Analyse von Bildresiduen mit Machine Learning im Rahmen von Kamerakalibrierungen[1]

Image Residual Analysis with Machine Learning within the Context of Camera Calibrations

Waldemar Kisser, Frank Boochs, Dietrich Paulus

Photogrammetrie ermöglicht es, Objekte mithilfe von Digitalbildern zu vermessen. Bei optimalen Messbedingungen sind Qualitätsunterschiede der abgeleiteten Maße vor allem auf die mathematische Modellierung des verwendeten Sensors und des Objektivs zurückzuführen. Photogrammetrische Kalibrierungen erfolgen meist mittels Bündelblockausgleichung. Diese gestattet es, vielerlei statistische Kennzahlen abzuleiten. Eine tiefer gehende Analyse der berechneten Parameter, Standardabweichungen, Korrelationen und deren Verteilungen kann Aufschluss darüber geben, ob das verwendete Kalibriermodell Schwächen aufweist. Solche Defizite können sich durch systematische Restfehler im Bild- oder Objektraum äußern. Da solche Restfehler auch zu Ungenauigkeiten in den daraus abgeleiteten Informationen führen können, ist deren mathematischer Nachweis und anschließende Kompensation zur Erzielung höchster Genauigkeiten unausweichlich. Neueste Ansätze nutzen Korrekturterme, um Residuensystematiken schon während der Bündelblockausgleichung zu modellieren. Dieser Beitrag beschreibt, wie auch Machine-Learning-Techniken dabei helfen können, verbliebene systematische Abweichungen in Bildresiduen nachzuweisen, ohne dass hierzu ein Eingriff in die Bündelblockausgleichung notwendig ist. Dies wird im ersten Schritt anhand von Beispieldaten erläutert. Im zweiten Schritt wird die Wirkung dieser Vorgehensweise an einer realen Kamerakalibrierung verdeutlicht. Abschließend erfolgt eine Diskussion der im Zuge dieser Arbeit erzielten Resultate und möglicher Eignung dieses Verfahrens in der Praxis.

Schlüsselwörter: Machine Learning, Residuenanalyse, Photogrammetrie

Photogrammetry uses digital images to extract geometric object information. In case of optimal measurement conditions the result quality is determined by the mathematical model, sensor and lens configurations. Bundle adjustment is a wide used approach in order to carry out photogrammetric tasks as well as camera and lens calibrations. Additionally, it yields various statistic parameters. Further standard deviations, correlation coefficients and distribution analysis helps users to disclose insufficient mathematical modelling. These deficiencies can manifest themselves through residuals in the image or object space. Since they also lower the quality of all derived information, one should carefully track and compensate these remaining errors in order to achieve highest accuracies. Most recent approaches model such errors "on-the-fly" within the bundle adjustment by adding or altering parameters. This paper describes how machine learning techniques can be used to track systematic residual patterns without changes in the bundle adjustment source code. Initially the idea is described with tiny examples. The second step applies this technique to a real camera calibration scenario. Finally, the achieved results as well as the suitability and practicability of such methods are discussed.

Keywords: *Machine learning, residual analysis, photogrammetry*

[1] Überarbeiteter und erweiterter Beitrag der Oldenburger 3D-Tage 2016

1 MOTIVATION

Fortschreitende Technik sorgt dafür, dass Messinstrumente jeglicher Art immer größere Datenmengen an den Nutzer liefern, zum Beispiel die zunehmende Auflösung bei Kamerasensoren oder zunehmend dichter werdende Punktwolken bei Laserscannern. Manuelle oder semiautomatisierte Auswertung, die früher mit wenigen Klicks und Kontrollen erledigt werden konnte, ist aus diesem Grund mit steigendem Aufwand verbunden. Deshalb sind Industrie und Forschung stets bestrebt, Verfahren und Algorithmen zu finden, um zeitaufwendige Auswerteaufgaben vollständig zu automatisieren. Um einen übersichtlichen Einstieg in das hier vorgestellte Verfahren zu gewährleisten, werden zunächst einzelne Machine-Learning-Begriffe erläutert. Hierbei wird deren Bedeutung hauptsächlich im Kontext der praktischen Auswertung betrachtet.

2 MACHINE LEARNING

Machine Learning (ML) ist eine junge Wissenschaft, die ähnlich wie die Ausgleichungsrechnung auf Erkenntnissen aus der Statistik, Stochastik, Numerik und der mathematischen Optimierung basiert. Trotz der vergleichsweise kurzen Periode ist es gelungen, sich derart zu etablieren, dass ML heute aus vielen Bereichen unseres Lebens nicht wegzudenken ist. Dabei ist der Kern von ML sehr einfach gestaltet. /Abu-Mostafa et al. 2012/ nennen lediglich drei Voraussetzungen:

- Es muss tatsächlich ein Muster in den Daten vorliegen.
- Das Muster lässt sich nicht mathematisch beschreiben.
- Es müssen genügend Daten zur Verfügung stehen.

Gleichzeitig verweisen die Autoren darauf, dass die ersten zwei Forderungen vergleichsweise unkritisch sind. Wenn kein Muster vorliegt, dann entspricht die Fehlerrate des Lernprozesses einem Zufallsschätzer. So ist die Voraussagerate einer fairen Münze bei hinreichend großer Stichprobe exakt 50 %. Liegt eine mathematische Beschreibung für das Muster vor, so ist der Nutzer besser beraten, es zu implementieren. Theoretisch ist es dann auch möglich, dieses Muster mit ML zu lernen. Allerdings wird der Datenbedarf in der Regel wesentlich höher sein, um Ergebnisse mit vergleichbarer Qualität wie die der direkten Lösung zu erreichen. Der dritte Punkt ist tatsächlich eine Einschränkung. Um den Datenbedarf abschätzen zu können, werden zunächst die im ML gängigen Begriffe wie Klassifizierung, Growth-Function usw. kurz anhand von Beispielen erläutert.

2.1 Klassifizierung

Als Klassifizierung wird im ML ein Vorgang bezeichnet, währenddessen ein- oder mehrdimensionale Punkte eines Datensatzes anhand ihrer Merkmale (Features) in eine oder mehrere Kategorien sortiert werden. Für diese ziemlich allgemein formulierte Aufgabe gibt es eine Vielzahl mathematischer Ansätze und Lösungen. In diesem Beitrag wurden zur Lösung Ansätze aus der Kategorie „Überwachtes Lernen“ verwendet. Solche Methoden sind vor allem für Aufgabenstellungen geeignet, in denen die Klassenzugehörigkeit für den Großteil des Datensatzes bekannt ist, während die noch nicht klassifizierten Punkte aufgrund ihrer Features in die bekannten Klassen einsortiert werden sollen.

Ein simples Beispiel mit $n = 20$ Punkten sei in *Abb. 1* (links) dargestellt. In einem ebenen Koordinatensystem aufgezeichnet, besteht der Datensatz (zunächst) nur aus zwei Klassen $\boldsymbol{y} = [y_1 \;\; y_2]^T$ – gefüllter und ungefüllter Kreis – mit je zwei Features pro Punkt $\boldsymbol{x} = [x_1 \;\; x_2]^T \in \mathbb{R}^2$. Auch ist ein Punkt (Kreuz) enthalten, dessen Klassenzugehörigkeit zunächst unbekannt ist. Dessen Einordnung sollte nun aufgrund der aktuell zur Verfügung stehenden Daten möglichst sinnvoll erfolgen. Eine naheliegende Lösung dieser Aufgabe könnte gemäß der rechten Seite geschehen: Beide Anhäufungen werden mit einer imaginären Linie M getrennt. Die über M liegenden Punkte gehören anscheinend zur Klasse der gefüllten Kreise, während die darunter liegenden Punkte zur ungefüllten Kreismenge gehören. Dieser Logik folgend liegt es nahe, dass der noch unbekannte Punkt zur Klasse der gefüllten Kreise gehört.

2.2 Growth-Function als Indikator für Modellkomplexität

Abb. 1 beschreibt nur ein simples Beispiel. Das verwendete Linienmodell M lässt sich durch die Hesse'sche Normalenform mit nur drei[1] Parametern definieren. Nun stellt sich die Frage, ob sich bei Verwendung von zwei Klassen alle möglichen Punktmuster mit einer einzigen Linie reproduzieren lassen. Diese Annahme ist falsch. Es lassen sich sehr einfach Datensätze (*Abb. 2* links) mit mehr als drei Punkten konstruieren, deren Muster sich nicht durch eine einzige Linie reproduzieren lässt. Diese Eigenschaft wird als Bruchpunkt eines Schätzers bezeichnet. Sie wird als Punktanzahl angegeben, ab der das gewählte mathematische Modell nicht mehr alle möglichen Muster erzeugen kann. Zu beachten ist, dass dieser Begriff

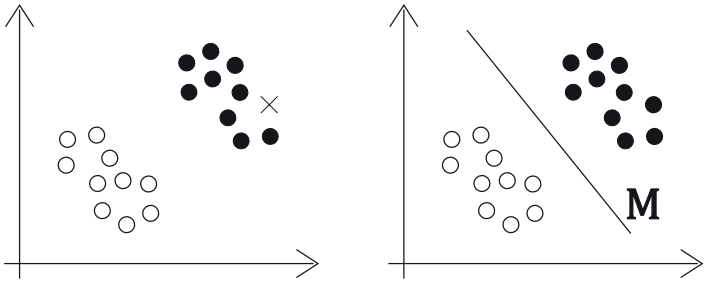

Abb. 1 | Links: Datensatz bestehend aus zwei Klassen und einem noch nicht klassifizierten Punkt. Rechts: Klassifizierung durch lineares Modell

[1] Die Beschreibung einer Linie mit nur zwei Parametern (z. B. Punkt und Steigung) ist bei numerischen Anwendungen selten sinnvoll, da eine zunehmende Steigung sich äußerst ungünstig auf die Konditionszahl des Gleichungssystems auswirkt. Auch resultiert eine Klassentrennung entlang der *Y*-Achse in unendlicher Steigung, die ebenfalls praktisch gesehen nicht sinnvoll wiederverwertbar ist.

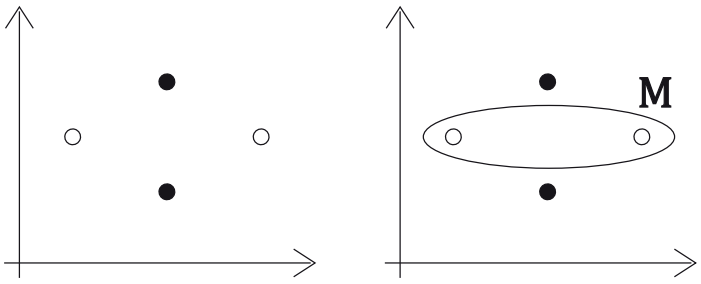

Abb. 2 | Links: Nicht linear separierbarer Datensatz. Rechts: Klassifizierung durch ein komplexeres Modell /Vapnik 2000, S. 81/

im ML anders definiert ist als in der Ausgleichungsrechnung. Im ML ist es die zuvor beschriebene Fähigkeit, eine bestimmte Menge zu „zerschmettern" (engl: shattered set). In der Ausgleichungsrechnung geht es um die Toleranz eines Schätzers in Verbindung mit Ausreißern. Der Bruchpunkt einer 2D-Linie ist nach dieser Definition vier. Zur korrekten Klassifizierung ist somit ein Vorgang notwendig, welcher die Features oder Parameter des mathematischen Modells so erweitert, dass eine Trennung der einzelnen Klassen geschehen kann. Im rechten Teil der Abbildung wird das Modell mit zusätzlichen Parametern erweitert, um dieses Muster erzeugen zu können. Wiederum lassen sich in der Ebene Muster erzeugen, die sich nicht durch die fünf Ellipsenparameter trennen lassen und somit Modelle mit mehr Parametern erfordern. Aus diesem Gedankengang definierten /Vapnik & Chervoneniks 1968/ eine Growth-Function G. Der Wert von G ist abhängig von der Anzahl n zur Verfügung stehender Punkte und der VC-Dimension d_{vc} (≈ Anzahl unabhängiger Parameter eines Modells). Vereinfacht dargestellt gibt G an, wie viele voneinander verschiedene Muster ein mathematisches Modell maximal erzeugen kann.

2.3 Hoeffding-Ungleichung

Das vorhergehende Beispiel verdeutlicht, dass komplexere Muster nur dann erfolgreich reproduziert werden, wenn das mathematische Modell genügend Freiheitsgrade in Form von Parametern besitzt. Nun ist aber auch aus der Ausgleichungsrechnung bekannt, dass für eine brauchbare Schätzung für jeden unabhängigen Parameter (optimistisch gesehen) etwa zehn oder mehr unabhängige Beobachtungen vorliegen sollten. Dieser Gedankengang lässt sich elegant mit der Hoeffding-Ungleichung (*1*) ausdrücken /Hoeffding 1963/:

$$\mathbb{P}\left[\left|W - N\right| > \varepsilon\right] \leq 2\,\mathrm{e}^{-2n\varepsilon^2}. \tag{1}$$

Die Wahrscheinlichkeit $\mathbb{P}$, dass der Differenzbetrag $|W-N|$ zwischen der wahren, jedoch stets unbekannten Fehlerrate (Out Sample Error W) und der im Training numerisch bestimmten Fehlerrate (In Sample Error N) größer ist als ein vorgegebener Schwellenwert (ε; $0 < \varepsilon < 1$), sollte möglichst gering sein. $\mathbb{P}$ ist abhängig von ε^2 und der Anzahl der Messpunkte n. Während des Schätzprozesses ist nur N bekannt. Es stellt eine Art Proxy für W dar und ist gleichzeitig diejenige Größe, nach der minimiert wird. Aus diesem Grund ist es wünschenswert, dass die Abweichung $|W-N|$ nicht nur so gering wie möglich ausfällt, sondern auch, dass diese Aussage so sicher wie möglich ist. Deshalb ist der Nutzer auch stets bemüht, den Term auf der rechten Seite der Ungleichung so gering wie möglich zu halten. Andernfalls wäre der Schätzprozess ebenfalls sinnlos. Möchte der Nutzer mehr Sicherheit auf der linken Seite, so wählt er für ε einen kleinen Wert. Mit abnehmendem ε wächst jedoch das Risiko einer ungünstigen Schätzung auf der rechten Seite mit ε^2 quadratisch. Es gibt noch eine weitere, wesentliche Einschränkung. Diese Ungleichung gilt nur für einen Versuch. In der Regel unternimmt der Schätzalgorithmus mehrere Versuche und verläuft wie folgt: Die Parameter werden mit zufällig gewählten Zahlen [0, 1] initialisiert. Bei jedem Versuch werden die Parameter so anpasst, dass sie zunehmend besser zu den Beobachtungen passen. Dabei bedient sich der Algorithmus bei den Parametern aus der Menge aller möglichen Hypothesen. Diese kann unendlich groß sein. Jeder erneute Versuch steigert aber auch das Risiko, dass der Algorithmus die Idiosynkrasie (deterministisches und statistisches Rauschen, Ausreißer usw.) der ihm vorgelegten Daten auswendig lernt, anstatt sie auf einer höheren Abstraktionsebene zu verstehen. Um dieses Risiko entsprechend zu bewerten, muss die Ungleichung für k Versuche ($k \in \mathbb{N}_{>0}$) wie folgt angepasst werden:

$$\mathbb{P}\left[\left|W - N\right| > \varepsilon\right] \leq 2\,k\,\mathrm{e}^{-2n\varepsilon^2}. \tag{2}$$

Damit kann k sehr große Werte annehmen. Der einzige gegenwirkende Faktor, eine zunehmende Beobachtungszahl, geht nur linear in die Ungleichung ein. Hinzu kommt, dass selbst ein simpler Schätzer wie die Linie unendlich viele Hypothesen erzeugen kann und damit einen k-Wert von ∞ besitzt. Mit diesen Eingangsvoraussetzungen wäre ML äußerst impraktikabel.

2.4 VC-Ungleichung

Den für ML wohl wichtigsten Beitrag haben Vapnik & Chervonenkis (VC) geleistet. Die Autoren präsentierten eine Argumentationskette, um die konservativ angesetzte Schranke der Hoeffding-Ungleichung aufzuweichen und herabzusenken. Damit lässt sich nachvollziehen, dass k in Ungleichung (*2*) wie folgt durch G ersetzt werden kann:

$$\mathbb{P}\left[\left|W - N\right| > \varepsilon\right] \leq 4\,G\left(2n, d_{vc}\right)\mathrm{e}^{-\frac{n}{8}\varepsilon^2}. \tag{3}$$

Übertragen auf den praktischen ML-Fall bedeutet dies, dass k nicht mehr unendlich, sondern im schlechtesten Fall 2^n beträgt. Letzteres ist in praktischen Fällen eher eine Ausnahme und gilt nur für Modelle, die keinen Bruchpunkt besitzen. Dieser Fall sollte ohnehin vermieden werden, da G über die Exponentialfunktion dominiert. In allen anderen Fällen ist G ein Polynom, welches asymptotisch der Exponentialfunktion unterliegt und somit einen Lernprozess ermöglicht. Begründet wurde der Tausch von k in G vor allem damit, dass die Ungleichung (*2*) zwar gültig ist, aber zunächst einmal nur die obere Schranke darstellt. Um diese Schranke zu erreichen, müsste der unglücklichste aller Fälle eintreten, nämlich dass sich alle ungünstigen Ereignisse $|W-N| > \varepsilon$ stets aufaddieren und es zu keiner Überlappung der ungünstigen Ereignisse kommt.

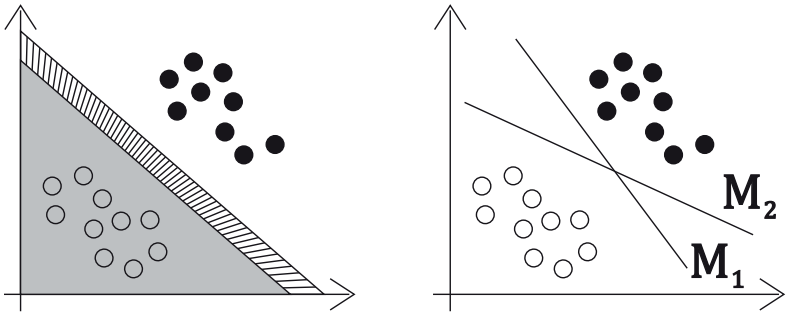

Abb. 3 | Links: Wahrer Verlauf des Klassifizierungsmodells. Die schraffierte Fläche entspricht einer kleinen Änderung. Rechts: Klassifizierung durch zwei lineare Modelle

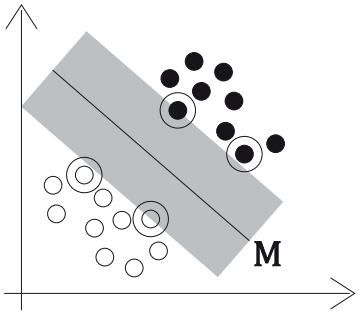

Abb. 4 | Trennung der Klassen mit SVM

In praktischen Fällen überlappen sich die schlechten Ereignisse jedoch sehr stark.

Die gefüllte Fläche in *Abb. 3* (links) zeigt beispielhaft den stets unbekannten Verlauf der wahren Hypothese. Die schraffierte Fläche soll eine leichte Veränderung dieses Modells darstellen. Es ist naheliegend, dass ein Großteil aller Ereignisse, die für das gefüllte Modell gelten, wegen der kleinen Änderung wahrscheinlich auch für das schraffierte Modell gültig sind. Somit müsste es auch eine starke Überlappung der schlechten Ereignisse geben. Weiterhin sind in praktischen Fällen alle Datensätze begrenzt, sodass Änderungen nur dann sichtbar sind, wenn ein Punkt in diesem Bereich liegt.

Abb. 3 (rechts) zeigt eine tatsächliche Schätzung. Die wahre Hypothese bleibt unsichtbar. M_1 und M_2 sind zwei voneinander verschiedene Hypothesen, die ein und dasselbe Ergebnis produzieren, da es trotz Änderung in den Parametern zu keiner Änderung in N kommt. Es ist offensichtlich, dass sich damit die Anzahl der Versuche von $k = \infty$ auf G, nämlich die modellbezogene maximale Anzahl der darstellbaren Muster reduziert, die wiederum höchstens 2^n beträgt. Gleichzeitig wurde in Abschnitt 2.2 deutlich, dass einfache Linien-Schätzer schon mit mehr als drei Punkten nicht alle Muster erzeugen können und die Werte G somit in der Regel wesentlich kleiner ausfallen als die obere Schranke. Hiermit wird es nochmals deutlich, dass die Anzahl der zur Verfügung stehenden Punkte als einzige bekannte Größe einer möglichen Generalisierung durch den Lernprozess im Wege steht.

Ein weiterer sehr gewichtiger Vorteil der VC-Theorie ist, dass sie auf möglichst wenigen Annahmen aufbaut. Dadurch ist sie unabhängig vom Lernalgorithmus, der Verteilung des Trainingsdatensatzes und der zur lernenden, aber stets unbekannten wahren Hypothese.

2.5 SUPPORT-VECTOR-MACHINES

Die in den vorhergehenden Punkten angerissenen Grundgedanken hatten /Boser et al. 1992/ im Sinn, als sie in ihrer Arbeit einen Algorithmus namens Support-Vector-Machines (SVM) vorschlugen. Dessen Lagrange-Formulierung

$$L(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} y_i y_j \alpha_i \alpha_j \boldsymbol{x}_i^{\mathrm{T}} \boldsymbol{x}_j$$

$$\text{mit} \sum_{i=1}^{n} \alpha_i y_i = 0 \text{ und } 0 \leq \alpha_i \leq c. \qquad (4)$$

mit $\boldsymbol{x} \in \mathbb{R}^m$ berechnet das Modell in *Abb. 4* so, dass der Orthogonalabstand beider Klassen maximal ausfällt. Die Konstante c ist ein Regularisierungsparameter und wird in der Regel per Brute-Force oder durch heuristische Verfahren ermittelt. Diejenigen Punkte, deren Lagrangekoeffizienten $\alpha_i \neq 0$ sind, werden als Stützvektoren (SV) bezeichnet. Sie bilden den Grenzverlauf des Modells. In *Abb. 4* wären dies die umrandeten Punkte. Deren Anzahl entspricht ungefähr der Anzahl unabhängiger Parameter und wird als VC-Dimension d_{vc} bezeichnet. Durch geschickte mathematische Formulierung einer optimalen Klassenabgrenzung gilt in praktischen Fällen $d_{vc} \ll n$. Hierdurch nimmt G wiederum geringere Werte an, ohne dabei bei komplexen Mustern an Flexibilität einzubüßen. Ein äußerst günstiger Umstand, weswegen SVM auch binnen kürzester Zeit in vielen Aufgaben zum Standardschätzer erhoben wurde. Kurz nach dessen Einführung schrieben /Bottou et al. 1994/, dass SVM ohne jegliche Modifikationen bei der Erkennung handgeschriebener Postleitzahlen Ergebnisse vergleichbarer Qualität liefert, wie die besten und zur damaligen Zeit noch handoptimierten Neuronalen Netze. Zusätzlich beschreibt /Vapnik 1998/ in seinem Buch die Herleitung der für SVM gültigen Ungleichung (*5*), einer deutlich günstigeren Form der Ungleichung (*3*):

$$\mathbb{E}[W] \leq \frac{\mathbb{E}[\text{Anzahl}(SV) \neq 0]}{n - 1}. \qquad (5)$$

Damit ist es mithilfe von SVM sogar möglich, (aus mehreren Durchläufen) den Klassifikationsfehler der wahren Hypothese ohne ε und N nach oben einzugrenzen.

2.6 MULTI-CLASS-SVM

Praktische Anwendungen sind zunächst selten – wie zu Beginn dieses Beitrags beschrieben – auf Ja/Nein-Entscheidungen limitiert und erfordern daher in der Regel mehr als nur zwei Klassen. Dazu gehört zum Beispiel die Trennung aller vorkommenden Buchstaben und Zahlen während einer Zeichenerkennung. Auch nichttrivial trennbare Klassengruppen mit sehr komplexen Mustern, wie das Risiko eines Herzinfarkts oder die Ausfallwahrscheinlichkeit einer Anlage, kommen ebenfalls vor. *Abb. 5* zeigt einen Datensatz mit drei Klassen (K_1, K_2, K_3). Mit SVM sind damit verschiedene Lösungen denkbar. Zum einen lässt sich mit einer „Einer-gegen-alle"-Strategie eine einzelne Klasse von allen anderen Klassen abgrenzen. Trainiert wird dann in der Kombination: $\{(K_1 | K_2 \cup K_3)$,

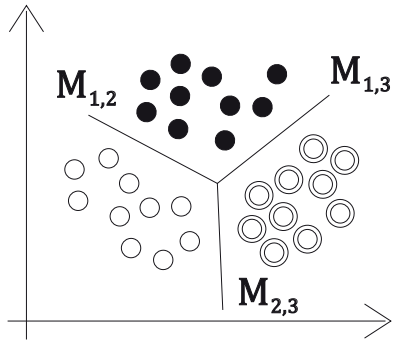

Abb. 5 | „Alle Paare“-Training mit drei Klassen

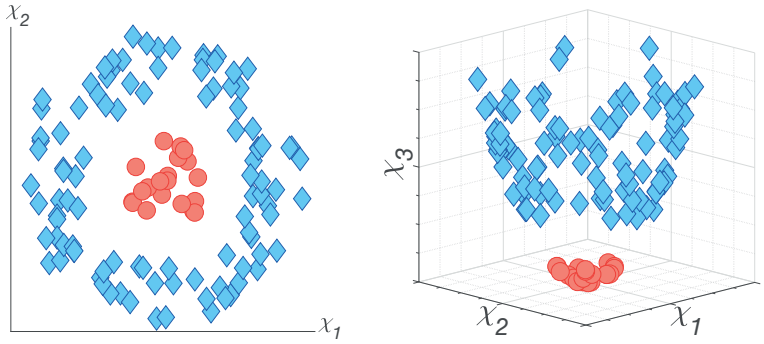

Abb. 6 | Links: Datensatz nicht linear trennbar. Rechts: Erweiterung des Feature-Raums durch Kernel-Tricks

$(K_2 | K_1 \cup K_3)$, $(K_3 | K_1 \cup K_2)\}$. Ebenfalls denkbar ist eine „Alle Paare“-Konfiguration: $\{(K_1 | K_2), (K_1 | K_3), (K_2 | K_3)\}$ nach /Hastie & Tibshirani 1998/. In beiden Fällen kann es in den aufgestellten Modellen aufgrund von Ausreißern und Rauschen zu Widersprüchen in einzelnen Punkten kommen. In diesem Fall beanspruchen zwei oder mehr Modelle einen Punkt für sich. Die Auflösung solcher Widersprüche ist bis heute ein aktives Forschungsgebiet. Auch gibt es keinen eindeutigen Konsens darüber, welches Verfahren die besten Ergebnisse bringt. Stand der Technik ist, die Fehler durch eine geschickte Codierung zu reduzieren/korrigieren. Dabei verwendet „Error-Correcting-Output-Codes (ECOC-SVM)“ nach /Dietterich & Ghulum 1995/ eine Hamming-Codierung, während „Error-Correcting-Tournaments“ nach /Beygelzimer et al. 2009/ zur Korrektur Filter-Trees einsetzen. Unabhängig vom Korrekturverfahren scheint dennoch ein großer Teil der Nutzer eine „Alle-Paare“-Konfiguration zu bevorzugen. Dies hat praktische Gründe. SVM besitzt zwar eine konvexe Zielfunktion, gehört aber dennoch aufgrund der Nebenbedingungen in (*4*) gemäß /Platt 1998/ in die Komplexitätsklasse $\sim O(n^2)$, die mittels quadratischer Programmierung gelöst werden muss. Durch die „Alle-Paare“-Aufteilung entstehen damit viele Aufgaben mit kleineren Datensätzen, die sich parallel auf CPU/GPU deutlich effizienter lösen lassen.

2.7 KERNEL-METHODEN

Analog zur Erweiterung der Parameter in Abschnitt 2.2 sind auch Erweiterungen des Feature-Raums denkbar, um somit eine Klassifizierung mit wenigen Parametern zu gestatten. Hierzu bedient sich der Nutzer sogenannter Kernel-Tricks nach /Guyon et al. 1993/. Ein Kernel $\boldsymbol{K}$ ersetzt das innere Produkt $\boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{x}_j$ der Lagrange-Funktion in der Gl. (*4*) mit $\boldsymbol{z}_i^{\mathrm{T}}\boldsymbol{z}_j = \boldsymbol{K}(\boldsymbol{x}_i, \boldsymbol{x}_j)$. $\boldsymbol{K}$ ist eine Matrix, welche die Mercer-Bedingungen Symmetrie und positive Semidefinitheit erfüllen muss. Zwei in der Praxis oft genutzte Kernel sind der Polynomkernel der Gl. (*6*) und Gauß-Kernel (*7*).

$$(1 + \boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{x}_j)^d = (1 + \boldsymbol{x}_{i,1}\boldsymbol{x}_{j,1} + \boldsymbol{x}_{i,2}\boldsymbol{x}_{j,2} + \cdots + \boldsymbol{x}_{i,m}\boldsymbol{x}_{j,m})^d; \quad d > 0. \tag{6}$$

$$\mathrm{e}^{-\gamma\|x_i - x_j\|^2} = \mathrm{e}^{-x_i^2}\mathrm{e}^{x_i x_j}\mathrm{e}^{-x_j^2} = \mathrm{e}^{-x_i^2}\sum_{t=0}^{\infty}\frac{2^t\,(\boldsymbol{x}_i)^t\,(\boldsymbol{x}_j)^t}{t!}\mathrm{e}^{-x_j^2}; \quad \gamma > 0. \tag{7}$$

Abb. 6 (links) zeigt einen ebenen Datensatz. Dieses Muster lässt sich nicht durch eine einzige Linie klassifizieren. Nun wird eine Transformation auf $\boldsymbol{x}$ angewendet, sodass $\boldsymbol{K}$: $\boldsymbol{x} \rightarrow \boldsymbol{z}$, $\mathbb{R}^2 \rightarrow \mathbb{R}^3$ mit $\boldsymbol{z} = (x_1 \mid x_2 \mid x_1^2 + x_2^2)^{\mathrm{T}}$. Nach dieser Transformation ist in *Abb. 6* (rechts) deutlich erkennbar, dass das Muster nun durch die Erweiterung des Feature-Raums um nur eine Dimension (nur mit vier Ebenen-Parametern) getrennt werden kann – ohne eine nennenswerte Steigerung der SV-Zahl.

Vapnik beschreibt diese Eigenschaft noch eindrucksvoller in seinem Buch, aus dem *Tab. 1* entnommen wurde. Es handelt sich um eine Zeichenerkennung mittels 16×16-px-Bildern, sodass $\boldsymbol{x} \in \mathbb{R}^{256}$. Durch eine Steigerung des Polynomgrads auf 7 in Gl. (*6*) lässt sich der Feature-Raum auf die Dimension $\sim 1 \cdot 10^{16}$ ausdehnen, ohne dass es bei dem eingesetzten SVM-Schätzer zum nennenswerten Overfitting kommt. Der Gauß-Kernel der Gl. (*7*) ist ebenfalls bemerkenswert. Er lässt sich nämlich algebraisch so umformulieren, dass daraus seine unendlich-dimensionale[2] Transformation $\boldsymbol{z} \in \mathbb{R}^{\infty}$ ersichtlich ist. Die Wahl eines Kernels und dessen Parameter ist ebenso eine praktische Aufgabe. Hierbei werden wie zuvor Grid-Search oder heuristische Verfahren angewendet, um eine optimale Konfiguration zu finden.

Ordnung (d)	Dimension z	Anzahl SV	N
1	256	282	8,9 %
2	~ 33 000	227	4,7 %
3	$\sim 1 \cdot 10^6$	274	4,0 %
4	$\sim 1 \cdot 10^9$	321	4,2 %
5	$\sim 1 \cdot 10^{12}$	374	4,3 %
6	$\sim 1 \cdot 10^{14}$	377	4,5 %
7	$\sim 1 \cdot 10^{16}$	422	4,5 %

Tab. 1 | Gegenüberstellung von Polynomgrad, Dimension z und Anzahl an Stützvektoren einer Klassifizierung /Vapnik 1998, S. 501/

[2] Wobei Terme höherer Dimension aufgrund der Division mit $t!$ auch sehr stark abklingen.

2.8 BEISPIEL

Der praktische Ablauf einer Machine-Learning-Schätzung soll im Folgenden anhand der Boolschen UND-Funktion verdeutlicht werden:

$$\text{UND: } \boldsymbol{X}_\text{T} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \\ 1 & 1 \end{bmatrix}, \mathrm{f}\left(\boldsymbol{X}_\text{T}\right) = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}. \tag{8}$$

$$\boldsymbol{u} = \begin{bmatrix} 0 & 0 \end{bmatrix}, f\left(\boldsymbol{u}\right) = ? \tag{9}$$

Die Messdaten werden gemäß Gl. (*8*) in einer $n \times 2$-Trainings-Matrix $\boldsymbol{X}_\text{T}$ zusammengefasst, die damit verbundene Klassenzugehörigkeit in $f(\boldsymbol{X}_\text{T})$. Es handelt sich um eine klassische Wahrheitstabelle. Die erste Zeile in $\boldsymbol{X}_\text{T}$ entspricht FALSCH und WAHR bzw. 0 und 1. In der UND-Funktion ergibt diese Kombination: FALSCH. Somit wird die erste Zeile in $f(\boldsymbol{X}_\text{T})$ in die Klasse 0 eingeordnet. Entsprechend werden auch die übrigen zwei Zeilen ausgewertet.

$$\text{training } (\boldsymbol{X}_\text{T}, f(\boldsymbol{X}_\text{T})) \longrightarrow M, \tag{10}$$

$$\text{validierung } (M) \longrightarrow \text{Fehlerrate}, \tag{11}$$

$$\text{klassifizierung } (M, \boldsymbol{u}) \longrightarrow 0. \tag{12}$$

Die Features und deren Klassenzugehörigkeiten werden in Gl. (*10*) verwendet, um ein Schätzmodell zu berechnen. In diesem Schritt werden auch die optimalen Kernel und deren Parameter, wie γ oder d, ermittelt. Im nächsten Schritt wird die Fehlerrate von M bestimmt. Dieser Schritt ist aus vielerlei Gründen nicht zu vernachlässigen. Zum einen besitzen viele ML-Algorithmen keine konvexe Zielfunktion und können in ein lokales Minimum konvergieren. Auch kann es durchaus vorkommen, dass die verwendete Datengrundlage, Wahl der Features oder des Schätzers keine ausreichende Ableitung von Mustern gestattet, ohne dass sich dieser Umstand in der Trainingsphase verdächtig äußert. Der dritte, ebenfalls kritische Punkt ist die Neigung einiger Schätzmodelle (insbesondere Neuronaler Netze) zum Overfitting. Es handelt sich um eine Eigenschaft, bei der während der Schätzung nicht nur allgemeine Eigenschaften des Musters, sondern auch die Idiosynkrasien durch den Schätzer modelliert werden. Ist ein Muster vorhanden, so sollte bei hinreichend großen Datensätzen die ermittelte Fehlerrate in der Validierung unter der eines Zufallsschätzers liegen und ähnliche Beträge wie der im Training ermittelten Fehlerrate N annehmen. Nach erfolgreichem Abschluss der vorhergehenden Schritte kann das Modell im letzten Schritt auch „im Feld" für eine Klassifizierung von Punkten verwendet werden, deren Klassenzugehörigkeit noch unbekannt ist.

Die Modellschätzung der UND-Funktion lässt sich ebenfalls visuell interpretieren. *Abb. 7* stellt diese in einem ebenen Koordinatensystem dar. Die Zeilen in $\boldsymbol{X}_\text{T}$ entsprechen den Koordinaten (01), (10) und (11). Der SVM-Algorithmus berechnet M so, dass die resultierenden Parameter der Gerade die Klassen 0 und 1 mit maximalem Orthogonalabstand trennen. Während der Berechnung werden alle Punkte über der Gerade zur Klasse 1 gruppiert, unter der Gerade zur Klasse 0. Damit gehört der Punkt u wie erwartet zur Klasse 0.

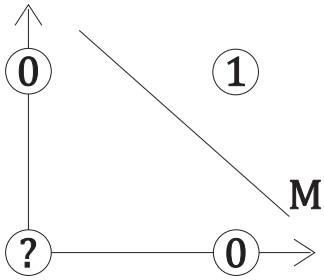

Abb. 7 | Darstellung der UND-Funktion in einem Koordinatensystem

3 AUSWERTUNG

Während der Kalibrierung jeglicher Geräte, seien es Laserscanner, Distanzmesser oder Kameras, entsteht oft ein Dilemma: Auf der einen Seite gibt es gute Gründe, beim Kalibrieren die Daten nicht direkt anzuschauen, sondern nur deren generelle Eigenschaften zu kennen. Zum Beispiel Monotonie, Symmetrie oder etwaige Grenzwerte. All diese Eigenschaften können in die Berechnung der Kalibrierparameter einfließen. Andererseits ist es auch sinnvoll zu wissen, wie gut das berechnete Modell zu den Messdaten passt. Eventuell lässt sich das Instrument mit anderen Methoden noch präziser kalibrieren. Beide Standpunkte sind nachvollziehbar. Oft überwiegt jedoch der zweite Fall und es wird nach Parametern und neuartigen Ansätzen gesucht, die besser zum gegebenen Datensatz passen. Das damit verbundene Data-Snooping-Risiko wird in diesem Fall in Kauf genommen.

Abb. 8 (links) zeigt einen synthetischen Kalibrierdatensatz. Die Abszissenachse beschreibt die Messwerte, die Ordinatenachse die Residuen (Abweichungen zum Kalibriermodell). Eine manuelle oder semi-automatisierte Auswertung erfolgt dann in der Regel

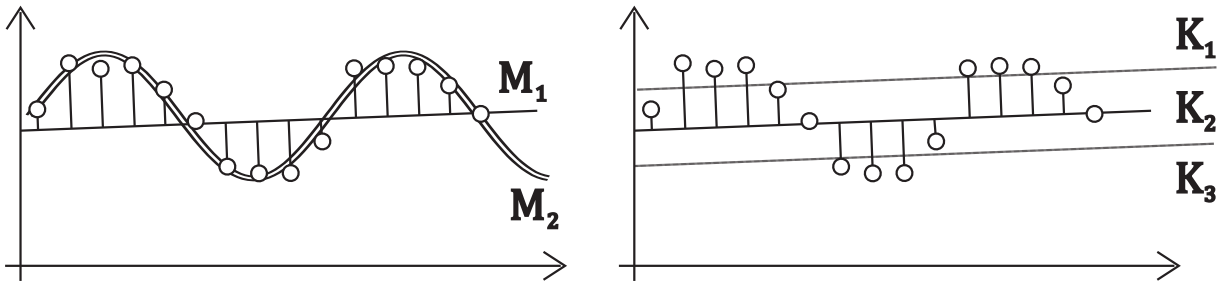

Abb. 8 | Links: Modellierung der Residuen-Systematik als Schwingung. Rechts: Einteilung der Residuen in Klassen

wie folgt: Zuerst wird ein Standard-Kalibriermodell (hier: ausgleichende Gerade M_1) verwendet. Danach werden die Residuen im Verlauf des Kalibriermodells dargestellt. Es handelt sich um eine visuelle Interpretationshilfe, die hervorhebt, ob es trotz Kalibrierung noch ein Rest-Muster in den Residuen gibt. Um eine Spezifikation mit höherer Genauigkeit zu erreichen, wird im zweiten Schritt ein komplexeres Modell (hier: periodische Schwingung M_2) berechnet, welches zusätzlich auch das visuell gefundene Muster in das Modell aufnimmt.

Vorhergehende Beispiele besitzen eine diskrete Klassenzuordnung der einzelnen Punkte. Demgegenüber sind die Residuen in *Abb. 8* reellwertige Zahlen. Da die Stärken von ML-Ansätzen vor allem in der Klassifizierung und weniger in der Regression liegen, wird vor Auswertung mittels ML deshalb eine manuelle Einteilung in Klassen durchgeführt. Die in diesem Beitrag verwendete Methode ist in *Abb. 8* (rechts) dargestellt. Hierbei werden die Residuen je Dimension anhand ihres Betrags und des Vorzeichens in Klassen eingeteilt, wobei die Anzahl der Punkte dazu verwendet werden kann, um abzuschätzen, wie viele Klassen für den gegebenen Datensatz sinnvoll sind. Alternativ wäre auch der Einsatz von „unüberwachten" Lernmethoden wie Nächste-Nachbar-Klassifikation denkbar, um eine sinnvolle Klassenanzahl zu bestimmen.

3.1 Entstehung der zweidimensionalen Bildresiduen

In der Nahbereichsphotogrammetrie (vgl. /Luhmann 2010/) ist der Kalibrierprozess mittels Bündelblockausgleichung durch die VDI/VDE 2634 standardisiert. *Abb. 9* (links) zeigt den dazu vorgeschlagenen Kalibrierkubus, wobei die Messpunkte durch schwarze Kreise auf weißem Hintergrund realisiert wurden. Diese Kreismarken werden mithilfe von Bildverarbeitung detektiert und als zweidimensionale Beobachtungen in eine darauf folgende Bündelblockausgleichung eingeführt. Allerdings handelt es sich hierbei lediglich um eine Einzelaufnahme. Je nachdem, was für die jeweilige Kamera und Objektiv-Kombination sinnvoll ist, kann es Hunderte solcher Aufnahmen geben. Diese werden dann wie in *Abb. 9* (rechts) gezeigt, aus mehreren Standpunkten sowie Höhen und Winkeln aufgenommen. Während der anschließenden Kalibrierung gilt die zuvor schon beschriebene Vorgehensweise. Zunächst wird mit einem Standardmodell kalibriert. In der Nahbereichsphotogrammetrie ist dies ein Lochkameramodell mit Objektiv.

Die in *Abb. 10* (links) gezeigten Residuen (stark überzeichnet) entsprechen den Abweichungen zu diesem Standardmodell. Allerdings lassen sich auf diese Weise dargestellte Abweichungen kaum visuell interpretieren, da es sich um Einzelbilder handelt. Der vollständige Datensatz verfügt nämlich meistens über mehr als hundert Bilder. Der Nutzer könnte zwar zwischen einzelnen Bildern hin- und

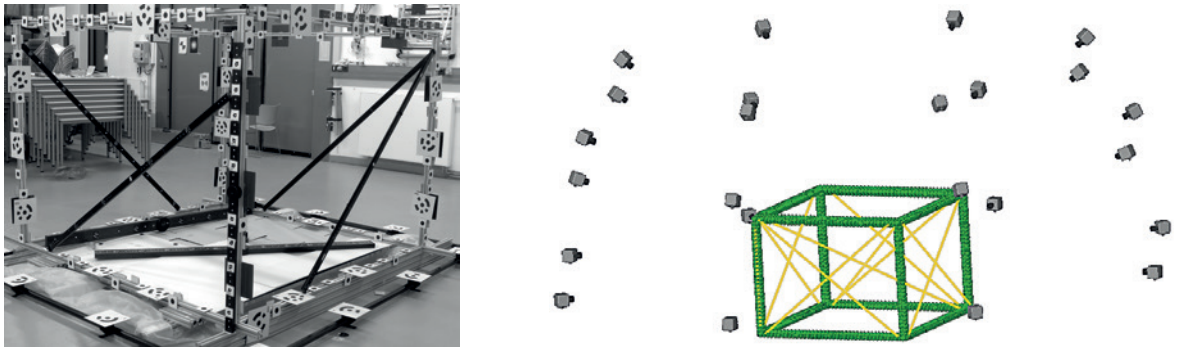

Abb. 9 | Links: Photogrammetrischer Kalibrierkörper nach VDI/VDE 2634. Rechts: Beispielhafte Anordnung einer Kamerakalibrierung

herschalten und dabei versuchen, mögliche Systematiken zu interpretieren. Abhängigkeiten zwischen zwei oder mehreren Bildern werden damit jedoch nur mit hohem Aufwand entdeckt. Zusätzlich besteht immer die Gefahr, dass Nutzer systematische Muster sehen, die sich statistisch nicht nachweisen lassen. Aufgrund der hohen Beobachtungsanzahl ist es auch nicht einfacher, wenn alle Residuen zusammengefasst in einer Übersicht wie in *Abb. 10* (rechts) dargestellt werden.

Die Residuen (*vx*, *vy*) der Bündelblockausgleichung entstehen für jeden Messpunkt in beiden Koordinatenachsen des Bildraums. Damit gibt es verschiedene Möglichkeiten, um Klassen zu bilden. Im hier vorgeführten Beispiel erfolgt die Gruppierung nach Koordinatenachse und Wert. Alternativ wären auch Gruppierungen über den Orientierungswinkel und Betrag denkbar. Als Features kommen alle Informationen infrage, die einen direkten Einfluss auf die Verbesserung haben könnten. Gl. (*13*) zeigt eine mögliche Featurewahl. Zur Verbesserung *vx* der Bildkoordinate (Biko) mit dem Index *i* gehört die Verbesserung vy_i, Objektkoordinate (Obko) und deren Oberflächennormale (Norm) *i*, äußere Orientierung (EOR & Quat) *j* und Innere Orientierung (IOR) *k*.

$$\boldsymbol{X}_{\mathrm{T}vx,i,j,k} = \Big[\overbrace{(x,y)_i}^{\text{Biko}} \Big| \overbrace{(vy)_i}^{\text{Resi}} \Big| \overbrace{(X,Y,Z)_i}^{\text{Obko}} \Big| \overbrace{(nx,ny,nz)_i}^{\text{Norm}} \Big| \overbrace{(X0,Y0,Z0)_j}^{\text{EOR}} \Big| \overbrace{(a,b,c,d)_j}^{\text{Quat}} \Big| \overbrace{(c,xh,yh,\ldots)_k}^{\text{IOR}} \Big]. \tag{13}$$

Features, die sich über den ganzen Datensatz nicht ändern und somit konstante Spalten in der Trainingsmatrix verursachen, müssen vor der Schätzung entfernt werden. Dies gilt zum Beispiel für innere Orientierungen, wenn nur ein einziges Objektiv kalibriert wird. Bei ebenen Kalibrierkörpern, deren Oberflächennormale im Rahmen der Messgenauigkeit für alle Objektkoordinaten identisch ist, kann auf deren Verwendung als Feature ebenfalls verzichtet werden. Durch gezieltes Ein- und Ausschalten einzelner Features lässt sich auch der Einfluss einzelner Komponenten auf jeweilige Verbesserungen nachweisen.

4 AUSWERTUNG

Der erste Schritt jeder Machine-Learning-Auswertung ist eine Teilung des Datensatzes. Dies geschieht, bevor auch nur eine einzige Operation an den Messwerten durchgeführt wird. Aus dem Gesamtdatensatz wird eine zufällig gewählte Stichprobe entnommen. Deren Umfang liegt in der Regel zwischen 15 % und 20 % und bleibt bis zur finalen Validierung unangetastet. Sie bildet eine Art Versicherung gegen Fehlinterpretationen und falsch gewählte Modellparameter. Aus der Teilung entstehen somit ein Trainingsdatensatz mit ~ 80 % und der Testdatensatz mit ~ 20 %.

Mit dem Trainingsdatensatz werden die Modellparameter des Schätzers bestimmt. In diesem Beitrag erfolgt die Schätzung mit einem Mehrklassen-ECOC-SVM und Gauß-Kernel. Die zum Datensatz optimal passenden Kernel-Parameter wurden mithilfe von Brute-Force-Methoden (Grid-Search) ermittelt. Die Prüfung von eventuell vorkommenden Überparametrisierungen geschieht durch zehnfache Kreuzvalidierung nach /Stone 1974/. Nach der Trainingsphase wird im abschließenden Test die dazu gehörende Fehlerrate *N* mit der Fehlerrate aus dem Testdatensatz verglichen. Dies ist der

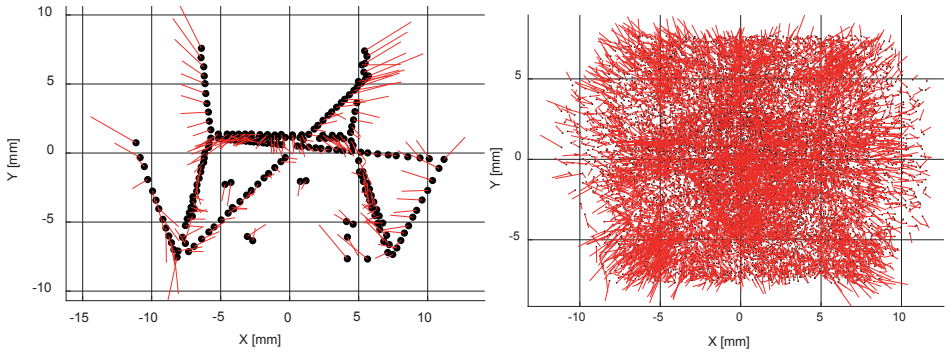

Abb. 10 | Links: Skalierte Bildresiduen; Einzelbild einer Serie. Rechts: Alle Residuen in einem Bild zusammengefasst

erste und gleichzeitig auch der einzige Schritt, in dem der Testdatensatz verwendet werden darf. Beide Größen müssen ungefähr übereinstimmen. Im Zweifelsfall gilt die höhere Fehlerrate aus dem Testdatensatz, da dieses Ergebnis nicht durch das Training und das damit verbundene Durchprobieren kontaminiert ist.

Abb. 11 zeigt eine Industriekamera zum hier ausgewerteten Datensatz. Deren Objektiv verfügt über eine ausgeprägte Radialverzeichnung. Es handelt sich um einen von mehreren Prototypen mit verschiedenen Sensor-Objektiv-Kombinationen, die im Rahmen einer Kooperation zu untersuchen sind. Bei dieser Kamera-Objektiv-Kombination stellt sich insbesondere die Frage, ob eine solche Konfiguration sinnvoll ohne Fisheye-Parameter kalibriert werden kann. Ist dies nicht der Fall, so müsste der Hersteller seine Standard-Kalibrierroutinen und Protokolle anpassen. Neben einem zweiten Schätzer, Kreuzvalidierung und Ungleichung (*5*) können auch Wahrheitstabellen verwendet werden, um die ML-Schätzung zu kontrollieren. Diese bieten eine schnelle Übersicht darüber, wie sicher die einzelnen Klassen getrennt sind.

Tab. 2 fasst den Testdatensatz zur Verbesserung x zusammen. Die erste Zeile gehört zur Klasse 1. Von insgesamt 85 (= Summe der ersten Zeile) Verbesserungen, die in dieser Klasse gruppiert sind, können 41 durch das berechnete Modell tatsächlich auch als Klasse 1 eingeordnet werden. Zahlen außerhalb der Hauptdiagonale sind in dieser Darstellungsweise falsch eingestuft. Damit liegt die Übereinstimmung der Klasse 1 bei $41/85 \approx 48$ %. Im Schnitt über alle Klassen beider Verbesserungen liegt die Übereinstimmung der Klassifizierung bei ≈ 55 %. Ein Zufallsschätzer wird nach Berücksichtigung der Klassengrößen eine Übereinstimmung von ≈ 17 % erreichen. Auch die Fehlerrate im Testdatensatz, welcher nicht am Training teilnimmt, liegt in derselben Größenordnung. Dies ist ein deutlicher Hinweis auf verbleibende Systematiken.

Im nächsten Schritt wird das SVM-Modell dazu verwendet, um die Bildmessungen entsprechend der Klassengruppierung in zehn diskreten Schritten zu korrigieren. Nach der Korrektur und einer erneuten Bündelblockausgleichung mit korrigierten Bildmessungen verbessern sich die Testwerte. RMS_{xyz} der Objektkoordinaten sinkt von ≈ 170 µm auf ≈ 75 µm, RMS der Strecken von ≈ 70 µm auf ≈ 36 µm. Durch den Genauigkeitsgewinn nach Korrektur der Messwerte mit dem geschätzten Modell erhärtet sich die Vermutung, dass nach der ersten Bündelblockausgleichung statistisch signifikante Restsystematiken im Bildraum verbleiben. Dadurch ist es notwendig, das bisherige Kalibriermodell für diese Sensor-Objektiv-Kombination zu wechseln.

Abb. 11 | Einzelbild eines Prototyps mit starker Radialverzeichnung

Kl.	1	2	3	4	5	6	7	8	9	10
1	41	16	16	4	3	1	2	0	0	2
2	4	29	78	21	7	0	0	0	0	0
3	3	19	232	201	21	5	1	1	0	0
4	0	3	100	597	296	35	1	0	0	0
5	1	4	33	239	843	346	19	3	1	0
6	0	0	3	22	321	924	246	17	2	2
7	1	0	0	4	27	291	603	97	3	0
8	1	0	0	2	3	21	202	208	11	2
9	0	0	0	1	3	7	22	59	41	4
10	1	0	0	0	4	1	11	13	37	51

Tab. 2 | Wahrheitsmatrix für den *vx*-Testdatensatz der Kamerakalibrierung

5 FAZIT

Zusammenfassend lässt sich feststellen, dass die in den vorhergehenden Absätzen erläuterte Methode geeignet ist, um vorhandene Restsystematiken in den Bildresiduen nachzuweisen. Jedoch bestehen auch Einschränkungen, deren sich der Anwender stets bewusst sein muss.

Ein Großteil der mit ML verbundenen Mathematik basiert auf Beweisen, die binäre Schätzer annehmen. Hierzu gehören auch die VC-Ungleichung und der später auf dieser Theorie aufbauende und in dieser Arbeit verwendete SVM-Schätzer. Diese unterscheiden wiederum hauptsächlich zwischen Klassen. Erfahrungswerte zeigen, dass hier die Stärken von ML liegen. Aus diesem Grund werden reellwertige Residuen zunächst in Klassen gruppiert. Der Datenbedarf ist ein weiterer, nicht zu vernachlässigender Nachteil, der sich aus der VC-Ungleichung ergibt. Der nächste Nachteil ergibt sich aus bedingt interpretierbaren Parametern. Verglichen mit Bündelblockausgleichungen ist das geschätzte ML-Modell eine Art Blackbox. In der Photogrammetrie ist es allgemein bekannt, dass die Koeffizienten A_1, A_2, A_3 Komponenten der radialsymmetrischen Verzeichnung darstellen. Sie haben ihren physikalischen Ursprung und lassen sich sehr wohl geometrisch deuten. Dagegen verstehen sich Parameter eines SVM-Schätzers in der Regel wie folgt: Die Features gehören zur Klasse 1, weil dadurch der größte Orthogonalabstand zwischen beiden Klassen erreicht wurde. Viele Nachteile relativieren sich jedoch, wenn diese Methode als Ergänzung zu bestehenden Tools gesehen wird, welche bereits für den Nachweis von verbleibenden Systematiken vorgeschlagen wurden. Durch eine Kombination lassen sich bestehende Vermutungen zunehmend erhärten oder widerlegen.

Auf der Habenseite steht das mittlerweile sehr gut erforschte mathematische Fundament. Es ermöglicht automatisierte Entscheidungen, weitgehend unabhängig vom

Nutzer und auch unabhängig davon, wie viele Messpunkte zum Kalibrieren verwendet werden. Das Verfahren selbst ist auch flexibel in der Feature-Wahl. Der Nutzer kann gezielt einzelne Komponenten an- und ausschalten, um bei Bedarf mögliche Abhängigkeiten zu untersuchen. Das Schätzverfahren, ob SVM, Neuronale Netzwerke oder andere, kann ebenfalls frei vom Nutzer gewählt werden. Implementierte ML-Algorithmen sind auch mit geringem Aufwand in Prozesse außerhalb von Kamerakalibrierungen übertragbar. Etablierte ML-Schätzer wie SVM basieren meistens auf simplen Gleichungen und sind deshalb vergleichsweise einfach selbst zu implementieren. Diese Hürde muss ebenfalls nicht genommen werden, da hierfür bereits genügend kommerzielle wie freie Bibliotheken und Entwicklungsumgebungen zur Verfügung stehen. Zum Beispiel Matlab, TensorFlow, Torch, libSVM oder Theano, die von führenden Technologiefirmen wie Mathworks, Google, Facebook und vielen wissenschaftlichen Einrichtungen verwendet und gepflegt werden.

DANKSAGUNG

An dieser Stelle bedanken sich die Autoren bei der Stiftung Rheinland-Pfalz für Innovation, die diese Untersuchungen im Rahmen des Projekts „Optimierte geometrisch/physikalische Modellierung digitaler Industriemesskameras im Einsatz der industriellen Qualitätskontrolle“ (MIndKam), Förderkennzeichen: 961 – 386261/1055, finanziert hat.

LITERATUR

Abu-Mostafa, Y. S.; Magdon-Ismail, M.; Lin, H. (2012): Learning From Data. AMLBook.

Beygelzimer, A.; Langford, J.; Ravikumar, P. (2009): Error-Correcting Tournaments. In: International Conference on Algorithmic Learning Theory, 3. – 5. Oktober 2009, University of Porto, Portugal. Springer, Berlin/Heidelberg, 247 – 262.

Boser, B. E.; Guyon, I. M.; Vapnik, V. N. (1992): A training algorithm for optimal margin classifiers. In: Proceedings of the 5th Annual ACM Workshop on Computational Learning Theory, 27. – 29. Juli 1992, Pittsburgh, PA. ACM Press, New York, 144 – 152.

Bottou, L.; Cortes, C.; Denker, J. S.; Drucker, H.; Guyon, I.; Jackel, L. D.; LeCun, Y.; Muller, U. A.; Sackinger, E.; Simard, P; Vapnik, V. (1994): Comparison of classifier methods: A case study in handwritten digit recognition. In: Proceedings of the 12th IAPR International Conference on Pattern Recognition, 9. – 13. Oktober 1994, Jerusalem, Israel. Vol. 2, 77 – 82.

Dietterich, T. G.; Ghulum, B. (1995): Solving Multiclass Learning Problems via Error-Correcting Output Codes. In: Journal of Artificial Intelligence Research, (1995)2, 263 – 286.

Guyon, I.; Boser, B.; Vapnik, V. (1993): Automatic Capacity Tuning of Very Large VC-dimension Classifiers. In: Hanson, S. J; Cowan, J. D.; Giles, C. L. (Hrsg.): Advances in Neural Information Processing Systems. Morgan Kaufmann, San Francisco, 147 – 155.

Hastie, T.; Tibshirani, R. (1998): Classification by pairwise coupling. In: The Annals of Statistics, 26(1998)2, 451 – 471.

Hoeffding, W. (1963): Probability Inequalities for Sums of Bounded Random Variable. In: Journal of the American Statistical Association, 58(1963), 13 – 30.

Luhmann, T. (2010): Nahbereichsphotogrammetrie. Grundlagen, Methoden und Anwendungen. 3. Auflage. Wichmann, Berlin/Offenbach.

Platt, J. C. (1998): Sequential Minimal Optimization: A Fast Algorithm for Training Support Vector Machines. Technical Report MSR-TR-98-14, Microsoft Research.

Stone, M. (1974): Cross-Validatory Choice and Assessment of Statistical Predictions. In: Journal of the Royal Statistical Society, Series B (Methodological), 36(1974)2, 111 – 147.

Vapnik, V. N.; Chervoneniks, A. J. (1968): Uniform Convergence of Frequencies of Occurrence of Events to Their Probabilities. In: Soviet Mathematics Doklady, 9(1968)4 [Übersetzung], 915 – 918.

Vapnik, V. N. (1998): Statistical Learning Theory. Wiley, Hoboken, NJ.

Vapnik, V. N. (2000): The Nature of Statistical Learning Theory. 2. Auflage. Springer, New York.

M. Sc. Waldemar Kisser

HOCHSCHULE MAINZ – UNIVERSITY OF APPLIED SCIENCES I3MAINZ, INSTITUT FÜR RAUMBEZOGENE INFORMATIONS- UND MESSTECHNIK

Lucy-Hillebrand Str. 2 | 55128 Mainz
waldemar.kisser@hs-mainz.de

Prof. Dr.-Ing. Frank Boochs

HOCHSCHULE MAINZ – UNIVERSITY OF APPLIED SCIENCES I3MAINZ, INSTITUT FÜR RAUMBEZOGENE INFORMATIONS- UND MESSTECHNIK

Lucy-Hillebrand Str. 2 | 55128 Mainz
frank.boochs@hs-mainz.de

Prof. Dr.-Ing. Dietrich Paulus

UNIVERSITÄT KOBLENZ-LANDAU
INSTITUT FÜR COMPUTERVISUALISTIK

Universitätsstraße 1 | 56070 Koblenz
paulus@uni-koblenz.de