet. Als Besonderheit können sie durch ihre angebrachte Methode auch verschiedene Variationen innerhalb einer Art unterscheiden. Zur Segmentierung der Adern nutzen sie eine erweiterte Form der Hit or Miss Transform auf Graustufenbildern. Für die Extraktion der Features nutzen sie die Implementierung von Price et al. (2011), die die grafische Oberfläche „Leaf GUI“ zur Ermittlung und Darstellung biologischer Merkmale der Nervatur von Blättern erstellten. Price et al. wandeln dazu Bilder von Blattnerven in eine Netzwerkpräsentation mit Knoten und Kanten um. Mit ihrer Methode erzielen sie eine durchschnittliche Erkennungsrate von 92% bei einer Klassifikation innerhalb eines Datensatzes mit drei Unterarten von Sojabohnen und 67% bei Klassifikation von im Feld getätigten Fotos (Larese et al. 2014).

3.4. Entwurf des Systems zur Klassifizierung

3.4.1. Datensammlung

Dem Entwurfsmuster aus Kapitel 2.3.2 folgend, müssen als Grundlage für das zu entwickelnde System zunächst Daten gesammelt werden. Im vorliegenden Fall müssen Bilder von Blättern akquiriert werden.

Es sind entsprechende Bild-Datenbanken verfügbar, die unter freier Lizenz zum Training für die automatische Bestimmung genutzt werden können. Unter anderem veröffentlichten Wu et al. (2007) den im Rahmen ihrer Arbeit veröffentlichten Flavia-Datensatz. Der Datensatz beinhaltet 1800 Bilder in einer Auflösung von 1600 x 1200 Pixeln von Blättern 32 unterschiedlicher, hauptsächlich chinesischer Arten. Die Bilder liegen vorverarbeitet vor. Die Blätter sind bereits freigestellt und die Blattstiele wurden entfernt (siehe Abbildung 10a).

Der Leafsnap-Datensatz von Kumar et al. ist ebenfalls frei erhältlich. Die zugrundeliegenden Trainingsdaten umfassen mit 800 x 600 Pixeln niedrig aufgelöste Fotografien von 185 in den nordöstlichen USA beheimateten Baumarten. Die Bilder sind sowohl in ihrer ursprünglichen Form und in binarisierter Form erhältlich (siehe Abbildung 10b).

Leafsnap- und Flavia-Datensatz wurden als Datengrundlage gewählt, da die beiden zugrundeliegenden Arbeiten vielfach von Veröffentlichungen anderer Autoren zitiert werden, die die Datensätze oftmals auch als Grundlage für die Tests ihrer eigenen erarbeiteten Methoden nutzen. Somit ergibt sich die Möglichkeit die vorliegende Arbeit mit anderen Arbeiten vergleichen zu können.

Abbildung 25 - Beispielfoto aus der Flavia-Datenbank (Wu et al., 2007)

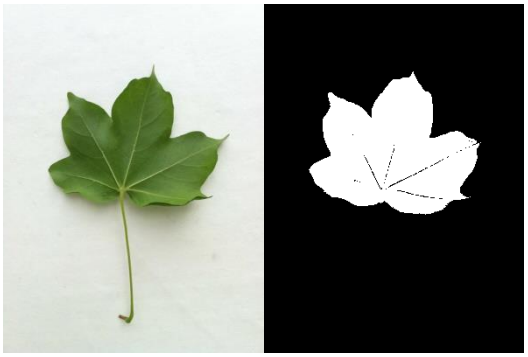


Abbildung 26 - unbearbeitetes (links) und binarisiertes Blatt (rechts) aus dem Leafsnap-Datensatz (Kumar et al., 2012)

Es soll in dieser Arbeit aber auch geprüft werden, ob sich die Blattunterseite besser eignet, um bestimmte Merkmale, im Besonderen die Blattadern, erkennen und trainieren zu können. Der Flavia-Datensatz enthält allerdings nur Fotografien von den Blattoberseiten. Im Leafsnap-Datensatz sind Blattober- und -unterseite im Datensatz vermischt mit verschiedener Anzahl und ohne erkennbare Ordnung.

Für die Evaluierung des geplanten Systems wurde im Oktober 2016 zu Beginn der Recherche- und Konzeptphase zusätzlich ein eigener Datensatz erstellt. Er enthält Fotografien 25 verschiedener Spezies mit je 40 Blättern. Von jedem Blatt wurden sowohl Ober- als auch Unterseite

aufgenommen und geordnet abgelegt. Dies macht es möglich den Datensatz in zwei gleiche Teile zu separieren, die Bilder derselben Blätter, aber verschiedener Blattseiten enthalten. Dies ermöglicht eine Prüfung der Abhängigkeit der Performanz des Systems und speziell der texturbeschreibenden Features von der betrachteten Blattseite.

Abbildung 27 - Oberseite eines Blattes von Malus Domestica (Klarapfel) aus dem selbsterstellten Datensatz

Abbildung 28- Unterseite eines Blattes von Malus Domestica (Klarapfel) aus dem selbsterstellten Datensatz

Bei den fotografierten Arten handelt es sich hauptsächlich um in Deutschland beheimatete Pflanzen STIMMT OFFENSICHTLICH NICHT, WAS KANN ICH HIER SCHREIBEN?.. *(verweisen auf Anhang, Tabelle der Spezies)*

Bei der Auswahl der Arten wurde versucht, eine möglichst hohe Variabilität der Blattform zwischen den Arten zu erhalten. Es wurden aber auch Blätter verschiedener Arten gewählt, die sehr ähnliche globale Formen aufweisen, sich aber in Details wie den Blatträndern unterscheiden, um die Grenzbereiche des Systems zu testen.

Die Bilder wurden hochauflösend mit 3000 x 2000 Pixeln aufgenommen, um auch sehr feine Details der Blatttextur und der Blatträndern beschreiben zu können. Bei der Kamera handelte es sich um eine Nikon D70S mit einem Tamron Aspherial XR Di II Objektiv mit 18 - 200 mm Brennweite- Für die Aufnahmen wurde der Aufbau aus Abbildung 29 verwendet. Die Kamera wurde an einem Stativ befestigt mit der Objektivöffnung nach unten in Richtung des Blattes schauend. Als Hintergrund für die zu fotografierenden Blätter diente ein weißes DIN A4 Blatt. In Abbildung 27 und Abbildung 28 sind Beispielfotos des so erstellten Datensatzes zu sehen.

Abbildung 29 - Versuchsaufbau zur Aufnahme der Blätter des eigenen Datensatzes

Im Laufe der Recherchearbeit wurde auch das Swedish Leaf Dataset als für die Systemevaluation geeignet ermittelt. Es wird vor allem für den Vergleich des Systems mit dem Independent Inner Distance Shape Context herangezogen, da Zhao et al. (2015) unter anderem diesen Datensatz für ihre Tests verwendeten. Söderkvist (2001) erstellte es im Rahmen seiner Master-Thesis. Es enthält 15 Spezies mit je 75 Beispielbildern. Die Bilder sind mit bis zu 3000 x 2000 Pixeln sehr hoch aufgelöst. Allerdings sind auch hier Ober- und Unterseiten der Blätter ungeordnet miteinander vermengt. Söderkvist erzielte auf seinem Datensatz unter Nutzung geometrischer Merkmale eine durchschnittliche Erkennungsrate von unter 70%. kommt in Ergebnisse

3.4.2. Segmentierung der Blätter

Die Blätter des selbsterstellten Datensatzes müssen zunächst vom Hintergrund freigestellt werden. Auch die Bilder des Leafsnap-Algorithmus von Kumar et. al. (2012) müssen vor einer Untersuchung der Blattnervatur segmentiert werden. Neben den unbearbeiteten Bildern sind lediglich Binärbilder vorhanden (siehe Abbildung 10b). Diesen fehlen notwendige Informationen der Blatttextur, da sich die Autoren in ihrer Arbeit auf die Blattkontur konzentrieren und die Textur dafür nicht benötigen. Kumar et. al. (2012) beschreiben in ihrer Veröffentlichung einen eigenen Algorithmus einer farbbasierten Segmentierung. Sie führen an, dass kanten- und

regionsbasierte Verfahren gegenüber ihrem Verfahren langsamer arbeiten und Details der Blattränder verloren gehen können.

Für die Blätter des selbsterstellten Datensatzes sollte es aber genügen, ein Schwellwert-Algorithmus basierend auf den Graustufen zu verwenden, um das Blatt freizustellen ohne Details der Kontur zu verlieren. Wie in Abbildung 27 und Abbildung 28 beispielhaft zu sehen, ist der Hintergrund im Bildmaterial farbneutral und gleichmäßig ausgeleuchtet, so dass sich die Blätter deutlich von ihm abheben. Dies bedeutet, dass sich auch in einem umgewandelten Graustufenbild zwei deutlich voneinander getrennte Maxima zwischen Vorder- und Hintergrund im Histogramm ergeben. In Abbildung 30 ist beispielhaft das Graustufen-Histogramm eines Fotos aus dem Datensatz abgebildet. Der rechte Peak stellt die Hintergrundpixel und der linke flachere Peak die Pixel des Blattes dar. Die Verteilung der Graustufenwerte ist bimodal.

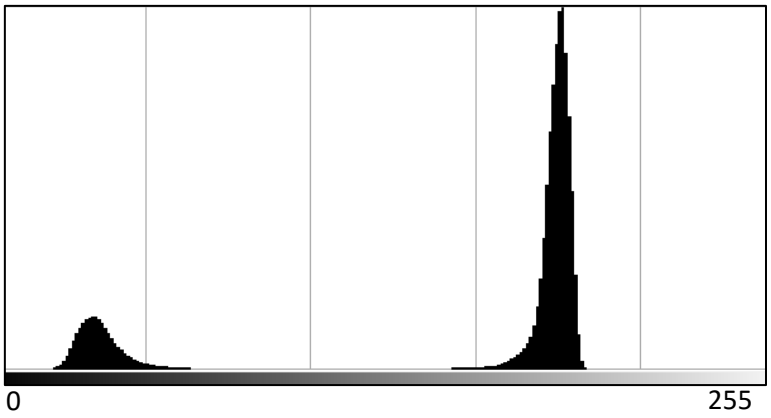


Abbildung 30 - Histogramm der Graustufen des Blattfotos aus Abbildung 27, Screenshot aus der Bildbearbeitungssoftware „GNU Image Manipulation Program“, Version 2.8.14

Zur Binarisierung soll daher die Otsu-Methode dienen, die mit bimodaler Graustufenverteilung gute automatisierte Schwellwerte liefert. Laut Burger & Burge (2015, S. 278) arbeitet die Otsu-Methode *„sehr effizient“* und liefert in der Praxis *„recht brauchbare Ergebnisse“*.

In den zu erstellenden Binärbildern wird das Blatt zum Vordergrund gezählt und weiß codiert. Der Hintergrund wird mit 0, also schwarz codiert.

Die Binärbilder werden als Masken genutzt, um die Blätter inklusive dargestellter Blatttextur freizustellen und getrennt abzuspeichern. In diesen Bildern wird der Hintergrund allerdings weiß codiert werden. Dies ist die verbreitetste Darstellungsform und wird auch bei den Datensätzen Flavia und Swedish Leaf Dataset auf diese Art angewendet.

3.4.3. Extraktion und Beschreibung der Merkmale

3.4.3.1. Wahl der Features

Es kann nicht davon ausgegangen werden, dass die als Ausgangsmaterial dienenden Blätter stets aus dem gleichen Winkel mit derselben Ausrichtung und Entfernung fotografiert werden. Um dennoch eine stabile Klassifikation gegenüber Änderungen der Rotation, Translation und Skalierung zu erreichen, gibt es mehrere Möglichkeiten.

Die Trainingsbilder können repliziert und mit der Transformation verändert werden, gegenüber der man eine Invarianz herstellen will (Bishop, 2006, S. 261). Der Klassifikator kann so während des Trainings auch die Variabilität der Transformation innerhalb des Sets mitlernen. Eine weitere Möglichkeit um Varianzen zu verringern ist es, die Bilder sowohl vor dem Training, als auch vor der Klassifizierung automatisiert einheitlich auszurichten, zu skalieren und zu beschneiden.

In dieser Arbeit soll eine Stabilität gegenüber Transformationen aber vor allem dadurch erreicht werden, dass die Features möglichst so ausgewählt werden, dass ihre Deskriptoren bereits viele Invarianzen vorweisen. Alle Klassifikatoren, die diese Features nutzen, zeigen somit zwangsläufig dieselben Invarianzen (Bishop, 2006, S. 262).

Wie in Kapitel 2.1.2.2 angeführt, ist die Form der Blätter in der Botanik ein wichtiges Unterscheidungsmerkmal für die Bestimmung einer Pflanze. Daher soll sie auch bei der automatisierten Bestimmung eine besondere Beachtung finden. Daneben ist die Blatttextur von Bedeutung, die Informationen über die Blattnervatur enthält.

Form und Textur sollen als voneinander unabhängige Merkmalskategorien beschrieben werden, um ihre Bedeutung zur Unterscheidbarkeit zwischen den Arten eigenständig beurteilen zu können. Die gewählten Features und Maßnahmen zur Minimierung des Einflusses der jeweils anderen Kategorie werden in Kapitel 3.4.3.2 beziehungsweise 3.4.3.3 aufgeführt.

Davon ausgenommen sind SIFT und SURF. Sie ergänzen die eigenständigen Betrachtungen von Form und Textur durch eine gleichzeitige Beschreibung. Sie sollen hauptsächlich eingebaut werden, um die Performanz von Kombinationen aus getrennten Form- und Texturmerkmalen mit einer globalen Beschreibung zu vergleichen.

Bei SIFT (Scale Invariant Feature Transform) und SURF (Speeded Up Robust Features) handelt es sich um Deskriptoren, die interessante Bildpunkte, die Keypoints, innerhalb eines Bildes lokalisieren und diese und ihre Umgebung beschreiben. Beide Algorithmen sind skalierungs- und rotationsinvariant. Während SIFT die Keypoints mit kombinierten Laplace-Gauß-Filtern

detektiert (Burger & Burge, 2015, S. 645), ermittelt SURF diese durch die Faltung von Integralbildern mit einem Box-Filter (Khan, McCane & Wyvill, 2011, S. 502).

Laut Khan et al. (2011, S. 501) liefern beide eine gute Leistung bei der Featuredetektion. In ihrer Evaluation der Performanz von SIFT und SURF auf einem Datensatz mit verschiedenartig transformierten Bildern kommen Khan et al. zu dem Schluss, dass die deskriptive Leistung von SURF ähnlich gut ist wie die von SIFT. Bei den Invarianzen hinsichtlich Skalierung, großer Unschärfe und Blickwinkel schneidet SIFT in ihren Tests allerdings besser ab.

mit oder ohne Blattstiel? laut Grundlagen (Anheftung) auch Aussagekraft

Im eigenen und den fremden Datensätzen beinhaltet ausschließlich gestielte Blätter, was nicht in der Absicht lag.

3.4.3.2. Beschreibung der Blattform

Die Beschreibung der Form erfolgt auf Basis von Binärbildern, die keine Informationen über die Textur mehr enthalten. Dadurch erübrigt sich ein etwaiger Einfluss der Textur auf die Formbeschreibung.

Der Beschreibung der Form soll der Inner Distance Shape Context (IDSC) von Ling & Jacobs (2007) dienen, da er sehr gute Erkennungsraten liefert (siehe Kapitel 3.3) und auch von anderen Autoren als sehr robust mit starker deskriptiver Leistung beschrieben wird (Cao et al., 2016, S. 52). Da auch die feinen Strukturen der Blattränder berücksichtigt werden sollen, wird der IDSC um den Independent Inner Distance Shape Context (I-IDSC) von Zhao et al. (2015) erweitert.

IDSC und I-IDSC werden als getrennte Module umgesetzt, so dass beide eigenständig zur Deskription verwendet werden können. So können die Ergebnisse miteinander verglichen werden und die Bedeutung der Blattränder für die Beschreibung der Form geprüft werden.

Als formbeschreibende Features eignen sich zudem die sogenannten Momente, die statistische geometrische Merkmale, wie Fläche und Schwerpunkt berechnen können (Burger & Burge, 2015, S. 251). Dazu zählen die Hu-Momente, die diese Merkmale *„geschickt kombinieren“* und weitgehend invariant unter Translation, Skalierung und Rotation sind (Burger & Burge, 2015, S. 257).

Anzumerken ist, dass bei zusammengesetzten Blättern nicht die Form der einzelnen Blättchen für sich betrachtet werden soll, sondern in ihrer Komposition, da sie aus botanischer Sicht auch zusammen das Blatt bilden.

3.4.3.3. Beschreibung der Blatttextur

Die Extraktion von Features aus der Blatttextur soll grundsätzlich auf zwei verschiedene Arten erfolgen.

Der erste Weg baut auf dem Vorwissen auf, dass Blätter über den Verlauf der Nerven durch die Spreite unterschieden werden können. Die Nerven variieren in Anzahl, Stärke und Anordnung zueinander. Um die Struktur des Nervennetzes zu untersuchen, müssen die Blattnerven zunächst von der restlichen Fläche der Blattspreite freigestellt werden. Wie es Katyal (2012) in seiner Arbeit vorschlägt, lassen sich Blattadern gut durch Gabor-Filter vom Hintergrund abheben. Seine Abhandlung konzentriert sich dabei lediglich auf die Hervorhebung der Adern, beinhaltet aber keine konkreten Methoden zur Segmentierung.

Die Binarisierung der Nerven kann durch morphologische Öffnungs- und Schließungsoperationen aus dem mit Gaborfiltern gefalteten Bild erfolgen. Die segmentierten Blattnerven lassen sich dann mit morphologischen Merkmalen beschreiben, wie dem Verhältnis der Fläche der Nerven zur Fläche des Blattes.

Um weitere relevante Merkmale wie die Länge der Nerven, die Anzahl der starken Nerven und die Anordnung beschreiben zu können, muss das Binärbild der Adern durch eine Menge von Linien oder als Graph repräsentiert werden. Die in Kapitel 3.3 erwähnte Leaf-GUI von Price et al. (2011), die Bilder des Nervennetzes in eine Graphenrepräsentation überführen kann, kommt dazu leider nicht in Frage. Sie ist zwar frei nutzbar für den akademischen Gebrauch, allerdings für die kommerzielle Software Matlab umgesetzt, die in dieser Arbeit keine Verwendung finden wird.

Ein Einfluss der Blattform auf die Beschreibung der Blattnervatur ist nicht vermeidbar und muss in Kauf genommen werden, da der Verlauf und die Länge der Nerven auch immer die globale Form wiedergeben. Die Entfernung des Blattrandes aus der Betrachtung kann den Einfluss aber verringern.

Der andere Weg ist die Textur in ihrer Gänze ohne den Vorverarbeitungsschritt der Nervensegmentierung zu betrachten. Dabei werden nur die Graustufen, nicht aber der Farbraum berücksichtigt, da nur Blätter in ihrer Grundform betrachtet werden sollen. Relevanz hätte die Farbe vor allem bei der Unterscheidung von Blüten. Blätter sind in der meist grün aufgrund ihres Chlorophyllgehaltes. Es gibt zwar Variationen von Grüntönen zwischen den Arten, diese sind aber in Fotos in der Regel nicht zuverlässig unterscheidbar. Sie können aufgrund unterschiedlicher Belichtungen und Kameraeinstellungen auch zwischen Aufnahmen derselben Art stark variieren. Dies kann zwar theoretisch über den Abgleich mit einer

mitaufgenommenen Farbskala ausgeglichen werden, ist aber wenig praktikabel und im verwendeten Bildmaterial größtenteils nicht geschehen. Nur Leafsnap enthält Laborfotos mit Farbskalen.

Zur Beschreibung der Textur sollen die Local Binary Patterns zum Einsatz kommen, aufgrund ihrer Invarianz gegenüber Schwankungen im Graustufenbereich und ihrer „hohen deskriptiven Leistung“ (nach Arun et al., 2013, S. 3).

Um den Einfluss der Form auf die Texturbeschreibung zu minimieren, extrahieren Larese et al. (2014) einen blattrandfreien Texturausschnitt der Blattspreite, einen Patch, den sie anschließend beschreiben. In Anlehnung daran sollen in dieser Arbeit neben der globalen Textur auch eine oder mehrere aus der Textur geschnittene Patches beschrieben werden.

3.4.3.4. Codebooks

Einige der gewählten Features sind nicht direkt als Eingangsdaten für den Klassifizierer geeignet, da ihre Daten ungeordnet vorliegen und somit schwer vergleich- und trainierbar sind. Sie müssen zunächst in Featurevektoren mit fester Ordnung und Größe überführt werden.

Bei SIFT und SURF ist im Vorhinein nicht bekannt welche und wieviele Keypoints aus einem Bild extrahiert werden. Um die Keypoints geordnet zu beschreiben, eignet sich das als Bag of Visual Words bekannte Verfahren, wie es erstmals von Sivic & Zisserman (2009) beschrieben wurde, in Anlehnung an die Bag of Words-Methode zur Analyse von Texten. In ihrer Arbeit nutzen sie SIFT zur Extraktion und Deskription der Keypoints. Aus den Keypoints bilden sie ein visuelles Vokabular mit Hilfe des in Kapitel 2.3.5 beschriebenen k-Means Clustering. Die ermittelten Clusterzentren werden als visuelle Wörter des Bildes verstanden. Die Menge der visuellen Wörter bildet damit ein Wörterbuch, das für alle zu trainierenden und klassifizierenden Bilder gleichermaßen genutzt wird. Anhand des visuellen Wörterbuches wird die Häufigkeit des Auftauchens der einzelnen Wörter im Bild gezählt. Das gebildete Histogramm dient dann als Eingangsdaten für das Klassifizierungsmodell.

Gleiches gilt für den Inner Distance Shape Context (IDSC) und seine Erweiterung durch den Independent Inner Distance Shape Context. Ihre Deskription liefert eine Reihe von Punktmerkmalen ohne Ordnung. In der ursprünglichen Version des IDSC erfolgt die Klassifizierung mittels dynamischer Programmierung. Da hier aber stattdessen neuronale Netze verwendet sollen, müssen auch die extrahierten Shape Contexts zunächst transformiert werden.

Zhao et al. (2015) schlagen dazu vor, mit einem sogenannten Sparse Dictionary die extrahierten Shape Contexts in ihre wiederkehrenden atomaren Bestandteile zu zerlegen und ähnlich wie beim Bag of Visual Words Histogramme der Häufigkeit ihres Vorkommens innerhalb des Bildes zu erstellen.

Sowohl das Sparse Dictionary, als auch das mit k-Means gebildete visulle Wörterbuch sollen für die Transformation der ungeordneten Features umgesetzt werden.

3.4.4. Wahl der Klassifizierungs-Modelle

Wie in der Zielstellung erwähnt, soll ein Hauptaugenmerk der Arbeit auf dem Aufbau, der geeigneten Parametrisierung und dem Training eines neuronalen Netzes liegen.

Bei der Klassifizierung von Blättern haben laut Wu et al. (2007, S. 1) künstliche neuronale Netze wie Multilayer Perceptron (MLP) und Probabilistic Neural Networks (PNN) die schnellste Geschwindigkeit und höchste Genauigkeit unter den bis dato 2007 verwendeten Modellen. Sie selbst nutzen ein PNN aufgrund der simplen Struktur und hohen Rechengeschwindigkeit.

In dieser Arbeit soll ein MLP die Klassifizierung übernehmen. Der Vorteil des MLP gegenüber einfachen Perzeptrons ohne versteckte Schichten ist, dass sie nicht-lineare Abbildungen der Ausgangs- auf die Eingangsdaten bilden können und somit auch Muster in linear nicht trennbaren Eingangsdaten erkennen können. Die Backpropagation wird als Lernmethode eingesetzt, da es das meist genutzte Verfahren und somit vielfach erprobt ist. Zudem ist es universell einsetzbar für eine Vielzahl an Approximationsaufgaben, nicht nur für die Klassifizierung. (Ertel, 2016, S. 291)

Aufgrund der Nicht-Linearität der Abbildung wird darauf zu achten sein, dass eventuelles Overfitting während des Trainings erkannt und verhindert wird. Sollte sich die Rechengeschwindigkeit des MLP als problematisch für die zahlreichen durchzuführenden Tests herausstellen, soll es durch das simpler strukturierte PNN ersetzt werden.

Ein gefaltetes neuronales Netz wäre unnötig komplex für die gestellten Anforderungen. Es ist eher geeignet um selbsttätig Merkmale aus Bildern zu ermitteln und zu erlernen und benötigt dafür auch eine deutlich größere Anzahl an Bilddaten je Klasse als Eingang (Keller et al., 2016, S. 52). Der Umfang der verwendeten Datensätze reicht dafür nicht aus. Zudem sollen gerade die mit dem botanischen Vorwissen erstellten Features im Fokus der Arbeit stehen.

Als Vergleichs-Modell zum MLP wird eine Support-Vektor-Maschine (SVM) zum Einsatz kommen. Dieses Modell wird gewählt, weil die als Eingangsdaten erwarteten Featurevektoren die gleiche Struktur und Dimension wie beim MLP haben können. Es können somit die

gleichen Daten für MLP und SVM für Training und Klassifizierung verwendet werden, ohne sie vorher transformieren zu müssen. SVM vereint die Vorteile von linearen und nicht-linearen Modellen (Ertel, 2016, S. 298), wodurch die Gefahr einer Überanpassung geringer ist als beim MLP.

3.5. Datenhaltung

Die zu implementierende Datenhaltung dient der Persistenz, dem Austausch und der Wiederverwendbarkeit der Ergebnisse aus den einzelnen Modulen des Systems.

Die grundsätzliche Überlegung dabei ist, dass die Daten zentral an einem Ort gehalten werden sollen und einfach identifiziert und abgerufen werden können. Verstreut gespeicherte Einzelergebnisse aus den Modulen wären unübersichtlich, schwer zuordenbar und könnten leicht verloren gehen. Die Datenhaltung sollte zudem, wenn möglich, unabhängig von der verwendeten Programmiersprache sein. Falls zukünftig geplante Erweiterungen und Anpassungen des Systems es notwendig machen sollten, einen Wechsel der Programmiersprache des Systems oder einzelner Komponenten zu vollziehen, gestaltet sich dies einfacher ohne bereits gesammelte Daten zu verlieren. Zudem ist eine Nutzung der Daten in mit anderen Sprachen geschriebenen Anwendungen so theoretisch möglich.

Die unbearbeiteten und die segmentierten Bilder an sich werden nicht extra in der Datenhaltung abgelegt. Dort sollen aus Gründen des Speicherbedarfs nur die nach Pflanzen- und Featureart geordneten, extrahierten Merkmalsrepräsentationen der Bilder abgelegt werden. Zudem sollen die trainierten Klassifizierer mit allen genutzten Parametern und ermittelten Gewichtungen beziehungsweise Hyperebenen gespeichert werden, um sie nach dem Training nach beliebiger Zeit wieder laden und auf neue Bilder anwenden zu können. Zu den zu speichernden Daten zählen auch die erstellten Codebooks. Die Menge ihrer visuellen Wörter muss im Training und zur Klassifikation gleich sein. Um beide Schritte zeitlich getrennt voneinander durchführen zu können, müssen sie das gemeinsam benutzte Codebook jederzeit konsistent aus der Datenhaltung abrufen können.

Es stehen verschiedene Datenhaltungssysteme zur Auswahl, die der zentralen, geordneten Speicherung dienen können.

Dateibasierte Möglichkeiten sind unter anderen die Nutzung des HDF5 Formats und die einfache Serialisierung der Daten mit Mitteln der jeweiligen Programmiersprache. Da die Serialisierung sprachenabhängig ist und in der Regel eine Aufspaltung der Daten in einzelne verteilte Dateien nach sich zieht, kommt sie hier nicht in Frage.

HDF5 ist ein erweiterbares Datenmodell, das für die effiziente Speicherung und Verwaltung von komplexen Daten konzipiert ist Es unterstützt je nach Erweiterung eine Vielzahl verschiedener Datentypen (HDF Group, 2017). Zudem ist es ein sehr gebräuchliches Datenformat im wissenschaftlichen Bereich.

Der Vorteil der Nutzung eines SQL- oder NoSQL-Datenbankssystems ist vor allem, dass ein zentraler Zugriff nicht nur über das Dateisystem, sondern auch über einen Server in einem Netzwerk möglich ist. No-SQL bietet eine gute Skalierbarkeit bei großen ungeordneten Datenmengen, während sich SQL besser eignet, um Daten strukturiert abzulegen. Für die Bedürfnisse des geplanten Systems ist SQL besser geeignet, allerdings auch unnötig komplex. Einfache Zuordnungen von Daten zu bestimmten Kategorien lassen sich auch über HDF5 abbilden.

Da im Rahmen der Arbeit keine Client-Server Anwendung umgesetzt werden soll, scheint HDF5 von den genannten Möglichkeiten als am besten für die Datenhaltung geeignet.

3.6. Grafische Oberfläche und Testumgebung

Neben dem Klassifizierungs-System soll eine grafische Oberfläche erstellt werden, mit der sich die Komponenten einzeln und in Kombination steuern, visualisieren und evaluieren lassen.

Die Funktionalität der Segmentierung soll sich anhand ausgewählter Bilder testen lassen, indem Zwischenbilder und Endresultate der Filterungen und Schwellwertverfahren dargestellt werden.

Es ist geplant, aus der Oberfläche heraus die Extraktion der Features zu starten und in der Datenhaltung abzulegen. Zur Wahrung der Übersichtlichkeit soll sie eine Visualisierung aller bereits in der Datenhaltung hinterlegten Features, der zugehörigen Spezies und der für die Transformation der Daten verwendeten Codebooks bieten. Training und Evaluation sollen sich mit innerhalb der Oberfläche ausgewählten Features und Parametern starten lassen.

Für eine verbesserte Planbarkeit der Evaluation soll es zudem die Möglichkeit einer Projektierung geben. Es sollen Projekte angelegt werden können, die sich bezüglich der verwendeten Bilddatensätze und Parameter unterscheiden.

Zusätzlich zur Oberfläche sollen Unittests sicherstellen, dass bei Änderungen an den Systemkomponenten während der Entwicklung die erwarteten Kernfunktionalitäten erhalten bleiben.

3.7. Modularisierung des Systems

Das System soll streng modular aufgebaut werden, um eine Austauschbarkeit und Wiederverwendbarkeit der einzelnen Module zu gewährleisten.

Das System teilt sich der Aufgabe der Komponenten entsprechend in die Module Segmentierung, Features, Codebooks, Klassifizierer, Datenhaltung und grafische Oberfläche auf.

Abbildung der Systemkomponenten?

Die Module sollen möglichst stark voneinander getrennt werden. Segmentierung, Features, Klassifizierer und Codebooks gehören zur Modellschicht und sollen keine Kenntnis von der Datenhaltung und der Oberfläche haben. Um die Module der Modellschicht zu persistieren, benötigt die Datenhaltung wiederum Kenntnis ihrer Module, aber nicht von der Oberfläche. Die Oberfläche hat die Aufgaben eines Controllers und Zugriff auf Methoden sowohl der Modellschicht, als auch der Datenhaltung.

Um die Austauschbarkeit und Erweiterbarkeit der Module der Modellschicht sicherzustellen, erben diese von Interfaces, die die nach außen hin benötigten Funktionen vorschreiben. Da auch die Datenhaltung in Hinblick auf mögliche zukünftige Erweiterungen austauschbar sein soll, erhält auch sie ein Interface. Bei der Oberfläche ist dies natürlich nicht der Fall, da sie der Steuerung der Komponenten dient, und alle anderen Module keine Kenntnis über sie haben.

4. Implementierung

4.1. Programmierumgebung und Programmbibliotheken

kein Matlab, da kommerziell

zunächst java mit opencv, schnell n grenzen, mehr bibliotheken für python

java an sich schneller als python, aber mit cython möglich performantes bla zu schreiben

in Kapitel 3 rein!

Binärbilder trotzdem in 8Bit, definition: 0 Hintergrund >0 (1 oder 255) Vordergrund, opencv liefert beim thresholding 255

nicht Java sondern Python

OpenCV Konturpunkte, Binarisierung

skimage und opnecv, eigentlich alle benutzen numpy arrays als represäntation von Bildern, benutzt: 8bit farbrepräsentation

versuch schleifen zu vermeiden mit numpy, indizierung über mehrere Indizes

4.2. allgemeine Entscheidungen

alle Features erwarten segmentiertes Bild, alle Segmentierer liefern segmentiertes Bild (Vereinheitlichung), einfache binarisierung mit umcodierung hintergrund zu schwarz und simplem threshold, aber achtung jpg

Unsupervised Feature genannt, um anzuzeigen, dass unsupervised lernverfahren folgen muss clustering oder codebook nötig

Segmentierte Bilder, Hintergrund weiß (abgeguckt von allen anderen)

einfache Binarisierung: da Hintergrund mit weiß codiert, gleich schwarz setzen (Binärbild Codierung!) und Rest auf 1 bzw. 255

Achtung: jpg macht Probleme! (irgendwo anders hier)

Feature Klasse übernimmt Vorbereitung des Bildes, delegiert an Unterklassen, wo eigentliche Berechnungen stattfinden

4.3. Systemarchitektur

abweichend von kapitel, keine interfaces sonder abstrakte klassen

4.4. Segmentierung

Ausgang immer maskiertes Bild

croppen wichtig, skalierung wichtig

Die Segmentierung mittels Otsu-Thresholding wurde mit Hilfe der OpenCV-Bibliothek umgesetzt. Zunächst wird das Ausgangsbild im RGB-Farbraum eingelesen und in ein Graustufenbild umgewandelt. ??? Allerdings war das Ergebnis entgegen der Vermutung aus Kapitel unzureichend für eine anschließende Weiterverarbeitung. In einigen Bildern verfälschen lange Schatten die Kontur des binarisierten Blattes. Da bei der Aufnahme des Testdatensatzes

teilweise ein Blitzlicht verwendet wurde, sind die vom Blatt geworfenen Schatten sehr dunkel. Da auch das Blatt dunkler ist als der Hintergrund, werden die Pixel der Schatten dem Vordergrund zugeordnet. (siehe Bilder)

Da die Fotos der Blätter vor einem neutralen farbarmen Hintergrund aufgenommen wurden, lassen sich die Blätter vor allem in der Sättigung von diesem unterscheiden, der eigentliche Farbwert ist vernachlässigbar. Somit ist der HSV-Farbraum, der die Helligkeit von der Farbsättigung und dem Farbwert trennt, für die Freistellung der farbigen Blätter besser geeignet als der RGB-Farbraum (siehe Abbildung). Ein Otsu-Thresholding des Sättigungsanteils trennt in einer Vielzahl der Bilder

Warum kmeans? Bilder raussuchen, Kmeans im Farbraum, mehr informationen

Es ist also notwendig

Zudem fransen die schattigen Ränder der Blätter aus.

Löcher schließen, Löcher sollen Merkmal der Textur, aber nicht der Form sein

Hintergrund eingefärbt vom Umgebungslicht. Einige Fotos in Dämmerung geschossen, farbiges Licht

Auch dies ergab

value dazu

Bestehendes Problem: farbige Schatten. Die Schatten sind hauptächlich grau mit einer geringen Sättigung. zwei Lichtquellen: Umgebungsbeleuchtung, Blitz

Da aber auf einigen Es verbleiben dennoch farbige Schatten,

Generell erfolgt vor allen Segmentierungen, bei denen Faltungen mit festen Kernelgrößen vorgenommen werden, zunächst eine Skalierung des Bildes auf eine vordefinierte Größe. So wird verhindert, dass die strukturierenden Elemente unterschiedlichen Einfluss haben (Erosion, Dilation).

kein Gauss, damit Kontur scharf bleibt (-> Segmentation verursacht deswegen Löcher)

DSC_6982.JPG, DSC_6813.JPG - Beispiel dafür, dass einfaches Otsu und k-means nicht funktionieren

bei Durchsicht der segmentierten Blätter stelle sich heraus…

H - viel noise

S + V - wenig noise, bisschen harte schatten

nur S - weniger schatten, noisy, da fragmente der schatten

Schwierigkeit bei entfernung stiel entfernt (vielleicht bezug auf Plant identification using leaf shapes—A pattern counting approach)

Stiel wichtig (Grundlage), Details von Blattgrund und Blattspitze kann verloren gehen

Entscheidung: Stiel bleibt dran

normalisieren, da nur erhältnis der werte zueinander entscheidend, nicht höhe der werte (wie oft taucht pattern x auf im verh. zu allen anderen)

Shape allgemein: immer längste kontur nehmen, falls mehrere Konturen, kleine Objekte bei Segmentierung zurückbleiben

Abweichend von der blabla, die Bilder nicht vorher zu transformieren, werden die Bilder beschnitten, Hintergrund außerhalb der bounding box ist unnötige Bildinformation und muss nicht gespeichert werden. Features können bei Bedarf das Bild auch skalieren, da einzelne Features wie die Beschreibung der Blattnervatur (Kapitel 4.5.4) feste Kernelgrößen verwenden, deren Funktionsweise somit auch von der Auflösung des Bildes und der Größe seiner Elemente abhängen.

4.5. Features

4.5.1. Feature-Klasse

können sich selbst auf eine vordefinierte einheiltiche größe skalieren

keine kenntnis view nicht, als Kniff erhalten dictionary um zwischenbilder darin zu speichern, damit view sie darstellen kann

haben id, dient der datenhaltung zur eindeutigen Identifizierung/Zuordnung aus welchem Blatt sie extrahiert

4.5.2. Inner Distance Shape Context

Für Python sind keine Bibliotheken verfügbar, die Methoden zur Extraktion des Inner Distance Shape Context (IDSC) bereitstellen. Es ist Quellcode der Veröffentlichung von Ling & Jacobs (2007), geschrieben in Matlab, online abrufbar. Da sich Matlab-Code schwer auf die genutzte Umgebung mit Python übertragen lässt, wird der Algorithmus anhand der Beschreibungen des veröffentlichten Papers selbst implementiert.

Der Ausgangspunkt für die Berechnung jedweder Shape Contexts ist die aus einem Binärbild ermittelte Kontur. Die Konturpunkte werden mit Hilfe der OpenCV-Bibliothek ermittelt. Diesen wird anschließend eine vordefinierte Anzahl an Punkten, die in regelmäßigen Abständen verteilt sind, entnommen. Keine besonderen Punkte

Die euklidischen Distanzen zwischen allen entnommenen Punkten werden berechnet und in einer mit Nullen initialisierten Distanzmatrix gehalten. Abweichend vom ursprünglichen Konzept des Shape Contexts, werden wie bei IDSC nur innere Distanzen berücksichtigt. Dies bedeutet, dass die direkte Linie zwischen zwei betrachteten Punkten komplett innerhalb der Form liegen muss, damit sie innerhalb der Form als verbunden gelten können. Da der Hintergrund im Binärbild mit Nullen codiert ist, wird dazu programmatisch überprüft, ob Zwischenpunkte auf dieser Linie nur Werte größer Null annehmen. Ist dies der Fall, wird die Distanz in die Matrix eingetragen. Die verbleibenden Nullen geben an, dass keine Verbindung zwischen den jeweilig indizierten Punkten besteht.

Die Distanzmatrix bildet den Eingangsgraphen für die Ermittlung der kürzesten Wege zwischen den Punkten der Kontur, wobei die euklidische Distanz die Kantenlänge angibt. Über den Floyd-Warshall-Algorithmus der scipy-Bibliothek wird eine Matrix erzeugt , die paarweise die kürzesten Wege enthält. Jeder kürzeste Weg ist die minimale Summe der Länge der Kanten im ungerichteten Graph, die abgelaufen werden müssen, um den Punkt zu erreichen. Gibt es keine Kante zwischen zwei Punkten, ist der kürzeste Weg, wie in der Distanzmatrix, gleich Null, beziehungsweise unendlich.

Für jeden aus der Kontur entnommenen Punkte wird anschließend ein Histogramm erstellt. Das Histogramm beschreibt das Verhältnis zu allen anderen Punkten im Hinblick auf die

relative Orientierung und die Länge des kürzesten Weges zu ihnen. Die relative Orientierung wird als Winkel zwischen der an der Kontur anliegenden Tangente am betrachteten Punkt und dem Zielpunkt definiert.

Für die Zuordnung der Punkte in das Histogramm werden Winkel und Wegelängen als Polarkoordinaten in jeweils acht Klassen unterteilt. Der Wertebereich der Winkel liegt zwischen 0 und 2π, der der Wegelängen zwischen 0 und dem Logarithmus der maximalen Distanz. Als maximale Distanz wird die Diagonale des Ausgangsbildes angenommen.

Jeder nicht erreichbare Punkt wird der Histogramm-Klasse zugeordnet, die alle Punkte enthält, die im Hinblick auf kürzeste Wege den größten Abstand zum betrachteten Punkt haben. Somit ist das Histogramm jedes Punktes vollständig beschreibend, die Summe seiner Werte entspricht immer der Anzahl der entnommenen Punkte minus den betrachteten Punkt. Um den Wertebereich des Histogramms unabhängig von der gewählten Anzahl der Punkte zu halten, wird die L1-Norm, also die Summennorm darauf angewendet.

Histogrammformel hier hin

Um die Konturen von Objekten anhand ihrer ermittelten Inner Distance Shape Contexts zu vergleichen und zu klassifizieren, verwenden bla et al, dynamische Programmierung. Da die Histogramme nicht geordnet, irgendwas Dies eignet sich nicht für die angestrebte Lösung, die Klassifizierung mit neuronalen Netzen oder Support Vector Machines durchzuführen.

Die erzeugten Histogramme können aber als sich wiederholende Muster gleicher Dimension angenommen werden. Diese Muster können als einzelne Fragmente für ein Bag-of-words-Modell dienen und zu visuellen Wörtern zusammengefügt werden, die wiederum die Form beschreiben. Diese Aufgabe übernehmen die in Kapitel ??? beschriebenen Codebooks.

Die einzelnen Histogramme mit einer Dimension von 8 x 8 werden konkateniert, so dass das Endergebnis der Feature-Extraktion eine Matrix der Dimension Anzahl der Punkte x 8 x 8 ist.

4.5.3. Multilevel Inner Distance Shape Context

Auf Basis der Arbeit von Zhao et al. (2015) wurde der im vorigen Kapitel 4.5.2 beschriebene Algorithmus um den Independent Inner Distance Shape Context (I-IDSC) erweitert, aus der Motivation heraus, die feinen Details der Konturen unabhängig von der Gesamtkontur beschreiben zu können. Auch in diesem Fall waren weder Bibliotheken noch Quellcode verfügbar, so dass sich die Umsetzung nur an den Inhalten des veröffentlichten Papers orientiert. Die Implementierung weicht in einigen Punkten von der des I-IDSC, wie er von Zhao et al. beschrieben wird, ab, sowohl in der Berechnung, als auch der Transformation in einen für

das neuronale Netz geeigneten Featurevektor (siehe Kapitel 4.6.1). Daher wurde mit „Multilevel Inner Distance Shape Context“ ein anderer Name für den Algorithmus gewählt.

Wie auch beim Independent Inner Distance Shape Context, werden zunächst mehrere Stufen des binarisierten Bildes gebildet. Auf der niedrigsten, der feinsten, Stufe bleibt das Ausgangsbild erhalten, um alle Details der Kontur abzubilden. Auf den weiteren Stufen wird das Ausgangsbild mit einem Gauß-Filter $G_\sigma(x,y)$ gefaltet und das Bild somit geglättet (siehe 2.2.3.2). Zwischen den Stufen steigt die Standardabweichung σ stetig an, so dass auch der Grad der Glättung ansteigt und die Form des Objektes so an Details verliert. Konkret wird σ, beginnend mit $\sigma = 8$ auf der zweiten Stufe, von Stufe zu Stufe verachtfacht. Thresholding!

Die maximalen Distanzen bestimmen zusätzlich die Größe der Nachbarschaft eines Punktes in Bezug auf die maximale Länge der kürzesten Wege. Auf dem feinsten Detailgrad wird nur die unmittelbare Nachbarschaft betrachtet, die maximale Distanz muss dementsprechend klein sein. Auf dem gröbsten Detailgrad sollen alle Punkte der Kontur als benachbart angenommen werden. Auf dieser Stufe entspricht die Berechnung dem ursprünglichen IDSC, allerdings angewandt auf eine stark vereinfachte Form.

Die geglätteten Konturen dienen als Eingangspunkte des Algorithmus des Inner Distance Shape Context aus Kapitel 0. Der Algorithmus wurde dahingehend modifiziert, dass Punkte außerhalb der maximalen Distanz ignoriert werden, anstatt sie der letzten Histogramm-Klasse zuzuordnen. Das Histogramm ist somit nicht vollständig beschreibend, sondern beschreibt nur die Nachbarschaft der Punkte.

Abweichend vom Independent Inner Distance Shape Context von Zhao et al. werden die einzelnen Glättungs-Stufen unabhängig voneinander behandelt. Die Konturpunkte zwischen den Stufen sind nicht deckungsgleich. Stattdessen werden mehr Punkte betrachtet, je feiner der Detailgrad ist. Der Grund dafür ist, dass die Gesamtzahl an Konturpunkten erhöht werden muss, damit für eine detaillierte Beschreibung einer kleinen Nachbarschaft genug Punkte in diesem Ausschnitt der Kontur zur Verfügung stehen.

Die resultierenden Histogramme der verschiedenen Stufen sind als Basiselemente, aufgrund der verschieden definierten Nachbarschaften, nicht direkt miteinander kombinierbar. Daher wird für jede Stufe ein eigenes Codebook benötigt.

4.5.4. Segmentierung und Beschreibung der Blattnervatur

Probleme bereits bei Beschreibung von Linien, kein geeigneter rotationsinvarianter Linien-Deskriptor in Bibliotheken, opencv bietet einen, aber nicht in python binding

zeitlich nicht möglich eigenen zu implementieren oder c++ line descriptor in python einzubinden

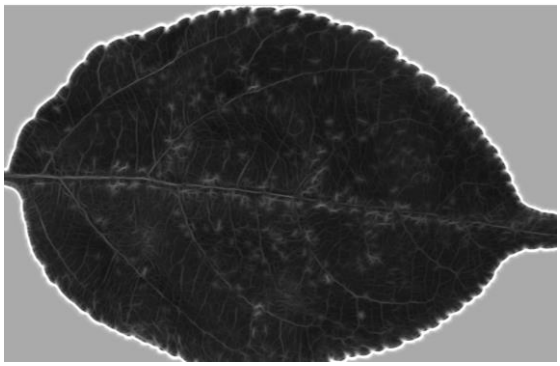


Abbildung 31 - Mit Gaborfiltern gefaltetes Graustufenbild der Blattoberseite von Malus Domestica aus Abbildung 27

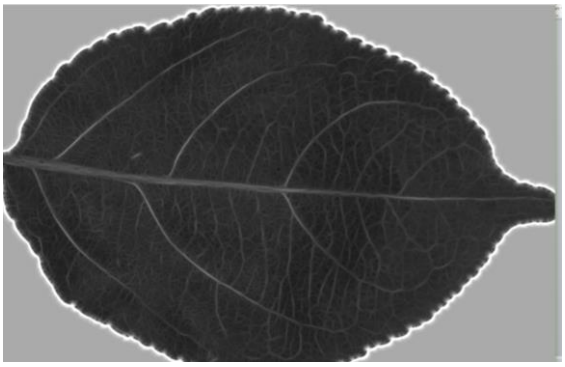


Abbildung 32 - Mit Gaborfiltern gefaltetes Graustufenbild der Blattunterseite von Malus Domestica aus Abbildung 28

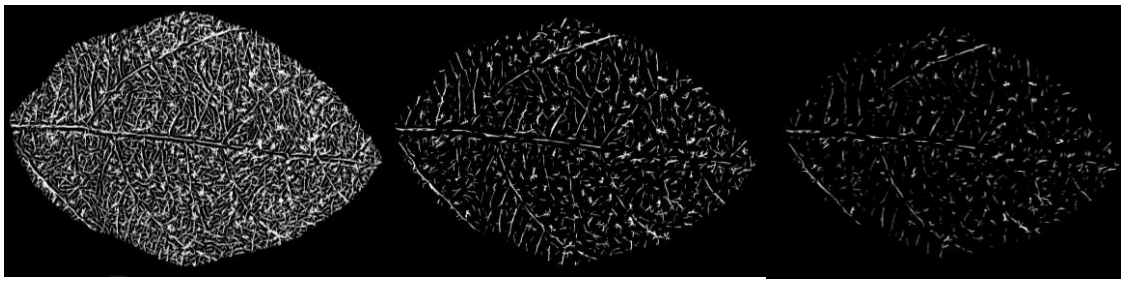


Abbildung 33 - freigestellte Blattadern aus dem Gaborbild der Blattoberseite aus Abbildung 31 mit Hilfe morphologischer Operationen mit von links nach rechts ansteigender Größe des Filterkernels

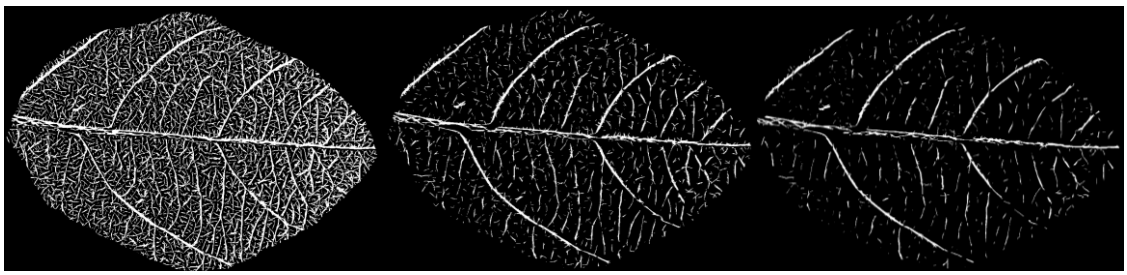


Abbildung 34 - freigestellte Blattadern aus dem Gaborbild der Blattunterseite aus Abbildung 32 mit Hilfe morphologischer Operationen mit von links nach rechts ansteigender Größe des Filterkernels

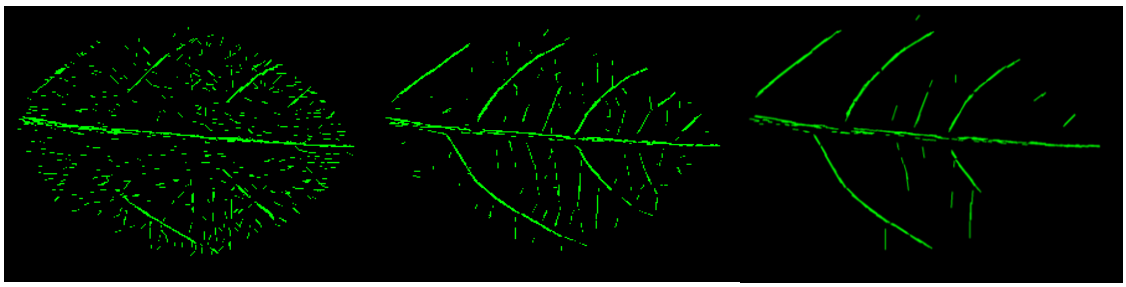


Abbildung 35 - Hough-Linien (grün), extrahiert aus den morphologischen Faltungen aus Abbildung 34 (in gleicher Reihenfolge)

extrahiert Linien (grün) aus Unterseite des Blattes aus morph. bildern

Adern hervorgehoben durch Gaborfilterung

Hough Lines nicht optimiert, viele einzelne Linien, weiter bearbeiten und parallele Linien auf starken Andern entfernen, beziehungsweise vorher ausdünnen der zusammenhängenden Pixel, da eh kein Deskriptor (v.a. linkes Bild)

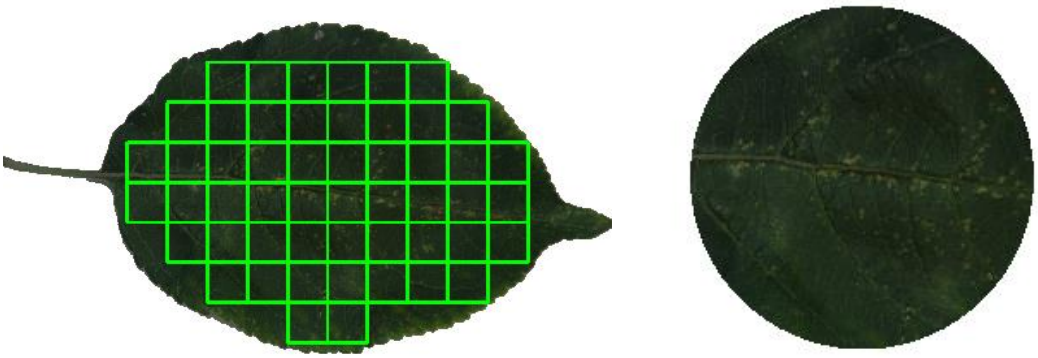
viel noise auf oberseite, da gabor mit verwendeten filterkernels auch auf flecken stark anspricht

Hauptnerven treden deutlicher hervor bei Rückseite

in abb. ?ß zu sehen gabor spricht stark auf Blattrand an, entfernen aus gabor gefiltertem bild durch gaußfilter auf hintergrundmaske, maske neu anwenden

Leaf-GUI zwar frei für akademischen Gebrauch, aber in matlab geschrieben, schwer einzubinden.

Eigener Weg Hough Lines extrahieren und die Linien beschreiben



kreisrunder Patch

Local Binary Pattern arbeiten nicht zufriedenstellend

Die erwähnten Gaborfilter eignen sich nicht nur zur Hervorhebung der Kanten, sondern auch zur Beschreibung von Texturen. Zu dem Zweck werden Filterbanken mit verschiedenen Gabor-Filtern gebildet und ihre

da Gaborfilter generell auf Gradienten reagieren auch geeignet um Adern zu beschreiben

4.5.5. Sift und Surf

noch zusätzlich dazugenommen, um Form und dingens zu

4.6. Codebooks

Die Klassifizierer benötigen als Eingangsdaten eindimensionale Feature-Vektoren, die statistische Eigenschaften eines Objektes in einer festen Reihenfolge beschreiben. Um ungeordnete Features zu beschreiben, können diese als Fragmente visueller Wörter aufgefasst werden. Über die Häufigkeit des Auftretens der visuellen Wörter, können Histogramme gebildet werden, welche dann wiederum als eindimensionale geordnete Feature-Vektoren nutzbar sind. Die Aufgabe der Bestimmung geeigneter visueller Codewörter übernehmen die im Folgenden beschriebenen Codebooks. (kommt in 3 rein)

Zunächst bei Extraktion der Features direkt angewendet, eigenes Feature für jeden Typ, getrennt gespeichert, on the fly codebook erstellung

später getrennt, da Extraktion lange dauert und Codebooks auf die gleichen Ausgangsdaten der Features zugreifen, für bessere Vergleichbarkeit. -> viel Speicherbedarf, on the fly noch nicht vollständig, da bisher keine möglichkeit, benutzten codebook typ mitzuspeichern (für predict wichtig), achtung: mischung vermeiden, on the fly ist deaktiviert

4.6.1. Sparse dictionary learning

Speziell für die Umwandlung der Histogramme des Inner Distance Shape Context wurde ein Wörterbuch für das sogenannte „sparse dictionary learning“ umgesetzt. Die Aufgabe des Wörterbuchs ist es, die Menge der extrahierten Shape-Context-Vektoren zu reduzieren, indem diese in ihre Atome zerlegt werden. Mit Atomen werden die Linearkombinationen bezeichnet, aus denen sich die Vektoren komponieren lassen.

Löst das Problem der Formel

Vorteil: online dictionary (wird nicht benutzt), Dictionary kann erweitert werden

Angelehnt an die Arbeit von Zhao et al. sind die Kombinationen der Atome des Wörterbuchs sich wiederholende Muster und können somit als visuelle Wörter betrachtet werden. und in einem Histogramm gezählt. Abweichend nicht darauf geachtet, dass nur bestimmte Zahl auftauchen darf. einfacher halten und universeller anwendbar

Atom als visuelles Wort

Die Repräsentation ist übererfüllt,

Dictionary. KMeans zusätzlich

erwartete Dimension

4.7. Multilayer Perceptron

mit Keras umgesetzt

zunächst für Evaluation und Test der Features nur simpel

später erweitert um Anpassbarkeit der Parameter

4.8. Support Vector Machine

linear

Parameter Standardeinstellungen

4.9. Datenhaltung

teilweise pickle, was eigentlich nicht erwünscht war, da abhängig von der programmiersprache python

Feature muss entscheiden welche Merkmale es braucht und sie entsprechend zur Speicherung zur Verfügung stellen und serialisieren, wenn keine numpy arrays

keras extra, erweist sich als Vorteil, dass HDF5 genommen

4.10. Oberfläche

4.11. Evaluation

Performanz der Features

Vergleich mit anderen Arbeiten

Parametrisierung des MLP

5. Ergebnis

local binary pattern keine Aussagekraft, auch in Kombination nicht

Fokus lag auf Form, da schon in Entwicklungsphase absehbar war, dass Formbeschreibungen besser performen

leider keine

weder

MLP liefert beste Performanz. SVM auch gut und im Training deutlich schneller. Aber SVM Parametrisierung auch nicht getestet, da Fokus auf NN

möglicherweise

5.1. Genauigkeit (welche Metrik?)

Vergleich verschiedene Blattformen

Vergleich steigende Anzahl an Klassen

Vergleich Vorder/Rückseite

Vergleich steigende Anzahl an Trainingsblättern (Erkennungsrate)

Kombination der Features

Länge des Codebooks

5.2. Aufwandsanalyse?

6. Zusammenfassung

6.1. Ausblick

- Gleiche Spezies untersuchen – verschiedene Sorten (möglicherweise gleiche Blattform, aber Unterschiede in Blattadern), im Datenset nur Apfel (?)
- bugs/usability in Oberfläche (war für Zwecke hier egal9
- Smartphone App

auf Feldfotos testen, mit anderen Aufnahmebedingungen

Optimierung IDSC: Performance generell (in C schreiben, python zu langsam, opencv interface?)

Textur integrieren in IDSC (SPT)

Trainingsfotos: Beleuchtung, eine diffuse, unfarbige Lichtquelle, DSC_5821

sql besser geeignet für simultane zugriffe, besser auszulagern, auch möglich wenn hdf5 in einem prozess geöffnet

Varianten des codebooks parallel: zurück zu on the fly

Parametrisierung des MLP über Oberfläche

features un- und verarbeitet speicherbar, müssen auch speichern welches codebook verwendet wurde, gibt Datenhaltung nicht her

bekannte Fehler: gelegentlicher hdf5 error beim Zugriff auf Datei

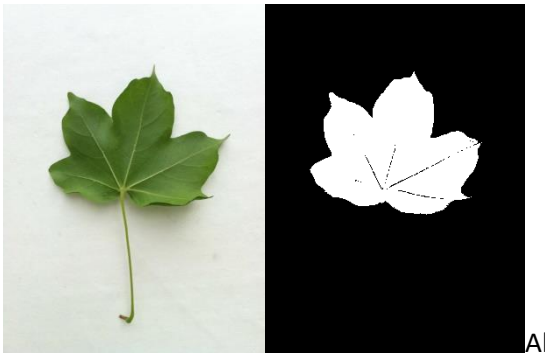
„H5F_open - unable to open file“

weitere tests auf dem Datensatz bezüglich Textur Blattseiten, eentuelle Vorteile konnten hier nicht herausgearbeitet werden, Veröffentlichung, kein vergleichbarer Datensatz während Recherche entdeckt

Anhang: Links zu den Blättern! kommen nicht auf DVD

mehr Unittests

I. Abbildungsverzeichnis



II. Tabellenverzeichnis

Es konnten keine Einträge für ein Abbildungsverzeichnis gefunden werden.

III. Codelistings

Es konnten keine Einträge für ein Abbildungsverzeichnis gefunden werden.

IV. Literaturverzeichnis

Arun, C. H., Sam Emmanuel, W. R., & Christopher Durairaj, D. (2013). Texture Feature Extraction for Identification of Medicinal Plants and Comparison of Different Classifiers. *International Journal of Computer Applications*, *62*(12), 975–8887.

Belongie, S., Malik, J., & Puzicha, J. (2002). Shape matching and object recognition using shape contexts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *24*(4), 509–522. https://doi.org/10.1109/34.993558

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. New York: Springer

Science+Business Media.

Burdick, H. E. (1997). *Digital Imaging- Theory and Applications*. New York: McGraw-Hill.

Burger, W., & Burge, M. J. (2015). *Digitale Bildverarbeitung: eine algorithmische Einführung mit Java* (3. Aufl.). Berlin, Heidelberg: Springer-Verlag. https://doi.org/10.1007/978-3-642-04604-9

Cao, J., Wang, B., & Brown, D. (2016). Similarity based leaf image retrieval using multiscale R-angle description. *Information Sciences*, *374*, 51–64. https://doi.org/10.1016/j.ins.2016.09.023

Chaki, J., Parekh, R., & Bhattacharya, S. (2015). Plant leaf recognition using texture and shape features with neural classifiers. *Pattern Recognition Letters*, *58*, 61–68. https://doi.org/10.1016/j.patrec.2015.02.010

Duda, R. O., Hart, P. E., & Stork, D. G. (2001). *Pattern Classification* (2. Aufl.). New York: John Wiley & Sons.

Ertel, W. (2016). *Grundkurs Künstliche Intelligenz* (4. Aufl.). Wiesbaden: Springer Fachmedien. https://doi.org/10.1007/978-3-8348-9989-7

HDF Group. (2017). HDF Group - HDF5. Abgerufen 1. März 2017, von https://support.hdfgroup.org/HDF5/

Jähne, B. (2012). *Digitale Bildverarbeitung: und Bildgewinnung (German Edition)* (7. Aufl.). Heidelberg: Springer Verlag.

Katyal, V. (2012). *Leaf vein segmentation using Odd Gabor filters and morphological operations*. Amity University Indien.

Keller, J. M., Liu, D., & Fogel, D. B. (2016). *Fundamentals of Computational Intelligence*. New Jersey: John Wiley & Sons.

Khan, N. Y., McCane, B., & Wyvill, G. (2011). SIFT and SURF performance evaluation against various image deformations on benchmark dataset. In *Proceedings - 2011 International Conference on Digital Image Computing: Techniques and Applications, DICTA 2011* (S. 501–506). https://doi.org/10.1109/DICTA.2011.90

Kumar, N., Belhumeur, P. N., Biswas, A., Jacobs, D. W., Kress, W. J., Lopez, I., & Soares, J. V. B. (2012). Leafsnap: A Computer Vision System for Automatic Plant Species Identification. *Proceedings of the 12th European Conference on Computer Vision (ECCV)*.

Larese, M. G., Bayá, A. E., Craviotto, R. M., Arango, M. R., Gallo, C., & Granitto, P. M. (2014).

Multiscale recognition of legume varieties based on leaf venation images. *Expert Systems with Applications*, *41*(10), 4638–4647. https://doi.org/10.1016/j.eswa.2014.01.029

Ling, H. L. H., & Jacobs, D. W. (2007). Shape Classification Using the Inner-Distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *29*(2), 286–299. https://doi.org/10.1109/TPAMI.2007.41

Price, C. A., Symonova, O., Mileyko, Y., Hilley, T., & Weitz, J. S. (2011). Leaf Extraction and Analysis Framework Graphical User Interface: Segmenting and Analyzing the Structure of Leaf Veins and Areoles. *Plant Physiology*, *155*(1), 236–245. https://doi.org/10.1104/pp.110.162834

Rothmaler, W. (1967). *Exkursionsflora von Deutschland - Gefäßpflanzen* (6. Aufl.). Berlin: Volk und Wissen volkseigener Verlag.

Rothmaler, W. (2011). *Exkursionsflora von Deutschland - Gefäßpflanzen* (20. Aufl.). Heidelberg: Spektrum Akademischer Verlag.

Sivic, J., & Zisserman, A. (2009). Efficient visual search of videos cast as text retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *31*(4), 591–606. https://doi.org/10.1109/TPAMI.2008.111

Söderkvist, O. J. O. (2001). *Computer Vision Classification of Leaves from Swedish Trees*. Linköping University.

Welle, E. F. (2014). *Kleines Repetitorium der Botanik* (16. Aufl.). Hamburg: Verlag Dr. Felix Büchner - Handwerk und Technik.

Wikipedia, C. (2017). HSL and HSV. Abgerufen 22. Februar 2017, von https://en.wikipedia.org/w/index.php?title=HSL_and_HSV&oldid=765756890#/media/File:HSV_color_solid_cylinder_alpha_lowgamma.png

Wu, S. G., Bao, F. S., Xu, E. Y., Wang, Y. X., Chang, Y. F., & Xiang, Q. L. (2007). A leaf recognition algorithm for plant classification using probabilistic neural network. *ISSPIT 2007 - 2007 IEEE International Symposium on Signal Processing and Information Technology*, (December), 11–16. https://doi.org/10.1109/ISSPIT.2007.4458016

Zhao, C., Chan, S. S. F., Cham, W.-K., & Chu, L. M. (2015). Plant identification using leaf shapes—A pattern counting approach. *Pattern Recognition*, *48*(10), 3203–3215. https://doi.org/10.1016/j.patcog.2015.04.004

V. Anhang

a. Installationsanleitung

-theano.orc (Windows: unter %USERPROFILE%):

[*global]*

device = gpu

- CUDA Toolkit installieren

u.U in theano.orc (wenn cl.exe nicht gefunden oder Value '2008' is not defined for option 'cl-version')

[*nvcc]*

compiler_bindir=E:\Programme\Microsoft Visual Studio 12.0\VC\bin

utf-8 not supported: in theano.compat ändern (fett):

```
def decode_iter(itr):
    for x in itr:
        yield x.decode('gbk')
```

header-Datei fehlt: in das angegebene Verzeichnis kopieren