Erfassung und Bewertung von Substanzmerkmalen von Fahrbahnoberflächen mittels digitaler Bildverarbeitung und neuronaler Netze

Von der Naturwissenschaftlichen
Fakultät der

Gottfried Wilhelm Leibniz Universität Hannover

zur Erlangung des Grades

Doktor der Naturwissenschaften

- Dr. rer. nat. -

genehmigte Dissertation
von

Dipl.-Chem. David Geissler

geboren am 15.08.1977, in Hannover

2008

Referent: Prof. Dr. Bernd Hitzmann

Korreferent: Prof. Dr. Thomas Scheper

Tag der Promotion: 20.05.2008

Selbständigkeitserklärung

Hiermit versichere ich, die vorliegende Dissertation selbständig und nur unter Verwendung der angegebenen Hilfsmittel angefertigt zu haben. Ich versichere ferner, dass die Dissertation nicht schon als Diplomarbeit oder ähnliche Prüfungsarbeit verwendet worden ist.

Widmung

Ich bin nicht das Musterbeispiel eines Enkelkindes, neige dazu, mich nur zum Geburtstag zu melden, oder warte darauf, dass wir uns zu Weihnachten sehen. Dennoch hat meine französische Oma von ihrem einzigen Enkelkind die für Großeltern typische allerbeste Meinung. Auch hat sie sich immer sehr für meinen Werdegang und auch meine Promotion interessiert. Eine „Schwiegerenkelin“, die sie gerne kennengelernt hätte, kann ich ihr zu diesem Zeitpunkt leider nicht vorstellen (oder vielleicht doch), aber hiermit widme ich ihr diese Arbeit.

Dédicace

Je ne suis pas le meilleur des petits fils, j'ai tendance à seulement lui téléphoner pour son anniversaire, ou j’attends qu'on se revoie à Noël. Malgré tout, pour ma Mamie j'ai toujours été son formidable petit-fils. Elle s'est toujours intéressée à ma « carrière », spécialement à ma thèse de doctorat. Pour l’instant, je ne peux pas lui présenter la « belle-petite-fille » qu'elle aurait bien aimé connaître (ou peut-être que si), mais je lui dédie cette thèse de doctorat.

Danksagung

Diese Arbeit entstand im Zeitraum vom August 2003 bis März 2007 im Institut für Technische Chemie an der Universität Hannover unter der Leitung von Prof. Dr. Bernd Hitzmann, dem ich an dieser Stelle für das interessante Thema und die aufmerksame Betreuung danken will.

Außerdem danke ich Prof. Dr. Thomas Scheper für die Übernahme des Korreferats.

Ich danke Herrn Dr. Schmidt von der Firma IWS Messtechnik, dessen Projekt die Grundlage der vorliegenden Arbeit darstellt, und Herrn Villaret sowie seinen Mitarbeitern von der Villaret Ingenieurgesellschaft mbH für die Zusammenarbeit bei diesem Projekt.

Den Mitarbeitern des Institutes für Technische Chemie, mit denen ich im Laufe meiner Arbeit zu tun hatte, danke ich für ihre Hilfe und Unterstützung. An den Arbeitskreis Hitzmann geht mein besonderer Dank für die guten gemeinsamen Zeiten, sowohl am als auch jenseits des Arbeitsplatzes.

Abseits der Arbeit danke ich besonders meinen Freunden vom Volleyball. In schwierigen Zeiten habe ich dort die nötige Ablenkung gefunden, um den Kopf wieder frei zu kriegen, und in guten Zeiten einfach den nötigen Ausgleich zum Job vorm Rechner.

Zu guter Letzt danke ich besonders meinen Eltern. Ihre Unterstützung durch das gesamte Studium und darüber hinaus während der Promotion hat diese Arbeit ermöglicht.

Kurzfassung

In Deutschland gibt es ca. 30.000 km Autobahnen und Fernstraßen. Um eine rechtzeitige Ausbesserung von Schäden zu ermöglichen, ist eine regelmäßige Kontrolle notwendig, bei der Risse gefunden und bewertet werden.

Zu diesem Zweck wurde in dieser Arbeit ein Mustererkennungssystem aufgebaut, welches mit Mitteln der digitalen Bildbearbeitung und durch den Einsatz neuronaler Netze, ausgehend von digitalen Bildern, eine systematische, objektive und automatisierte Bildauswertung ermöglicht. Ein Mensch, der ein Bild betrachtet, sieht scheinbar spontan und mühelos Strukturen und Muster. Allerdings handelt es sich hierbei bereits um eine sehr komplexe Analyse. Sie basiert auf der Fähigkeit des menschlichen Gehirns visuelle Daten zu erfassen, zu abstrahieren und Muster wiederzuerkennen sowie jahrelanger Erfahrung. Für den Aufbau eines automatisierten Systems, welches dies leistet, wurde ein entsprechender Mustererkennungsprozess entwickelt.

Hierfür standen 924 Bilder eines 250 m langen asphaltierten Straßenabschnittes zur Verfügung. Die Bewertung des Bildmaterials wurde von Ingenieuren aus dem Straßenbau vorgenommen. Ein einzelnes Bild (ca. 1 m x 1,4 m) besteht aus 5 Millionen Pixel. Dies kann nicht in dieser Form direkt ausgewertet werden. Zunächst erfolgt eine Bildbearbeitung. Diese beinhaltet eine Reduktion der Farbtiefe und eine Korrektur der Ausleuchtung. Darauf folgt die Bildverarbeitung, beginnend mit der Segmentierung des Bildes. Potenziell interessante Strukturen und Objekte werden als solche erkannt, identifiziert und vom Hintergrund isoliert. Ist dies geschehen, werden die so gefundenen Objekte näher analysiert, um Kenngrößen zu extrahieren, welche die Objekte in knapper Form charakterisieren. Auf der Basis dieser Kenngrößen erfolgt dann im letzten Schritt eine Klassifizierung durch ein neuronales Netz.

Hierbei wurden im Mittel Fehlerquoten von 15-20 % Prozent erreicht. Durch eine weitere Optimierung des Systems konnte bei der Erkennung von Schäden ein Fehler von unter 10 % erreicht werden.

Schlagworte: digitale Bildbearbeitung, digitale Bildverarbeitung, digitale Filter, Segmentierung, neuronale Netze, Mustererkennung

Abstract

In Germany there are approximately 30.000 km of highways. In order to maintain and to repair these roads, they have to be monitored regularly in order to find and judge fissures in time.

For this task a pattern recognition system was established, based on digital image processing and neuronal networks. It provides a systematic, objective and automated analysis for digital images. When humans look at images they seem to recognize pattern and structures spontaneously without any effort. However this is a very complex analysis. It is based on the ability of the human brain to understand visual data, to abstract and to recognize patterns and on many years of experience. In order to build a system providing these abilities, a suitable pattern recognition system was designed.

For this task 924 images of a 250 m long road section were available. The evaluation of these images was done by road construction engineers. One images has a resolution of 5 million pixels. This amount of data cannot be evaluated directly. Preprocessing is needed such as reducing the color depth and correcting the lightning of the images. Further the images have to be segmented. This means that potentially interesting structures and objects that might represent fissures, have to be found and isolated from the background. When this is done the objects found have to be analyzed in order to extract features, characteristic values describing these objects in a lean form. Based on these extracted features a classification is done in the last step, using a neuronal feedforward network.

This way a mean error rate of 15-20 % was achieved. With further optimization error rates below 10 % were achieved for the detection of damages.

Keywords: digital image processing, digital filter, segmentation, neuronal networks, pattern recognition

Inhaltsverzeichnis

1 Einleitung

In der Technischen Chemie steht weniger die Chemie sondern die Umsetzung und Realisierung von chemischen Verfahren im Vordergrund. Beginnend im Labormaßstab, geht dies über den Technikumsmaßstab bis hin zum industriellen Maßstab. Um Prozesse und Vorgänge, die im kleinen Maßstab realisiert werden können, in einen industriellen Maßstab überführen zu können, bedarf es verschiedenster Mess- und Regelungstechniken. So werden bei Bioprozessen online häufig die Temperatur, der pH-Wert und die Zusammensetzung des Abgasstroms gemessen. Ebenso wird der gelöste Sauerstoff- sowie Kohlendioxid-gehalt gemessen. Mit zusätzlichem apparativem Aufwand lässt sich mit einem FIA-System ein Substrat wie Glucose online messen. Durch die Offline-Analyse von Proben kann letztlich jede gewünschte Komponente quantifiziert werden, aber ein besonderes Augenmerk liegt auf Methoden, die online messen und nicht invasiv sind. In den letzten Jahren hat in dieser Hinsicht die 2D-Fluoreszenzspektroskopie als optische Methode an Bedeutung gewonnen. Ein weiterer Ansatz ist die Realisierung eines Insitu-Mikroskops, welches mikroskopische Aufnahmen am laufenden Prozess ermöglicht. Die Masse an Daten in Form von digitalen Bildern kann es wiederum erforderlich machen, diese systematisch und automatisiert mit statistischen Methoden und Mitteln der digitalen Bildverarbeitung hinsichtlich Zellzahl, Zellgröße oder anderen relevanten Kriterien auszuwerten.

Im Rahmen dieser Arbeit geht es um die Bewertung von Bitumenoberflächen und die Auffindung von Rissen, basierend auf digitalen Aufnahmen. Bei den zu untersuchenden Oberflächen handelt es sich um den Belag deutscher Fernstraßen und Autobahnen. Zur Wartung und um eine rechtzeitige Ausbesserung von Schäden zu ermöglichen, ist eine regelmäßige Kontrolle notwendig. Derzeit erfolgt die Überwachung durch Straßenmeister oder mit Hilfe eines Videowagens. Die VHS-Aufnahmen werden im Nachhinein von fachkundigem Personal am Bildschirm ausgewertet. Eine solche Bewertung ist allerdings sehr subjektiv und fehlerbehaftet. Daher gibt es Bemühungen, die VHS-Technik durch hochauflösende,

handelsübliche Fotokameras zu ersetzen. Die einzelnen Bilder sollen dann einer automatisierten Auswertung durch ein Computersystem übergeben werden.

Das Bildmaterial liegt mit einer Auflösung von fünf Megapixeln vor. Ein Bild zeigt ungefähr einen 1 m x 1,4 m Straßenabschnitt. Diese große Datenmenge kann nicht ohne Vorverarbeitung direkt zur Auswertung herangezogen werden. Daher wurde in dieser Arbeit ein Mustererkennungssystem aufgebaut, welches die digitale Vorverarbeitung des Bildmaterials, die Extraktion von Kenngrößen und eine Beurteilung auf der Basis dieser Kenngrößen durch neuronale Netze umfasst. Am Ende steht eine systematische, objektive und automatisierte Bildauswertung.

2 Theorie

2.1 Mustererkennung

Die Erkennung und Identifizierung von Mustern ist für den Menschen eine selbstverständliche, alltägliche Erfahrung. Von klein auf lernen wir Formen und Farben zu unterscheiden, Objekte zu identifizieren, Sprache zu verstehen und Vieles mehr. All dies, was dem Menschen selbstverständlich erscheint, ist das Ergebnis einer langen Evolution und eines jahrelangen individuellen Trainings. Tatsächlich ist die Erkennung wie auch immer gearteter Muster ein sehr schwieriger und komplexer Vorgang.

Seit jeher gibt es das Bestreben, natürliche Systeme durch mechanische, technische und elektronische Hilfsmittel zu imitieren. So sind heute Sicherheitssysteme im Einsatz, die auf der Identifizierung von Fingerabdrücken oder Gesichtern basieren, Handys werden über Sprachbefehle gesteuert und Texte werden mittels sogenannter OCR-Systeme (optical character recognition) digitalisiert. Trotz dieser Fortschritte haben diese Systeme auch Grenzen, die verdeutlichen, wie komplex und schwierig diese Aufgaben sind. Fingerabdrücke und Gesichter werden nur erkannt, wenn genormte Aufnahmen guter Qualität vorliegen, die Spracherkennung versagt, wenn sie nicht auf den jeweiligen Benutzer trainiert wurde, und die Texterkennung erkennt keine Handschriften, die die meisten Mensch lesen könnte.

Je mehr man sich mit den Problemen der Mustererkennung und der Lösung der dabei auftauchenden Probleme befasst, desto deutlicher wird, welche Vorgänge bei der Mustererkennung ablaufen (1).

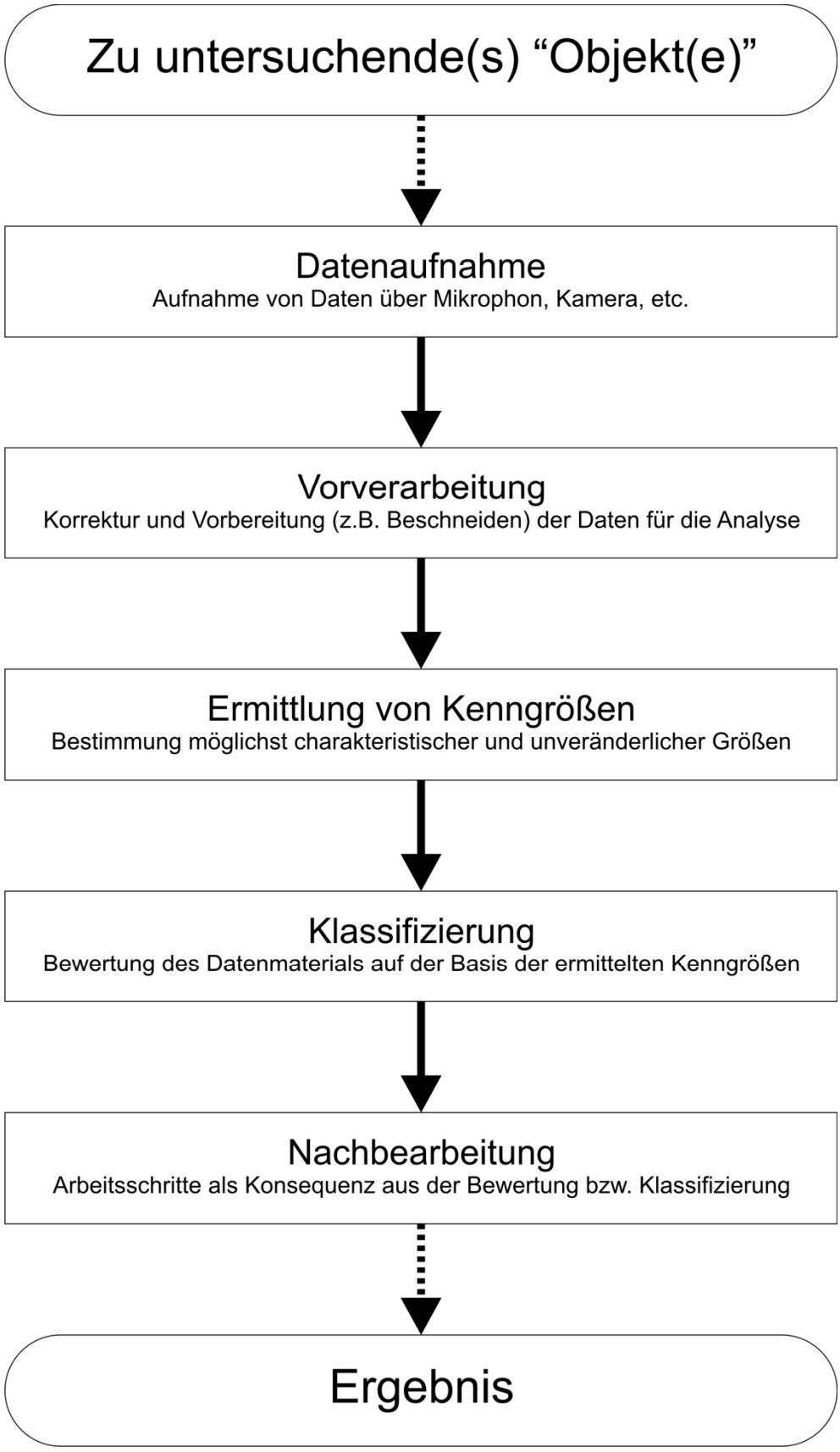


Abbildung 1: Schematischer Ablauf eines Mustererkennungsprozesses

Unabhängig von der Art der zu identifizierenden Muster lässt sich ein solcher Prozess in die folgenden, auch in Abbildung 1 dargestellten, wesentlichen Arbeitsschritte einteilen:

- Datenaufnahme
- Vorverarbeitung
- Ermittlung von Kenngrößen
- Klassifizierung
- Nachbearbeitung

2.1.1 Datenaufnahme

Über ein Mikrophon, eine Kamera oder ein anderes geeignetes Gerät werden Daten aufgenommen, in denen die relevanten bzw. auszuwertenden Informationen enthalten sind.

Hierbei ist wichtig, dass die Datenaufnahme der Aufgabenstellung entsprechend gestaltet wird. Eine Aufnahme sollte möglichst frei von unnötigen und vermeidbaren Verfremdungen sein und eine ausreichende Auflösung aufweisen. Bei bestimmten Systemen, wie zum Beispiel der Gesichtserkennung, ist auch eine Aufnahme unter definierten Rahmenbedingungen erforderlich.

2.1.2 Vorverarbeitung

Die aufgenommenen Daten müssen gegebenenfalls für die weitere Verarbeitung vorbereitet werden. Das kann die Korrektur von Aufnahmefehlern, das Beschneiden auf relevante Teildatensätze oder eine Form der Normierung beinhalten. Dies ist nicht der Fall, wenn zum Beispiel die Aufnahme eines einzelnen Fingerabdrucks vorliegt, die unter gewissen Vorgaben angefertigt wurde. Bei der automatisierten Videoüberwachung eines Fließbandes jedoch, auf dem zum Beispiel Äpfel und Birnen sortiert werden, müssen diese auf der Aufnahme zunächst isoliert werden. Dann kann jedes Objekt für sich untersucht werden, um zu unterscheiden, ob es sich um das eine oder das andere handelt.

Allerdings kann dies bereits zu einem Problem werden. Hat man es mit einem deutlich erkennbaren Objekt vor einem einfarbigen Hintergrund zu tun, kann

dieses gut isoliert werden. Probleme treten dann auf, wenn sich mehrere Objekte überlagern oder nicht ohne weiteres vom Hintergrund getrennt werden können. In diesen Fällen setzt die erfolgreiche Isolierung einzelner Objekte bereits gewisse Kenntnisse voraus. Kenntnisse, die man gegebenenfalls noch gar nicht hat, wenn es darum geht zu entscheiden mit was für einem Objekt man es eigentlich zu tun hat.

2.1.3 Ermittlung von Kenngrößen

Digitale Bilder, als Beispiel, besitzen heute eine Auflösung von mehreren Megapixeln. Diese Datenmenge kann nicht direkt ausgewertet werden. Zunächst muss nach Formen, Strukturen und markanten Punkten gesucht werden. Die für die zu identifizierenden Objekte charakteristischen Eigenschaften müssen ermittelt werden. Informationen über deren Vorhandensein, Lage und Größe können dann als Kenngrößen zusammengetragen werden.

Merkmale, die je nach Lage des Objekts verdeckt sein können, sind hierfür jedoch ungeeignet. Die Merkmale müssen gegebenenfalls auch größenunabhängig sein, da ein kleiner Apfel nicht weniger ein Apfel ist, nur weil er kleiner ist als ein anderer. Und auch feste Formen können bei dreidimensionalen Objekten, bedingt durch perspektivische Verzerrungen, sehr unterschiedlich erscheinen. Die Herausforderung besteht also darin, wirklich kennzeichnende und möglichst unveränderliche Größen zu finden.

2.1.4 Klassifizierung

Auf der Basis der ermittelten Kenngrößen wird letztlich eine Entscheidung getroffen. Die Grenze zwischen der Ermittlung der Kenngrößen und der endgültigen Klassifizierung ist dabei nicht klar definiert. Je besser die Ermittlung von Kenngrößen funktioniert - dem kann bereits eine gewisse Interpretation der Daten zugrunde liegen - desto einfacher wird die Klassifizierung. In den einfachsten Fällen kann sich dies auf die Entscheidung beschränken, ob das Objekt dem einen oder dem anderen Kriterium genügt, ob es eine bestimmte Grenze unter- oder überschreitet. In anderen Fällen hingegen liegt es an der Klassifizierung auf der Basis einer Menge nicht eindeutiger Kenngrößen eine korrekte Bewertung vorzunehmen.

In der Praxis letztlich wird man es meistens mit mehreren Größen, zwischen denen man abwägen muss, zu tun haben. Die Klassifizierung kann in solchen Fällen zum Beispiel auf der Basis einer multilinearen Regression, neuronaler Netze oder unter Verwendung von Fuzzy Logic realisiert werden.

2.1.5 Nachbearbeitung

Als letzter Schritt eines Mustererkennungsprozesses kann noch eine Nachbearbeitung folgen. Dabei können zum Beispiel, abhängig vom Ergebnis der Klassifizierung, bestimmte Prozesse in Gang gesetzt werden. Bei der Spracherkennung eines Handys würde dies bedeuten, dass entweder die angeforderte Nummer gewählt oder der Benutzer wird zur Wiederholung seines Befehls aufgefordert wird, wenn keine entsprechende Nummer gefunden werden konnte.

Allerdings ist die Grenze zwischen Klassifizierung und Nachbearbeitung fließend. Liegen mehrere Klassifizierungen vor, entspricht dies einer Anfrage bei mehreren Experten, welche gegebenenfalls unterschiedliche Antworten geben. Das Abwägen der verschiedenen Antworten zur Findung einer eindeutigen Aussage ist dann im Grunde wieder eine Frage der Klassifizierung.

2.2 Digitale Bildbearbeitung

2.2.1 Filter und Konvolution

Die digitale Bildbearbeitung bietet eine Vielzahl von Möglichkeiten (2)(3). Diese reichen von einfachen Bildmanipulationen, wie dem Aufhellen eines Bildes, bis hin zu komplizierten Algorithmen mit dem Ziel, bestimmte Informationen aus einem Bild zu gewinnen. Eine grundlegende, aber schon sehr weitreichende, Form der Bildmanipulation sind die so genannten Filter. Sie werden eingesetzt, um Fehler zu korrigieren, Rauschen zu entfernen, den Kontrast zu verbessern oder um bestimmte Merkmale, wie zum Beispiel Kanten oder andere für die Verarbeitung relevante Merkmale, hervorzuheben. Die Filter lassen sich in drei Klassen aufteilen.

Bei der ersten Klasse handelt es sich um Punktoperatoren. Bei diesen Filtern wird für die Berechnung des neuen Farbwertes eines Pixels lediglich die Farbinformation des Pixels benötigt. In diese Klasse fallen zum Beispiel Gamma-, Helligkeits- und Kontrastkorrektur. Eine automatische Kontrastverbesserung wäre bereits ein Grenzfall, da zunächst das gesamte Bild analysiert wird, um anhand eines Histogramms zu bestimmen, wie die vorhandenen Farbwerte optimal über den gesamten Wertebereich verteilt werden können.

Bei der zweiten Klasse handelt es sich um lokale Operatoren. Diese berechnen den neuen Farbwert eines Pixels unter Einbeziehung der Farbwerte der Pixel in der näheren Umgebung. In diese Klasse fallen eine Vielzahl von Weichzeichnern bzw. Unschärfe-Filter (Mittelwert-Filter, Gauß-Filter, etc.) oder die Kantenerkennung mit dem Sobel-Operator.

Bei der letzten Klasse handelt es sich um die globalen Operatoren. Ein Beispiel für globale Operatoren ist die Fouriertransformation eines Bildes, mit der das Bild vom normalen Abbildungsraum in den Frequenzraum überführt wird.

Grenzwertfilter

Der Grenzwertfilter in seiner einfachsten Form gehört zu der Gruppe der Punktoperatoren. Er ist in der digitalen Bildverarbeitung eine gängige Methode, wenn es darum geht, ein Objekt, welches sich in seiner Helligkeit deutlich vom Hintergrund absetzt, von eben diesem zu trennen. Das Ergebnis ist meist ein schwarz-weiß Bild in dem schwarze Pixel den Hintergrund bzw. uninteressante Bereiche und weiße Pixel die gesuchten Objekte bzw. interessante Bereiche markieren (ggf. auch umgekehrt).

Ausgangsmaterial sind meist Graustufen-Bilder. Liegt ein Farbbild vor, kann dieses im Vorfeld in ein Graustufen-Bild konvertiert werden. Alternativ kann der Grenzwertfilter auf nur einen der Farbkanäle (i. A. rot, grün und blau) angewandt werden. Voraussetzung für eine erfolgreiche Trennung ist aber eine entsprechende Helligkeitsverteilung.

Abbildung 2 zeigt zwei Beispiel-Histogramme. Abbildung 2a zeigt ein Histogramm mit einer eindeutigen Helligkeitsverteilung. Hier kann ein klarer Grenzwert definiert und das Bild entsprechend unterteilt werden. Abbildung 2b hingegen weist eine Helligkeitsverteilung auf, bei der es sehr unwahrscheinlich ist, dass das Bild auf diesem Weg erfolgreich unterteilt wird.

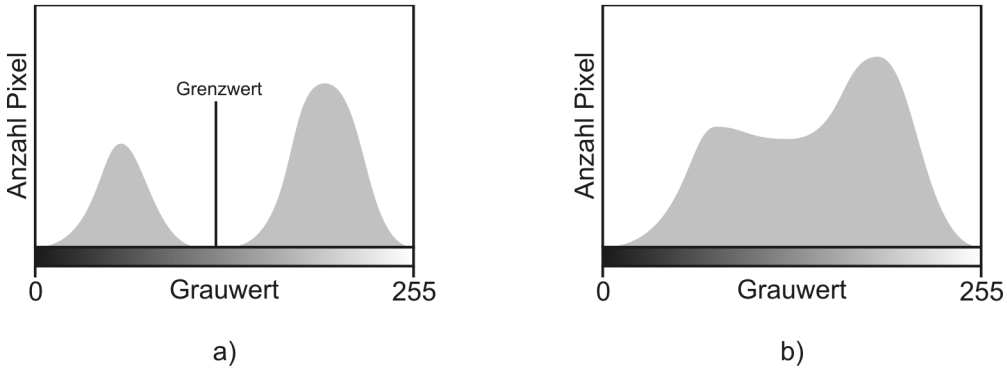


Abbildung 2: Beispiel-Histogramme: a) Eindeutige Helligkeitsverteilung b) Stark überlappende Helligkeitsverteilung

Konvolutionsfilter

Bei den Konvolutionsfiltern handelt es sich um lokale Operatoren. Konvolution ist eine einfache mathematische Operation, bei der eine Wichtungsmatrix, in diesem Zusammenhang Konvolutionskern ***K*** genannt, über ein Bild ***B*** bewegt wird. Die einzelnen Matrixelemente geben an, wie die jeweiligen umliegenden Pixel verrechnet werden müssen. Um für einen zentralen Pixel einen neuen Farbwert zu berechnen bietet es sich an, einen Kern mit ungerader Spalten- und Reihenzahl, wie in Abbildung 3 dargestellt, zu verwenden.

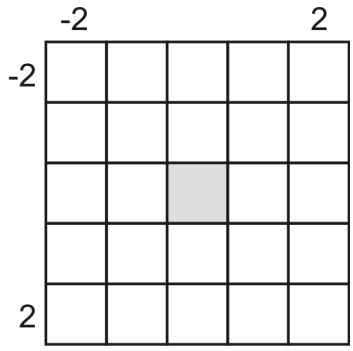


Abbildung 3: Ein Konvolutionskern ***K*** mit einem Radius *r* von 2

Dabei liegt das Matrixelement 0,0 (grau markiert) auf dem Bildpunkt, für den ein neuer Wert berechnet werden soll. Dieser Ausgabewert a für eine Position x/y wird dann gemäß Gleichung 1 berechnet. Alle umliegenden Pixel b im Bereich des Konvolutionskerns mit dem Radius r werden mit dem entsprechenden Matrixelement k multipliziert und aufsummiert.

$$a_{x,y} = \sum_{i=-r}^{r} \sum_{j=-r}^{r} b_{x+i,y+j} k_{i,j} \tag{1}$$

Daraus folgt auch, dass die Konvolution für einen Bildrand, der dem Radius des Kerns entspricht, nicht definiert ist, da Pixel außerhalb des Bildes in die Berechnung einfließen würden. Wie mit diesen Bildbereichen verfahren wird, muss separat definiert werden. Die Pixel können unverändert bleiben, pauschal auf einen Wert von 0 oder 255 gesetzt werden, oder die Pixel am Bildrand werden gespiegelt, um die Konvolution bis zum Bildrand durchführen zu können.

Abbildung 4 zeigt einige gängige Konvolutionskerne. Bei den Kernen a) und b) handelt es sich um die beiden Kerne des Sobeloperators (4)(5). Der erste dient zum Auffinden von Kanten in x-Richtung und der zweite in y-Richtung. Durch eine Kombination beider Kerne können Kanten gefunden und ihre Ausrichtung bestimmt werde. Kern c) stellt einen Mittelwertfilter dar und d) ist der Kern für einen Gaußschen Unschärfe-Filter.

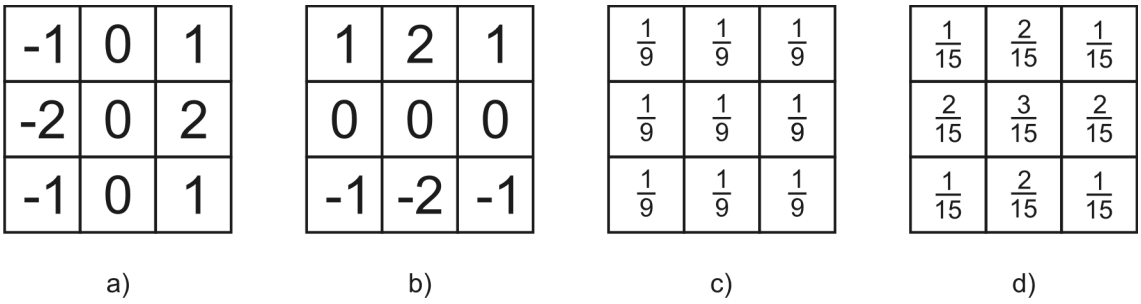


Abbildung 4: Verschiedene Konvolutionskerne: a) Sobel x-Gradient b) Sobel y-Gradient c) Mittelwert-Filter d) Gauß-Filter

2.3 Neuronale Netze

Künstliche neuronale Netze (KNN), im Englischen *artificial neural networks* (ANN), sind der technische Ansatz, ein biologisches System zum Lösen komplexer Probleme zu imitieren (6)(7). Als Vorbild hierzu dient das Nervensystem lebender Organismen. Letztlich handelt es sich bei einem neuronalen Netz um ein signalverarbeitendes System auf der Basis kleiner vernetzter Recheneinheiten. Durch die Anpassung von Struktur und Verbindungsstärken können die Netze auf eine Vielzahl von Aufgaben trainiert werden. Genauere Ausführungen zur Anwendung sind bei Zell (8) nachzulesen.

Im menschlichen Gehirn sind ca. 10^{10} bis 10^{11} Neuronen bzw. Nervenzellen zu einem komplexen Netzwerk verbunden. Gemessen daran, sind künstliche neuronale Netze eine stark vereinfachte Annäherung. Die Anzahl verwendeter Neuronen in neuronalen Netzen ist deutlich kleiner. Außerdem bleiben exakte chemische und zeitliche Vorgänge sowie die chemische Interaktion räumlich benachbarter Nervenzellen unberücksichtigt. Für die Bearbeitung wissenschaftlicher Problemstellungen wäre eine vollständige Modellierung weder effizient noch erforderlich (8)(9). Beibehalten werden im Wesentlichen die massive Parallelität einfacher Elemente (Neuronen), die über gerichtete Verbindungen kommunizieren. Hierbei werden keine Symbole oder Datenstrukturen übermittelt, sondern einfache Signale, die über eine Wichtung der Verbindungen, welche im Rahmen eines Lernprozesses angepasst werden können. Genauere Ausführungen

zu den Unterschieden zwischen neuronalen Netzen und ihren biologischen Vorbildern sind bei Zell zu finden (8).

2.3.1 Das Neuron

Das Neuron ist als kleinste Recheneinheit eines neuronalen Netzes eine stark idealisierte Nervenzelle. Abbildung 5 zeigt den schematischen Aufbau einer Wirbeltier-Nervenzelle mit ihren vier wesentlichen Bausteinen (10).

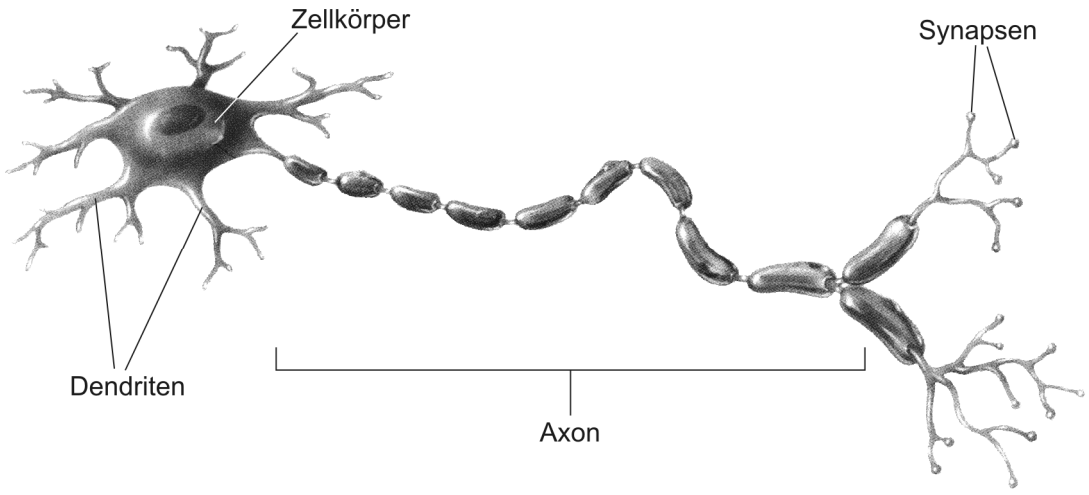


Abbildung 5: Schematischer Aufbau einer myelinisierten Wirbeltier-Nervenzelle (11)

Dendriten

Die Dendriten sind die Schnittstelle für eingehende Informationen. Es sind auch andere Verbindungen zwischen einzelnen Nervenzellen möglich, aber im Allgemeinen sind es die Dendriten, die für die Aufnahme von Informationen und ihre Weiterleitung zum Zellkörper verantwortlich sind.

Zellkörper

Der Zellkörper beinhaltet alles, was die Zelle zum existieren braucht. Hier befinden sich die Organellen, und hier findet der Stoffwechsel statt. Darüber hinaus werden im Zellkörper alle eingehenden Signale gesammelt, verarbeitet und schließlich weitergeleitet.

Axon

Das Axon ist die abgehende Verbindung der Zelle, welche die verarbeiteten Signale in Form elektrischer Impulse weiterleitet. Es ist stark verzweigt und nimmt mit anderen Nervenzellen Kontakt auf.

Synapse

Die Synapsen stellen die Enden des stark verzweigten Axons dar. Sie können verstärkend oder hemmend wirken. Somit liegt ein wesentlicher Teil der Funktionalität der Nervenzellen in diesen Verbindungen, da die Stärke der interzellularen Verbindungen wesentlichen Einfluss auf die Funktion des Netzwerkes hat.

Wie bereits angesprochen, sind die Vorgänge in Nervenzelle zu komplex, als dass es sinnvoll wäre, die exakte biologische Funktion zu simulieren. Abbildung 6 zeigt den vereinfachten schematischen Aufbau eines Neurons in einem künstlichen neuronalen Netz (12).

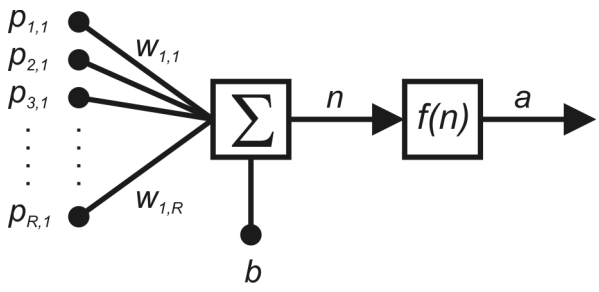


Abbildung 6: Schematischer Aufbau eines Neurons in einem künstlichen neuronalen Netz[1]

Das Neuron bekommt einen Eingangsvektor $\boldsymbol{p}$ mit R Elementen, welcher mit einem Wichtungsvektor $\boldsymbol{w}$ gleicher Dimension multipliziert wird. Dies entspricht den synaptischen Verbindungen an den Dendriten einer realen Nervenzelle.

[1] Die Nomenklatur der einzelnen Größen und Parameter entspricht der Neuronale Netze Toolbox Version 4 in Matlab 6.5

Im Zellkörper werden alle gewichteten Eingangssignale aufsummiert und ggf. ein Bias b addiert. Das neue summierte Signal n (auch netinput genannt) durchläuft dann die Transferfunktion $f(n)$. Im einfachsten Fall handelt es sich um eine lineare Transferfunktion (vergl. Abbildung 7), was bedeutet, dass das aufsummierte Signal unverändert weitergeleitet wird.

Das transformierte Ausgangssignal a letztlich entspricht der Information, die über das Axon an weitere Nevenzellen weitergeleitet wird.

Die mathematische Arbeitsweise eines Neurons wird durch Gleichung 2 beschrieben.

$$a = f\left(\boldsymbol{wp} + b\right) \tag{2}$$

Jeder Eingangswert $p_{i,1}$ wird mit einem entsprechenden Wichtungsfaktor $w_{1,i}$ multipliziert und aufsummiert, was der Matrixmultiplikation der beiden Vektoren entspricht (Gleichung 3).

$$\boldsymbol{wp} = p_{1,1}w_{1,1} + p_{2,1}w_{1,2} + \ldots + p_{R,1}w_{1,R} \tag{3}$$

Hierzu addiert man noch den Bias b und erhält das sogenannte Netinput n. Dies wird an die Transferfunktion $f(n)$ übergeben und man erhält a, die Ausgabe des Neurons.

2.3.2 Transferfunktionen

Neben den Verbindungsstärken (definiert durch die Wichtungsfaktoren) wird das Verhalten eines Neurons durch die Transferfunktion bestimmt. Ein Auszug häufig verwendeter Funktionen ist in Abbildung 7 gegeben. Je nach Anwendung kommen verschiedene dieser Transferfunktionen zum Einsatz.

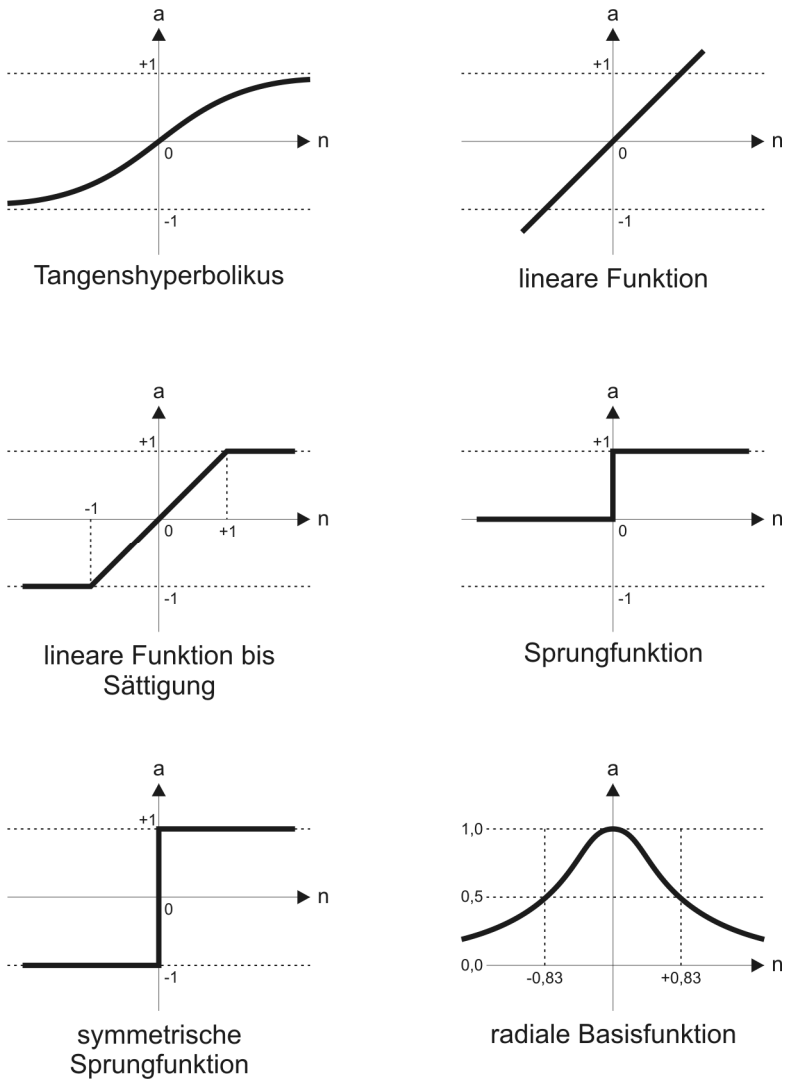


Abbildung 7: Auszug häufig verwendeter Transferfunktionen

Die einfachste Funktion ist die schon angesprochene lineare Funktion. Sie verändert das Ausgangssignal des Neurons nicht weiter und kommt in linearen Filtern oder in der linearen Ausgabeschicht anderer Netzwerke zum Einsatz.

Die Stufenfunktion begrenzt die Ausgabe eines Neurons auf die Werte 0 und 1 (die symmetrische Stufenfunktion auf -1 und 1). Sie kommt zum Beispiel in Percepton-Netzwerken zum Einsatz, wenn Neuronen eine Klassifizierungs-Entscheidung zu treffen haben.

Die Tangenshyperbolikus-Funktion staucht die Eingabe auf einen Wertebereich zwischen -1 und 1. Die Funktion hält die Ausgabewerte der Neuronen innerhalb eines bestimmten Wertebereichs und ist zudem differenzierbar. Daher findet sie oft in Backpropagation-Netzwerken Verwendung. Im Gegensatz zu einer linearen Transferfunktion mit Sättigung ist sie darüberhinaus auch in der Lage, nicht lineare Probleme zu beschreiben.

Die radiale Basisfunktion ist der Kern einer weiteren Klasse von Netzwerken, den Radial Basis Netzwerken. Solche Neuronen arbeiten nicht nach dem zuvor beschriebenen Prinzip. Die Eingaben werden hier nicht mit den Wichtungsfaktoren multipliziert. Stattdessen arbeitet die Transferfunktion mit der Differenz zwischen Eingabe und Wichtung.

2.3.3 Aufbau von Netzwerken

Ein einzelnes Neuron kann nicht viel ausrichten. Mit einer linearen Transferfunktion entspräche es einem multilinearen Regressionsmodell. Allerdings können mehrere der vorgestellten Neuronen zu Schichten zusammengefasst werden (12). Abbildung 8 zeigt eine solche Schicht.

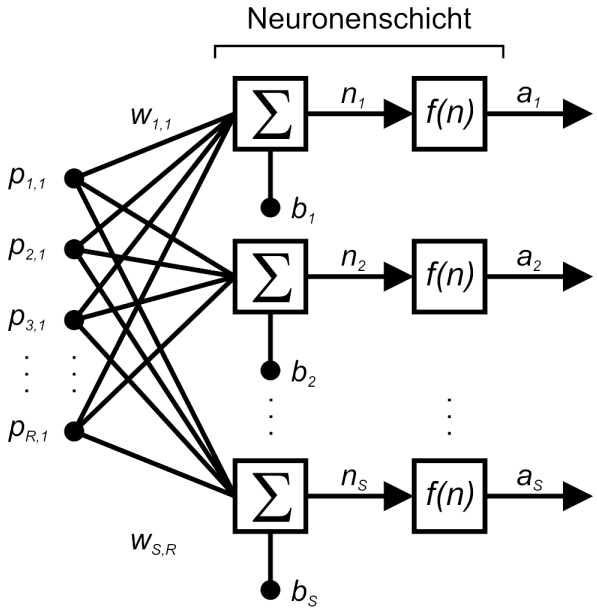


Abbildung 8: Schicht aus *S* Neuronen

Diese Schicht besteht aus *S* Neuronen. Der Eingangsvektor $\boldsymbol{p}$ wird nun mit einer Matrix von Wichtungsfaktoren $\boldsymbol{W}$ multipliziert, deren Spaltenzahl der Dimension *R* des Eingangs-Vektors und dessen Reihenzahl der Anzahl an Neuronen *S* entspricht. Jedes der *S* Neuronen addiert einen entsprechenden Bias und liefert über seine Transferfunktion einen entsprechenden Ausgabewert. Als Ausgabe der Schicht erhält man nun einen Vektor, der mit der Dimension *S* der Neuronenzahl der Schicht entspricht.

$$\boldsymbol{a} = f\left(\boldsymbol{W}\boldsymbol{p} + \boldsymbol{b}\right) \tag{4}$$

Diese Schichten wiederum lassen sich zu mehrschichtigen Netzen kombinieren. Die Ausgabe einer Schicht dient dabei als Eingabe für eine folgende Schicht. Dies ist in Abbildung 9 schematisch dargestellt.

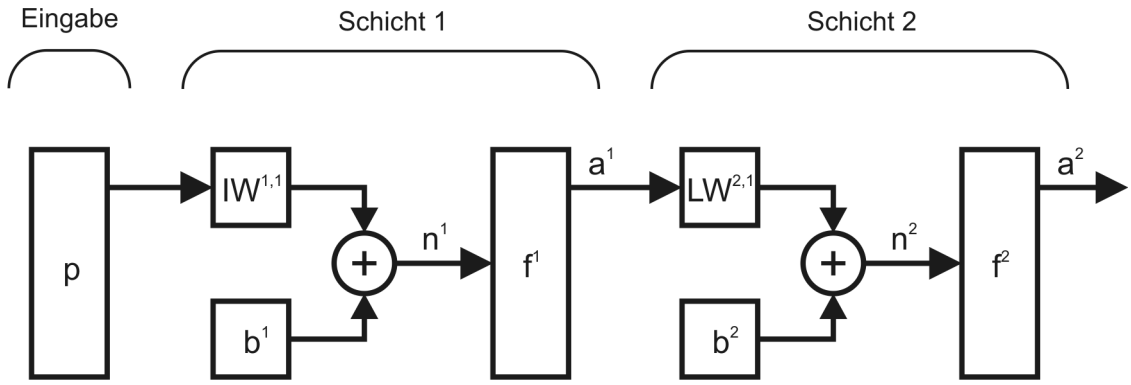


Abbildung 9: Ein neuronales Netz aus zwei Schichten

Die Netzeingabe $\boldsymbol{p}$ geht an die erste Schicht und wird dort, wie zuvor beschrieben, verarbeitet. Solche netzinternen Schichten werden als verdeckte Schichten bezeichnet. Die Wichtungsmatrix wird in diesem Zusammenhang $\boldsymbol{IW}$ (input weights) genannt, da sie sich auf die Netz-Eingabe (input) bezieht. Die Ausgabe der ersten (verdeckten) Schicht $\boldsymbol{a^1}$ dient dann als Eingabe für die zweite Schicht. Innerhalb des Netzes werden die verwendeten Wichtungsmatrizen mit $\boldsymbol{LW}$ (layer weights) bezeichnet. In diesem Fall $\boldsymbol{LW^{2,1}}$, da es sich um die Wichtungsmatrix der

zweiten Schicht handelt und die Ausgabe der ersten Schicht gewichtet wird. Die Ausgabe der zweiten Schicht könnte wiederum als Eingabe einer dritten Schicht dienen. In diesem Beispiel ist dies die letzte Schicht, und somit entspricht die Ausgabe der zweiten Schicht $\boldsymbol{a^2}$ der Netz-Ausgabe $\boldsymbol{y}$. Diese letzte Schicht wird entsprechend auch als Ausgabeschicht bezeichnet. Die mathematische Beschreibung für die vollständige Berechnung der Netzausgabe ist in Gleichung 5 dargestellt.

$$\boldsymbol{a}^2 = f^2\left(\boldsymbol{LW}^{2,1} f^1\left(\boldsymbol{IW}^{1,1}\boldsymbol{p}+\boldsymbol{b}^1\right)+\boldsymbol{b}^2\right)=\boldsymbol{y} \tag{5}$$

Nach diesem Prinzip lassen sich beliebige, schichtweise verbundene feedforward Netze aufbauen. Alle Neuronen einer Schicht sind mit allen Neuronen der folgenden Schicht verbunden und es gibt keine Rückkopplungen.

Meistens kommen in den verdeckten Schichten solcher Netze sigmoide Transferfunktionen zum Einsatz. Dies ermöglicht es dem Netz, sowohl lineare als auch nichtlineare Zusammenhänge zu lernen und zu beschreiben. In der Ausgabeschicht können über die lineare Funktion beliebige Werte ausgegeben werden. Will man hingegen die Ausgabe limitieren, kann auch hier eine sigmoide Funktion oder die lineare Funktion mit Sättigung eingesetzt werden.

2.3.4 Training neuronaler Netze

Bevor ein neuronales Netz zum Einsatz kommen kann, muss es trainiert werden. Hierbei kann zwischen unüberwachtem, bestärkendem und überwachtem Lernen unterschieden werden. Beim unüberwachten Lernen werden dem neuronalen Netz nur Eingaben präsentiert und es erhält keine Rückmeldung über die Richtigkeit der Ausgabe. Diese Methode kommt unter anderem bei *self organizing maps* zum Einsatz (8)(13)(12). Hierbei können zum Beispiel ähnliche Eingaben in Klassen eingeteilt werden. Die nächste Stufe ist das bestärkende Lernen. Hierbei wird dem Netz zwar kein Ausgabemuster zum Vergleich geboten, es erhält aber eine Rückmeldung, ob die Ausgabe richtig oder falsch war. Beim überwachten Lernen wird den neuronalen Netzen zu jedem Eingabemuster die korrekte Ausgabe präsentiert. Abbildung 10 illustriert das grundlegende Prinzip dieses Trainings.

Das neuronale Netz wird mit einer bestimmten Eingabe versorgt, aus der eine entsprechende Ausgabe erzeugt wird. Die vom Netz berechnete Ausgabe wird daraufhin mit einer bestimmten Zielvorgabe verglichen. Im letzten Schritt werden die Parameter des neuronalen Netzes, die Wichtungsfaktoren, angepasst, um eine Annäherung der Ausgabe an die gewünschte Zielvorgabe zu erreichen. Ein solcher Durchgang (Berechnung der Ausgabe, Vergleich mit den Zielwerten, Anpassung der Wichtungsfaktoren) wird Epoche oder Generation genannt.

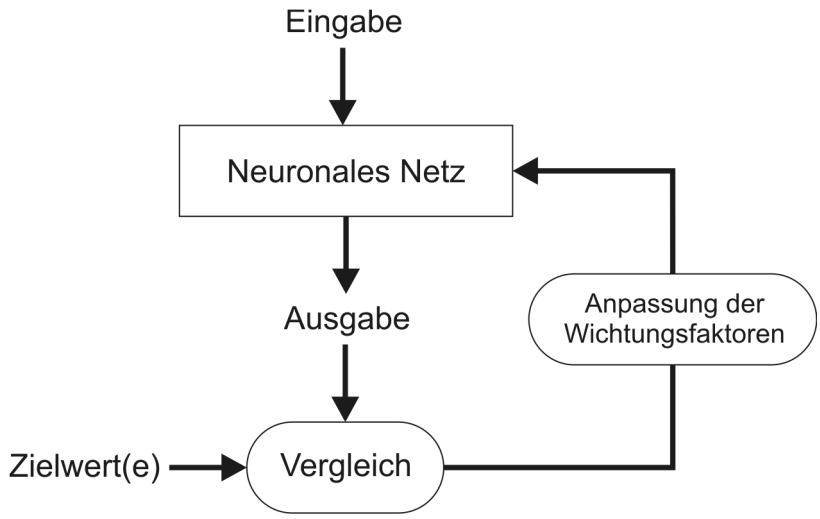


Abbildung 10: Trainingsprinzip eines neuronalen Netzes beim überwachten lernen

Durch die wiederholte Präsentation von Eingaben und zugehörigen Zielvorgaben wird das Netz soweit angepasst, dass es den Zusammenhang zwischen diesen korrekt wiedergeben kann. Bei richtigem Training ist ein solches Netz auch in der Lage, neue Muster, die nicht Teil des Lernprozesses waren, korrekt zu bewerten.

2.3.5 Lernregeln

Die Lernregel ist es, die es dem neuronalen Netz im Laufe des Trainings erlaubt sich anzupassen. Sie ist somit ein zentraler Punkt des Trainings neuronaler Netze. Es gibt verschiedene Möglichkeiten des Lernens. Es können neue Verbindungen zwischen Neuronen geschaffen werden, die Funktionsweise der Neuronen kann

verändert werden oder Neuronen können hinzugefügt oder entfernt werden. Die mit Abstand häufigste Methode ist allerdings die Modifikation der Wichtungsfaktoren (8). Diese impliziert in gewisser Weise auch das Löschen existierender bzw. die Erschaffung neuer Verbindungen durch entsprechende Werte für die Wichtungsfaktoren (0: gelöschte Verbindung, ungleich 0: bestehende Verbindung).

Für die Anpassung der Wichtungsfaktoren wurden im Laufe der Zeit verschiedene Lernregeln formuliert. Die erste, die Hebbsche Lernregel, wurde 1949 von Donald O. Hebb formuliert (14). Ihr folgte die Delta-Regel, oder auch Widrow-Hoff-Regel (15), und als Verallgemeinerung der Delta-Regel auf mehrschichtige Netze mit nichtlinearen Transferfunktionen letztlich die Backpropagation-Regel.

Backpropagation ist ein Gradientenverfahren. Für ein hypothetisches Netz mit nur zwei Parametern ließe sich der Fehler des Netzes, eine Funktion seiner zwei Parameter, als Fehlerfläche darstellen. Ziel ist es dann, sich durch Anpassung der Parameter auf dieser Fehlerfläche möglichst schnell entlang der Richtung des steilsten Abstiegs zu einem Fehlerminimum zu bewegen.

Während des Trainings eines neuronalen Netzes gilt es, den Fehler über alle zum Training präsentierten Muster zu minimieren. Dabei werden in einem Trainingsschritt, einer Generation oder Epoche, alle Wichtungsfaktoren des Netzes entsprechend angepasst. In welcher Richtung eine Minimierung des Fehlers zu erreichen ist, kann aus den verwendeten Transferfunktionen, bzw. aus deren Ableitung ermittelt werden. Dies ist der Grund, warum bei Backpropagation stetige, vollständig differenzierbare Transferfunktionen eingesetzt werden. Nach diesem Prinzip lässt sich die folgende allgemeine *Backpropagation-Regel* in Gleichung 6 ableiten.

$$\Delta w_{ij} = \eta a_i \delta_j = \begin{cases} f'(n_j)(t_j - a_j) & \text{falls } j \text{ Ausgabezelle ist} \\ f'(n_j)\sum_k \delta_k w_{jk} & \text{falls } j \text{ verdeckte Zelle ist} \end{cases} \quad (6)$$

Mit Hilfe von Gleichung 6 kann der Fehler für jedes einzelne Neuron, bzw. die daraus resultierende Anpassung seiner Wichtungsfaktoren berechnet werden. Der

Index j verweist auf die Neuronen der zu berechnenden Schicht, i auf die vorangegangene, k auf die nachfolgende Schicht. Die Veränderung der Wichtungsfaktoren Δw_{ij} wird aus den gewichteten Fehlern δ_j, multipliziert mit den Ausgaben der vorangegangenen Schicht a_i und einer Lernrate η berechnet. Hier muss noch unterschieden werden, ob es sich bei j um die Ausgabeschicht oder eine verdeckte Schicht handelt. Im Falle der Ausgabeschicht kann δ_j als Differenz zwischen Netzausgabe und Zielwert berechnet werden. Anderenfalls berechnet sich das δ_j aus den gewichteten Fehlern der nachfolgenden Neuronen. n_j ist die sogenannte Netzeingabe, die gewichteten und aufsummierten Ausgaben der vorangegangenen Neuronen. Eine ausführliche Herleitung der Backpropagation-Regel ist bei Zell zu finden (8).

Probleme und Variationen

Ein grundlegendes Problem von Gradientenverfahren ist, dass sie Gefahr laufen, in einem lokalen Minimum hängen zu bleiben und das globale Minimum nicht finden. Dafür gibt es keine allgemeingültige Lösung. Es hängt stark von der Anwendung und ihrer Komplexität ab, inwieweit dies zum Problem wird oder ausreichend gute Lösungen gefunden werden.

Weitere Probleme sind Plateaus, auf denen das neuronale Netz auf Grund eines Gradienten von 0 stagniert, oder schmale Schluchten, aus denen es wieder heraus springt. Solche Probleme können durch die Einführung eines Momentum-Terms oder durch Anpassung der Lernrate behoben werden.

Trainingsfunktionen unter MATLAB

Die Neuronale Netze Toolbox unter MATLAB bietet für das Training von Backpropagation-Netzen die Funktion *trainlm*. Dabei handelt es sich um den Levenberg-Marquardt-Algorithmus (16)(17)(18), der sich als schneller Algorithmus für das Training von Netzen mittlerer Größe bewährt hat (19). Eine weitere davon abgeleitete Variation ist die Trainingsfunktion *trainbr*. Sie verbessert das Training durch sogenannte *bayesian regularization* (20)(21). Hierbei wird versucht die Wichtungsfaktoren klein zu halten, da kleine

Wichtungsfaktoren zu einem gleichmäßigeren Verhalten des Netzes führen. Extreme Reaktionen und die Gefahr des Overfittings werden somit reduziert (12).

2.4 Training und Validation

Werden neuronale Netze einem überwachten Training unterzogen, bekommen sie eine Eingabe (Input) zusammen mit der Zielvorgabe (Target), wie der Input zu bewerten ist. Hierbei stellt sich das Problem, dass die hoch variablen Netze in der Lage sind große Datensätze auswendig zu lernen. Der Datensatz, an dem gelernt wurde, kann dann einwandfrei wiedergegeben werden. Allerdings ist das Netz dann nicht mehr in der Lage zu Verallgemeinern und neue Daten korrekt zu bewerten. Um dieses Phänomen, das sogenannte *Overfitting*, einzuschränken, gibt es in Matlab spezielle Funktionen für das Training der neuronalen Netze. Darüber hinaus gibt es verschiedene Validationsmethoden. Im Einzelnen handelte es sich hier um die Crossvalidation, die Validation mit einem Validationsdatensatz und die Validation mit einem Validations- und einem Testdatensatz (22)(23)(24).

2.4.1 Training mit Crossvalidation

Die Crossvalidation kommt zum Einsatz, wenn nur ein kleiner Datensatz zur Verfügung steht. Da in so einem Falle alle Daten für die Kalibration benötigt werden, muss sowohl die Kalibration als auch die Validation an ein und demselben Datensatz vorgenommen werden. Um dennoch eine Validierung zu ermöglichen, wird eine Vielzahl von Kalibrationen durchgeführt, bei denen systematisch jedes Element des Datensatzes (*Sample*) ein Mal ausgelassen wird. Dieses Sample, welches jeweils nicht in die Kalibration mit einfließt, wird dann zur Validierung vorhergesagt. Da dieses Sample nicht Teil der Kalibration war, ist dies eine unabhängige Vorhersage eines neuen Samples. Aufgrund der Vielzahl erforderlicher Kalibrationen ist dieses Verfahren sehr zeitaufwendig.

2.4.2 Training mit einem Validationsdatensatz

Stehen mehr Daten zur Verfügung, kann das Datenmaterial in zwei Datensätze unterteilt werden. Der erste Datensatz dient zur Kalibration. Mit dem zweiten

Datensatz kann zur Validation eine unabhängige Vorhersage gemacht werden, um die Güte des trainierten Netzes zu beurteilen. Zu Beginn des Trainings fallen die Ergebnisse in der Validation in der Regel schlecht aus, da das System noch nicht ausreichend trainiert wurde. Im Laufe des Trainings sollten sich die Ergebnisse bessern. Bei längerem Training wiederum werden sich die Ergebnisse in der Validation wieder verschlechtern. Das neuronale Netz wird zunehmend trainiert, lernt aber nur noch auswendig. Die Fähigkeit zu verallgemeinern und neue Muster korrekt zu bewerten geht dabei verloren. Es kommt zum *Overfitting*. Das Training wird daher abgebrochen, wenn ein Fehlerminimum in der Validation erreicht ist.

2.4.3 Training mit Validations- und Testdatensatz

Bei der Arbeit mit einem Validationsdatensatz erfolgt das Training mit dem Kalibrationsdatensatz, die Bewertung und der Abbruch des Trainings anhand des Validationsdatensatzes. In gewisser Weise erfolgt das Training also mit der gesamten Datenbasis. Ist die Datenbasis ausreichend groß, so kann auch ein Training mit Kalibrations- Validations- und Testdatensatz durchgeführt werden. Das Abbruchkriterium für das Training ist nach wie vor das Fehlerminimum in der Validation, die Beurteilung des Netzes erfolgt allerdings davon unabhängig anhand der Vorhersage des Testdatensatzes.

2.5 Der NIPALS-Algorithmus

Der NIPALS-Algorithmus wurde von H. Wold (25)(26) für die Durchführung von Hauptkomponentenanalysen (PCA) und Partial Least Square Regressionen (PLS) entwickelt und ist das meistgenutzte Werkzeug zur Berechnung der Hauptkomponenten eines Datensatzes. Er ist langsamer als die Singulärwertzerlegung (SVD), liefert aber die numerisch genaueren Ergebnisse als die SVD.

Die Hauptkomponenten werden sukzessive berechnet, wobei jede neue Hauptkomponente ein Maximum der verbliebenen Varianz im Datensatz beschreibt und orthogonal zu den bereits berechneten Hauptkomponenten ist.

Zunächst wird die Datenmatrix $\boldsymbol{X}$ zentriert. Die Zerlegung von $\boldsymbol{X}$ in einen Loadings-Vektor $\boldsymbol{v}$ und einen Scores-Vektor $\boldsymbol{u}$ erfolgt in folgenden iterativen Schritten:

Schritt 1

Aus der Matrix $\boldsymbol{X}$ wird ein Spaltenvektor $\boldsymbol{x_i}$ ausgewählt und in $\boldsymbol{u}$ kopiert.

$$\boldsymbol{u} = \boldsymbol{x_i} \tag{7}$$

Üblicherweise wird hier der Vektor mit dem größten Betrag gewählt.

Schritt 2

Um die korrespondierenden Loadings $\boldsymbol{v}$ zu berechnen, wird die Matrix $\boldsymbol{X}$ auf $\boldsymbol{u}$ projeziert.

$$\boldsymbol{v} = \frac{\boldsymbol{X}^T\boldsymbol{u}}{\left|\boldsymbol{X}^T\boldsymbol{u}\right|} \tag{8}$$

Der Vektor $\boldsymbol{v}$ ist normiert und hat einen Betrag von 1.

Schritt 3

Durch Projektion der Datenmatrix $\boldsymbol{X}$ auf die Loadings $\boldsymbol{v}$ erhält man die neuen Scores $\boldsymbol{u_{neu}}$.

$$\boldsymbol{u_{neu}} = \boldsymbol{X}\boldsymbol{v} \tag{9}$$

Schritt 4

Berechnung der Differenz d zwischen den alten und den neuen Scores.

$$d = \left|\boldsymbol{u_{neu}} - \boldsymbol{u}\right| \tag{10}$$

Unterschreitet d eine definierte Grenze (z. B. 10^{-5}), konvergiert der Algorithmus und die neue Hauptkomponente ist gefunden. Ist dies nicht der Fall, werden die Schritte 1 bis 4 mit $\boldsymbol{u} = \boldsymbol{u_{neu}}$ wiederholt.

Schritt 5

Sofern weitere Hauptkomponenten berechnet werden sollen, zieht man die berechnete Hauptkomponente $\boldsymbol{uv}^T$ von der Datenmatrix $\boldsymbol{X}$ ab und beginnt erneut mit Schritt 1.

$$\boldsymbol{X} = \boldsymbol{X} - \boldsymbol{uv}^T \tag{11}$$

Weitere ausführliche Informationen zum Einsatz des NIPALS-Algorithmus in der multivariaten Datenanalyse sind in der Literatur (23) zu finden.

2.6 Optimierungsverfahren

In dieser Arbeit wird ein Mustererkennungsprozess unter Einsatz verschiedener Filter und unterschiedlicher neuronaler Netze aufgebaut. Sowohl die Arbeitsweise der Filter, als auch die der neuronalen Netze beinhaltet variable Parameter. Die erfolgreiche Arbeit ist also, neben dem Aufbau eines geeigneten Systems, auch von der Wahl optimaler Parameter abhängig. Um diese zu finden, kann eine Parameteroptimierung durchgeführt werden.

Bei Optimierungsverfahren wird zwischen globalen und lokalen Optimierungsverfahren unterschieden (22). Lokale Optimierungsverfahren sind meist Gradientenverfahren, welche nur das nächste Optimum finden. Das globale Optimum hingegen wird oft nur bei der Wahl geeigneter Startwerte gefunden.. Beispiele hierfür sind die *Respond-Surface-Methode* (27) oder die *Box-Wilson-Optimierung* (28). Der *genetische Algorithmus* (GA) hingegen ist ein Beispiel für globale Optimierungsverfahren (29).

2.6.1 Der Simplex-Algorithmus

Bei der Simplex-Optimierung (30)(31) handelt es sich um ein lokales Optimierungsverfahren, welches sich, ausgehend von gegebenen Startparametern, durch systematische Generierung neuer Parameterkombinationen auf ein Optimum zu bewegt. Es besteht hierbei die Gefahr, dass der Simplex in ein lokales Minimum in der Nähe der Startparameter läuft. Das Ergebnis des Simplex ist also

stark von den Startparametern abhängig. Neben dem Simplex-Algorithmus mit fester Schrittweite gibt es auch einen Simplex-Algorithmus mit variabler Schrittweite, wie er von J. A. Nelder und R. Mead beschrieben wird (30). Dieser kann durch Veränderung der Schrittweite sowohl weitläufige Bereiche in großen Schritten überbrücken, als sich auch im Zielbereich mit zunehmend kleineren Schritten dem Optimum annähern.

Beim so genannten Simplex handelt es sich um eine geometrische Figur, die für m Parameter $m + 1$ Eckpunkte (Parameterkombinationen) besitzt. Diese Methode ist nach Belieben auf m Dimensionen anwendbar, soll hier aber an einem anschaulichen Beispiel mit zwei Parametern erläutert werden. Der Simplex ist in diesem Fall ein Dreieck. Die Qualität jedes Punktes wird ermittelt, wobei der beste Punkt mit $\boldsymbol{b}$, der nächstbeste Punkt mit $\boldsymbol{n}$ und der schlechteste Punkt mit $\boldsymbol{s}$ bezeichnet wird. Der schlechteste Punkt $\boldsymbol{s}$ wird nach Gleichung 12 am Schwerpunkt $\boldsymbol{p}$ der anderen Punkte (Zentroiden) gespiegelt, und das Resultat $\boldsymbol{r}$ wird neu bewertet. Der Schwerpunkt berechnet sich nach Gleichung 13, wobei $\boldsymbol{v}$ die einzelnen Punkte darstellt, $\boldsymbol{i}$ den Index des zu spiegelnden Punktes und $\boldsymbol{j}$ die Indizes aller anderen Punkte.

$$\boldsymbol{r} = \boldsymbol{p} + (\boldsymbol{p} - \boldsymbol{s}) \tag{12}$$

$$\boldsymbol{p} = \frac{1}{m} \sum_{j \neq i}^{m} \boldsymbol{v}_j \tag{13}$$

Die Schrittweite des Simplex mit fester Schrittweite ist durch die Spiegelung festgelegt. Bei der erweiterten Version des Algorithmus mit variabler Schrittweite werden nach der Bewertung des neuen Punktes vier Fälle unterschieden. Diese sind in Abbildung 11 für die Optimierung von zwei Parametern dargestellt (24).

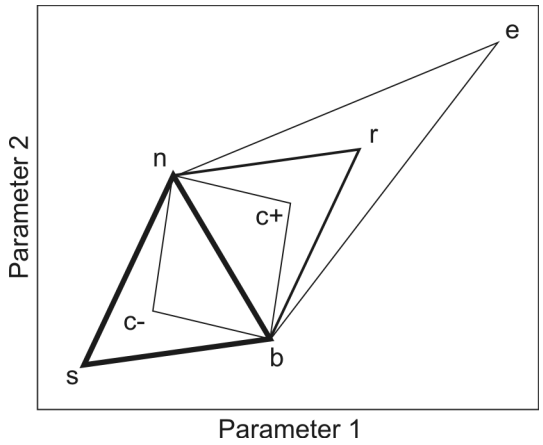


Abbildung 11: Möglichkeiten beim Simplex mit variabler Schrittweite

Fall 1:

Der gespiegelte Punkt $\boldsymbol{r}$ ist besser als die Punkte $\boldsymbol{b}$ und $\boldsymbol{n}$. In diesem Fall wird dies als ein Schritt in die richtige Richtung interpretiert, und der Simplex wird zum Punkt $\boldsymbol{e}$ expandiert. Der neue Simplex ist somit $\boldsymbol{bne}$.

$$\boldsymbol{e} = \boldsymbol{p} + \alpha(\boldsymbol{p} - \boldsymbol{s}) \quad \text{mit} \quad \alpha > 1{,}5 \tag{14}$$

Fall 2:

Der neue Punkt $\boldsymbol{r}$ ist besser als Punkt $\boldsymbol{n}$, aber schlechter als Punkt $\boldsymbol{b}$. In diesem Fall wird der Simplex $\boldsymbol{bnr}$ beibehalten.

Fall 3:

Der neue Punkt $\boldsymbol{r}$ ist schlechter als Punkt $\boldsymbol{n}$ aber besser als Punkt $\boldsymbol{s}$. In diesem Fall erfolgt eine positive Kontraktion zum Punkt $\boldsymbol{c}_+$ und der neue Simplex lautet $\boldsymbol{bnc}_+$.

$$\boldsymbol{c}_+ = \boldsymbol{p} + \beta(\boldsymbol{p} - \boldsymbol{s}) \quad \text{mit} \quad 0 < \beta < 0{,}5 \tag{15}$$

Fall 4:

Der neue Punkt $\boldsymbol{r}$ ist schlechter als die drei ursprünglichen Punkte. In diesem Fall erfolgt eine negative Kontraktion zum Punkt $\boldsymbol{c}_-$, es resultiert der Simplex $\boldsymbol{bnc}_-$.

$$\boldsymbol{c}_- = \boldsymbol{p} - \gamma(\boldsymbol{p} - \boldsymbol{s}) \quad \text{mit} \quad 0 < \gamma < 0{,}5 \tag{16}$$

Die Abbildung 12 veranschaulicht, wie sich der Simplex dem Optimum nähert. Ausgehend von den übergebenen Startparametern wird der erste Simplex, die Punkte 1 - 3, erzeugt. Der schlechteste Punkt wird gespiegelt und ergibt Punkt 4.

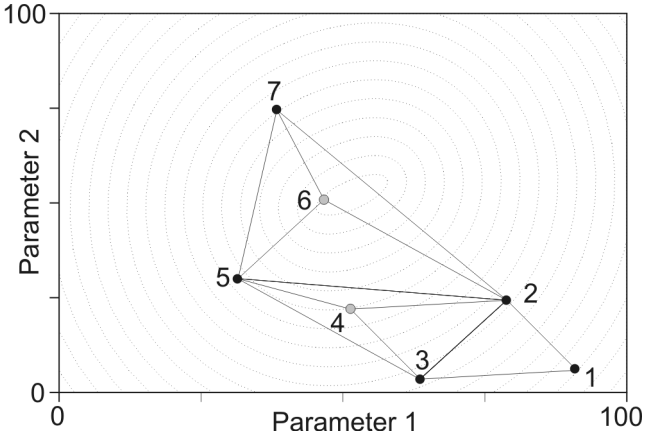


Abbildung 12: Die ersten sieben Schritte einer Simplex-Optimierung mit variabler Schrittweite bei der Optimierung von zwei Parametern

Da dieser Punkt besser ist, als die bisher vorhandenen, wird der Simplex auf Punkt 5 expandiert. Beim nun vorliegenden Simplex wird Punkt 3 als schlechtester Punkt identifiziert und in Folge dessen auf Punkt 6 gespiegelt. Dieser wiederum, besser als alle anderen, wird zum Punkt 7 expandiert. So umkreist der Simplex das Optimum und konvergiert letztlich bis die Abbruchkriterien erreicht sind. Dies ist der Fall, wenn der Durchmesser des Simplex eine gewisse Grenze *TolX* unterschreitet, die Verbesserung der Antwortfunktion eine gewisse Grenze *TolF* unterschreitet oder eine maximale Anzahl von Iterationsschritten *MaxIter* erreicht wurde.

3 Material und Methoden

3.1 Das Bildmaterial

Der Arbeit liegen Bilder zugrunde, die mit einer herkömmlichen Digitalkamera (Sony Cybershot) gemacht wurden. Sie haben eine Größe von 5 Megapixeln (2560x1920 Pixel), eine Farbtiefe von 24 Bit und liegen als komprimierte JPG-Dateien vor. Ein Bild zeigt jeweils einen ca. 1 m x 1,4 m großen Straßenausschnitt. Für eine normierte senkrechte Bildaufnahme wurde ein Handwagen mit fest installierter Kamera konstruiert. Die aufgenommenen Bilder stammen von einem 250 Meter langen, asphaltierten Straßenabschnitt. Unter den 1216 zur Verfügung stehenden Bildern wurde zunächst eine Vorauswahl getroffen. Um die Untersuchung auf das Problem der Risserkennung, bzw. Erkennung von Schadensmerkmalen, zu konzentrieren, wurden nur die Bilder herangezogen, welche keine weiteren störenden Elemente wie Schattenwurf oder Fahrbahnmarkierungen aufwiesen. Somit standen 924 Bilder für die Etablierung eines Mustererkennungsprozesses zur Erfassung von Schadensmerkmalen zur Verfügung. Das Bildmaterial, sowie eine fachkundige Beurteilung der Bilder wurden von der Firma IWS Messtechnik (Hambühren) und der Villaret Ingenieurgesellschaft mbH (Hönow) bereitgestellt.

3.2 Soft- und Hardware

Die Entwicklung des Mustererkennungsprozesses erfolgte in Matlab 6.5 Release 13 (The MathWorks Inc) mit der Neuronale Netze Toolbox 4.0.1. In einem folgenden Arbeitsschritt wurde das System aufgrund der Geschwindigkeitsvorteile kompilierter Programmiersprachen in Delphi 7 (Borland) und Turbo Delphi (CodeGear) umgesetzt.

Betrieben wurden die Programme auf einem PC mit Intel Pentium 4 Prozessor mit 3,2 Gigahertz und einer Precision WorkStation 530 MT von Dell mit zwei Inter Xeon 1.8 Prozessoren mit je 1,8 Gigahertz.

3.3 Die Mustererkennung

3.3.1 Bildausschnitt

Die aufgenommenen Bilder zeigen neben einem Ausschnitt der Asphaltoberfläche auch Teile des Messwagens. Daher wird zunächst ein geeigneter Bildausschnitt gewählt, auf dem der Rahmen des Messwagens nicht zu sehen ist. Dieser Bildausschnitt umfasst 2188x1717 (3,8 Mio.) Pixel. Ein entsprechender Bildausschnitt ist in Abbildung 13 veranschaulicht.

Abbildung 13: Rohbild und Auswahl eines geeigneten Bildausschnittes

3.3.2 Reduktion der Farbtiefe

Des Weiteren ist die Bedeutung der drei Farbkanäle vernachlässigbar klein. Dies ist in Abbildung 14, in der ein Beispielbild mit seinen drei Farbkanälen dargestellt wird, bereits mit bloßem Auge zu erkennen.

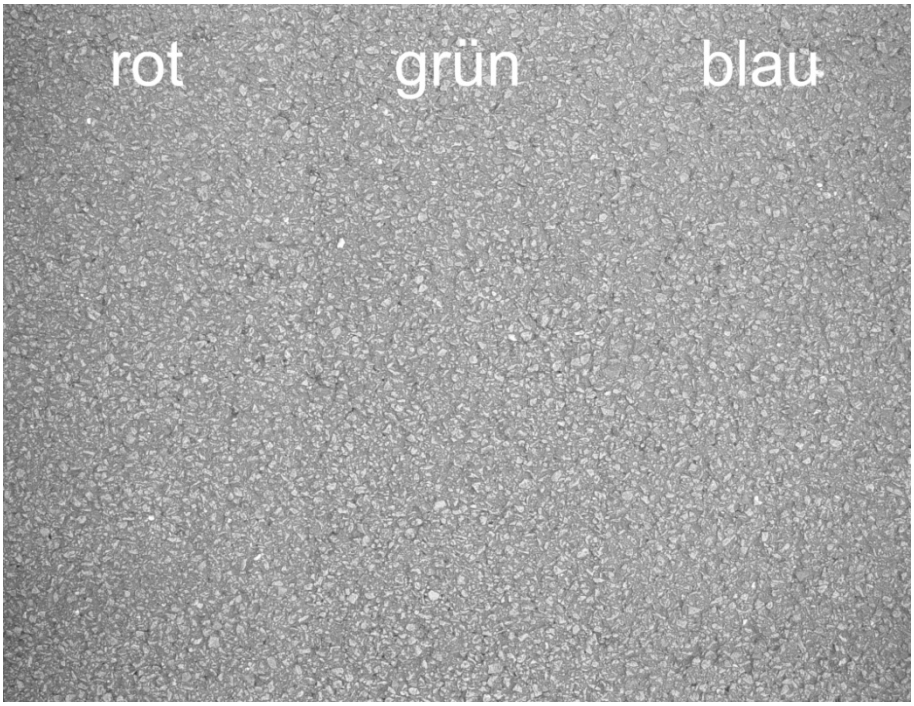


Abbildung 14: Vergleich der drei Farbkanäle des original Bildmaterials

Das linke Drittel des Bildes zeigt den roten Farbkanal, vom mittleren Drittel des Bildes ist der grüne Farbkanal und vom rechten Drittel des Bildes der blaue Farbkanal dargestellt. Mit bloßem Auge ist kaum ein Unterschied zu erkennen. Um dies genauer zu erfassen kann zunächst für jeden einzelnen Pixel P nach Gleichung 17 ein mittlerer Farbwert $\bar{P}$ berechnet werden.

$$\bar{P}_{x,y} = \frac{P_{x,y,r} + P_{x,y,g} + P_{x,y,b}}{3} \tag{17}$$

Die Indizes r, g und b bezeichnen die drei Farbkanäle, x und y die jeweilige Position des Pixels. Nach Gleichung 18 wird die Standardabweichung über die drei Farbkanäle berechnet.

$$StdAbw_{x,y} = \frac{\left(P_{x,y,r} - \overline{P}_{x,y}\right)^2 + \left(P_{x,y,g} - \overline{P}_{x,y}\right)^2 + \left(P_{x,y,b} - \overline{P}_{x,y}\right)^2}{3} \tag{18}$$

Nach Gleichung 19 kann letztlich die Standardabweichung über die drei Farbkanäle für das gesamte Bild berechnet werden. Hierbei ist n die Anzahl Pixel, die sich aus der Höhe und der Breite des Bildes ergibt.

$$StdAbw_{Bild} = \frac{\sum StdAbw_{x,y}}{n} \tag{19}$$

3.3.3 Helligkeitskorrektur

Bedingt durch einen zentralen Blitz bei der Kamera ist die Ausleuchtung der Bilder sehr ungleichmäßig. Abbildung 15 zeigt deutlich die parabolische Abnahme der Helligkeit zum Rand des Bildes hin. Um diese Grafik zu erhalten, wird das Bild in Zeilen und Spalten unterteilt. Über die entstehenden Segmente wird dann jeweils der mittlere Grauwert berechnet.

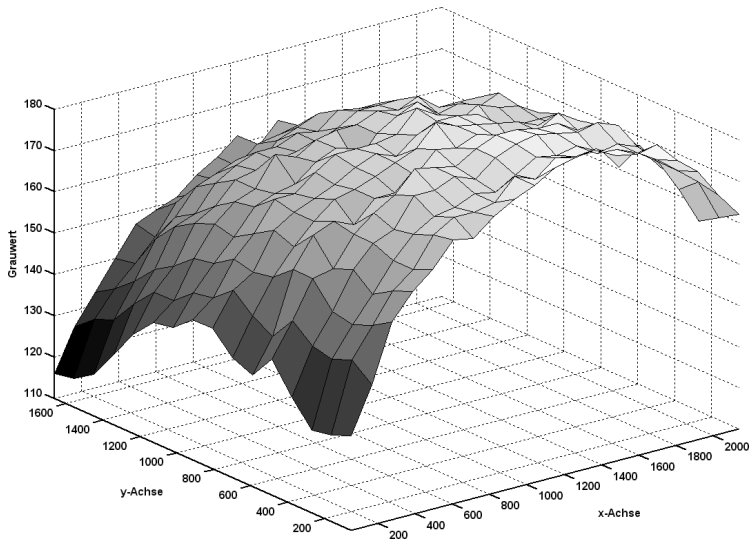


Abbildung 15: Parabolische Ausleuchtung der Bilder vor der Korrektur

Um diese ungleichmäßige Ausleuchtung zu kompensieren, wird eine zweidimensionale quadratische Korrekturfunktion (Gleichung 20) an die Werte aus Abbildung 15 angepasst.

$$G(x,y) = m - f_x\left(x - d_x\right)^2 - f_y\left(y - d_y\right)^2 \tag{20}$$

G ist der zu berechnende Grauwert für eine bestimmte Position (x, y). Die Parameter d_x und d_y verschieben das Maximum der Parabel, f_x und f_y bestimmen die Ausprägung der Parabel in x- und y-Richtung und m ermöglicht einen Offset. Die Anpassung der Funktion aus Gleichung 20 erfolgt mit dem Simplex-Algorithmus.

Wird die so ermittelte Korrekturfunktion vom Bild abgezogen und ein Offset (der gewünschte Mittelwert für das Bild) addiert, so erhält man Bilder mit ausgewogener Ausleuchtung und einem definierten mittleren Grauwert.

3.3.4 Adaptiver Grenzwertfilter

Im nächsten Schritt wird das Bild in ein Schwarzweißbild konvertiert. Zu diesem Zweck muss eine Grenze definiert werden. Pixel, welche dunkler als die angegebene Grenze sind, werden durch schwarz Pixel ersetzt, hellere durch weiße Pixel. Um diese Konvertierung flexibel zu halten, wird die Grenze bildabhängig definiert. Zu diesem Zweck wird die Standardabweichung in der Verteilung der Grauwerte berechnet.

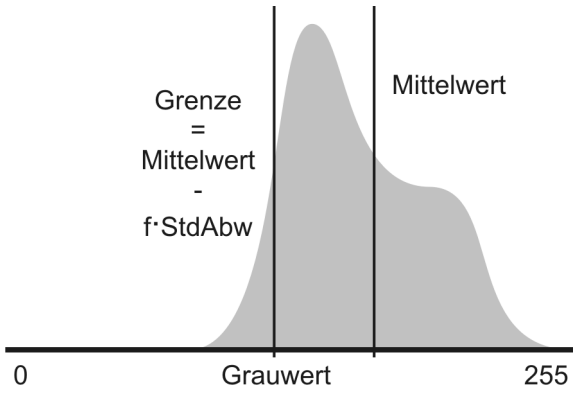


Abbildung 16: Bildabhängige Bestimmung des Grenzwertes für den adaptiven Grenzwertfilter

Die Grenze ergibt sich schließlich als Mittelwert, abzüglich einem Vielfachen f der Standardabweichung. Dies ist für ein Beispiel an einem Histogramm in Abbildung 16 dargestellt. Diese Art der Grenzwertfilterung erlaubt eine flexible Verarbeitung von Bildern unterschiedlicher Helligkeit und mit unterschiedlichem Kontrast.

3.3.5 Objektfilter

Als Ergebnis des Grenzwertfilters erhält man ein Schwarzweißbild. Dieses zeigt eine Vielzahl schwarzer Pixel oder Pixelhaufen, welche als Objekte betrachtet werden. Nicht all diese Objekte sind für die weitere Betrachtung relevant. Insbesondere wird es eine große Menge besonders kleiner Objekte geben, die mit großer Wahrscheinlichkeit lediglich nicht relevante Unregelmäßigkeiten in der Asphaltoberfläche darstellen. Um die weiteren Untersuchungen auf größere Objekte zu beschränken, müssen die Objekte zunächst als solche gefunden und ihre Größe bestimmt werden. Objekte, die eine Mindestgröße unterschreiten, werden aussortiert, während größere Objekte später charakterisiert werden können.

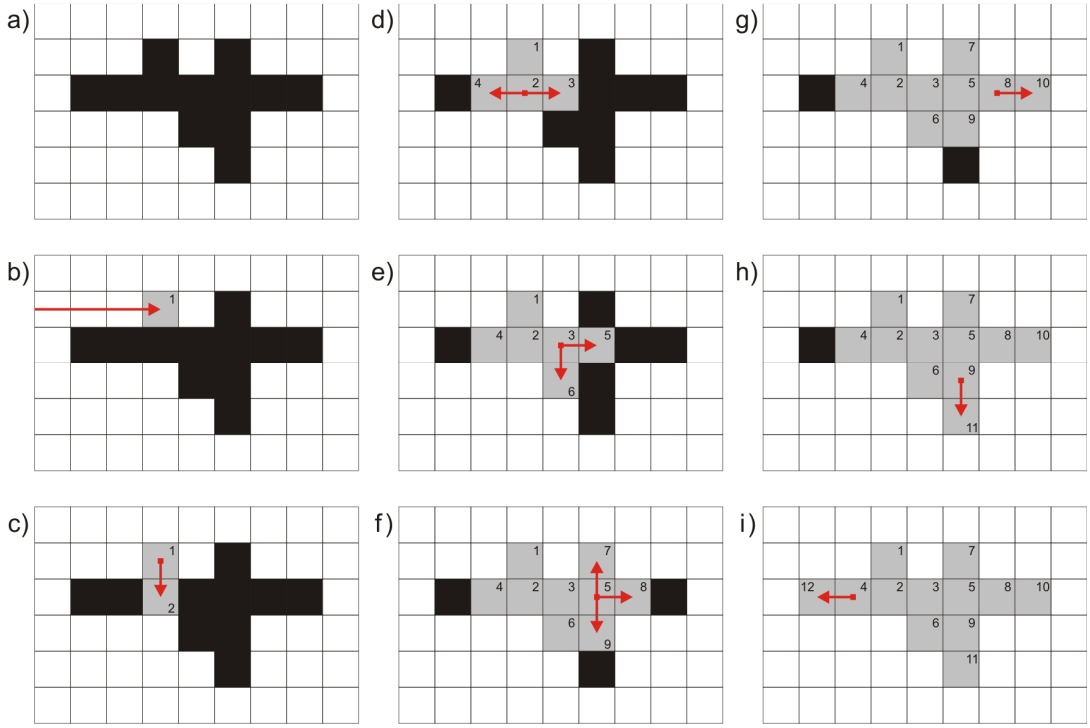


Abbildung 17: Rekursiver Algorithmus zur Erfassung aller Pixel eines Objektes

Um ein Objekt mit all seinen Pixeln zu erfassen, kommt in Matlab ein rekursiver Algorithmus zum Einsatz. Abbildung 17 zeigt die Vorgehensweise dieses Algorithmus. Zunächst wird das Bild von oben nach unten und von links nach rechts nach einem schwarzen Pixel abgesucht. Ist dieser gefunden, dient er als Ausgangspunkt für den rekursiven Algorithmus. Jeder gefundene Pixel wird grau markiert und kann somit kein zweites Mal gefunden werden. Außerdem wird jeder gefundene Pixel in einer Liste gespeichert. Nun kommt die rekursive Funktion zum Einsatz, die von einem gegebenen Pixel aus die Umgebung nach weiteren schwarzen Pixeln absucht. Werden solche gefunden, werden sie grau markiert und die Funktion ruft sich für jeden gefundenen Pixel selber auf. Ausgehend von Pixel 1 (b) wird demnach Pixel 2 gefunden (c). Von Pixel 2 aus (d) werden die Pixel 3 und 4 gefunden. An dieser Stelle verzweigt sich die Suche. Zunächst wird die Suchfunktion für den Pixel 3 aufgerufen und die rekursive Suche verzweigt sich weiter auf der rechten Seite (e, f) bis sie bei den Pixeln 7 (f), 10 (g) und 11 (h) die einzelnen Zweige enden, da die jeweiligen Pixel keine weiteren schwarzen Nachbarn haben. Letztlich kehrt die Funktion zur ersten Verzweigung zurück (i), von wo aus der letzte Pixel 12 gefunden wird. Somit ist das gesamte Objekt mit all seinen Pixeln erfasst.

Unter normalen Bedingungen arbeitet die rekursive Objektfindung zuverlässig und zeichnet sich in der Programmierung dadurch aus, dass sie elegant, kompakt und übersichtlich ist. Allerdings steigt mit der Rekursionstiefe der Speicheraufwand, da für jeden gefundenen Pixel eine Instanz der Suchfunktion auf dem Stack liegt. Treten unter extremen Bedingungen (bei zu hoch gewählter Grenze im Grenzwertfilter) sehr große Objekte auf, kann dies im schlimmsten Fall zum Absturz des Programms führen. Um dies zu vermeiden, wurde in Delphi ebenso ein iterativer Ansatz implementiert. Die Markierung und Speicherung gefundener Pixel ist identisch. Auch die Suchfunktion arbeitet im wesentlichen wie bereits beschrieben, ruft sich allerdings nicht selber auf. Stattdessen wird eine temporäre Pixelliste geführt, in der die Pixel gespeichert werden, von denen aus noch gesucht werden muss. In einer WHILE-Schleife wird diese Liste fortlaufend erweitert und abgearbeitet bis sie leer ist.

Für das Beispiel aus Abbildung 17 würde dies bedeuten, dass zu Beginn Pixel 1 in der temporären Liste gespeichert wird (b). Im nächsten Schritt wird Pixel 1 abgearbeitet (aus der Liste wieder entfernt) und durch den von dort aus gefundenen Pixel 2 ersetzt (c). Im dritten Schritt (d) wird Pixel 2 aus der Liste entfernt, und im Gegenzug werden die Pixel 4 und 3 hinzugefügt. Pixel 3 wiederum wird durch Pixel 6 und 5 ersetzt (e). Hat ein Pixel keine weiteren Nachbarn (Pixel 7, 10 und 11), verkürzt sich die Liste wieder. Mit Erreichen von Pixel 12 (i) wird der letzte Pixel aus der temporären Liste entfernt und keine weiteren Pixel hinzugefügt. Die temporäre Liste ist leer und der Algorithmus endet. Der Speicheraufwand wird hierbei auf ein Minimum reduziert. Es wird lediglich Speicherplatz für die Koordinaten der „losen Enden“ in der temporären Pixelliste beansprucht.

Wurde ein Objekt vollständig erfasst, wird die systematische Suche im Bild nach weiteren Objekten vom Ausgangspunkt des zuletzt gefundenen Objektes aus fortgesetzt. Nach diesem Schritt wird das Bild nicht weiter benötigt. Alle folgenden Arbeitsschritte bauen auf die nun vorliegende Liste von Objekten mit den Koordinaten aller dazugehörigen Pixel auf.

3.3.6 Verknüpfung von Objekten

Bedingt durch die strukturierte Oberfläche des Asphaltes kann es sehr oft passieren, dass größere Risse durch einzelne Steine unterbrochen erscheinen und so nicht als Ganzes erfasst werden. Um solche Teilstücke wieder zu größeren und aussagekräftigeren Objekten zusammenzufügen, werden in einem nächsten Arbeitsschritt Objekte verknüpft.

Verknüpfung naheliegender Objekte

Das einfachste Kriterium für die Verknüpfung von Objekten ist ihre Nähe zueinander. Man nimmt an, dass die Unterbrechung der Risse durch die Struktur des Asphalts klein ist gegenüber dem Abstand zu zufällig verteilten Objekte oder auch zu anderen Rissen. Als Kriterium für die Verknüpfung von Objekten dient also lediglich ihre Entfernung zueinander. Abbildung 18a zeigt beispielhaft eine Auswahl von drei Objekten mit den Indizes eins bis drei, die es zu überprüfen gilt.

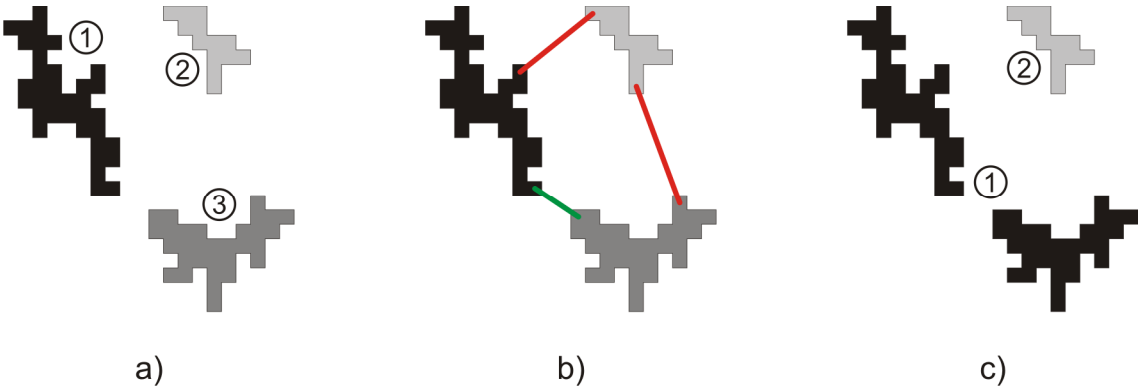


Abbildung 18: Drei Beispielobjekte (a), deren Entfernung überprüft wird (b), und die entsprechend verknüpft werden (c)

Jedes Objekt wird dabei pixelweise mit jedem anderen verglichen, um die kürzeste Entfernung zwischen den einzelnen Objekten zu ermitteln (Abbildung 18b). Wird ein definierter Mindestabstand unterschritten, werden zwei Objekte verknüpft. Dies geschieht, indem die Daten des Objektes mit dem höheren Index dem Objekt mit dem niedrigeren Index zugeordnet werden. Das Objekt mit dem höheren Index selbst wird daraufhin aus der Objektliste entfernt (Abbildung 18c).

Verknüpfung mittels Vektorisierung

Um zu verhindern, dass naheliegende Objekte verknüpft werden, die trotz ihrer Nähe im Grunde nichts miteinander zu tun haben, kann ein weiterer, differenzierterer Ansatz verfolgt werden. Zu diesem Zweck werden die Objekte, welche als „Pixelhaufen“ erfasst sind, vektorisiert, wie in Abbildung 19 gezeigt.

Die Objekte werden dann nicht mehr durch einzelne Pixel beschrieben, sondern durch Vektoren. Diese ermöglichen im späteren Verlauf die Bestimmung interessanter Kenngrößen:

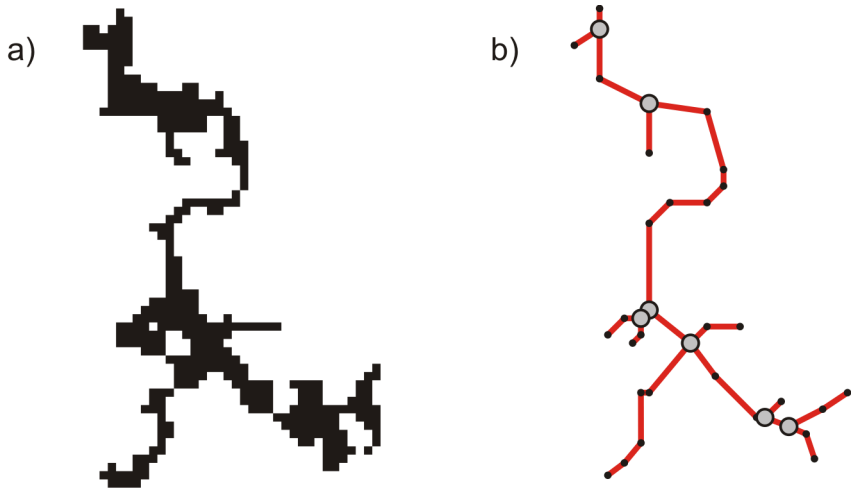


Abbildung 19: Vektorisierung eines aus einzelnen Pixeln bestehenden Objektes. a) Originalobjekt b) Vektorisiertes Objekt

Länge

Ohne Vektorisierung kann nur die Größe eines Objektes (Anzahl Pixel oder beanspruchte Rechtecksfläche) bestimmt werden und inwieweit das Objekt über eine bestimmte Ausrichtung verfügt. Durch die Vektorisierung kann konkret die Länge des Objektes, bzw. des potentiellen Rissabschnittes bestimmt werden. Ein U-Förmiger Rissabschnitt, der anderenfalls nur als kleines kompaktes Objekt in Erscheinung tritt, bekommt durch die Vektorisierung eine aussagekräftige Länge zugewiesen.

Verzweigung

In der vektorisierten Form des Risses sind Verzweigungen ersichtlich. Auch diese können für die Charakterisierung des Risses als Kenngröße herangezogen werden.

Ausrichtung

Im ersten Ansatz werden Objekte lediglich aufgrund ihrer räumlichen Nähe verbunden. Durch die Vektorisierung kann genauer bestimmt werden, wohin ein Objekt, bzw. einzelne Ausläufer des Objektes tatsächlich zeigen. Ausgehend von den Endpunkten eines Objektes kann dann, wie in Abbildung 20 veranschaulicht, zielgerichteter nach möglichen Verlängerungen gesucht werden.

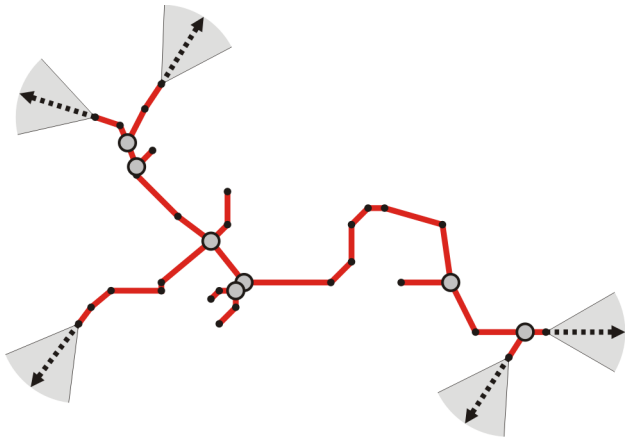


Abbildung 20: Gezielte Suche nach möglichen Rissfortsätzen mit einer gewissen Toleranz

3.3.7 Dilatation

Wie im vorangegangenen Kapitel angesprochen, hat die Oberflächenstruktur des Asphalts zur Folge, dass Risse nach der Anwendung der Filter unterbrochen erscheinen und in kleinere Objekte unterteilt werden, was die Auffindung größerer Strukturen erschwert. Statt der nachträglichen Verknüpfung von Objekten kann auch versucht werden, bereits vor der Grenzwertfilterung mögliche kleinere Unterbrechungen zu überbrücken.

Die Dilatation ist ein Verfahren, welches im Allgemeinen auf binäre Bilder angewandt wird und die Aufgabe hat, Regionen aus Vordergrundpixel (Pixel, die für den Betrachter von Interesse sind) schrittweise zu erweitern(32)(33). Abbildung 21 zeigt beispielhaft die Dilatation an einem Objekt.

Für jeden Punkt im Bild wird die 3x3-Nachbarschaft betrachtet und die Anzahl an Vordergrundpixeln (hier schwarz) gezählt. Dies kann im Prinzip als Konvolution betrachtet werden. Der 3x3 Konvolutionskern ist hierbei vollständig mit 1 bestückt. Wenn Hintergrundpixel mit 0 und Vordergrundpixel mit 1 bezeichnet werden, resultiert daraus die Zählung der Vordergrundpixel in einer 3x3 Nachbarschaft.

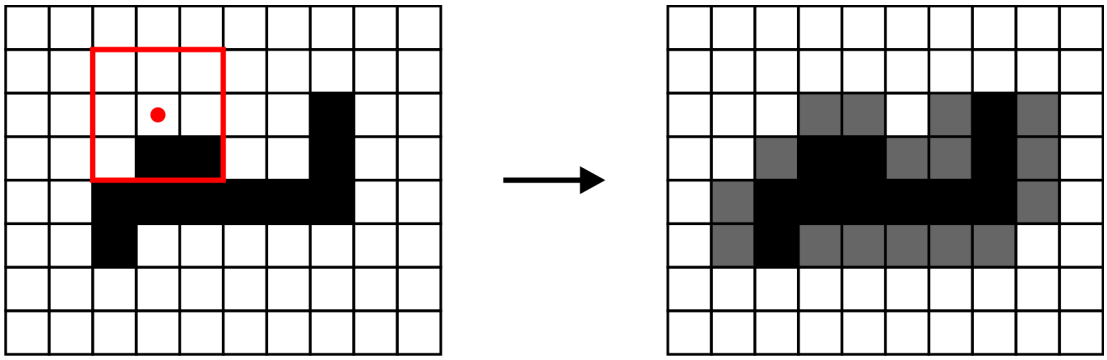


Abbildung 21: Dilatation mit einem quadratischen 3x3 Kern

Entsprechend der Anzahl schwarzer Pixel in seiner Nachbarschaft wird ein Hintergrundpixel weiß belassen, oder er wird zu einem schwarzen Pixel. In diesem Beispiel wurden alle weißen Pixel mit mindestens zwei schwarzen Nachbarn zu schwarzen Pixeln.

Abbildung 22 zeigt wie diese Methode bereits auf das Graustufenbild angewendet werden kann, um zwei benachbarte Objekte zu verknüpfen.

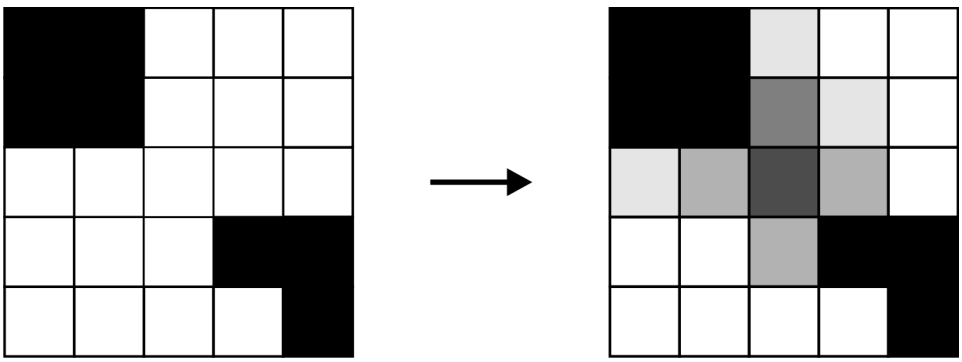


Abbildung 22: Überbrückung von Lücken durch eine Dilatation am Graustufenbild

Befindet sich ein weißer Pixel in unmittelbarer Nachbarschaft mehrerer schwarzer Pixel sollte dieser Pixel (dunkelgrau) ebenfalls erfasst werden. Dies kann ggf. auch für weitere Pixel (hellgrau) gelten.

Allerdings kann in einem Graustufenbild nicht eindeutig zwischen den Hintergrundpixeln (weiß) und Vordergrundpixeln (schwarz) unterschieden werden, da es noch 254 weitere Grauwerte gibt. Daher wurde ein modifizierter Ansatz verfolgt.

Zunächst wird ein Konvolutionsfilter angewendet. Der Konvolutionskern ***K*** ist in Abbildung 23 dargestellt.

r=2

1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1

Abbildung 23: Filtermatrix bzw. Konvolutionskern ***K*** mit einem Radius von *r*=2

Diese Matrix wird auf jeden Pixel *p* des Bildes angewandt. Der neue Wert für einen Pixel wird berechnet, indem jeder Pixel gemäß Gleichung 21 mit dem entsprechenden Matrix-Wert multipliziert wird und die Produkte addiert werden.

$$p_{x,y} = \sum_{i=-r}^{r} \sum_{j=-r}^{r} p_{x+i,y+j} k_{i,j} \tag{21}$$

Wird anschließend durch die Summe der Matrixwerte geteilt, entspricht dies einem Mittelwertfilter. Dies ist hier allerdings nicht erforderlich. Wie in Abbildung 24 schematisch angedeutet, wird die Matrix ***P*** (die vom Filter berechneten Werte) gemäß Gleichung 22 auf einen Wertebereich von null bis eins normiert.

$$\boldsymbol{P}_{neu} = \frac{\boldsymbol{P}_{alt} - p_{min}}{p_{max} - p_{min}} \tag{22}$$

Die Matrix, die man somit erhält, ist eine normierte Mittelwertmatrix, die mit Werten nahe null dunkle Bereiche und mit Werten nahe eins helle Bereiche

kennzeichnet. Diese wird, wie in Abbildung 25 gezeigt, mit dem Originalbild multipliziert, um das neue gefilterte Bild zu erhalten.

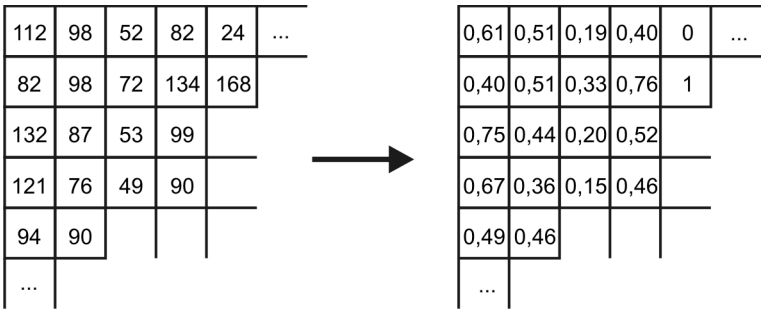


Abbildung 24: Ergebnis des Konvolutionsfilters mit anschließender Normierung auf einen Wertebereich von 0 bis 1

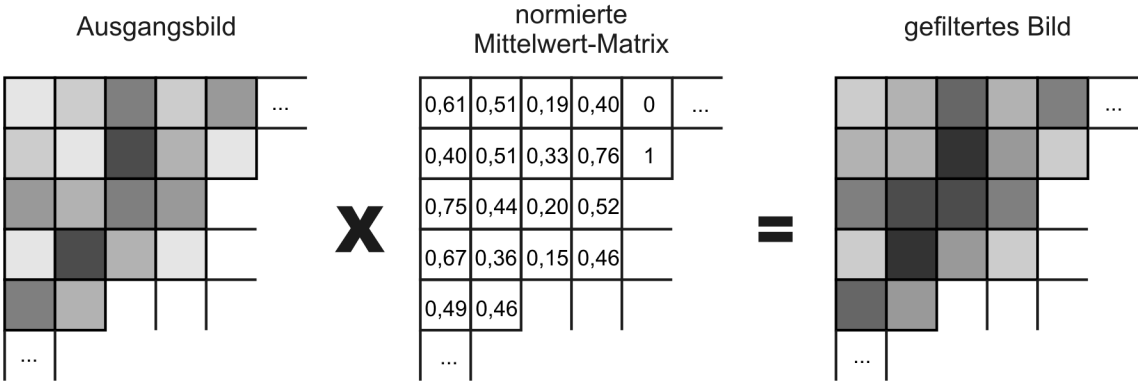


Abbildung 25: Verrechnen des Originalbildes mit der normierten Mittelwertmatrix

In diesem gefilterten Bild erscheinen Risse, die bereits vorher zu erkennen waren, noch dunkler. Aber auch Bereiche in Rissnähe, und insbesondere Bereiche zwischen zwei Rissen, werden merklich abgedunkelt. Wie bei einer klassischen Dilatation an einem Schwarzweißbild werden die Risse erweitert. Damit können sie bereits vor dem Grenzwertfilter „zusammenwachsen“ und ergeben nach der Filterung größere, aussagekräftigere Strukturen.

3.3.8 Bestimmung der Kenngrößen

Das Originalbild mit einem Informationsvolumen von 15 MB (5 Mio. Pixel, drei Farbkanäle) wird durch die bisherigen Arbeitsschritte auf ca. 1 MB reduziert. Dennoch ist auch diese Datenmenge noch zu groß und zu unspezifisch, um von einem neuronalen Netz ausgewertet zu werden. Daher werden aus den identifizierten Objekten zunächst Kenngrößen berechnet, die als Eingabe für das neuronale Netz dienen. Zu diesem Zweck wird das Bild, wie in Abbildung 26 gezeigt, in 30 Bildsegmente unterteilt (die Anzahl an Segmenten ist flexibel und kann angepasst werden).

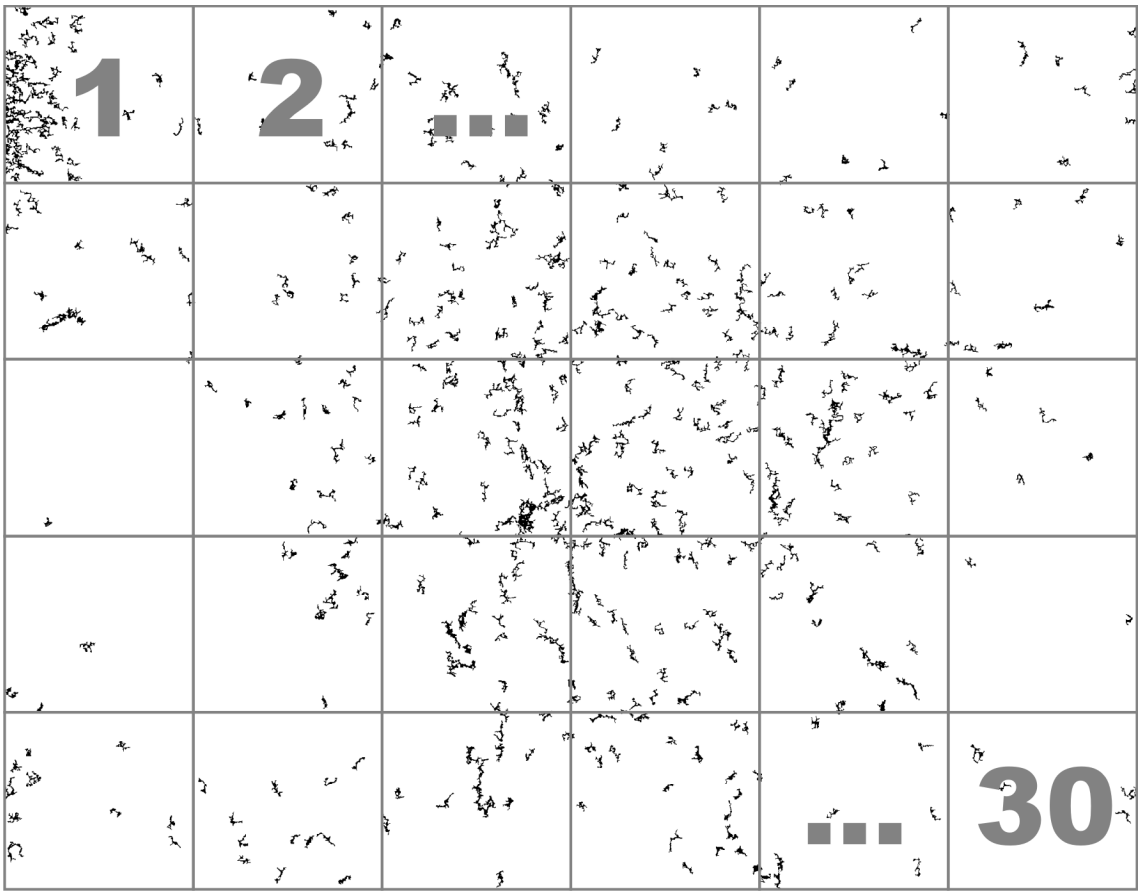


Abbildung 26: In dreißig Segmente unterteiltes Schwarzweißbild eines Asphaltphotos

Für jedes dieser Segmente werden, basierend auf den in diesem Segment befindlichen Objekten, sechs Kenngrößen bestimmt:

Objektzahl (1. Kenngröße)

Für jedes Bildsegment wird die Anzahl der darin enthaltenen Objekte ermittelt. Für die Position eines Objektes ist sein Schwerpunkt maßgeblich.

Bedeckungsgrad (2. Kenngröße)

Der Bedeckungsgrad ist der Quotient aus der Fläche (Pixelzahl) aller Objekte innerhalb eines Bildsegmentes und der Fläche des Bildsegmentes. Da die Größe der einzelnen Segmente identisch ist, wird als Maß für den Bedeckungsgrad die Summe der Pixel aller enthaltenen Objekte herangezogen.

Ausrichtung (3.-6. Kenngröße)

Die letzten vier Kenngrößen geben an, wie stark die Objekte innerhalb der jeweiligen Bildsegmente nach oben, unten, links oder rechts zu den möglichen Nachbarsegmenten hin orientiert sind. Dazu ist es zunächst erforderlich, jedes einzelne Objekt, wie in Abbildung 27 gezeigt, zu charakterisieren.

Abbildung 27: Zwei Beispielobjekte mit Angaben zu ihrer Ausrichtung, beschrieben durch Winkel und Korrelationskoeffizient

Zum einen wird der Korrelationskoeffizient zwischen den x- und y-Koordinaten der Pixel bestimmt. Dieser Wert gibt an, wie sehr die Ausrichtung des Objektes ausgeprägt ist. Da in diesem Zusammenhang negative Korrelationskoeffizienten keine Aussage haben, wird der Absolutwert der Korrelationskoeffizienten herangezogen. Diese haben einen Wertebereich von 0 bis 1. Die maximale Ausrichtung von 1 wäre gleichbedeutend mit einer Linie, während ein Wert von 0

einen Pixelhaufen ohne jegliche Ausrichtung beschreibt. Der Winkel des Objektes gibt schließlich an, in welche Richtung das Objekt ausgerichtet ist. Um ihn zu bestimmen, wird eine lineare Regression durchgeführt. Der Winkel wird in einem Wertebereich von -90° bis +90° angegeben, wobei eine senkrechte Ausrichtung einem Winkel von 0° entspricht. Basierend auf dem Winkel der Objekte und ihrer Position innerhalb des Segmentes, werden die absoluten Korrelationskoeffizienten der Objekte in den vier Ausrichtungskenngrößen (nach oben, unten, links oder rechts) des jeweiligen Segmentes aufsummiert.

Das ursprüngliche Bild mit 5 Mio. Bildpunkten/Parametern wird, bei einer Anzahl von 30 Bildsegmenten mit jeweils 6 Kenngrößen, somit auf einen Satz von 180 Parametern reduziert. Diese können im nächsten Schritt an ein neuronales Netz übergeben werden.

3.3.9 Die neuronalen Netze

Am Ende des Mustererkennungsprozesses steht die Klassifizierung. Für diese Aufgabe wurden in dieser Arbeit zwei feedforward backpropagation Netze verwendet und in verschiedenen Varianten trainiert. Die Netze werden hier als Netze der Kategorie 1 (Kat01) und der Kategorie 2 (Kat02) bezeichnet. Abbildung 28 stellt die beiden Netzwerkarchitekturen gegenüber.

Bei den Netzen vom Typ Kat01 handelt es sich um einfache feedforward Netze. Die 180 ermittelten Kenngrößen werden als ein Eingabevektor an das neuronale Netz übergeben. Das Netz rechnet mit einer verdeckten Schicht mit einer variablen Anzahl *n* an Neuronen und hat eine Ausgabeschicht mit einem Neuron, um einen Ausgabewert zu berechnen, der zur Beurteilung des vorliegenden Bildes dient. Bei diesem neuronalen Netz werden alle 180 Kenngrößen von einer Schicht verarbeitet. Im Grunde handelt es sich bei diesen 180 Kenngrößen allerdings um 30 Gruppen mit jeweils 6 Kenngrößen, die aus verschiedenen Bildsegmenten stammen. Und die Bewertung des ersten Bildsegmentes oben links sollte unabhängig sein von der Bewertung des 30. Bildsegmentes unten rechts. Um dieser Tatsache gerecht zu werden, wurden die Netze vom Typ Kat02 entwickelt.

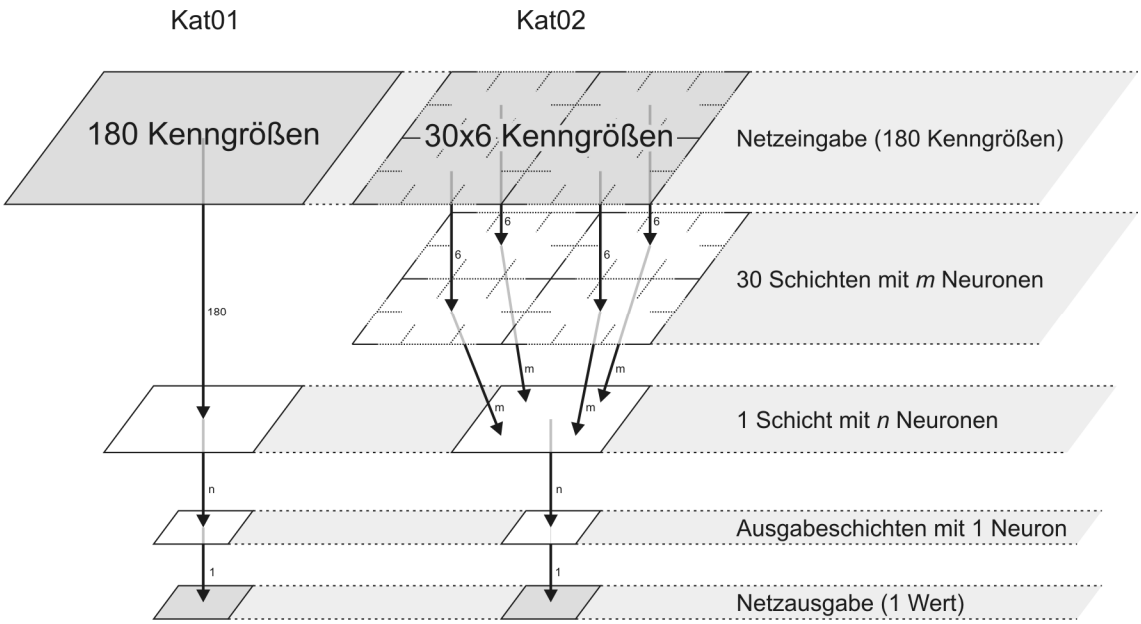


Abbildung 28: Gegenüberstellung der Netzarchitekturen Kat01 und Kat02. Die Unterteilung des Bildes in 30 Segmente und die daraus resultierende Struktur des Netzes vom Typ Kat02 ist der Übersicht halber durch vier Segmente dargestellt.

Bei den Kat02-Netzen wird nicht ein Eingabevektor mit 180 Kenngrößen übergeben, sondern, entsprechend der Anzahl an Bildsegmenten, 30 Eingabevektoren mit jeweils nur sechs Kenngrößen. Diese werden von 30 Schichten mit jeweils *m* Neuronen unabhängig voneinander verarbeitet. Dies spiegelt die Tatsache wieder, dass z.B. der Straßenzustand in der oberen linken Ecke unabhängig ist vom Straßenzustand in der unteren rechten Ecke. Erst in der nächsten Schicht mit *n* Neuronen werden die Ausgaben der vorangegangenen 30 Schichten zusammengeführt und gemeinsam weiterverarbeitet. Am Ende steht auch hier eine Ausgabeschicht mit einem Neuron für die Bewertung des vorliegenden Bildes.

Durch die segmentweise Verarbeitung der Kenngrößen haben die Netze der Kategorie 2 mehr Neuronen, aber weniger Wichtungsfaktoren. Ein Kat01-Netz

hätte mit 4 Neuronen bereits 724 Wichtungsfaktoren, ein Kat02-Netz mit jeweils 2 Neuronen in allen Schichten (62 Neuronen) hätte lediglich 482 Wichtungsfaktoren.

3.3.10 Simplexoptimierung

Der Mustererkennungsprozess beinhaltet eine Vielzahl von Parametern, die im Rahmen der Entwicklung zunächst empirisch bestimmt, bzw. festgelegt werden müssen. Die Leistung des Mustererkennungsprozesses kann verbessern werden, wenn die Parameter der einzelnen Arbeitsschritte optimiert werden. Mitunter sind die Parameter sehr sensitiv und gekoppelt. So wird die mittlere Größe der Objekte nach der Grenzwertfilterung deutlich zunehmen, wenn der Grenzwert gesenkt wird. Entsprechend muss dann auch eine höhere Grenze für den Objektfilter angesetzt werden. Eine wahllose Kombination von Parametern innerhalb bestimmter Grenzen, wie es bei vielen globalen Optimierungsverfahren, wie z.B. dem genetischen Algorithmus, der Fall ist, würde in den meisten Fällen zu keinen brauchbaren Ergebnissen führen. Allerdings sollte mit einem lokalen Optimierungsverfahren wie dem Simplex, ausgehend von geeigneten Parameterkombinationen, eine Verbesserung möglich sein. Abbildung 29 zeigt wie die Simplex-Optimierung implementiert wurde.

Der Simplex-Algorithmus, der, ausgehend von bestimmten Startparametern, die Optimierung vornimmt, wurde in Matlab implementiert. In jedem Iterationsschritt wird eine Fehlerfunktion aufgerufen, die die aktuelle Parameterkombination evaluiert. Im Rahmen dieser Fehlerfunktion wiederum wird ein Delphi-Programm aufgerufen, welches die Bilder, entsprechend den gewünschten Parametern, verarbeitet und die aus den verarbeiteten Bildern extrahierten Kenngrößen abspeichert. Diese werden dann von der ausführenden Matlab-Funktion eingelesen und zum Training neuronaler Netze eingesetzt. Da das Training aufgrund der zufälligen Initialisierung sehr unterschiedlich verlaufen kann, wird nicht nur ein Netz trainiert, sondern es werden in jedem Iterationsschritt 100 neuronale Netze trainiert und validiert. Wahlweise kann der Mittelwert der in der Validation erzielten Fehler oder aber das beste erzielte Ergebnis als Fehler der Fehlerfunktion herangezogen werden, um als Basis für die Entwicklung des

Simplex zu dienen. Schrumpft der Simplex unter eine bestimmte Größe, wird die Optimierung beendet und die optimierten Parameter als Ergebnis zurückgegeben.

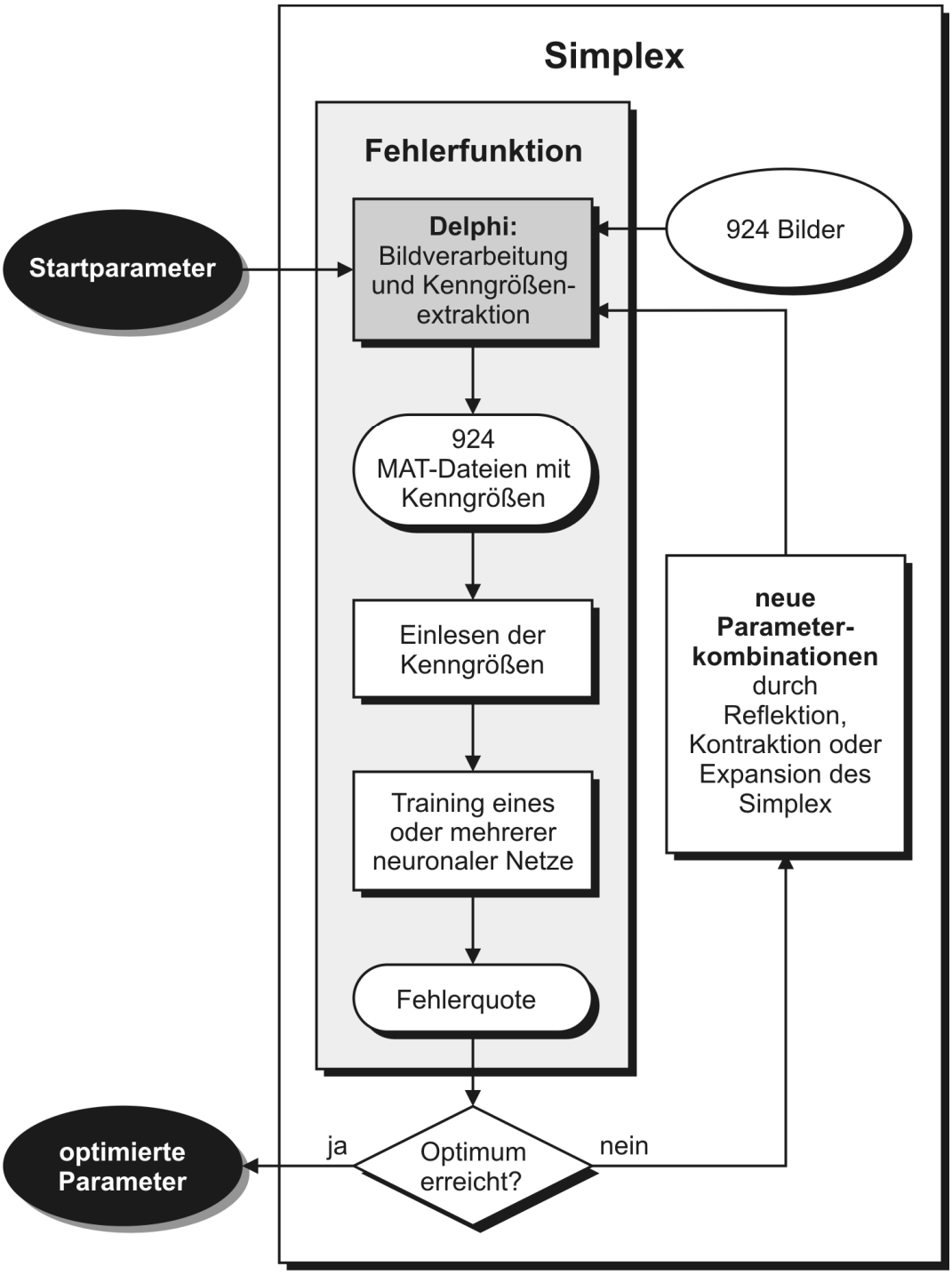


Abbildung 29: Optimierung der Parameter des Mustererkennungsprozesses mit Hilfe eines Simplex-Algorithmus

3.4 Beurteilung und Angabe von Fehlern

Um die Ergebnisse der neuronalen Netze zu beurteilen, müssen Fehlerwerte berechnet werden. In dieser Arbeit werden Fehler auf zwei Arten angegeben: als *RMSEP* oder als *FQ* (Fehlerquote). Mit dem RMSEP wird im Training wertungsfrei die Differenz zwischen Trainingsziel und Netzausgabe erfasst. Die Fehlerquote hingegen beinhaltet die Klassifizierung der Bilder als schadhaft oder schadfrei.

RMSEP (root mean square error of prediction)

Die Beurteilung der Bilder durch das Netz erfolgt, je nach Implementierung, auf einer Skala von 0 bzw. -1 (schadfrei) bis 1 (schadhaft). Der RMSEP ist die Wurzel der mittleren quadrierten Abweichung der Netz-Vorhersage vom Zielwert. Er hat die gleiche Einheit und Größenordnung wie die Werte selbst und ist somit direkt vergleichbar.

$$RMSEP = \sqrt{\frac{\sum_{i=1}^{n} \left(Netz_i - Zielwert_i \right)^2}{n}} \tag{23}$$

Innerhalb der neuronalen Netze wird im Laufe des Trainings intern mit dem SSE (sum square error) gerechnet. Dieser ähnelt dem RMSEP, ist aber lediglich die aufsummierte, quadrierte Abweichung.

$$SSE = \sum_{i=1}^{n} \left(Netz_i - Zielwert_i \right)^2 \tag{24}$$

Fehlerquote

Bei einem Wertebereich von –1 bis 1 zum Beispiel würde jede Bewertung kleiner 0 als schadfrei, jede Bewertung größer 0 als schadhaft behandelt werden. Diese

Bewertung kann dann mit der Vorgabe abgeglichen werden. Die Fehlerquote ergibt sich dann als prozentualer Anteil falsch bewerteter Bilder.

$$FQ = \frac{n_{falsch}}{n_{bewertet}} \cdot 100\%$$

Gegebenenfalls kann hier noch unterschieden werden nach der Fehlerquote bei der Beurteilung schadfreier Bilder $FQ_{schadfrei}$ und der Fehlerquote bei der Beurteilung schadhafter Bilder $FQ_{schadhaft}$. Im Sinne dieser Arbeit, schadhafte Stellen zu finden, wird angestrebt insbesondere $FQ_{schadhaft}$ zu minimieren.

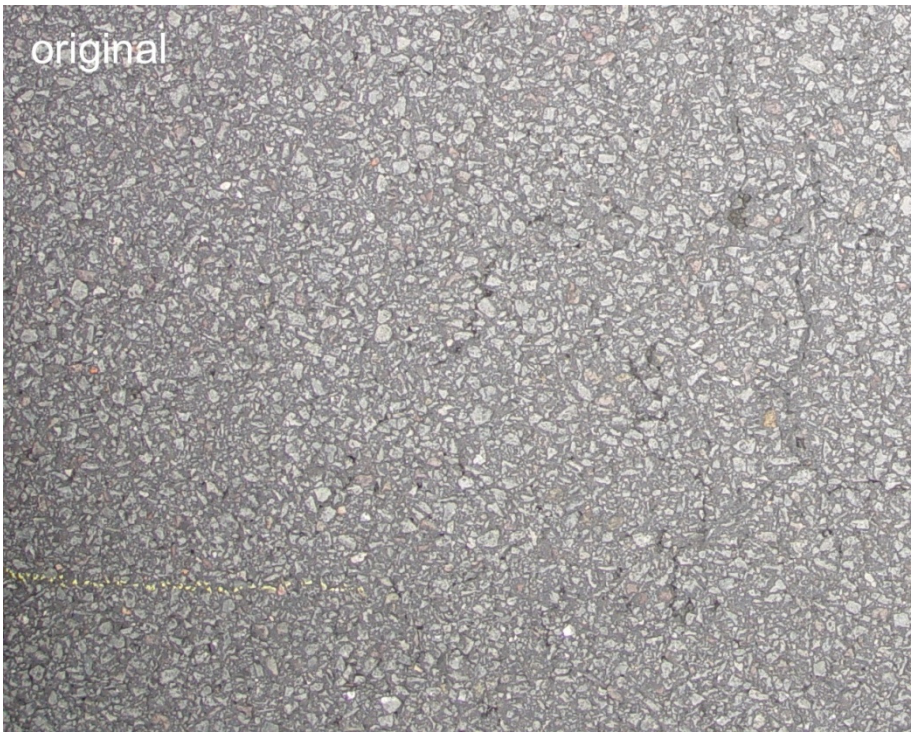
4 Ergebnisse

4.1 Reduktion der Farbtiefe

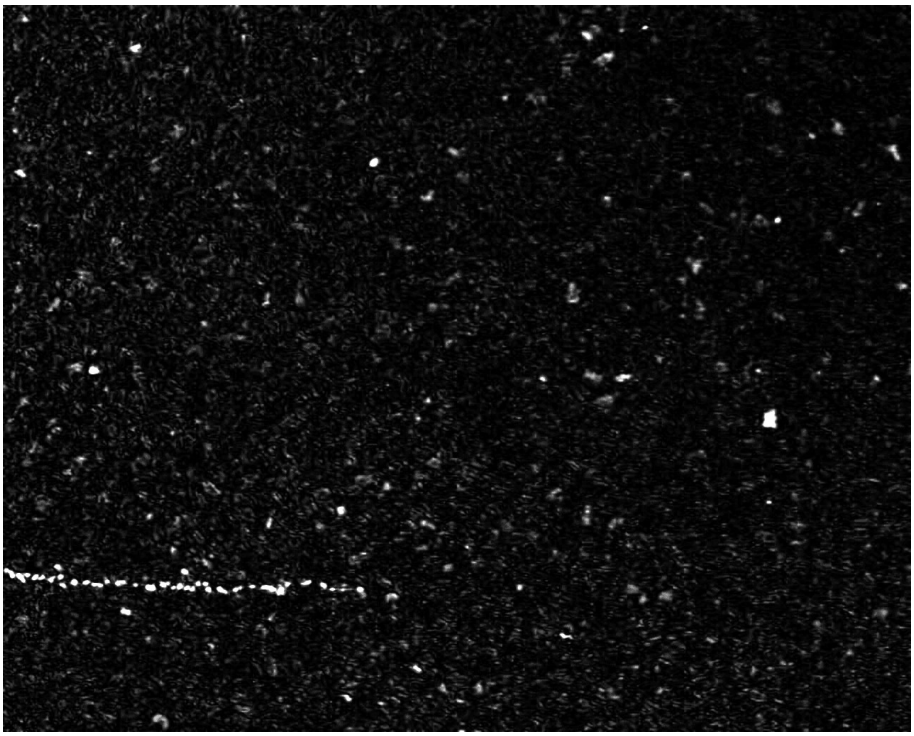
Der erste Schritt der Bildbearbeitung ist die Reduktion der Farbtiefe. Hierfür muss sichergestellt werden, dass keine relevanten Informationen verloren gehen.

Abbildung 30 zeigt das Ergebnis der Untersuchung der Standardabweichung über die drei Farbkanäle der JPG-Bilder. Sie verdeutlicht, worin die Unterschiede zwischen den Farbkanälen bestehen und inwiefern sie für diese Arbeit relevant sind. In Abbildung 30a ist ein Originalbild dargestellt, während Abbildung 30b die Standardabweichung über die drei Farbkanäle zeigt.

Im unteren linken Bildausschnitt in Abbildung 30a ist eine gelbe Farbspur zu erkennen. Ebenso fallen einige farbige Steine auf. Eben diese Merkmale sind es, die in Abbildung 30b in Erscheinung treten. Für das gesamte Bild beträgt die Standardabweichung 10 (einheitenlose Farbwerte von 0 bis 255). Bei einem mittleren Grauwert von 160 entspricht dies 6,3 %. Allerdings liegen über 70 % aller Pixel unter diesem Wert. Lediglich die angesprochenen Merkmale heben sich deutlich ab und erreichen Werte bis 1800. Da Farbspuren auf der Fahrbahn oder die tatsächliche Farbe einzelner Fahrbahnbestandteile für das Problem der Risserkennung momentan nicht von Bedeutung sind, ist die Standardabweichung als vernachlässigbar klein zu betrachten. Alle zur Verfügung stehenden Bilder werden daher in 8 Bit Graustufenbilder konvertiert und stellen die Grundlage aller folgenden Arbeitsschritte dar.



(a)

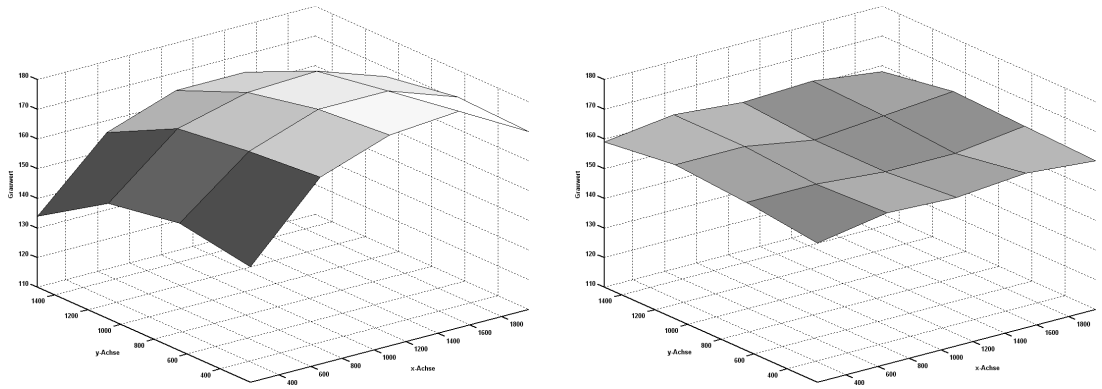


(b)

Abbildung 30: Darstellung der Standardabweichung über die drei Farbkanäle

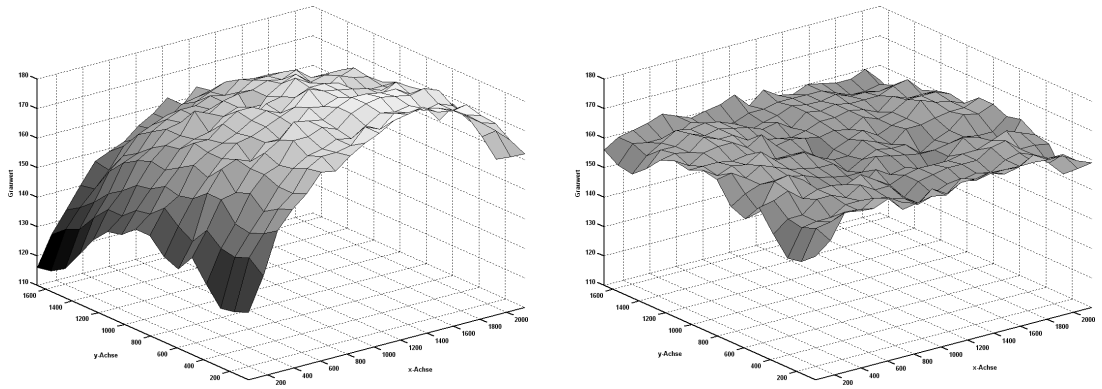
4.2 Helligkeitskorrektur

Die Ergebnisse der Helligkeitskorrektur sind abhängig von der Anzahl der Intervalle, über die die mittleren Grauwerte berechnet werden. Abbildung 31 zeigt, wie sich die Helligkeitsverteilung vor und nach der Korrektur darstellt, wenn 5 Intervalle in x-Richtung und 4 Intervalle in y-Richtung (20 Segmente) benutzt werden. Abbildung 32 stellt im Vergleich dazu die über 320 Segmente (20x16) berechnete Helligkeitsverteilung dar.



a) b)

Abbildung 31: Helligkeitsausgleich berechnet über 20 Segmente



a) b)

Abbildung 32: Helligkeitsausgleich berechnet über 320 Segmente

Je weniger Segmente man berechnet, desto gröber ist die Anpassung. Bei lediglich einem Segment könnte man nur noch den mittleren Grauwert des Bildes anpassen, jedoch nicht die ungleichmäßige Ausleuchtung korrigieren. Das andere Extrem wäre, mit jedem einzelnen Pixel zu rechnen. Dies wäre möglich, zieht jedoch einen beträchtlichen Rechenaufwand nach sich. Um eine geeignete Wahl für die Unterteilung des Bildes zu treffen, wurden die Anzahl der Intervalle in x-Richtung systematisch von 3 bis 60 variiert und die Parameter der angepassten Korrekturfunktion bewertet.

Stellvertretend für alle Parameter der Korrekturfunktion ist in Abbildung 33 die Abhängigkeit des Parameters d_x (vergl. Gleichung 20) von der Anzahl der Intervalle dargestellt. Die übrigen Parameter verhalten sich entsprechend.

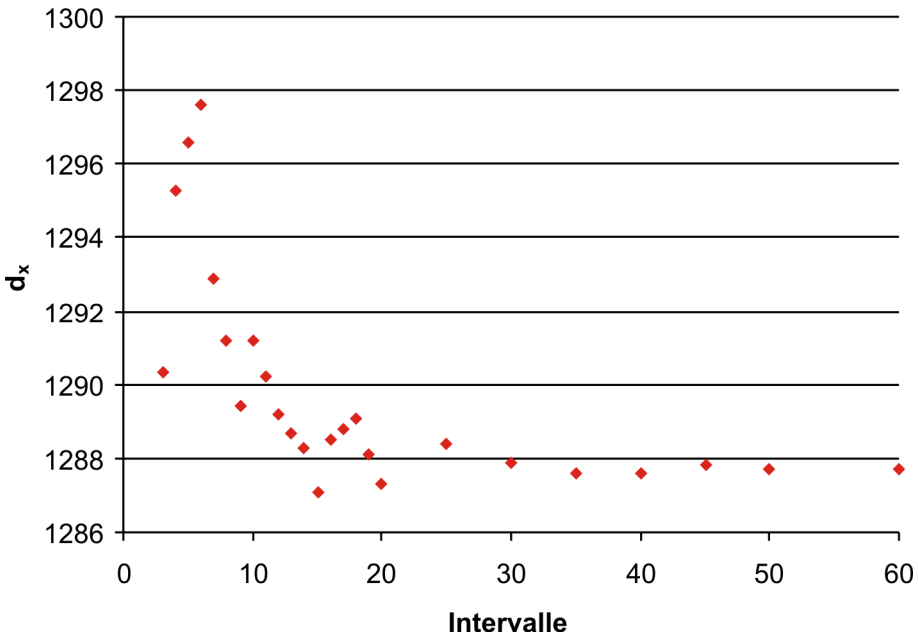


Abbildung 33: Optimierter Parameter der Korrekturfunktion in Abhängigkeit der Unterteilung des Bildes

Unter drei Intervallen ist eine Anpassung nicht möglich, da die Korrekturfunktion fünf Parameter hat. Somit werden mindestens ebensoviele Segmente benötigt, damit die Korrekturfunktion definiert ist. Im Bereich von drei bis circa 20

Intervallen verändern sich die Parameter sehr stark. Ab circa 20 Intervallen ändern sich die Parameter dann kaum noch. Entsprechend dem Seitenverhältnis der Bilder bedeuten 20 Spalten (20 Intervalle in x-Richtung) eine Unterteilung in 16 Zeilen. Daraus resultieren 320 Segmente für die Berechnung. Eine Unterteilung des Bildes in weniger Segmente führt zu einer unzuverlässigeren Anpassung der Korrekturfunktion, mehr Segmente hingegen erhöhen lediglich die Rechenzeit. Der mittlere Grauwert der korrigierten Bilder wird im Rahmen dieser Arbeit auf 160 festgelegt. Alle zur Verfügung stehenden Bilder wurden auf diese Art korrigiert und stellen die Grundlage aller folgenden Arbeitsschritte dar.

4.3 Dilatation

Um die Wirkungsweise der Dilatation zu verdeutlichen, zeigt Abbildung 34 einen direkten Vergleich der Segmentierung mit und ohne Dilatation.

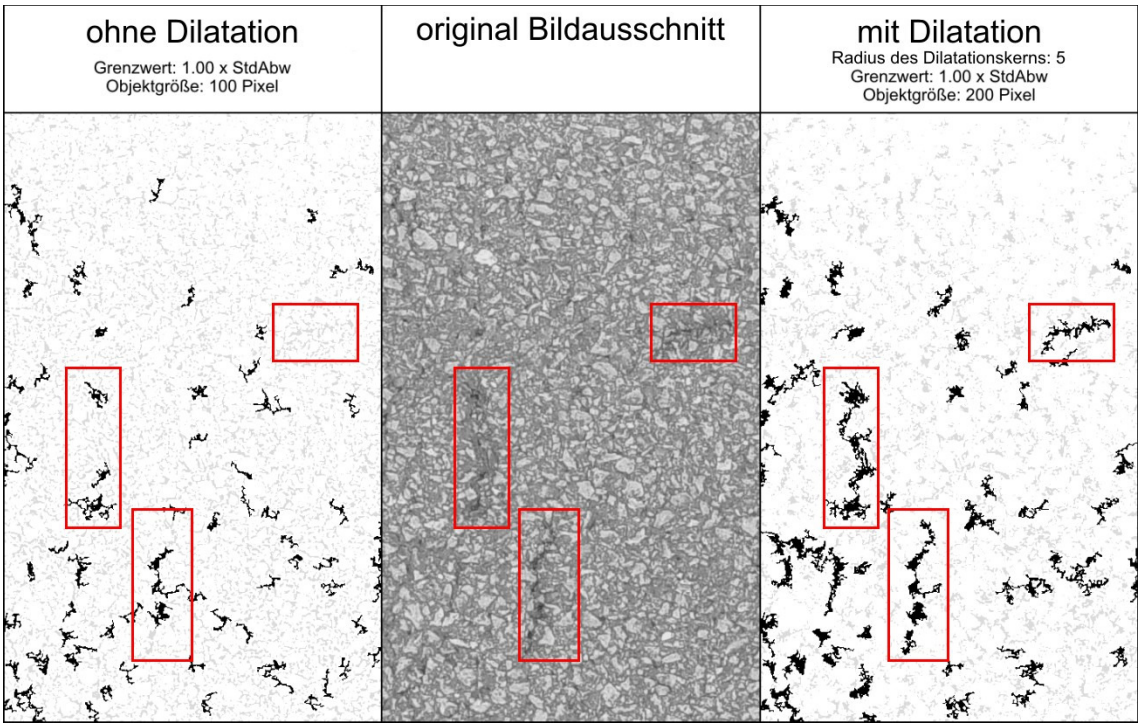


Abbildung 34: Ergebnisse der Segmentierung ohne und mit Dilatation

Eine empirisch gefundene Parameterkombination, die sich bewährt hat, ist der Faktor 1 für den adaptiven Grenzwertfilter und eine Objektgröße von mindestens 100 Pixel für den Objektfilter. Das Ergebnis der Segmentierung unter Verwendung dieser Parameterkombination sowie der originale Bildausschnitt sind in Abbildung 34 zu sehen. Darüber hinaus wird derselbe Bildausschnitt nach der Segmentierung unter Verwendung der Dilatation gezeigt. Der Vergleich ist insofern schwierig, da der Parameter des Objektfilters nicht der gleiche ist. Wenn aufgrund der Dilatation kleinere Objekte zu größeren zusammengefügt werden, muss die Mindestgröße des Objektfilters angepasst werden. In diesem Fall wurde ein Wert von 200 gewählt. Dennoch ist zu erkennen, dass die markantesten Stellen des originalen Bildausschnittes (rot umrandet) deutlich besser zur Geltung kommen. Der Riss oben rechts wird überhaupt erst durch den Einsatz der Dilatation erfasst.

4.4 Adaptiver Grenzwertfilter

Die vorliegenden Bilder sind die Aufnahmen eines Straßenabschnittes, die an einem Tag bei identischen Witterungsbedingungen aufgenommen wurden. Daher sind die Bilder untereinander vergleichbar. Um nun die Effektivität des adaptiven Grenzwertfilters zu überprüfen, wurde ein ausgewählter Bildausschnitt verfremdet, indem Helligkeit und Kontrast manipuliert wurden. Gefiltert wurden neben dem originalen Bild ein aufgehelltes Bild, ein abgedunkeltes Bild und ein Bild mit verstärktem Kontrast. Diese Variationen sind in der ersten Spalte von Abbildung 35 sehen. Die zweite Spalte zeigt das Ergebnis einer Grenzwertfilterung mit festgelegter Grenze. Wie zu erwarten, fallen die Ergebnisse sehr unterschiedlich aus. Während der gewählte Grenzwert beim Originalbild die interessanten Strukturen erkennen lässt, sind die Ergebnisse bei den manipulierten Bildern gänzlich unbrauchbar. Die Ergebnisse des adaptiven Grenzwertfilters jedoch sind, wie die dritte Spalte von Abbildung 35 zeigt, unabhängig von den abweichenden Bedingungen nahezu identisch. Dies gewährleistet eine zuverlässige Arbeit des Filters unter wechselnden Bedingungen.

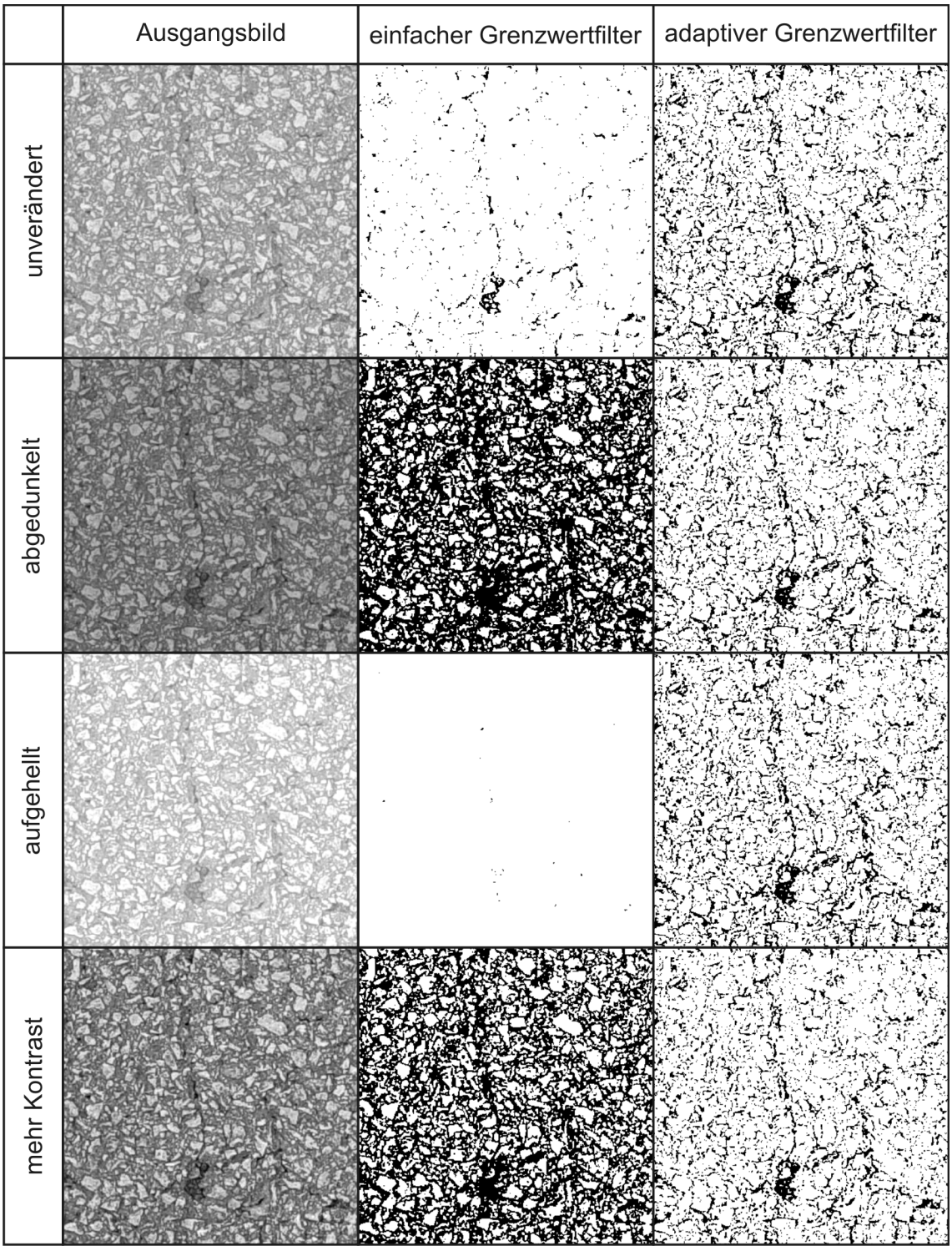


Abbildung 35: Vergleich eines einfachen Grenzwertfilters mit dem adaptiven Grenzwertfilter bei Bildern unterschiedlicher Helligkeit und mit unterschiedlichem Kontrast

4.5 Verknüpfung von Objekten

Zur Vektorisierung von Objekten wurde ein Tracer-Algorithmus geschrieben, welcher den potentiellen Riss „abläuft“ und sich rekursiv verzweigen kann. Dies wurde im gleichnamigen Programm *Tracer* implementiert. Abbildung 36 zeigt das **Tra**cer **De**bug **Ce**nter (TraDeCe), welches in der Testphase die zentrale Steuerung des Tracers darstellt und eine Überwachung seiner Fortschritte ermöglicht. In diesem Interface kann ein Tracer erschaffen und positioniert werden. Die Position des Tracers kann zu Testzwecken zufällig gewählt werden (RndPos) oder auf den ersten Pixel gesetzt werden, der bei einer zeilenweisen Suche von oben gefunden wird (StartPos). Die mit *Go* gestartete Suche erfolgt bei eingeschaltetem Debug-Modus schrittweise oder wahlweise bis zur vollständigen Erfassung des Objektes.

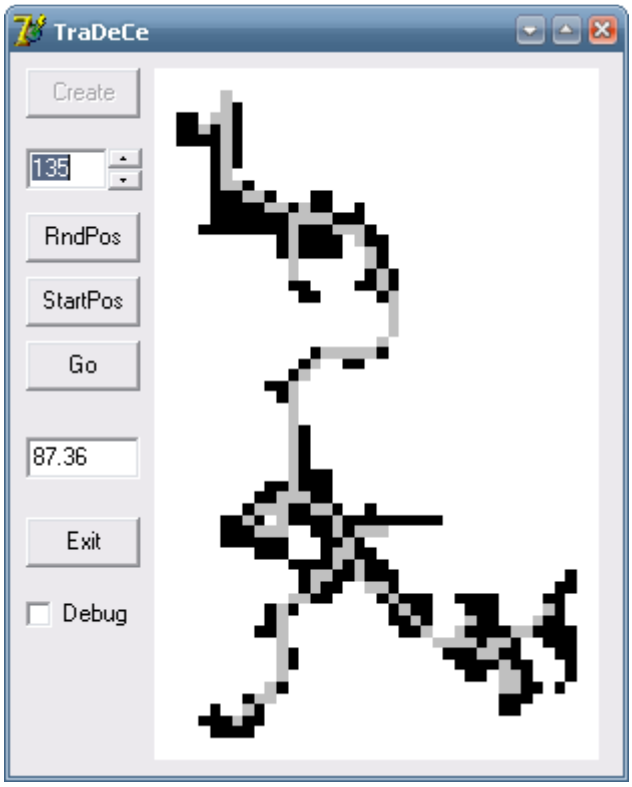


Abbildung 36: Das Tracer Debug Center (TraDeCe), in dem der Verlauf des Tracers dargestellt wird

Abbildung 37 zeigt das Statusfenster eines aktiven Tracers und veranschaulicht seine Arbeitsweise. Der Tracer hat eine bestimmte Laufrichtung, die durch den Pfeil gekennzeichnet wird, und in die er bestrebt ist weiterzulaufen. Der Zahlenkreis stellt dar, was der Tracer „sieht“, das heißt schwarze Pixel in der jeweiligen Richtung. Abgesucht wird dabei ein einstellbarer Suchradius. Werden in einer Richtung nur schwarze Pixel gefunden, resultiert daraus eine Pixeldichte von 1. Gibt es keinen schwarzen Pixel, führt dies zu einer Pixeldichte von 0. Als nächstes sucht der Tracer nach den Richtungen höchster Pixeldichte (gelb hervorgehoben), um sich gegebenenfalls zu verzweigen und in die entsprechenden Richtungen weiterzulaufen. Die Richtung, aus der der Tracer kommt, wird hierbei vernachlässigt.

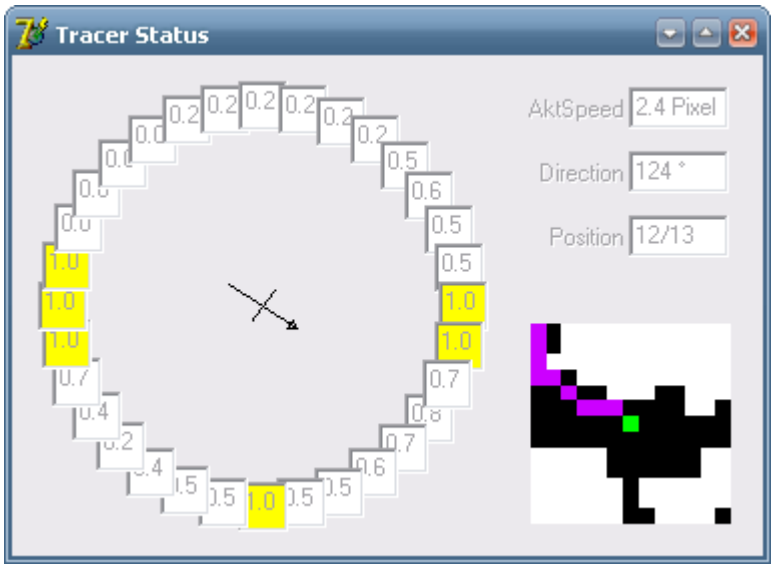


Abbildung 37: Statusfenster des Tracers, der seinen weiteren Weg sucht

Im dargestellten Beispiel in Abbildung 37 wurden drei Richtungen ausgemacht. Die eine wird als die Richtung identifiziert, aus der der Tracer kommt, die zweite als Laufrichtung, die weiter verfolgt wird. Die dritte Richtung führt zu einer Verzweigung. Zu diesem Zweck ruft der rekursiv arbeitende Tracer eine neue Instanz seiner selbst auf. Wie weit sich der Tracer im nächsten Schritt fortbewegt

ist den Statusinformationen in der oberen rechten Ecke zu entnehmen (AktSpeed). Diese setzt sich aus dem Suchradius, einer Geschwindigkeitskonstante und der Pixeldichte der jeweiligen Richtung zusammen. In der unteren rechten Ecke wird der Suchradius um den aktuellen Tracer (grüner Pixel) sowie abgelaufene Strecken (magenta Pixel) dargestellt.

Abbildung 38 zeigt das Ergebnis einer abgeschlossenen Vektorisierung.

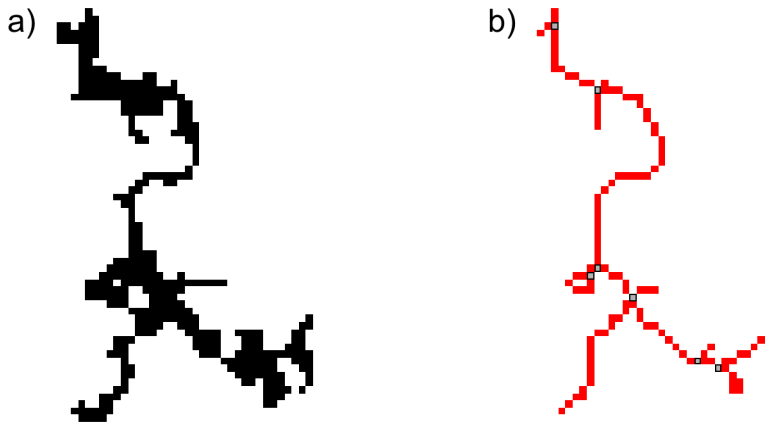


Abbildung 38: (a) Original und (b) vektorisiertes Objekt mit Verzweigungspunkten

Es hat sich allerdings gezeigt, dass dieses Procedere sehr zeitaufwändig und in dieser Form zu unzuverlässig ist. Durch Anpassung des Suchradius und der Laufgeschwindigkeit des Tracers kann gewährleistet werden, dass ein Objekt korrekt erfasst wird. Beim nächsten Objekt kann die Erfassung dann aber schon wieder unvollständig sein. Wird der Suchradius und die Laufgeschwindigkeit zu groß gewählt, kommt der Tracer gut voran, übersieht allerdings schmale Übergänge. Wird der Radius und die Laufgeschwindigkeit allerdings zu klein gewählt, verliert der Tracer die Orientierung. Der Tracer müsste also noch in die Lage versetzt werden, sich selbstständig anzupassen. Daher wird auf diese Form der Vektorisierung verzichtet, da die Verknüpfung naheliegender Objekte bereits gute Ergebnisse liefert (siehe Kapitel 4.6). Diese wird mit einer festen Entfernung von 20 Pixeln durchgeführt.

Um die Vektorisierung zu implementieren, wäre es in weiterführenden Arbeiten sinnvoll die Vektorisierung über eine Skelettierung der Objekte, z.B. mit dem Zhang/Suen-Algorithmus (2)(34)(35), zu realisieren. Bei der Skelettierung handelt es sich um ein Verfahren zur Ausdünnung binärer Bilder. Breite Linien oder auch flächige Objekte werden dabei auf ein *Skelett*, das Zentrum einer Linie oder die Mittelachse eines Objektes, reduziert.

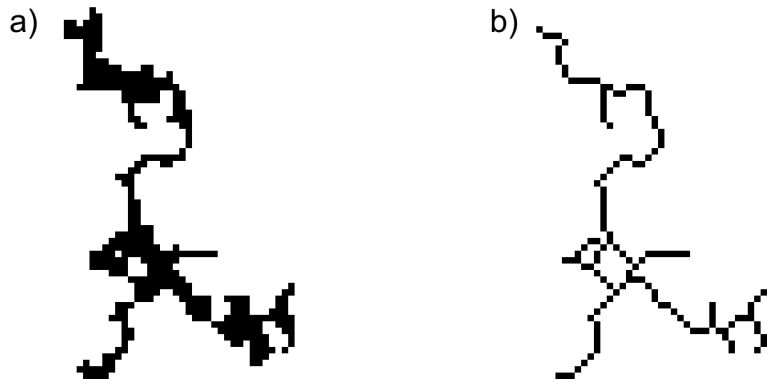


Abbildung 39: a) Ein Riss im Original und b) nach der Skelettierung mit dem Zhang/Suen-Algorithmus

Abbildung 39 zeigt den bereits zuvor gezeigten Rissabschnitt und das Ergebnis nach einer Skelettierung. Es handelt sich noch immer um eine pixelweise Beschreibung, und es ist ein weiterer Schritt erforderlich, um eine Beschreibung durch Vektoren mit Anfangspunkten, Endpunkten und Verzweigungen zu erhalten. Allerdings wurden hier die Daten bereits auf die relevante Struktur reduziert.

4.6 Kenngrößenextraktion

Abbildung 40 zeigt, wie ein Bildausschnitt letztlich vom Mustererkennungssystem gesehen wird. Zu diesem Zeitpunkt wurden der Grenzwertfilter, die Dilatation und der Objektfilter angewandt. Zurück bleiben die Objekte, die für die Auswertung durch das neuronale Netz als relevant erachtet wurden und nun mit Hilfe von

Kenngrößen erfasst werden. Die roten Kreise kennzeichnen hierbei den Schwerpunkt eines Objektes. Die roten Linien zeigen die Ausrichtung der Objekte, das heißt ihre Richtung (Winkel), und zum anderen wie ausgeprägt diese Ausrichtung ist (absoluter Korrelationskoeffizient). Je länger die Linie ist, desto ausgeprägter ist die Ausrichtung.

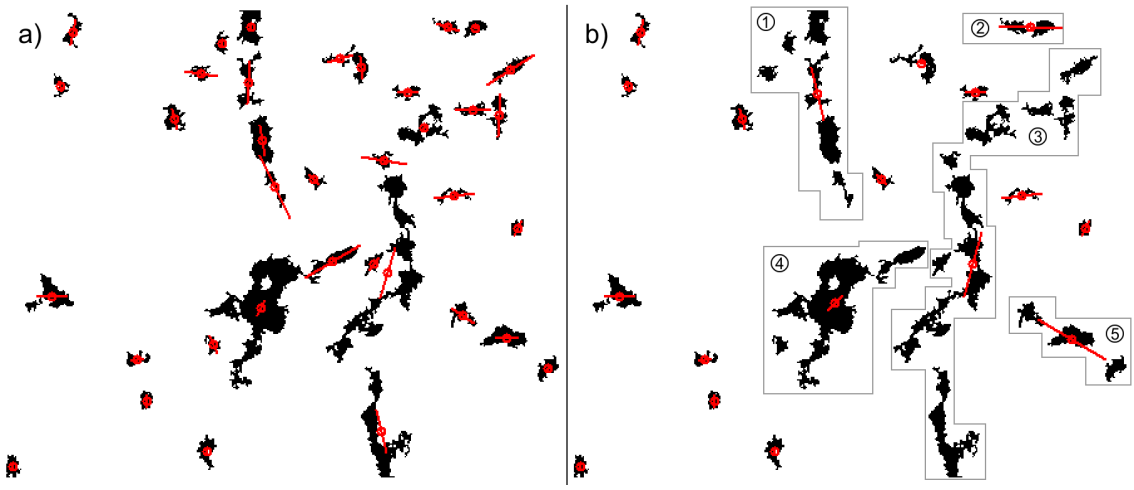


Abbildung 40: Visualisierung ermittelter Kenngrößen, a) ohne Verknüpfung von Objekten und b) mit Verknüpfung naheliegender Objekte

In Abbildung 40 werden die Ergebnisse mit und ohne Verknüpfung naheliegender Objekte gegenübergestellt. Abbildung 40a zeigt das Ergebnis, welches man ohne Verknüpfung erhält. Abbildung 40b zeigt, wie zahlreiche kleinere Objekte zu fünf großen Objekten verknüpft wurden. Aufgrund der Implementierung, die auf Geschwindigkeit optimiert ist, werden nicht alle Verbindungen registriert. Ein kleines Objekt oberhalb von Objekt 3 wurde nicht verbunden, und ebenso hätten Objekt 4 und 5 verbunden werden müssen. Dennoch ist zu erkennen wie kleine Objekte erfolgreich zu größeren Strukturen zusammengefasst wurden. Die Objekte 2 und 5 zeigen deutlich, wie kleinere Objekte zu größeren Objekten mit deutlicher ausgeprägter Ausrichtung verknüpft wurden. In den Objekten 1, 3 und 4 wurden mitunter einzelne kleine Objekte verknüpft, die bereits eine sehr deutliche Ausrichtung hatten und diese einbüßen. Dennoch wurden auch hier diverse

Objekte mit geringer Aussagekraft erfolgreich in größere aussagekräftigere Strukturen aufgenommen.

Abbildung 41 zeigt beispielhaft anhand von vier einzelnen Objekten, wie sie vom Mustererkennungssystem gesehen werden, wie diese bewertet und in die Kenngrößen eingerechnet werden.

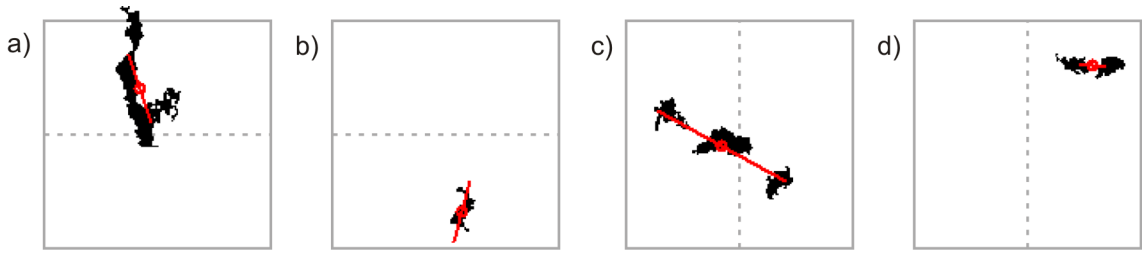


Abbildung 41: Zuordnung von Objekten innerhalb eines Segmentes zu den Kenngrößen 3 bis 6. a) Kenngröße 3 (nach oben), b) Kenngröße 4 (nach unten), c) Kenngröße 5 (nach links), d) Kenngröße 6 (nach rechts)

Bei den Objekten a) und b) handelt es sich jeweils um Objekte mit einem Winkel zwischen -45° und +45°. Diese werden als Objekte mit vertikaler Ausrichtung erkannt. Das Objekt a) liegt in der oberen Hälfte des Segmentes und wird somit zur Kenngröße 3 dieses Segmentes addiert, Objekt b) hingegen zur Kenngröße 4, da sein Schwerpunkt in der unteren Hälfte liegt. Entsprechend werden die Objekte c) und d) mit Winkeln kleiner -45° bzw. größer +45° als Objekte mit horizontaler Ausrichtung zu den Kenngrößen 5 und 6 addiert.

4.6.1 Verbesserungen in der Delphi-Implementierung

Mit der Delphi-Implementierung wurde die Kenngrößenextraktion verbessert. Die Charakterisierung der Objekte, wie sie erstmals in Matlab entwickelt wurde, basiert auf einer linearen Regression. Abbildung 42 zeigt zwei lineare Regressionen an einem Satz von 10 Datenpunkten.

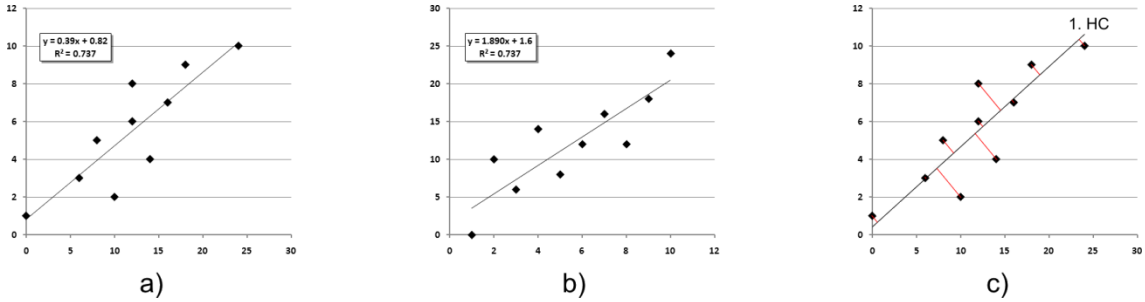


Abbildung 42: Lineare Regression an 10 Punkten. a) original Daten b) x- und y-Werte vertauscht. c) erste Hauptkomponente

Bei Abbildung 42a handelt es sich um den originalen Datensatz, während in Abbildung 42b die x- und y-Achsen vertauscht wurden. Wie in der Abbildung zu erkennen ist, sind die Ergebnisse, unabhängig von der Vertauschung der Achsen, nicht identisch. Im ersten Fall (a) wurde eine Steigung von 0,39 ermittelt. Nach Vertauschung der Achsen hätte die Steigung für den zweiten Fall (b) dem Kehrwert von 0,39, also 2,56 entsprechen sollen. Statt dessen erhält man eine Steigung von 1,89. Um unabhängig von der Ausrichtung des Objektes eine eindeutige Lösung zu finden, wird in der Delphi-Implementierung mit dem NIPALS-Algorithmus die erste Hauptkomponente berechnet. Damit wird die Richtung der größten Varianz bestimmt. Anstelle der Regressionskoeffizienten wird die durch die erste Hauptkomponente erklärte Varianz benutzt. Auch diese lässt sich, bezogen auf die Gesamtvarianz der Daten als Wert zwischen 0 und 1 angeben.

4.6.2 Relevanz der Kenngrößen

Es gibt kleine, einfache Netzwerke, wie zum Beispiel das XOR-Netzwerk, mit klarer Funktionsweise, die genau nachvollzogen werden kann. Im Allgemeinen ist es allerdings schwierig, und meist nicht sinnvoll, die Funktionsweise von Netzwerken, das heißt im Wesentlichen die Bedeutung einzelner Wichtungsfaktoren, nachzuvollziehen. Dennoch kann sich ein Blick auf die Wichtungsfaktoren der Eingangsgrößen lohnen, um unabhängig vom weiteren Verlauf der Berechnungen abzuschätzen, inwieweit die Eingangsgrößen für das Netz relevant sind. Da neuronale Netze sehr komplex und aufgrund der zufälligen

Initialisierung auch nur bedingt reproduzierbar sind, wurde zu diesem Zweck eine Vielzahl von Netzen trainiert. Netze, die nicht erfolgreich trainiert werden konnten, wurden aussortiert. Von den restlichen Netzen wurden die Absolutwerte der Wichtungsfaktoren der ersten verdeckten Schicht(en) getrennt nach Kenngrößen aufsummiert. Teilt man diese Werte durch die Anzahl aufsummierter Werte, erhält man pro Kenngröße einen mittleren Wichtungsfaktor. Hat eine Kenngröße einen mittleren Wichtungsfaktor nahe 0, so legt dies nahe, dass die entsprechende Kenngröße keinen nennenswerten Beitrag zur Vorhersage des Netzes liefert.

Abbildung 43 zeigt exemplarisch die nach Kenngrößen aufsummierten Wichtungsfaktoren für ein neuronales Netz. Bereits bei diesem einzelnen Netz scheint keine der Kenngrößen für das neuronale Netz vernachlässigbar zu sein. Im Mittel, über die große Zahl trainierter Netze, ist das Bild noch ausgeglichener. Die Untersuchung der Wichtungsfaktoren der ersten verdeckten Schicht(en) legt also nahe, dass alle bisher verwendeten Kenngrößen einen nennenswerten Beitrag zur Auswertung leisten.

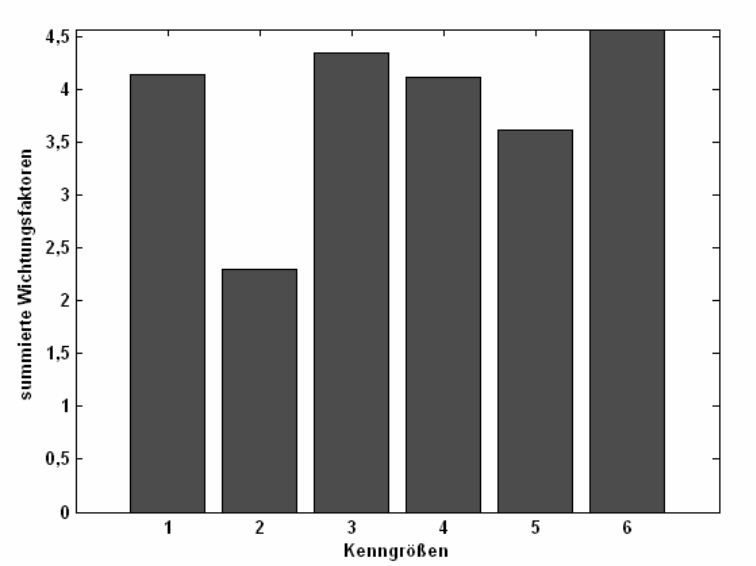
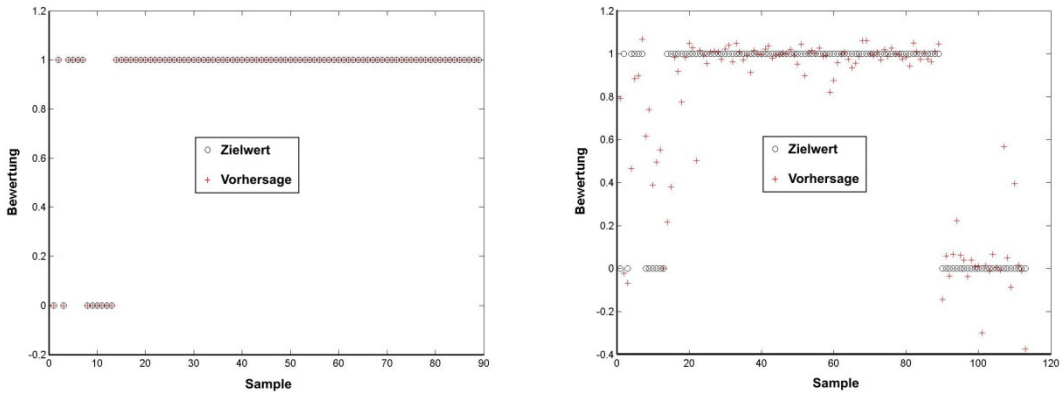


Abbildung 43: Mittlere absolute Wichtungsfaktoren für die sechs Kenngrößen

4.7 Die neuronalen Netze

4.7.1 Training mit und ohne Validation

In Abbildung 44 ist die Bedeutung einer Validation klar zu sehen. Abbildung 44a zeigt die Vorhersagen eines neuronalen Netzes aus einer frühen Phase an einem eingeschränkten Datensatz von nur 89 Samples (Bilder). Es zeigt die Vorhersagen, nachdem es mit eben diesen Samples trainiert wurde. Eine Bewertung von 0 bedeutet, dass es sich um einen schadfreien Asphaltabschnitt handelt, eine Bewertung von 1 hingegen deutet auf einen beschädigten Asphaltabschnitt. Es ist eindeutig, dass das Netz die vorliegenden Samples auswendiggelernt hat und diese einwandfrei wiedererkennt. Dies muss nicht schlecht sein. Es ist zumindest ein Zeichen, dass das Netz erfolgreich darauf trainiert werden kann, einen Zusammenhang zwischen den Eingangsdaten und den Zielwerten herzustellen. Allerdings besteht die Gefahr des *Overfittings*. Das Netzwerk kennt die vorliegenden Samples auswendig, doch es kann keine Aussage darüber getroffen werden, inwieweit das Netz in der Lage ist zu verallgemeinern. Vielmehr ist davon auszugehen, dass das Netz zu spezialisiert und nicht mehr in der Lage ist zu verallgemeinern.



(a) (b)

Abbildung 44: Netzvorhersagen nach Training a) ohne und b) mit Crossvalidation

Abbildung 44b hingegen zeigt das Ergebnis einer *Crossvalidation*. Zur Erhöhung der Datenbasis in dieser Projektphase wurden die 8 zur Verfügung stehenden schadfreien Bilder jeweils 3 Mal gespiegelt (in x, y und xy-Richtung), woraus eine Datenbasis von 113 Bildern resultiert. Für jedes der 113 Bilder wurde ein Netz ohne das jeweilige Bild trainiert. Mit diesem Netz wurde dann eine unabhängige Vorhersage über das ausgelassene Bild berechnet. Auf diese Art ist ausgeschlossen, dass das Netz sich das Bild gemerkt hat, und man erhält eine realistischere Einschätzung über die Fähigkeit des Netzes, unbekannte Bilder zu bewerten.

4.7.2 Aufbau ausgeglichener Datensätze

Ein weiterer Aspekt, welcher in der Anfangsphase deutlich wurde, ist die Wichtigkeit ausgeglichener Datensätze. Trotz eines vollständig bewerteten Datensatzes von 924 Bildern sind nur 104 dieser Bilder als schadhaft bewertet, während 820 Bilder als schadfrei bewertet sind. Erste Trainingsdurchläufe haben aber gezeigt, dass ein solches Verhältnis die Trainingsergebnisse verfälscht, da jedes Training dazu tendiert, pauschal alle Bilder als schadfrei zu beurteilen. Der Fehler wird damit, dem Anteil schadhafter Bilder entsprechend, 11 % betragen. Die folgenden Arbeiten unter Matlab und Delphi verfolgen unterschiedliche Ansätze.

Reduzierter Datensatz

Zunächst wurde unter Matlab mit reduzierten Datensätzen gearbeitet. Den 104 schadhaften Bildern wurde für diverse Trainingsdurchläufe eine zufällige Auswahl schadfreier Bilder zugeordnet. Diese wurden gleichmäßig auf einen Kalibrations- und einen Validationsdatensatz verteilt. Auf diese Art konnte mit 104 Bildern trainiert und mit 104 weiteren Bildern validiert werden.

Großer Datensatz durch Vervielfältigung der Daten

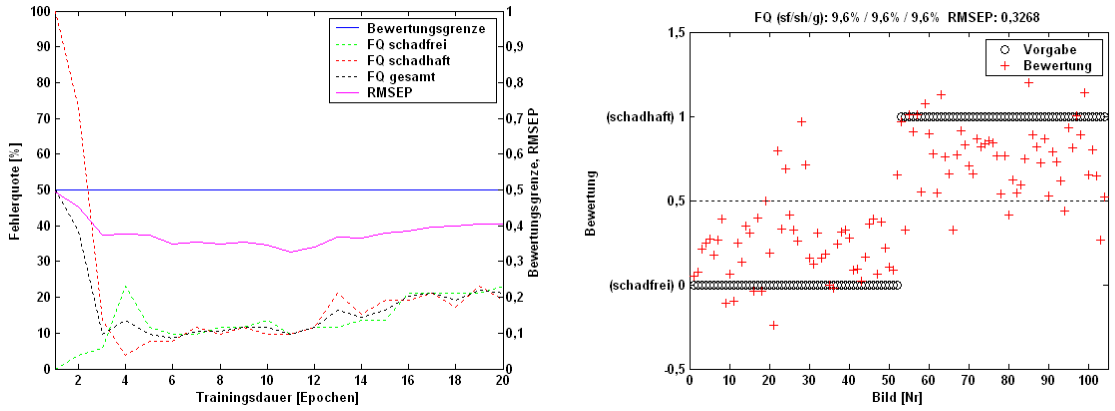
In späteren Matlab-Arbeiten wurde die Datenbasis aufgestockt, indem die schadhaften Bilder mehrfach ins Training einbezogen wurden. Die schadhaften Bilder wurden zunächst in Kalibrations- und Validationsdatensatz aufgeteilt und dann entsprechend der Anzahl schadfreier Bilder vervielfacht. Dies ermöglicht ein Training und eine Validation mit jeweils 820 Bildern.

Großer Datensatz durch Wichtung der Fehler

Der dritte Ansatz wurde unter Delphi realisiert. Da das Training der neuronalen Netze von Grund auf selber geschrieben ist, wurde hier nicht das Bildmaterial vervielfacht, sondern die Fehler der schadhaften Bilder entsprechend gewichtet.

Erfolgreiches Training mit ausgeglichenen Datensätzen

Abbildung 45 zeigt das erfolgreiche Training eines neuronalen Netzes an einem Datensatz mit 104 Bildern. In Abbildung 45a sind die Fehler in der Validation im Laufe des Trainings dargestellt. Die rote und die grüne Kurve zeigen die Fehlerquote bei der Beurteilung der schadhaften und der schadfreien Bilder. Grundlage für diese Fehlerquoten ist die Bewertungsgrenze von 0,5 (blaue Linie). Bilder mit einer Wertung unter 0,5 werden als schadfrei gewertet, darüber als schadhaft. Diese Bewertung ist allerdings sehr sprunghaft, wenn mehrere Bilder mit ihrer Bewertung in der Nähe der Grenze liegen. Daher richtet sich das Training nach dem RMSEP (magenta).



(a) (b)

Abbildung 45: Erfolgreiches Training mit großem Datensatz unter Matlab. a) Entwicklung der Fehler im Laufe des Trainings und b) Vorhersage des trainierten Netzes.

Hier ist deutlich zu sehen, wie das zunächst untrainierte Netz im Laufe des Trainings zunehmend bessere Resultate (geringere Fehler in der Validation) erzielt. Mit Fortschreiten des Trainings setzt dann allerdings auch das Overfitting

ein. Das Netz passt sich zunehmend dem Kalibrationsdatensatz an, verliert aber die Fähigkeit die Validationsdaten korrekt zu beschreiben.

Abbildung 45b zeigt letztlich die Vorhersage des trainierten Netzes, sortiert nach schadfreien (Bewertung 0) und schadhaften (Bewertung 1) Bildern. Dies verdeutlicht wie erfolgreich das Netz trainiert wurde. Sowohl bei der Bewertung der schadfreien, als auch bei der Bewertung der schadhaften Bilder liegt die Fehlerquote bei 9,6 %. Der RMSEP der Vorhersage liegt bei 0,327.

4.7.3 Die Netzstruktur

Um Aufschlüsse über eine geeignete Netzstruktur zu erhalten, wurde unter Matlab eine Vielzahl von Netzen trainiert. In einem Trainingsdurchgang wurden systematisch die Parameter für den Grenzwertfilter und den Objektfilter sowie die Anzahl an Neuronen in den neuronalen Netzen variiert. Die Parameter der Vorverarbeitung wurden, basierend auf den empirisch gefundenen Kombinationen 1.00/100 und 1.25/50, um jeweils ±10 % variiert. Die Anzahl verwendeter Neuronen variiert bei den Netzen vom Typ Kat01 zwischen 2 und 4 und bei den Netzen vom Typ Kat02 zwischen 1 und 3 für alle Schichten. Ein Trainingsdurchlauf umfasst 108 verschiedene trainierte neuronale Netze.

Für jeden Trainingsdurchgang wurde das beste erzielte Ergebnis herausgesucht und tabellarisch zusammengefasst. Diese Ergebnisse sind in Tabelle 1 und in Tabelle 2 zusammengefasst. Sie zeigen jeweils eine Zusammenfassung der besten erzielten Ergebnisse aus sieben Trainingsdurchläufen für die Netze vom Typ Kat01 und vom Typ Kat02. Angegeben sind die Parameter für die beiden Filter, die Fehlerquoten bei der Vorhersage schadfreier, schadhafter und aller Bilder (sf/sh/g), der RMSEP, die Anzahl verwendeter Neuronen und die Trainingsdauer in Epochen (Generationen).

Tabelle 1: Zusammenfassung der besten Ergebnisse von sieben Trainingsdurchläufen mit Netzen vom Typ Kat01.

	Grenzwert	Objektgröße	sf [%]	sh [%]	g [%]	RMSEP	Neuronen	Epochen
Durchgang 1	1.10	180	13.9	9.8	11.9	0.3235	3	5
Durchgang 2	0.90	200	14.9	11.7	13.3	0.3520	4	7
Durchgang 3	0.90	180	19.0	5.6	12.3	0.3350	3	4
Durchgang 4	1.10	180	16.8	11.7	14.3	0.3471	2	5
Durchgang 5	1.10	220	11.5	19.3	15.4	0.3367	3	8
Durchgang 6	0.90	200	17.1	11.7	14.4	0.3422	3	6
Durchgang 7	1.10	220	11.5	19.3	15.4	0.3367	3	8
Mittel	1.01	197	15.0	12.7	13.8	0.3390	3.0	6.1

Tabelle 2: Zusammenfassung der besten Ergebnisse von sieben Trainingsdurchläufen mit Netzen vom Typ Kat02.

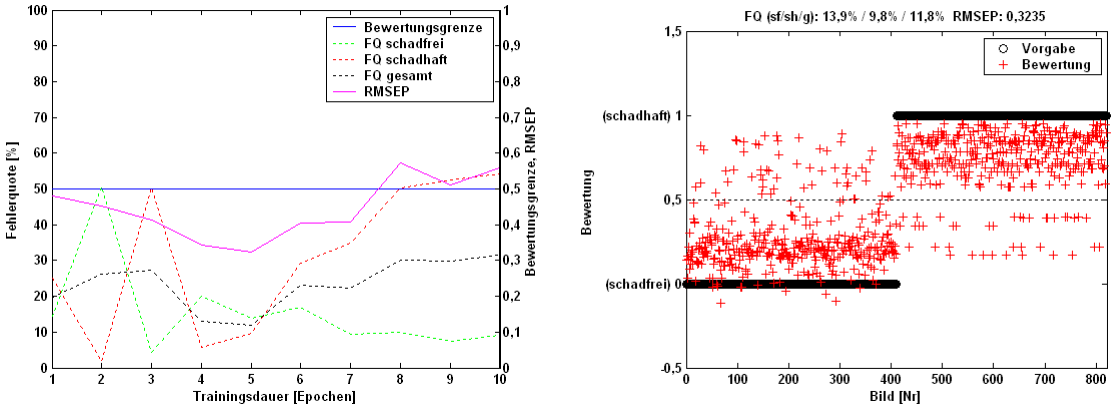
	Grenzwert	Objektgröße	sf [%]	sh [%]	g [%]	RMSEP	Neuronen	Epochen
Durchgang 1	1.00	200	15.9	7.6	11.8	0.3456	2	4
Durchgang 2	0.90	220	16.1	11.5	13.8	0.3376	3	4
Durchgang 3	1.10	200	22.0	9.5	15.8	0.3500	1	5
Durchgang 4	1.10	180	17.6	15.1	16.4	0.3655	2	5
Durchgang 5	1.10	200	13.2	17.6	15.4	0.3423	2	7
Durchgang 6	0.90	180	14.4	15.6	15.0	0.3483	1	6
Durchgang 7	1.10	200	13.2	17.6	15.4	0.3423	2	7
Mittel	1.03	197	16.1	13.5	14.8	0.3474	1.9	5.4

Die Ergebnisse für die Netze vom Typ Kat01 und Kat02 sind vergleichbar. Im Mittel liegen die Werte für den Grenzwertfilter ungefähr bei 1 und für den Objektfilter bei rund 200. Auch bei den erzielten Fehlern sind die Ergebnisse sehr ähnlich. Die RMSEP, die als Trainingskriterium dienen, liegen um 0,34. Dennoch zeigt sich in allen Werten, den Fehlerquoten und im RMSEP, dass die Netze vom Typ Kat01 die besseren Ergebnisse liefern. Allerdings sind die Netze vom Typ Kat02 zuverlässiger im Training. Die in Tabelle 1 grau markierten Durchgänge sind Trainingsdurchgänge, bei denen über den gesamten Trainingsdurchgang kein eindeutiges Optimum in der Validation auszumachen war. Dies war bei den Netzen vom Typ Kat02 nicht zu beobachten.

Die Trainingsdauer variiert, ist aber mit 5 bis 6 Epochen in beiden Fällen vergleichbar. Die Neuronenzahl ist auch nicht eindeutig festzulegen. Zahlreiche

Trainingsläufe deuten allerdings darauf hin, dass Netze vom Typ Kat01 mit ungefähr drei Neuronen in der verdeckten Schicht, Netze vom Typ Kat02 hingegen mit zwei Neuronen pro verdeckter Schicht auskommen.

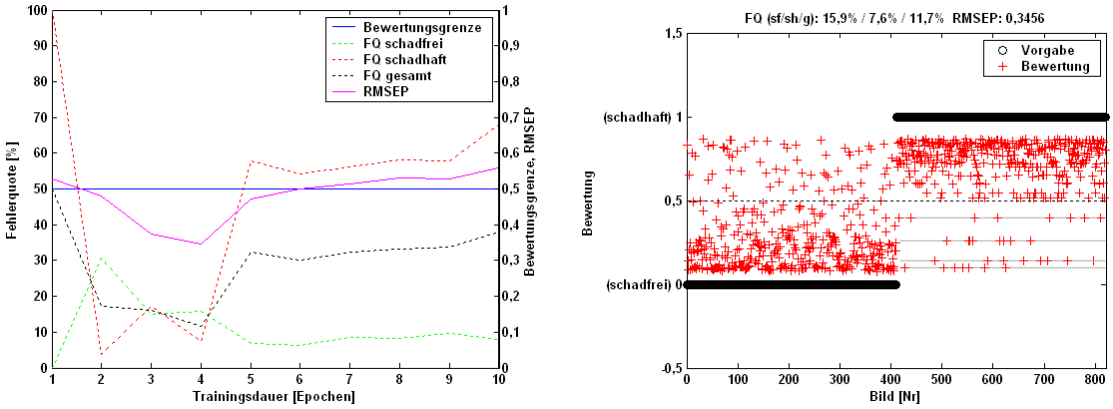
Die folgenden Abbildungen illustrieren die besten in dieser Phase der Arbeit erzielten Ergebnisse. Abbildung 46 zeigt das Trainingsverhalten und die entsprechende Vorhersage des besten Kat01 Netzes. Dieses Netz legt ein sehr ausgeprägtes Trainingsverhalten mit einem Optimum nach einer Trainingsdauer von 5 Epochen (Generationen) an den Tag. Auch in der Vorhersage ist das gute Ergebnis dieses Netzes deutlich zu erkennen.



a) b)

Abbildung 46: Bestes Ergebnis eines Netzes vom Typ Kat01 im Rahmen der Untersuchung zur Netzstruktur. a) Trainingsverlauf und b) Vorhersage des trainierten Netzes nach 5 Epochen

In Abbildung 47 sind die entsprechenden Ergebnisse des besten Kat02 Netzes dargestellt, welches unter Matlab trainiert wurde. Hier ist bei der Vorhersage deutlich zu erkennen, dass die schadhaften Bilder zur Aufstockung des Datensatzes vervielfältigt wurden. Die fehlerhaften Vorhersagen unter den schadhaften Bildern scheinen sich nicht statistisch zu verteilen, sondern auf einigen Niveaus zu häufen. Im Grunde handelt es sich hierbei nur um vier Bilder die verachtfacht wurden. Entsprechend sammeln sich die Fehlbeurteilungen, jeweils acht auf vier Niveaus.



a) b)

Abbildung 47: Bestes Ergebnis eines Netzes vom Typ Kat02 im Rahmen der Untersuchung zur Netzstruktur, a) Trainingsverlauf und b) Vorhersage des trainierten Netzes nach 4 Epochen

4.8 Parameteroptimierung

In Abbildung 48 sind die im Laufe der Optimierung durch einen Simplex-Algorithmus untersuchten Punkte dargestellt. Die Punkte, die am weitesten vom Zentrum entfernt sind, stammen aus den ersten vier Iterationsschritten. Da durch die erfolgten Reflektionen und Kontraktionen keine neuen besseren Punkte entstehen – der markierte Punkt aus der dritten Iteration hat keinen Bestand, da er durch die folgende Expansion ersetzt wird – fällt der Simplex schnell in sich zusammen. Im zehnten Iterationsschritt wurde die beste Parameterkombination dieser Optimierung gefunden. Läge dort tatsächlich ein vom Simplex ermitteltes Minimum, müsste der Simplex durch eine Folge von negativen Kontraktionen zunehmend schrumpfen. Die Ergebnisse der neuronalen Netze sind allerdings zu schwankend. Daher oszilliert der Simplex auf kleinem Raum durch eine endlose Folge von positiven und negativen Kontraktionen. Die gefundene Parameterkombination ist sicherlich eine gute Parameterkombination. Allerdings kann sie nur bedingt als Ergebnis einer erfolgreichen Optimierung betrachtet werden.

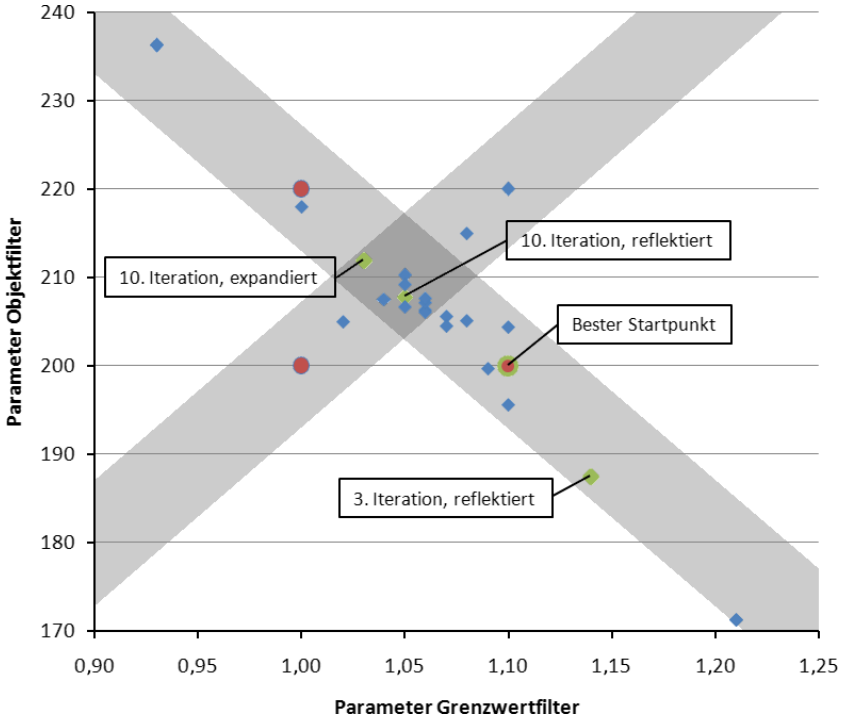


Abbildung 48: Verlauf der Simplex-Optimierung. Rot: Start-Simplex, grün: jeweils bester Punkt bis zu einer bestimmten Iteration.

Dazu bedürfte es reproduzierbarer Ergebnisse. Die systematische Parameteroptimierung ist demnach zu diesem Zeitpunkt noch nicht sinnvoll einsetzbar.

4.9 Die Delphi-Programme

4.9.1 Preprocess

Der erste Teil des Mustererkennungsprozesses, der in Delphi übertragen wurde, ist die Bildverarbeitung. Das entsprechende Programm wurde *Preprocess* genannt. Das Programm hat eine grafische Benutzeroberfläche, die es ermöglicht, Bilder zu laden und verschiedene Filter zur Segmentierung des Bildes anzuwenden. Abbildung 49 zeigt die Benutzeroberfläche des Programms.

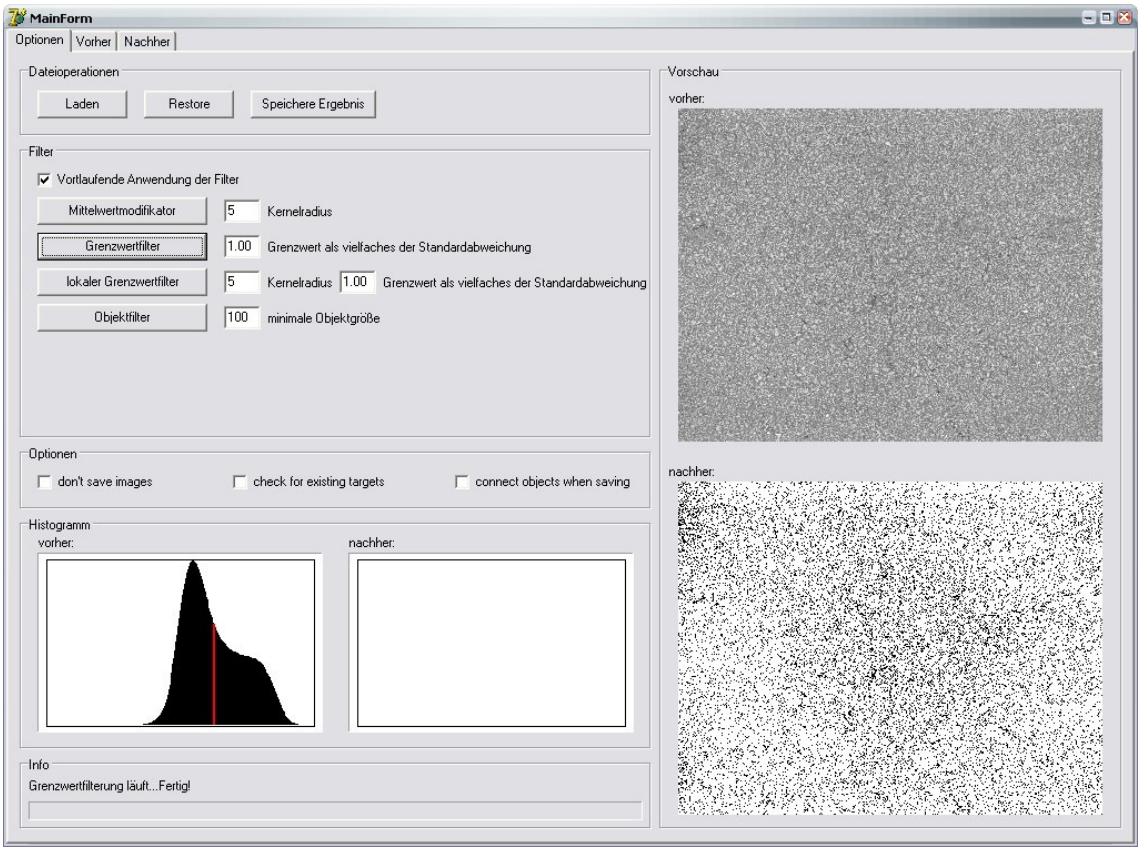


Abbildung 49: Interface des Delphi-Programms *Preprocess* für die grafische Vorverarbeitung der Bilder mit verschiedenen Filtern

Auf der ersten Seite (Register *Optionen*) bietet es grundlegende Dateioperationen, um ein Bild zu laden, es nach verschiedenen Filterversuchen wiederherzustellen und das Ergebnis zur späteren Begutachtung durch externe Programme zu speichern. Es stehen vier Filter zur Verfügung, die fortlaufend oder immer auf das Originalbild angewandt werden können. Die Parameter für die einzelnen Filter können frei gewählt werden. Außerdem werden sowohl eine Vorschau der Bilder als auch ihre Histogramme vor und nach Anwendung der Filter gezeigt. Für eine detailliertere Ansicht bietet das Programm unter den Registern *Vorher* und *Nachher* große Darstellungen des Bildes vor und nach der Anwendung des letzten Filters.

Mit diesem Programm kann die Arbeitsweise, beziehungsweise das Ergebnis verschiedener Filterkombinationen visualisiert werden. Auf diese Art können

empirisch Filter und Parameterkombinationen ermittelt werden, die geeignet scheinen, die gewünschten Merkmale zu isolieren.

Wurde eine bestimmte Parameterkombination festgelegt, kann das Programm auch mit Kommandozeilen-Parametern gestartet werden, um einen kompletten Datensatz in der gewünschten Art und Weise zu verarbeiten. Als Ergebnis wird für jedes Bild eine Pixelliste gespeichert, die die gefundenen Objekte beschreibt. Kenngrößen werden hier nicht bestimmt. Dies geschieht im aufrufenden Matlab-Programm.

4.9.2 Multithreadfilter

Der Multithreadfilter wurde ohne grafische Benutzeroberfläche, ausschließlich für die automatisierte Verarbeitung von Bildern geschrieben. Abbildung 50 zeigt den Multithreadfilter nach der Verarbeitung eines Test-Datensatzes von 10 Bildern. Er beherrscht neben der Bildverarbeitung auch die Kenngrößenextraktion. Die Bildverarbeitung beinhaltet die Dilatation, den Grenzwertfilter und den Objektfilter. Wahlweise kann über einen Kommandozeilenparameter die Verknüpfung naheliegender Objekte an oder ausgeschaltet werden.

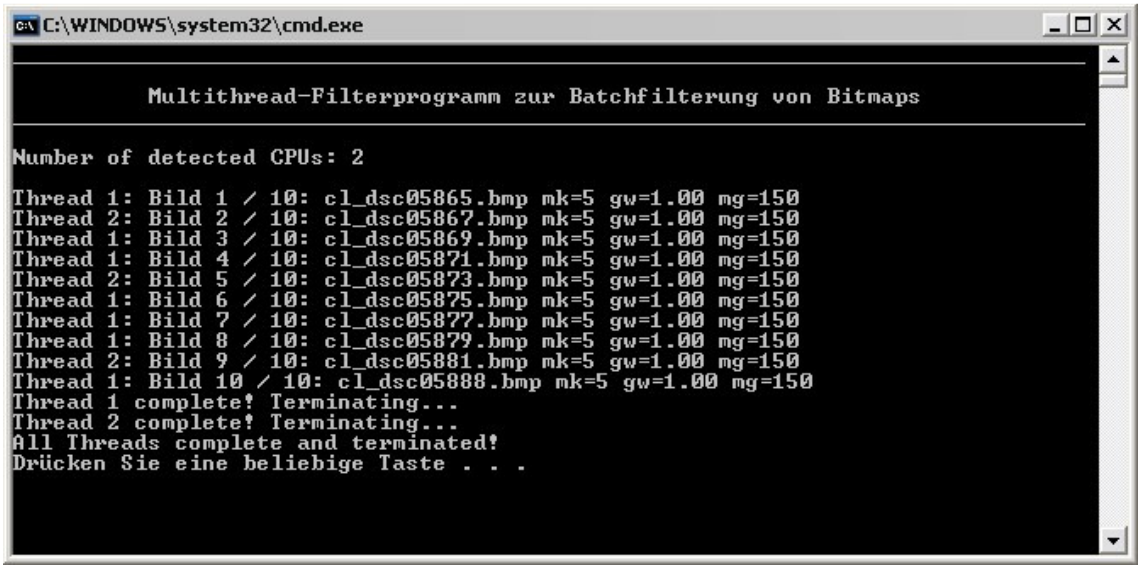


Abbildung 50: Konsolenausgabe des *Multithread-Filters* nach der Verarbeitung eines Test-Datensatzes von 10 Bildern

Es wird für jedes Bild eine Ergebnisdatei gespeichert. Entsprechend der gewählten Parameter wird eine Ordnerstruktur angelegt, und auch die Ergebnisdateien werden systematisch unter Benutzung der verwendeten Parameter benannt. Für folgende Filterung:

- Mittelwertbasierter Filter mit einem Radius von 5 (mk=5)
- Grenzwertfilter mit 1-facher Standardabweichung (gw=1.00)
- Objektfilter mit einer Mindestgröße von 150 (mg=150)

werden die Bilder im Unterverzeichnis:

.\mk5\gw1.00\mg150

gespeichert, wobei die Ergebnisdateien systematisch folgendermaßen benannt werden:

mg150_gw1.00_mk5_<ursprünglicher Bildname>.xxx

Die Ergebnisse können sowohl für die Weiterverarbeitung in Matlab (.mat) als auch in Delphi (.bin) gespeichert werden.

Um die volle Kapazität moderner Rechner zu unterstützen, wird die Arbeit, entsprechend der Anzahl vom System ausgewiesener CPUs, auf mehrere Threads verteilt.

4.9.3 Roadcompiler

Für die weitere Benutzung der Bilder unter Delphi (mit der *NNToolbox*) können die einzelnen Ergebnisdateien der Bilder in einer Datei zusammengefasst werden. Das Dateiformat speichert hierbei die aus den Dateinamen hervorgehenden Parameter der Bildverarbeitung. Der Datensatz kann über das Feld Kommentar (siehe Abbildung 51) mit weiteren Informationen versehen werden, z.B. über weitere Veränderungen, die am Bildmaterial vorgenommen wurden. Da aus der Namensgebung der Ergebnisdateien nicht hervorgeht, ob naheliegende Objekte verknüpft wurden, muss dies in diesem Programm noch angegeben werden. Daher wird das Kompilieren des Datensatzes mit den Knöpfen *not connected* oder *connected* gestartet.

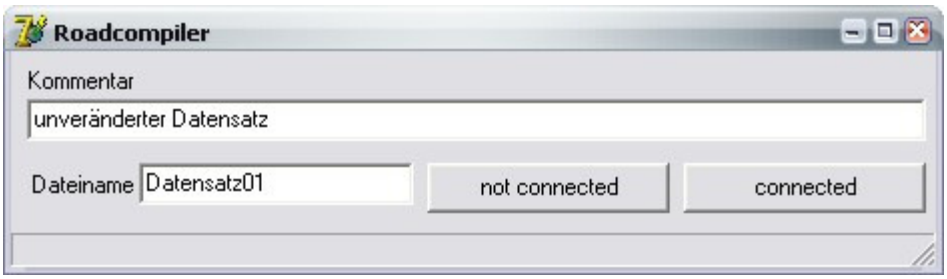


Abbildung 51: Benutzeroberfläche von *Roadcompiler*

4.9.4 NNToolbox

Mit der *NNToolbox* können feedforward Netze mit einer verdeckten Schicht und einer variablen Anzahl von Neuronen unter Verwendung von zwei verschiedenen Transferfunktionen (*tansig* oder *purelin*) trainiert werden. Das Training erfolgt mittels Backpropagation. Abbildung 52 zeigt das Hauptfenster des Programms.

Es können zuvor mit *Roadcompiler* kompilierte Datensätze geladen werden. Solche Datensätze, die im Programmverzeichnis vorliegen müssen, werden in einem Pulldown-Menü zur Auswahl angeboten. Anschließend kann der Datensatz nach Wunsch in bis zu drei Datensätze unterteilt werden: Kalibrations-, Validations- und Testdatensatz.

Wurden die Datensätze generiert, kann ein feedforward Netz trainiert werden. Das Programm unterstützt Netze mit einer verdeckten Schicht mit einstellbarer Anzahl an Neuronen. Für die verdeckte Schicht und für die Ausgabeschicht kann separat zwischen der *tansig* und der *purelin* Transferfunktion gewählt werden. Außerdem kann eine Lernrate angegeben werden. Es kann jederzeit ein neues Netz initialisiert werden oder das bestehende Netz trainiert werden.

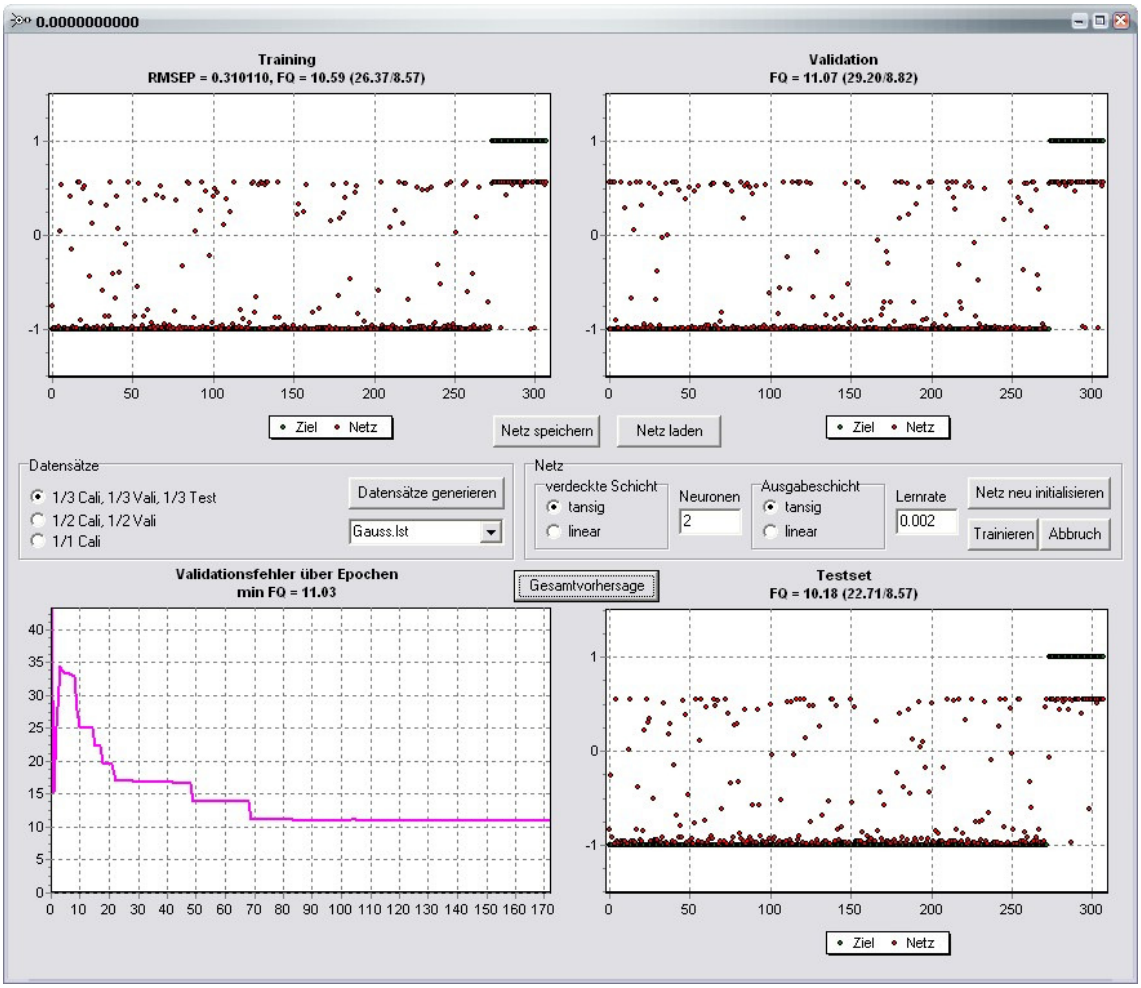


Abbildung 52: Interface des Delphi-Programms *NNToolbox* für das Training von feedforward backpropagation Netzen

Das Programm bietet vier Graphen in denen der Fortschritt des Trainings und die aktuellen Fehler beobachtet werden können. Der erste Graph (oben links) zeigt den Trainingsdatensatz. Sofern vorhanden zeigt der zweite Graph (oben rechts) die Netzvorhersage für den unabhängigen Validationsdatensatz. Er dient als Abbruchkriterium für das Training. Der Fehler in der Validation wird im dritten Graphen (unten links) dargestellt. Das Programm bricht im Minimum das Training nicht ab, speichert aber das neuronale Netz zu diesem Zeitpunkt automatisch ab. Die Vorhersage dieses Netzes ist es auch, die im vierten Graphen (unten rechts) dargestellt wird, sofern ein Testdatensatz generiert wurde.

Unabhängig von den automatisch gespeicherten Netzen sind rudimentäre Funktionen zum Speichern und Laden von Netzen vorhanden. Über den Knopf *Netz speichern* kann das neuronale Netz zu einem beliebigen Zeitpunkt gespeichert werden. Über den Knopf *Netz laden* kann ein neuronales Netz wieder geladen werden, um es zum Beispiel auf einen neuen Datensatz anzuwenden. Letztlich besteht über den Knopf *Gesamtvorhersage* noch die Möglichkeit, sich zu jedem Zeitpunkt eine Vorhersage des vollständigen Datensatzes anzeigen zu lassen (Abbildung 53).

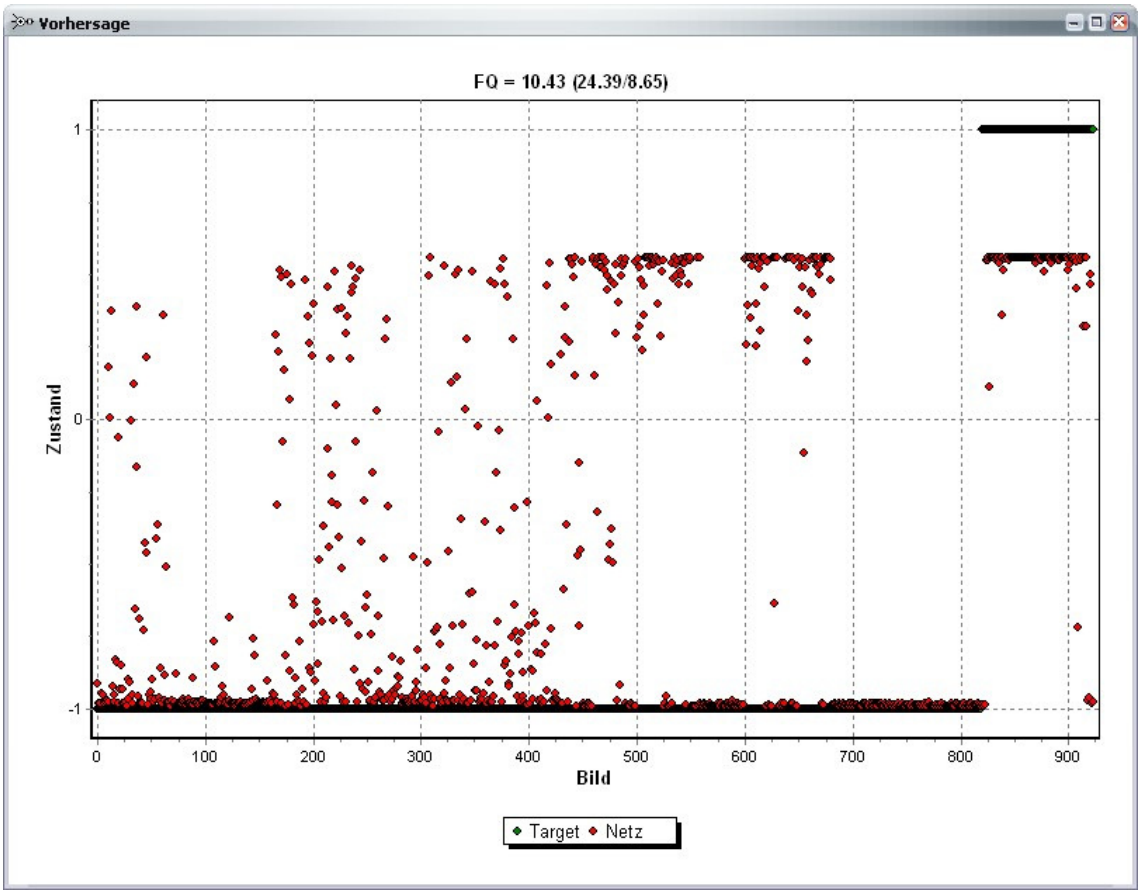


Abbildung 53: Vorhersage eines kompletten Datensatzes durch die *NNToolbox*

Das implementierte Backpropagation Training hat nicht die Komplexität der in Matlab zur Verfügung stehenden Algorithmen. Dennoch wurde zur Verbesserung

des Trainings eine variable Lernrate eingeführt. Diese wird in Abbildung 54 veranschaulicht. Gesucht ist hierbei ein Minimum auf den grauen Kurven.

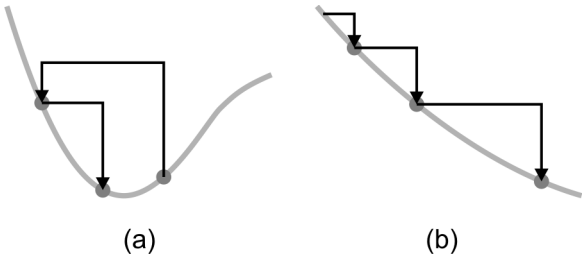


Abbildung 54: Unterschiedliche Schrittweite beim backpropagation Training durch eine variable Lernrate

Führt ein Trainingsschritt zu einer Verschlechterung (a) so wird die Lernrate halbiert. Das Training wird dadurch gebremst, aber es kann verhindert werden, dass der Algorithmus fortlaufend von einer Flanke des Minimums zur anderen springt. Stattdessen kann sich der Algorithmus mit zunehmend kleineren Schritten dem Minimum nähern. Auf der anderen Seite wird bei Verbesserungen (b) die Lernrate um 10 % erhöht. Auf diese Art ist der Algorithmus bei Schritten in die richtige Richtung in der Lage, die Lerngeschwindigkeit zu erhöhen.

4.10 Güte der Vorhersagen

4.10.1 Ergebnisse unter Matlab

In Kapitel 4.7.3 wurden die besten Ergebnisse der neuronalen Netzen vom Typ Kat01 und Kat02 aus mehreren Trainingsdurchläufen gezeigt. Ihre Fehler lagen für Kat01 Netze bei 13 % (Bewertung schadhafter Bilder) und 15 % (Bewertung schadfreier Bilder) und für Kat02 Netze bei 14 % respektive 16 %. In einzelnen Fällen ging die Fehlerquote bei der Bewertung schadhafter Bilder bis auf 5,6 % zurück. Liegt der Fehler bei den schadhaften Bildern allerdings sehr niedrig, so fällt der Fehler bei der Bewertung schadfreier Bildern mit 19 % entsprechend

höher aus. Außerdem muss bedacht werden, dass dort jeweils nur die besten Ergebnisse eines Trainingsdurchgangs berücksichtigt wurden. Die Netze, welche die jeweiligen Ergebnisse erbracht haben, basieren dabei nur auf einem Teil des Bildmaterials.

4.10.2 Ergebnisse unter Delphi

In der letzten Phase wurden neuronale Netze in Delphi implementiert. In dieser Implementierung stand, neben der unmittelbaren Kontrolle der Fehlerbewertung beim Training, eine bessere Visualisierung zur Verfügung. Ist ein Netz trainiert, kann eine Bewertung des gesamten Datenmaterials angezeigt werden. Abbildung 55 zeigt ein solches Ergebnis.

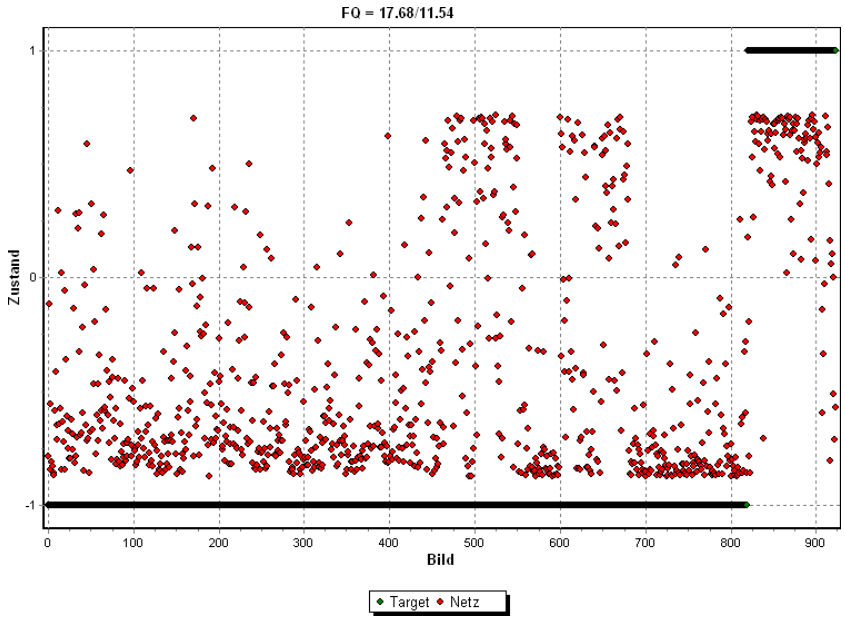


Abbildung 55: Bewertung des gesamten Datenmaterials (924 Bilder) durch ein trainiertes Netzes mit 2 Neuronen und einer linearen Ausgabeschicht.

Trainiert wurde hier ein Netz mit zwei Neuronen in der verdeckten Schicht. Die verdeckte Schicht arbeitet mit einer *tansig*-Funktion, während in der Ausgabeschicht eine *purelin*-Funktion zum Einsatz kommt. Die Fehlerquote bei der Beurteilung schadfreier Bilder liegt bei 17,7 %, bei der Beurteilung schadhafter

Bilder bei 11,5 %. Bedingt durch die unterschiedliche Anzahl an Bildern ergibt sich für den gesamten Datensatz eine Fehlerquote von 17,0 %.

Neben den Fehlerquoten ist in dieser Darstellung aber auch deutlich zu erkennen, dass bei der Beurteilung der schadfreien Bilder eine Häufung der Fehler im Bereich der Bilder 450 bis 550 und 600 bis 700 zu finden ist. Dieses Phänomen ist in unterschiedlicher Ausprägung bei praktisch allen Netzen zu beobachten. Es muss sich also um Bilder handeln, die sich in ihrer Charakteristik grundlegend von dem restlichen Datenmaterial unterscheiden. Dies erklärt sowohl die inkonsistenten Ergebnisse als auch vereinzelte außerordentlich gute Ergebnisse, die unter Matlab bei der Verwendung der reduzierten Datensätzen zu beobachten waren. Werden die Datensätz derart zusammengestellt, dass die problematischen Bilder weder in der Kalibration, noch in der Validation zum Einsatz kommen, werden sehr gute Ergebnisse erzielt. Kommen die problematischen Bilder in beiden Datensätzen vor, ist noch mit guten Ergebnissen zu rechnen. Allerdings versagt das Training nahezu vollständig, wenn die besagten Bilder nur in einem Datensatz auftauchen und somit kein repräsentatives Training möglich ist.

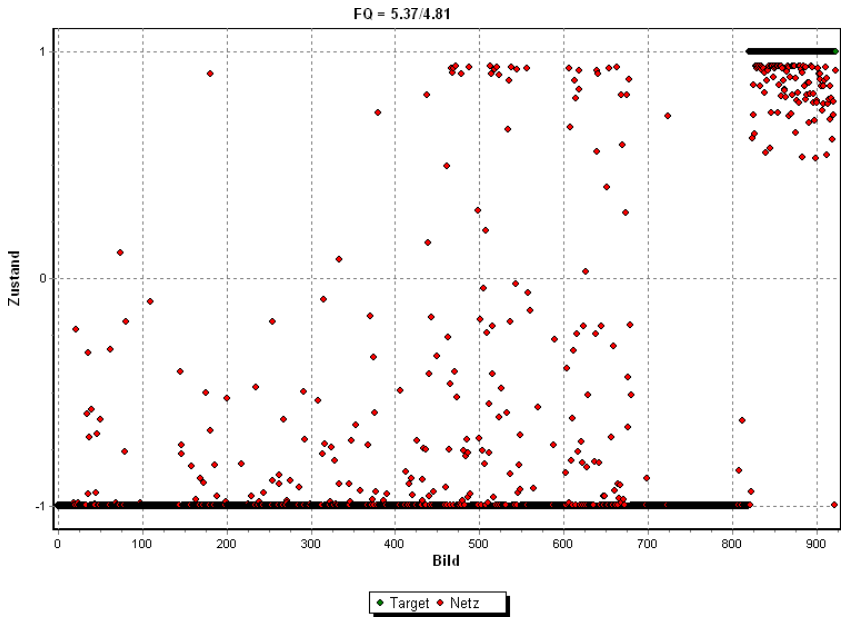


Abbildung 56: Bewertung des gesamten Datenmaterials (924 Bilder) durch ein Netz, welches mit dem gesamten Bildmaterial trainiert wurde.

Die Probleme, die diese Bilder bereiten, werden auch durch Abbildung 56 deutlich gezeigt. In diesem Fall wurde ein Netz mit dem vollständigen Datensatz trainiert. Die Fehlerquoten sind mit 5,4 % bei den schadfreien Bildern und 4,8 % bei den schadhaften Bildern entsprechend niedrig. Dennoch fällt auch hier die Häufung falscher Bewertungen im bereits erwähnten Bereich auf. Wodurch sich diese Bilder unterscheiden konnte ohne eingehende Untersuchung nicht geklärt werden.

In Abbildung 57 ist schließlich noch ein exemplarisches Ergebnis eines Netzes dargestellt, welches in seiner Ausgabeschicht eine Tansig-Funktion benutzt. Gemessen an den Fehlerquoten liefern solche Netze vergleichbare Ergebnisse. In diesem Fall liegt die Fehlerquote bei der Beurteilung schadfreier Bilder bei 17,1 %, bei der Beurteilung schadhafter Bilder bei 10,6 %. Bedingt durch die unterschiedliche Anzahl an Bildern ergibt sich in diesem Fall eine Gesamtfehlerquote von 16,3 %. Allerdings sind die Bewertungen weniger über den gesamten Wertebereich verteilt. Man sieht eine erheblich klarere Differenzierung. Über eine korrekte Klassifizierung hinaus entspricht diese Bewertung besser den tatsächlichen Vorgaben von –1 und 1.

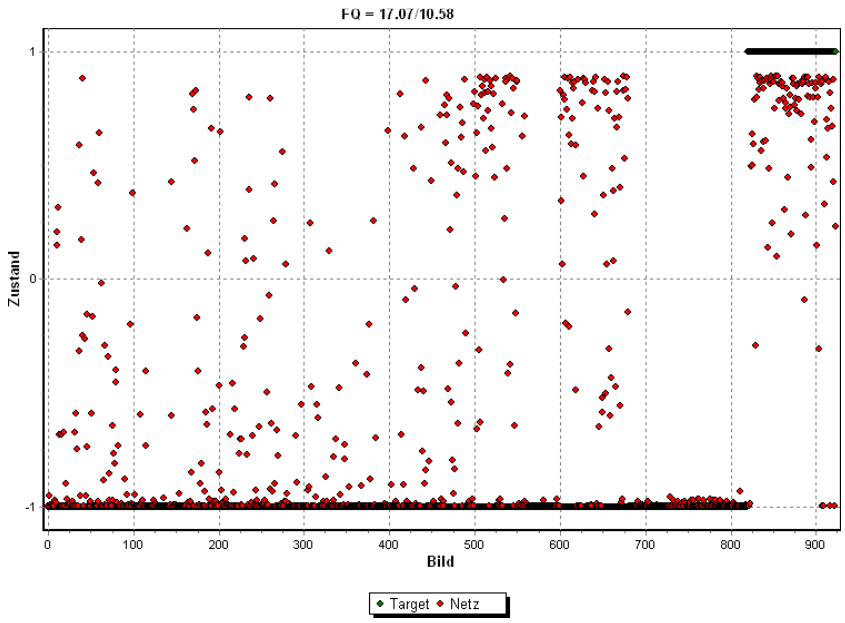


Abbildung 57: Differenzierte Bewertung des gesamten Datenmaterials (924 Bilder) durch ein trainiertes Netz mit 2 Neuronen und einer *tansig*-Ausgabeschicht.

Dies stellt eine deutliche Verbesserung bei der absoluten Beurteilung der Bilder dar. Der RMSEP lag beim Training unter Matlab, wo nur lineare Ausgabeschichten zum Einsatz kamen, im Mittel bei 0,34. Der RMSEP des Netzes, dessen Ergebnis in Abbildung 57 dargestellt ist, liegt hingegen knapp unter 0,2.

4.10.3 Auswertung eines 100 m-Abschnitts

Bei der Bewertung von Straßen wird nicht zwangsläufig jeder einzelne Quadratmeter bewertet. Die Bewertung der Straße erfolgt vielmehr Abschnittsweise. Dies hat zur Folge, dass fehlerhafte Bewertungen des neuronalen Netzes, die ja sowohl bei schadfreien als auch bei schadhaften Bildern auftreten, sich in gewissem Maße kompensieren können. Eine solche Bewertung einer Straße, bzw. eines 100 m-Abschnittes kann simuliert werden. Zugrundegelegt wurde ein neuronales Netz, welches bei der Beurteilung schadfreier Bilder eine Fehlerquote von 14,1 % und bei schadhaften Bildern eine Fehlerquote von 9,7 % aufweist.

Abbildung 58 zeigt die simulierte Bewertung eines 100 m-Abschnittes durch ein neuronales Netz.

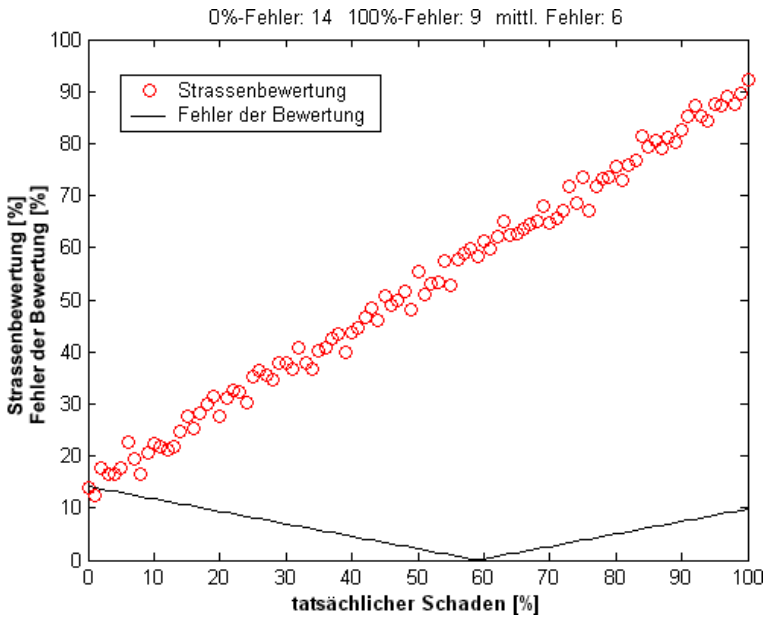


Abbildung 58: Bewertung eines 100m-Abschnitts

Dargestellt ist dabei die Bewertung des neuronalen Netzes (y-Achse) gegen den tatsächlichen Schaden auf dem 100 m-Abschnitt (x-Achse). Darüber hinaus wird der Fehler dargestellt, den das Netz bei der Bewertung des 100 m-Abschnittes macht. Bei einem völlig intakten Abschnitt entspricht der Fehler des Netzes statistisch dem Fehler bei der Beurteilung schadfreier Bilder, 14,1 %. Bei einem zu 100 % schadhaften Abschnitt hingegen entspricht der Fehler des Netzes dem Fehler bei der Beurteilung schadhafter Bilder, 9,7 %. Da der Fehler der Netzvorhersage für einen Straßenabschnitt einer Linearkombination der Einzelfehler (schadfrei/schadhaft) entspricht ist zur Mitte hin ein linearer Abfall zu sehen. Bei Straßenabschnitten mit einer durchschnittlichen Beschädigung von 59 % schließlich kompensieren sich die Fehler bei der Beurteilung schadhafter und schadfreier Bilder vollständig und der Netzfehler sinkt auf 0. Das Verhältnis 59 %:41 % entspricht dem Verhältnis der Einzelfehler bei der Beurteilung schadfreier und schadhafter Bilder (14,1 %:9,7 %). Der mittlere Fehler des neuronalen Netzes bei der Beurteilung aller möglichen 100 m-Abschnitte beträgt 6,1 % und liegt durch die sich kompensierenden Fehler unter den einzelnen Fehlerquoten für schadfreie bzw. schadhafte Bilder. Dies stellt allerding den günstigsten Fall dar. Im ungünstigsten Fall können sich die Einzelfehlerauch addieren.

4.10.4 Fazit

Steht ein ausgeglichener und vor allem repräsentativer Datensatz für das Training zur Verfügung, können Fehlerquoten von 15-20 % erreicht werden. Sobald das Netz trainiert ist, können diese Fehlerquoten im Sinne der Anwendung optimiert werden. Da für die Früherkennung von Schäden wichtig ist, dass diese rechtzeitig gefunden werden, kann die Grenze zwischen schadhaften und schadfreien Bildern verschoben werden. Dies erhöht zwar die Fehlerquote bei der Beurteilung schadfreier Oberflächen, gewährleistet aber auf der anderen Seite, dass Schäden zuverlässiger gefunden werden. Berechnungen haben darüber hinaus gezeigt, dass sich die Bewertungsfehler bei der Beurteilung größerer Abschnitte auch

kompensieren. Damit wurde im statistischen Mittel eine Fehlerquote von 6 % erreicht.

Eine Beurteilung von Asphaltoberflächen auf der Basis digitaler Bildverarbeitung und unter Einsatz neuronaler Netze möglich ist somit möglich.

5 Zusammenfassung

In dieser Arbeit konnte gezeigt werden, dass eine Bewertung von Fahrbahnoberflächen, insbesondere Asphaltoberflächen, auf der Basis digitaler Bildverarbeitung und mit dem Einsatz neuronaler Netze möglich ist. Zu diesem Zweck wurde ein Mustererkennungsprozess aufgebaut, der, ausgehend von den Aufnahmen einer handelsüblichen Digitalkamera, in mehreren Arbeitsschritten eine Bewertung des Bildmaterials und eine Klassifizierung als schadfrei bzw. schadhaft vornimmt.

Der in Matlab und Delphi implementierte Mustererkennungsprozess beginnt damit, dass das Bildmaterial bearbeitet wird (Abbildung 59). Erforderlich ist hierbei die Auswahl eines geeigneten Bildausschnittes sowie die Korrektur der technisch bedingten ungleichmäßigen Ausleuchtung des Bildes.

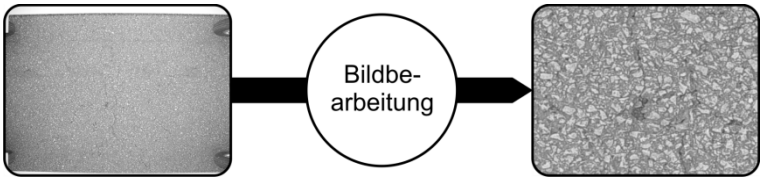


Abbildung 59: Auswahl eines Bildausschnittes und Korrektur der Bildausleuchtung

Der zweite Schritt ist die Bildverarbeitung (Abbildung 60). Hier wird das Bild durch die Anwendung verschiedener Filter segmentiert. Man erhält ein weißes Bild (Hintergrund) mit den potentiell interessante Strukturen (schwarz), die vom Hintergrund isoliert wurden und in der Folge untersucht werden sollen.

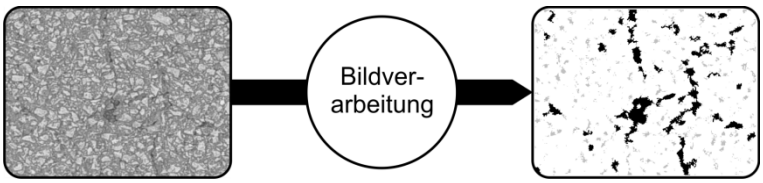


Abbildung 60: Segmentierung des Bildes durch die Anwendung verschiedener Filter

Auf der Basis der Vorverarbeitung erfolgt im dritten Schritt die Kenngrößenextraktion (Abbildung 61). Hier wird das Bild in einzelne

Bildabschnitte unterteilt, und es werden abschnittsweise aus dem segmentierten Bild charakteristische Kenngrößen berechnet.

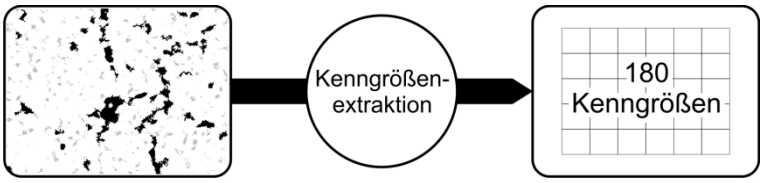


Abbildung 61: Extraktion von Kenngrößen aus den segmentierten Bildern zur Auswertung durch das neuronale Netz

Der letzte Schritt der Mustererkennung ist die Klassifizierung durch ein neuronales Netz (Abbildung 62). Zu diesem Zweck wurden verschiedene neuronale feedforward Netze mittels Backpropagation trainiert, um einen Zusammenhang zwischen den ermittelten Kenngrößen und dem Zustand des präsentierten Bildmaterials herzustellen. Man erhält hierbei für die Bilder eine Bewertung von schadfrei bis schadhaft.

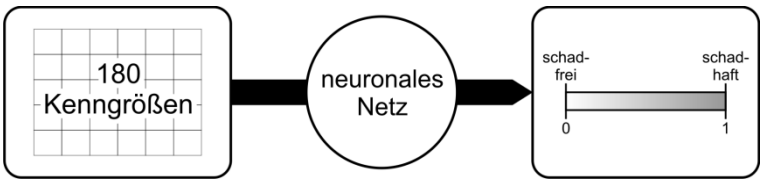


Abbildung 62: Bewertung des Bildmaterials durch das neuronale Netz und Klassifizierung als schadhaft oder schadfrei

Die Bewertung durch das neuronale Netz kann sehr unterschiedlich ausfallen. Je nach Bildmaterial und Zusammenstellung der Kalibrationsdatensätze werden mitunter schlechte, aber auch sehr gute Ergebnisse erzielt. Grundsätzlich ist die Segmentierung erfolgreich, und es können aussagekräftige Kenngrößen berechnet werden. Auch eine Klassifizierung auf der Basis von feedforward Netzen ist möglich. Es hat sich allerdings deutlich gezeigt, dass ein ausgeglichener und vor allem repräsentativer Datensatz für das erfolgreiche Training der Netze nötig ist. Ist dies gewährleistet, können Fehlerquoten von 15-20 % erreicht werden. Sobald das Netz trainiert ist, können diese Fehlerquoten im Sinne der Anwendung optimiert werden. Da für die Früherkennung wichtig ist, dass alle Schäden rechtzeitig gefunden werden, kann die Grenze zwischen schadhaften und

schadfreien Bildern verschoben werden. Dies erhöht zwar die Fehlerquote bei der Beurteilung schadfreier Oberflächen, gewährleistet aber auf der anderen Seite, dass Schäden zuverlässiger gefunden werden. Berechnungen haben darüber hinaus gezeigt, dass sich die Bewertungsfehler bei der Beurteilung größerer Abschnitte auch kompensieren. Damit wird eine Fehlerquote unter 10 % realistisch.

Insgesamt konnte gezeigt werden, dass eine Beurteilung von Asphaltoberflächen auf der Basis digitaler Bildverarbeitung und unter Einsatz neuronaler Netze möglich ist.

6 Ausblick

Um die Leistung des Mustererkennungsprozesses weiter zu verbessern, gibt es mehrere Ansatzpunkte an verschiedenen Punkten des Prozesses, von den Trainingsdaten bis hin zum neuronalen Netz.

Die Datenbasis

Die bisherigen Arbeiten basieren auf einem 250 m langen Straßenabschnitt. Dieser kann nicht repräsentativ für mehrere 10.000 km sein, insbesondere da bereits beim vorliegenden Bildmaterial zu erkennen ist, dass Asphaltoberflächen sehr unterschiedlich aussehen können. Und es bleibt zu klären, warum sich einige Bilder im Training so grundlegend vom restlichen Bildmaterial unterscheiden. Auch ist davon auszugehen, dass die Schadensmerkmale noch weit mehr variieren können, als das bisherige Bildmaterial zu zeigen vermag. Für ein fundierteres Training ist also umfangreicheres und repräsentativeres Bildmaterial notwendig.

Detailliertere Bewertung

Zum aktuellen Bildmaterial liegt ausschließlich eine Bewertung in den Kategorien schadfrei und schadhaft vor, wobei Bilder erst als schadhaft bewertet werden, wenn sie einen aus Straßenbausicht relevanten Schaden aufweisen. Es sollte untersucht werden, ob nicht eine detailliertere Bewertung mit verschiedenen Einstufungen zwischen 0 % (absolut schadfrei) bis 100 % (stark beschädigt) das Training verbessern könnte. Der Lernprozess sollte davon profitieren, wenn Bilder, die bereits erkennbare Beschädigungen aufweisen, auch eine entsprechende Bewertung erhielten.

Bildverarbeitung

Die Segmentierung der Bilder basiert momentan auf grundlegenden Methoden der digitalen Bildverarbeitung. Hier liegt noch deutliches Potential zur Verbesserung. Als Grundlage zur korrekteren Erfassung der Risse kann die Vektorisierung überarbeitet werden, basierend auf einer Skelettierung, damit sie effizient einsetzbar wird. Sollte dieser Mustererkennungsprozess auf Asphaltoberflächen unterschiedlicher Beschaffenheit angewendet werden, so müssen die Parameter

der Bildverarbeitung entsprechend angepasst werden. Idealerweise sollte das System selbst erkennen, mit welchen Oberflächen es konfrontiert wird. Hier wäre es denkbar, aus dem Histogramm der Bilder auf die Oberflächencharakteristik schließen zu können und den Mustererkennungsprozess entsprechend einzustellen.

Kenngrößenextraktion

Auch die Extraktion der Kenngrößen kann erweitert werden. So könnte abschnittsweise oder über das gesamte Bild eine Objektgrößenverteilung ermittelt werden. Wird eine Vektorisierung in den Prozess aufgenommen, erschließen sich weitere Merkmale, wie die tatsächliche Länge von Rissen oder das Maß an Verzweigung. Auch eine zielgerichtete Verknüpfung von potentiellen Rissabschnitten wird somit möglich.

Neuronale Netze

Auch bei den neuronalen Netzen sind noch Verbesserungen möglich. Es hat sich gezeigt, dass feedforward Netze mit relativ wenigen Neuronen für die Aufgabe geeignet sind. Die Art ihrer Anwendung könnte allerdings verbessert werden. In dieser Arbeit kamen zwei Netztopologien zum Einsatz. Im einen Fall wurden Kenngrößen des gesamten Bildes aus 30 Bildabschnitten gesammelt an eine einzelne Neuronenschicht übergeben. Dabei sollte ein Riss in der oberen linken Ecke völlig unabhängig von der Beschaffenheit der unteren rechten Ecke erkannt und beurteilt werden. Aus diesem Grund wurde das zweite neuronale Netz entworfen, in dem die 30 Bildabschnitte durch 30 Schichten zunächst unabhängig bewertet werden, bevor eine Gesamtwertung vorgenommen wird. Die Bewertung, die von den 30 einzelnen Schichten vorgenommen wird, ist im Prinzip aber die gleiche. Es sollte also vielmehr ein neuronales Netz sein, welches der Reihe nach alle Bildabschnitte bewertet, um aus der Summe dieser Wertungen auf den Zustand des Asphaltabschnittes zu schließen. Dies setzt allerdings eine entsprechend detaillierte Bewertung des Bildmaterials voraus.

Dass gute Ergebnisse erzielt werden können, konnte bereits gezeigt werden. Mit den angesprochenen Verbesserungen könnte dann die Zuverlässigkeit und die Flexibilität des Mustererkennungsprozesses nennenswert verbessert werden.

7 Literaturverzeichnis

1. **Duda, Richard O., Hart, Peter E. und Stork, David G.** *Pattern Classification.* 2. Edition. s.l. : John Wiley & Sons, 2001.

2. **Steinbrecher, Rainer.** *Bildverarbeitung in der Praxis.* s.l. : www.RST-Software.de, 2005.

3. Hypertext Image Processing Reference. [Online] [Zitat vom: 24. Mai 2007.] http://homepages.inf.ed.ac.uk/rbf/HIPR2/.

4. **Pratt, William K.** *Digital Image Processing.* s.l. : John Wiley & Sons, 1978.

5. *Image blockiness evaluation based on Sobel operator.* **Perra, C., Massidda, F. und Giusto, D.D.** International Conference on Image Processing : s.n., 2005. Bde. Volume 1, 11-14 Sept, S. 389-392.

6. **Aleksander, I. und Morton, H.** *An Introduction to neural computing.* 1. London : Capman and Hall, 1990.

7. **Zupan, Jure und Gateiger, Johann.** *Neural Networks for Chemists - An Introduction.* Weinheim : VCH, 1993.

8. **Zell, Andreas.** *Simulation neuronaler Netze.* 2. unveränderter Nachdruck. München : R. Oldenbourg Verlag, 1997.

9. **Cichocki, Andrzej und Unbehauen, Rolf.** *Neural Networks for Optimization and Signal Processing.* s.l. : Teubner, 1993.

10. **Stryer, L.** *Biochemie.* 4. Auflage. s.l. : Spektrum Akad. Verlag, 1996.

11. Nervous Tissue. [Online] [Zitat vom: 12. November 2006.] http://kentsimmons.uwinnipeg.ca/cm1504/15lab404/15lb4p7.htm.

12. **Demuth, Howard und Beale, Mark.** *Matlab Neural Network Toolbox Users Guide, Version 4.* 7. Edition. Natick, MA : The MathWorks, 2000.

13. **Ritter, Helge, Martinez, Thomas und Schulten, Klaus.** *Neuronale Netze - Eine Einführung in die Neuroinformatik selbstorganisierender Netzwerke.* s.l. : Addison-Wesley, 1990.

14. **Hebb, D.O.** *The Organization of Behavior.* New York : Wiley, 1949.

15. **Rumelhart, D.E. und McClelland, J.L.** *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Volume 1: Foundations.* s.l. : MIT Press, 1986.

16. **Marquardt, D.W.** An algorithm for least-squares estimation of nonlinear parameters. *Journal of the Society for Industrial and Applied Mathematics.* 1963, Bd. 11, 2, S. 431-441.

17. **Hagan, M.T. und Menhaj, M.** Training feed-forward networks with. *IEEE Transactions on Neural Networks.* 1999, Bd. 5, 6, S. 989–993.

18. **Hagan, M.T., Demuth, H.B. und Beale, M.H.** *Neural Network Design.* Boston, MA : PWS Publishing, 1996.

19. **Moller, M.F.** A scaled conjugate gradient algorithm for fast. *Neural Networks.* 1993, Bd. 6, S. 525–533.

20. *Gauss-Newton approximation to Bayesian regularization.* **Foresee, F.D. und Hagan, M.T.** Proceedings of the 1997 International Joint Conference on Neural Networks : s.n., 1997.

21. **MacKay, D.J.C.** Bayesian interpolation. *Neural Computation.* 1992, Bd. 4, 3, S. 415–447.

22. **Danzer, Klaus, et al.** *Chemometrik - Grundlagen und Anwendung.* Berlin Heidelberg : Springer-Verlag, 2001.

23. **Esbensen, Kim H.** *Multivariate Data Analysis - In Practice.* 4th Edition. s.l. : CAMO, 2000.

24. **Otto, Matthias.** *Chemometrie - Statistik und Computereinsatz in der Analytik.* Weinheim : VCH, 1997.

25. **Wold, H.** Estimation of principal components and related models by iterative least squares. *In Multivariate Analysis.* 1966, S. 391-420.

26. **H.Wold.** Path models with latent variables: The NIPALS approach. *In Quantitative Sociology: International perspectives on mathematical and statistical model building.* 1975, S. 307-357.

27. **Box, G.E.P. und Draper, N.R.** *Empirical Model-Building and Response Surfaces.* s.l. : Wiley, 1987.

28. **Box, G.E.P. und Wilson, K.B.** On the experimental attaintment of optimum conditions. *Journal of the Royal Statistical Society (Series B).* 1951, 13, S. 1-45.

29. **Goldberg, D.E.** *Genetic Algorithms in Search, Optimization, and.* s.l. : Addison-Wesley, 1989.

30. **Nelder, J.A. und Mead, R.** A Simplex-Method for functional minimization. *Computer Journal.* 1965, Bd. 7, 4, S. 308-313.

31. **Deming, S.N. und Morgan, S.L.** Teaching the fundamentals of experimental design. *Anal. Chim. Acta.* 1978, 150, S. 187.

32. **Gonzalez, Rafael C. und Woods, Richard E.** *Digital Image Processing.* s.l. : Addison-Wesley, 1992.

33. **Haralick, Robert M. und Shapiro, Linda G.** *Computer and Robot Vision, Volume I.* 1992 : Addison-Wesley.

34. **Stefanelli, R. und Rosenfeld, A.** Some Parallel Thinning Algorithms for Digital Pictures. *Journal of the ACM.* 1971, Bd. 18, 2, S. 255–264.

35. **Zhang, T.Y. und Suen, C.Y.** A Fast Parallel Algorithm for Thinning Digital Patterns. *Communications of the ACM,.* 1984, Bd. 27, 3, S. 236–239.

Abkürzungsverzeichnis

a ..... Ausgabewert eines Neurons

$\boldsymbol{a}$ ..... Ausgabevektor einer Schicht

A ..... Ausgabe bei Anwendung eines Konvolutionskerns

ANN ..... artificial neural network

b ..... bester Punkt eines Simplex

b ..... Bias eines Neurons

$\boldsymbol{b}$ ..... Biasvektor einer Schicht

$\boldsymbol{B}$ ..... Bild

c_+ ..... positiv kontrahierter Punkt

c_- ..... negativ kontrahierter Punkt

$d_{x/y}$ ..... Parameter der parabolischen Korrekturfunktion

e ..... expandierter Punkt des Simplex

$f(n)$ ..... Transferfunktion eines Neurons

$f_{x/y}$ ..... Parameter der parabolischen Korrekturfunktion

FQ ..... Fehlerquote

$G(x, y)$ ..... parabolisch Korrekturfunktion für die Helligkeitskorrektur

GA ..... genetischer Algorithmus

$\boldsymbol{IW}$ ..... Wichtungsmatrix der ersten Netzschicht

$\boldsymbol{K}$ ..... Konvolutionskern

KNN ..... künstliches neuronales Netz

$\boldsymbol{LW}$ ..... Wichtungsmatrix von Folgeschichten

$MaxIter$ ..... Maximale Anzahl von Iterationsschritten

n ..... zweitbester Punkt eines Simplex

n ..... Netinput eines Neurons

$\boldsymbol{n}$ ..... Netinput einer Schicht

NIPALS ..... nonlinear iterative partial least squares

p ..... Schwerpunkt des Simplex exklusive des schlechtesten Punktes

$\boldsymbol{p}$ ..... Eingabevektor des neuronalen Netzes

PCA ..... principal component analysis

PLS ..... partial least square regression

r ..... reflektierter Punkt eines Simplex

r ..... Radius

R ..... Anzahl der Elemente des Eingabevektors eines Neurons

$RMSEP$ ..... root mean square error of prediction

s ..... schlechtester Punkt eines Simplex

S ..... Anzahl Neuronen in einer Schicht

SVD ..... singular value decomposition / Singulärwertzerlegung

SSE ..... sum square error

t ..... Zielgröße (target)

$TolX$ ..... Minimaler Durchmesser des Simplex

$TolF$ ..... Minimale Veränderung der Antwortfunktion des Simplex

$\boldsymbol{u}$ ..... Score-Vektor

$\boldsymbol{v}$ ..... Loading-Vektor

$\boldsymbol{w}$ ..... Wichtungsvektor eines Neurons

$\boldsymbol{W}$ ..... Wichtungsmatrix einer Schicht

$\boldsymbol{X}$ ..... Datenmatrix

$\boldsymbol{x_i}$ ..... Spaltenvektor aus $\boldsymbol{X}$

Δw ..... Veränderung der Wichtungsfaktoren

α ..... Faktor für die Expansion des Simplex

β ..... Faktor für die positive Kontraktion des Simplex

γ ..... Faktor für die negative Kontraktion des Simplex

δ ..... gewichtete Fehler

η ..... Lernrate

Lebenslauf

Zur Person

Name	David Geissler
Geburtstag/-ort	15. August 1977 in Hannover
Familienstand	ledig, keine Kinder
Staatsangehörigkeit	deutsch (Vater: deutsch, Mutter: französisch)

Berufserfahrung

Seit 10/2007	Angestellter der BackUp GmbH, Gummersbach. Projekt: Softwareentwicklung für die Weatherford Oil Tool GmBH, Langenhagen.
08/2003 - 03/2007	Wissenschaftlicher Mitarbeiter am Institut für Technische Chemie, Universität Hannover. Tätigkeiten: Modellierung von Bioprozessen, statistische und multivariate Datenauswertung, Einsatz von Optimierungsmethoden (Simplex, GA,...), Modellierung mit neuronalen Netzen, digitale Bildverarbeitung, Programmentwicklung zur automatisierten Datenauswertung

Bildungsweg

06/2003 - 05/2008	Promotion Dissertation: „Erfassung und Bewertung von Substanzmerkmalen von Fahrbahnoberflächen mittels digitaler Bildverarbeitung und neuronaler Netze“ Note: sehr gut

09/1997 - 04/2003	Chemiestudium Diplomarbeit: „Auswertung von 2D-Fluoreszenzspektren aus Bioprozessen ohne Kalibrationsmessungen“ Note: sehr gut
05/1996	Abitur am Kurt-Schwitters-Gymnasium in Hannover

Zusatzqualifikationen

01/2004	Marketingseminar
06/2001	Qualitätssicherung in der chemischen Produktion
12/2000	Betriebsbeauftragte für Gewässerschutz - Fachkunde i. S. d. § 21 c Abs. 1 WHG
01/2000	Seminar Betrieblicher Umweltschutz

Wehrdienst

10/1996 - 08/1997	Wehrdienst beim LTG62 in Wunstorf

Sprachkenntnisse

Deutsch	Muttersprache
Französisch	Muttersprache
Englisch	Verhandlungssicher

EDV-Kenntnisse

Betriebssysteme	Windows
Textverarbeitung	MS Word / LaTeX
Tabellenkalkulation	MS Excel
Präsentation	MS Power Point

Grafik/Bildbearbeitung	CorelDRAW Graphics Suite
Datenanalyse	Origin / The Unscrambler
Programmiersprachen	C (Grundkenntnisse) / C# (Grundkenntnisse) / Delphi / Matlab / Simulink / Maple

Veröffentlichungen

D. Geissler, R. C. Ribeiro, H. H. Meyer und B. Hitzmann: *Chemometric Modelling based on 2D-Fluorescence Spectra without a Calibration Measurement.* BioPerspectives 2005, Wiesbaden (Poster).

D. Solle, D. Geissler, E. Stärk, T. Scheper und B. Hitzmann: *Chemometric Modelling based on 2D-Fluorescence Spectra without a Calibration Measurement.* Bioinformatics, 19(2):173-177, 2003.

D. Geissler, D. Solle, E. Stärk, T. Scheper, H. Märkl und B. Hitzmann: *A New Evaluation Method for 2-D Fluorescence Spectra Based On Theoretical Modeling.* Engineering in Life Sciences, 3(10): 397-400, 2003.

D. Geissler, D. Solle, E. Stärk, T. Scheper, H. Märkl und B. Hitzmann: *Berechnung chemometrischer Modelle basierend auf theoretischen Modellen von 2D-Fluoreszenzspektren.* Chemie Ingenieur Technik, S. 1167-1170, August 2002.