Wilhelm-Schickard-Institut für Informatik, Lehrstuhl für technische Informatik
Max-Planck-Institut für biologische Kybernetik

Kamerakalibrierung und Tiefenschätzung

Ein Vergleich von klassischer Bündelblockausgleichung und statistischen Lernalgorithmen

Fabian Sinz

15. März 2004

Inhaltsverzeichnis

1 Einleitung

Um die Position eines Punkts im Raum zu bestimmen genügen zwei Bilder, in denen dieser Punkt aus verschiedenen Perspektiven abgebildet ist. Durch die beiden verschiedenen Abbildungen ist genug Information vorhanden, um die Projektion dieses Punkts auf die Bildfläche rückgängig zu machen. Die Projektionsgleichungen, von denen jeweils zwei ein Modell einer Lochkamera bilden, besitzen nun mehrere Parameter die verschiedenen Eigenschaften der Kamera, wie z.B. der Brennweite, entsprechen. Diese müssen bekannt sein bzw. geschätzt werden, bevor sich die räumliche Position des Punkts wiederherstellen läßt. Die Ermittlung dieser Parameter wird auch *Kamerakalibrierung* genannt.

Unter Kalibrierung im eigentlichen Sinne versteht man die Bestimmung des Bildhauptpunkts, der Brennweite und weiterer Parameter zum Ausgleich von Linsenfehlern. Die Gesamtheit dieser Parameter wird weithin auch als *innere Orientierung* bezeichnet. Hinzu kommt die Bestimmung der *äußeren Orientierung*, welche die Lage und Ausrichtung des Kamerakoordinatensystems in einem übergeordneten Weltkoordinatensystem beschreibt.

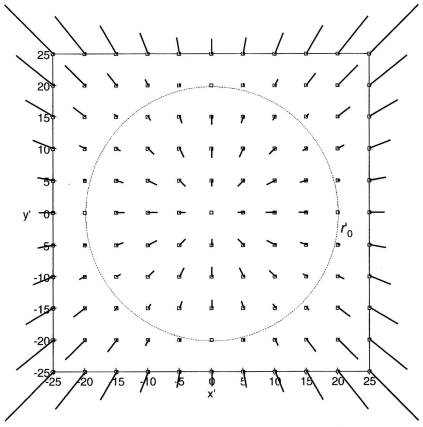

Abb. 1.0 Auswirkungen radial-symmetrischer Verzeichnungen (Abbildung aus [3] S.121)

Die Linsenverzeichnungen lassen sich grob in drei Gruppen unterteilen. Den größten Einfluß haben die radial-symmetrischen Verzeichnungen, welche durch Brechungsänderungen an der Linse des Objektivs entstehen. Sie sind sowohl von der aktuellen Fokussierung, als auch von der Entfernung des Objekts bei konstanter Fokussierung abhängig. Der Grad der radial-symmetrischen Verzeichnung nimmt mit dem Abstand vom Bildhauptpunkt zu und kann bei handelsüblichen Objektiven in den Bildecken bis zu 100 μm betragen (siehe auch [3] S. 119). Die zweite Gruppe, deren Einfluß aber bei weitem geringer ist, bilden die radial-asymmetrischen und tangentialen Verzeichnungen. Sie entstehen meist durch eine Dezentrierung der Linse im Objektiv. Die letzte Gruppe umfasst die Phänomene der Affinität und Scherung, welche Abweichungen des Bildkoordinatensystems von der Orthogonalität und der Gleichmaßstäbigkeit der Koordinatenachsen beschreiben.

Im Fall eines Stereokamerasystems kommt neben der absoluten Orientierung, welche das Gegenstück zur äußeren Orientierung der Einzelkamera darstellt, noch die relative Orientierung der beiden Kameras hinzu. Sie beschreibt die relative räumliche Translation und Rotation des einen Bildes zu dem des Stereopartners. Hat man die äußere Orientierung der beiden Kameras ermittelt, so kann man dadurch auf die absolute Orientierung des Stereosystems schließen. Im Allgemeinen gibt es jedoch mehrere Möglichkeiten die absolute und relative Orientierung von Stereokameras zu errechnen.

Zur Schätzung der inneren und äußeren Orientierung sind Bildpunkte, deren räumliche Position man kennt, sog. *Paßpunkte*, unabdingbar. Diese Aufteilung der Punkte in Ein- und erwünschte Ausgabe machen dieses Problem interessant für die Anwendung von Algorithmen des maschinellen Lernens. Im Rahmen dieser Studienarbeit soll nun der klassische Ansatz der Tiefen- bzw. Positionsbestimmung durch parameterisierte Kameramodelle mit der Leistung ausgewählter Lernalgorithmen für dieses Problem verglichen werden. Die Paßpunkte werden dabei mit einem Roboterarm aufgenommen, dessen Genauigkeit die Grundlage für die Möglichkeit liefert die Abbildung von Bild- auf Raumkoordinaten zu lernen.

Der erste Abschnitt dieser Arbeit befasst sich zunächst mit den Grundlagen beider Ansätze. Er gliedert sich demnach in zwei Teile, wobei der erste die Grundlagen der verwendeten Lernalgorithmen (Abschnitt 2.1), der zweite die der klassischen Kamerakalibrierung (Abschnitt 2.2) behandelt. Der nächste Abschnitt konkretisiert die theoretischen Grundlagen des vorhergehenden Kapitels und führt an einigen Stellen weitere Methoden ein, deren Rolle in dieser Arbeit nicht so bedeutend ist und so besser anhand der Probleme, die sie

lösen sollen, vorgestellt werden. Dazu gehören die Datenaquisition (Abschnitt 3.1), die tatsächliche Parameterisierung der Kameras (Abschnitt 3.2) und die Wahl bestimmter Parameter der Lernalgorithmen (Abschnitt 3.3). Der dritte Abschnitt behandelt die durch die verschiedenen Ansätze erhaltenen Ergebnisse (Abschnitt 4.1 - 4.3) und stellt einen Vergleich zwischen den Leistungen der einzelen Algorithmen an (Abschnitt 4.4). Eine Zusammenfassung und ein Ausblick auf Erweiterungen dieser Arbeit schließt diese Studienarbeit ab.

An dieser Stelle möchte ich auch gerne den Personen am Max-Planck-Institut für biologische Kybernetik in Tübingen (vor allem meinen beiden Betreuern Matthias Franz und Gökhan Bakır) danken, in deren Obhut diese Studienarbeit entstehen konnte und durch die ich in dieser Zeit einiges lernen konnte.

2 Allgemeine theoretische Grundlagen

Dieses Kapitel erläutert diejenigen theoretischen Grundlagen, die zum Verständnis der gewählten Lösungsansätze notwendig sind. Es gliedert sich in zwei Teile, von denen der erste die theoretischen Mittel für die Kalibrierung mit maschinellem Lernen, der zweite die für den modellbasierten Ansatz an die Hand gibt.

Im ersten Teil wird zunächst das Konzept des Kerns eingeführt. Daraufhin wird die *Support Vektor Maschine* und die darauf aufbauende *Support Vektor Regression*, zwei mit Kernen arbeitenden Lernalgorithmen, vorgestellt (vgl. [7]) und mit klassischen Lernverfahren verglichen. Die Einführung der *Ridge Regression* schließt diesen Teil des Grundlagenkapitels ab.

Der zweite Teil der theoretischen Grundlagen beschäftigt sich mit der modellbasierten Kamerakalibrierung. Hierzu wird im ersten Abschnitt ein parametrisiertes Lochkameramodell vorgestellt, das im zweiten Teil noch um die Modellierung von Linsenverzeichnungen erweitert wird. Die Kombination von zwei Einzelkameramodellen zu einem Stereokameramodell bildet den Abschluß des Grundlagenkapitels.

2.1 Grundlagen des maschinellen Lernens

2.1.1 Lernen mit Kernen

Beim überwachten Lernen ermittelt ein Algorithmus zu gegebenen Beispielen, bestehend aus Ein- und Ausgabe, durch Minimierung einer Fehlerfunktion eine sog. *Entscheidungsfunktion*, welche die Daten in geeigneter Weise beschreibt. Bei vielen dieser Algorithmen wird dieser die Klasse der linearen Funktionen, d.h. maximal $(n-1)$-dimensionale Hyperebenen in einem n-dimensionalen Raum, zugrunde gelegt.

Im Beispiel der binären Klassifikation sind die bekannten Daten gegeben als $(x, y) \in \mathbb{R}^n \times \{-1, 1\}$. 1 und -1 bezeichenen dabei die Klassenzugehörigkeit des Beispiels. Die gesuchte Entscheidungsfunktion

$$f : \begin{array}{c} \mathbb{R}^n \to \{-1, 1\} \\ \mathbf{x} \mapsto sgn(\langle \mathbf{w}, \mathbf{x} \rangle + b) \end{array}$$

soll die beiden Klassen räumlich voneinander trennen, so daß das Vorzeichen der Hyperebene zu gegebenen Beispiel seine Klassenzugehörigkeit anzeigt. Diese Trennung kann natürlich nur dann vollständig erfolgen, wenn die konvexen Hüllen der Datenpunkte der jeweiligen Klassen durch eine Ebene voneinander *linear separierbar* sind, also ihr Schnitt leer ist.

Ist dies nicht möglich, so muß man, will man lineare Funktionen als Entscheidungsfunktionen beibehalten, die Repräsentation der Daten ändern. Dies geschieht indem man sie in einen Raum abbildet, dessen Basisvektoren bestimmte Eigenschaften -sog. *Features*- der Eingabedaten repräsentieren. Die Wahl der Features ist eng verknüpft mit der Wahl der Funktionsklasse, mit der die Daten beschrieben werden sollen. Abb. 2.0 zeigt ein Beispiel einer solchen Abbildung. Die Eingabedaten $\mathbf{x}_i \in \mathbb{R}^2$ werden durch die Funktion

$$\Phi : \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \mapsto \begin{pmatrix} x_1^2 \\ x_2^2 \\ \sqrt{2} \cdot x_1 x_2 \end{pmatrix}$$

in einen höherdimensionalen *Feature-Raum* abgebildet, in welchem sie linear separierbar sind.

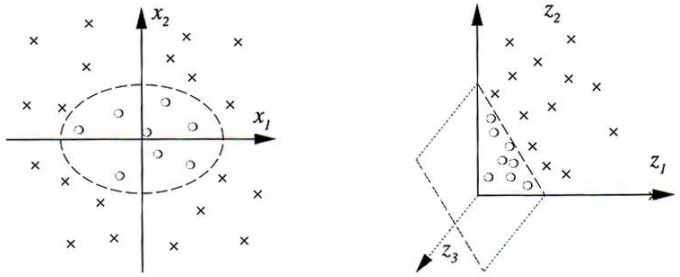

Abb. 2.0 Beispiel für eine Projektion in einen Feature Raum (Abbildung aus [7])

Würde man allgemein derartige Abbildungen für jedes Eingabebeispiel explizit ausführen, dann würden die jeweiligen Algorithmen v.a. für höherdimensionale Daten sehr schnell ineffizient. Viele Lernalgorithmen lassen sich nun aber derart formulieren, daß die Eingabebeispiele nur noch in Skalarprodukte in die Berechnung des Algorithmus eingehen. Ist dies der Fall, so läßt sich der *Kerneltrick* anwenden. Dabei wird die Abbildung $\Phi(x)$ nicht explizit berechnet, sondern nur das Skalarprodukt zwischen Bildern zweier Eingabebeispiele in dem eben erwähnten hochdimensionalen Raum. Dies kann effizient durch den *Kernel* -zu deutsch *Kern*- berechnet werden. Da sich im Deutschen für diese Art von Funktion die Bezeichnung *Kernel* durchgesetzt hat, wird im Folgenden der englische Ausdruck als Teil der Fachsprache verwandt. Dies dient auch dazu den *Kernel* sprachlich vom *Kern*, also dem Nullraum einer linearen Abbildung, abzugrenzen. Statt also erst die Abbildung in den Feature Space explizit auszuführen, um dann das Skalarprodukt zwischen den Bildern zweier Eingabebeispiele zu berechenen, liefert der Kernel effizient den Wert des Skalarprodukts. Ein (positiv semidefiniter) Kernel ist eine Funktion

$$k : \begin{array}{c} \mathcal{X} \times \mathcal{X} \to \mathbb{R} \\ (x, x') \mapsto \langle \mathbf{\Phi}(x), \mathbf{\Phi}(x') \rangle \end{array}$$

deren Gram-Matrix (siehe Definition 6.1.1) für alle $m \in \mathbb{N}$ und alle $x_1, \ldots, x_m$ *positiv semidefinit* ist (siehe Definition 6.1.2). Der Kernel erlaubt all jenen Algorithmen, die sich nur durch Skalarprodukte formulieren lassen, implizit und damit effizient in hochdimensionalen Feature-Räumen zu arbeiten.

Hierzu zwei Bemerkungen:

(i) Die Eingabemenge $\mathcal{X}$ muß nun nicht mehr zwingend Elemente eines Skalarproduktraums, sondern nur noch einer nicht-leere Menge sein. Es gibt Kernel, die direkt auf diskreten Objekten wie z.B. Graphen oder Zeichenketten arbeiten.

(ii) Der Feature-Raum, in den der Kernel implizit projeziert, wird fortan mit $\mathcal{H}$ bezeichnet. Darin kommt zum Ausdruck, daß dieser in der Regel ein *Hilbert-Raum*, d.h. ein vollständiger, unter dem Skalarprodukt abgeschlossener Raum ist. In dem Beispiel von Abb. 2.0 wären dann $\mathcal{X} = \mathbb{R}^2$ und $\mathcal{H} = \mathbb{R}^3$.

Im Folgenden werden nun noch drei Beispiele von Kerneln angeführt:

1. Das kanonische Skalarprodukt $\langle \mathbf{x}, \mathbf{x}' \rangle = \sum_i x_i x_i'$ wird als *linearer Kernel* bezeichnet. In diesem Fall ist $\mathcal{X} = \mathcal{H}_{linear}$ und eine vollständig trennende Hyperebene kann nur dann gefunden werden, wenn die Beispiele schon im Eingaberaum linear separierbar sind.

2. Unter dem *Radial-Basis-Function*- oder kurz *RBF-Kernel* versteht man die Abbildung

$$k(x, x') = \exp\left(-\frac{||x - x'||^2}{2\sigma^2}\right)$$

Abb. 2.1 zeigt den Graph eines RBF-Kernels für zweidimensionale Eingaben. Versteht man den Kernel als Ähnlichkeitsmaß zwischen Eingaben, dann "sitzt" der Datenpunkt, mit dem verglichen wird im Zentrum der Glockenkurve. Die Breite dieser Kurve, d.h. den Grad der Ähnlichkeit von entfernteren Beispielen, wird durch den Kernelparameter σ kontrolliert (je größer σ, desto breiter die Kurve, desto "ähnlicher" entferntere Datenpunkte).
Der Feature-Raum $\mathcal{H}_{RBF}$ des RBF-Kernels besitzt unendliche Dimension. Hier zeigt sich die Mächtigkeit des Kerneltricks, da eine explizite Projektion in einen unendlichdimensionalen Raum schlichtweg unmöglich ist.

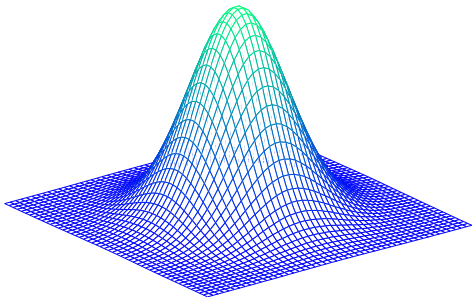

Abb. 2.1 RBF-Kernel für zweidimensionale Eingaben

3. Die Abbildung des *polynomiellen Kernels* ist gegeben durch

$$k(x, x') = (< x, x' > +c)^d$$

Er repräsentiert die Funktionsklasse der Poylnome vom Grad kleiner oder gleich d. Man unterscheidet zwischen dem *homogenen* und dem *heterogenen* polynomiellen Kernel. Beim homogenen polynomiellen Kernel ist $c = 0$. Er bildet in den Raum der Produkte über alle d-Permutationen der Koeffizienten von x ab. Für n-dimensionale Eingabebeispiele gilt für die Dimension des durch den homogenen polynomiellen Kernel aufgespannten Feature Raums

$$\dim(\mathcal{H}_{poly}) = \binom{d+n-1}{d} \in \mathcal{O}(n!)$$

Beim heterogenen polynomiellen Kernel ist $c \neq 0$. Er bildet in den Raum der Produkte über Permutationen bis einschließlich Grad d ab. Im Folgenden wird nur noch die heterogene Variante betrachtet.

2.1.2 Strukturelle Risikominimierung und Support Vektor Maschinen (SVM)

Sind zwei Klassen einer Trainingsmegen in einem Feature-Raum $\mathcal{H}$ linear separierbar, dann gibt es meist mehrere Hyperebenen, welche die Tennung bewerkstelligen könnten. Wäre die wahre Verteilung $P(x, y)$ der Daten bekannt, dann wäre diejenige Ebene optimal, die den Fehler für alle möglichen Eingaben minimiert. Dieser Fehler wird auch als *Risiko* bezeichnet und ist geben durch

$$R[f] = \int l(y, f(x))dP(x, y)$$

l bezeichnet hierbei die gewählte Fehlerfunktion. In Fall der Klassifizierung ist

$$l(y, f(x)) = \frac{1}{2}|f(x) - y|$$

Da P aber nicht bekannt ist, muß man sich damit begnügen, den Fehler auf den Trainingsdaten, das sog. *empirische Risiko* zu minimieren. Ist die Funktionsklasse aber zu mächtig, kann dies leicht dazu führen, daß der Lernalgorithmus die Daten "auswendig lernt"[1] und somit schlecht generalisiert. Dieses Phänomen wird auch als *Overfitting* bezeichnet. Die bloße Minimierung des empirischen Risikos wird also in vielen Fällen nicht zum Erfolg führen.

[1] Es ist immer möglich ein Interpolationspolynom zu finden, das alle Punkte genau trifft. In der Regel beschreibt ein solches Polynom aber nicht die den Daten zugrundeliegende Regularität.

Selbst im Grenzfall von unendlich vielen Trainingsbeispielen konvergiert zwar das empirische Risiko für eine feste Funktion f gegen das tatsächliche Risiko

$$\lim_{m\to\infty} P\left(|\mathcal{R}_{emp}[f] - \mathcal{R}[f]| > \epsilon\right) = 0 \quad \forall f \in \mathcal{F}$$

dies impliziert aber nicht, daß die im Grenzfall von unendlich vielen Trainingsbeispielen durch Minimierung des empirischen Risikos gefundene Funktion auch das tatsächliche Risiko minimiert.

$$P\left(\text{argmin}_f \mathcal{R}_{emp}[f] = \text{argmin}_f \mathcal{R}[f]\right) \neq 1$$

Nun existiert aber mit Wahrscheinlichkeit $(1-\delta)$ eine obere Schranke für (siehe auch [4] Vortrag von Bernhard Schölkopf) das Risiko

$$P\left(\mathcal{R}[f] \leq R_{emp}[f] + \phi(m,h,\delta) \quad \forall f \in \mathcal{F}\right) \geq 1-\delta$$

Der sog. *Konfidenzterm* ϕ hängt zusätzliche von der Anzahl der Trainingsbeispiele m und der *VC-Dimension* h ab. Sie ist gleich der höchsten Anzahl von Punkten, die man finden kann, so daß die gewählte Funktionsklasse in der Lage ist, jede beliebige Partition dieser Punkte durch eine Ebene voneinander zu trennen. Es ist nun klar, daß sich diejenige Hyperebene als optimal hervortut, die sowohl das empirische Risiko, als auch den Konfidenzterm ϕ minimiert. Diese gleichzeitige Minimierung von R_{emp} und ϕ wird als *strukturelle Risikominimierung* bezeichnet. Die Schönheit an dieser Schranke ist die Tatsache, daß sie für alle Funktionen der gewählten Funktionsklasse gilt (für eine ausführlichere Erläuterung siehe [7] 5.4).

Man kann zeigen, daß gerade die Ebene im Sinne der strukturellen Risikominimierung optimal ist, welche maximalen Abstand, engl. *maximal margin*, zu beiden Klassen der Trainingsmenge besitzt. Diese Ebene ist darüber hinaus eindeutig bestimmt und die Suche nach ihr führt zu einem konvexen, quadratischen Optimierungsproblem, für dessen Lösung effiziente Algorithmen existieren. Die Konvexität garantiert das Ausbleiben lokaler Optima. Da mit *margin* sowohl der Abstand, als auch die Grenze des Bereichs größten Abstands gemeint sein kann (ein Punkt kann "auf dem *margin* liegen"), wird im Folgenden der Englische Begriff verwendet. Für eine ausführlicher Einführung in die strukturelle Risikominimierung siehe [7] Kapitel 5 oder [6] Kapitel 4.

Die Suche nach eben beschriebener Hyperebene führt nun zu folgendem Optimierungsproblem

$$\underset{\mathbf{w}\in\mathcal{H}, b\in\mathbb{R}}{\texttt{maximiere}} \min\left\{||\mathbf{x}-\mathbf{x}_i|| \mid \mathbf{x}\in\mathcal{H}, \langle\mathbf{w},\mathbf{x}\rangle + b = 0, i = 1,\ldots,m\right\}$$

Eine dazu äquivalente Formulierung lautet

$$\begin{aligned}&\underset{\mathbf{w}\in\mathcal{H}, b\in\mathbb{R}}{\texttt{minimiere}} \tau(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|^2\\ &\texttt{unter der Bedingung, daß } y_i(\langle\mathbf{w},\mathbf{x}_i\rangle + b) \geq 1 \texttt{ für alle } i = 1,\ldots,m\end{aligned}$$

Die Zielfunktion $\tau(\mathbf{w})$ drückt gerade die Maximierung des *margin* aus. Dies kann man sich leicht intuitiv veranschaulichen, wenn man bedenkt, daß der Abstand eines Punktes zu einer Ebene E durch die Formel $d_E(\mathbf{x}) = \frac{\langle\mathbf{w},\mathbf{x}\rangle+b}{\|\mathbf{w}\|}$ gegeben ist. Durch die Minimierung der Norm des Gewichtsvektors wird der Term im Nenner klein und somit der Abstand groß. Die Nebenbedingungen erzwingen, daß sich alle Trainingsbeispiele auf der "richtigen" Seite der Hyperebene befinden, also korrekt klassifiziert werden. Ist z.B. $y_i = 1$ und die Klassifizierung von $\mathbf{x}_i$ falsch, d.h. $sgn(\langle\mathbf{w},\mathbf{x}_i\rangle + b) < 0$, dann ist die Nebenbedingung $y_i(\langle\mathbf{w},\mathbf{x}_i\rangle + b) < 0$ und die Nebenbedingung verletzt. Haben hingegen y_i und das Bild der Entscheidungsfunktion gleiches Vorzeichen, d.h. die Klasse von $\mathbf{x}_i$ wurde korrekt vorhergesagt, dann ist die Nebenbedingung erfüllt. Die Tatsache $y_i(\langle\mathbf{w},\mathbf{x}_i\rangle+b) \geq 1$ statt $y_i(\langle\mathbf{w},\mathbf{x}_i\rangle+b) \geq 0$ zu fordern hängt mit der Formulierung der Zielfunktion zusammen. Würde nur "≥ 0" gefordert, würde die Minimierung der Norm keinen Sinn haben (siehe hierzu auch [7] S. 12). Durch die Forderung "≥ 1" wird bewerkstellig, daß die Punkte außerhalb des *margin* der Hyperebene liegen.

Um dieses Optimierungsproblem mit Nebenbedingungen zu lösen werden *Lagrange'sche Multiplikatoren* $\alpha_i \geq 0$ benutzt, die zusammen mit der Zielfunktion und den Nebenbedingungen folgende *Lagrange-Funktion* ergeben.

$$L(\mathbf{w}, b, \alpha) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{m} \alpha_i (y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) - 1)$$

Die ursprüngliche von Lagrange formulierte Optimierung einer Funktion unter Nebenbedingungen ließ nur Nebenbedingungen in Form von Gleichungen zu. Dieser Ansatz wurde erst in den fünziger Jahren des letzten Jahrhunderts von Karush-Kuhn und Tucker zu den sog. KKT-Bedingungen erweitert, um eine Zielfunktion zu optimieren, deren Nebenbedingungen auch Ungleichungen sein konnten (siehe hierzu auch [6] Theorem 5.21). Um die gewünschte Lösung zu erhalten muß nach den KKT-Bedingungen die Lagrange-Funktion nach den *primalen Variablen* $\mathbf{w}, b$ minimiert und den *dualen Variablen* α_i maximiert werden. Daher müssen an dieser Stelle partiellen Ableitungen nach $\mathbf{w}$ und b gleich Null sein, also

$$\frac{\partial}{\partial \mathbf{w}} L(\mathbf{w}, b, \alpha) = 0 \text{ und } \frac{\partial}{\partial b} L(\mathbf{w}, b, \alpha) = 0$$

Damit die Lagrange-Funktion den KKT-Bedingungen genügt, muß darüber hinaus $\alpha_i \geq 0$ für alle $i = 1, \ldots, m$ gefordert werden. Dies führt zu

$$\mathbf{w} = \sum_{i=1}^{m} \alpha_i y_i \mathbf{x}_i \text{ und } \sum_{i=1}^{m} \alpha_i y_i = 0$$

Setzt man nun die beiden erhaltenen Bedingungen in die Lagrange-Funktion ein, dann ergibt sich daraus die sog. *duale Form*, welche nun nur noch eine Funktion der Lagrange'schen Multiplikatoren α_i ist.

$$\underset{\alpha \in \mathbb{R}^m}{\texttt{maximiere}} \; W(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle$$

$$\texttt{unter der Bedingung, daß } \alpha_i \geq 0 \texttt{ für alle } i = 1, \ldots, m \texttt{ und } \sum_{i=1}^{m} \alpha_i y_i = 0$$

Die Entscheidungsfunktion wird dann zu

$$\begin{aligned} f(\mathbf{x}) = sgn(\langle \mathbf{w}, \mathbf{x} \rangle + b) &= sgn\left(\left\langle \sum_{i=1}^{m} \alpha_i y_i \mathbf{x}_i, \mathbf{x} \right\rangle + b \right) \\ &= sgn\left(\sum_{i=1}^{m} \alpha_i y_i \langle \mathbf{x}_i, \mathbf{x} \rangle + b \right) \end{aligned}$$

wobei man b aus einem Teil der KKT-Bedingungen (siehe wieder [6] Theorem 5.21)

$$\alpha_i (y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) - 1) = 0 \texttt{ für alle } i = 1, \ldots, m$$

erhält.

Betrachtet man die Entscheidungsfunktion, so wird klar, daß nur diejenigen Trainingsbeispiele in die Auswertung eingehen, deren Lagrange'scher Multiplikator ungleich Null ist. Das sind die Vektoren, bei deren Nebenbedingung Gleichheit vorliegt und die somit auf dem Rand der separierenden Hyperebene liegen. Diese sog. *Supportvektoren* sind genau die Beispiele, welche die Ausrichtung der Hyperebene festlegen. Würde man alle andere Trainingsbeispiele in ihrem Halbraum durcheinanderwürfeln und nur die Supportvektoren an ihrer Stelle belassen, dann würde der eben beschriebene Algorithmus auf diesen Daten exakt dieselbe Hyperebene als Lösung erbringen.

Wie man ebenfalls leicht sehen kann, gehen die Trainingsbeispiele nur durch Skalarprodukte $\langle \mathbf{x}_i, \mathbf{x}_j \rangle$ in das Optimierungsproblem ein, so daß diese durch Kernelfunktionen $\langle \mathbf{\Phi}(x_i), \mathbf{\Phi}(x_j) \rangle = k(x_i, x_j)$ ersetzt werden können und der Algorithmus somit implizit in dem durch den Kernel und den Trainingsbeispielen aufgespannten Feature-Raum arbeitet.

In der Praxis kann es vorkommen, daß die Datenwolken der beiden Klassen überlappen, und es somit keine trennende Hyperebene gibt. Dies kann von Tatsache rühren, daß die beiden Klassen "von Natur aus" überlappen oder daß die Überlappung durch Störungen, wie z.B. Meßfehler, hervorgerufen werden. Aus diesem Grund werden Schlupfvariablen $\xi_i \geq 0$ eingeführt, die durch Umformulierung der Nebenbedingungen ein gewisses Maß an Fehlern zulassen sollen. Abb. 2.2 zeigt die geometrische Interpretation der Schlupfvariablen. Die Nebenbedingungen werden durch sie zu

$$y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq 1 - \xi_i \text{ für alle } i = 1, \ldots, m$$

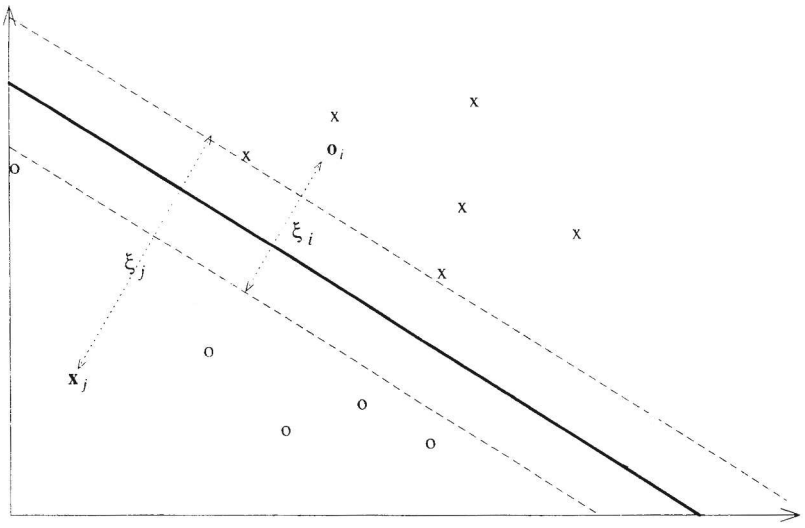

Abb. 2.2 *margin* und Schlupfvariablen eines Klassifikationsproblems (Abb. aus [6] S. 16)

Da die Fehler natürlich so gering wie möglich gehalten werden sollen, wird die Zielfunktion der primalen Form erweitert zu

$$\tau(\mathbf{w}, \xi) = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{m} \xi_i$$

Hier ist C ein Strafterm, der den Einfluß der Fehler kontrolliert. Je kleiner der Wert ist, desto mehr Fehler werden zugelassen, da sie durch das kleine C nicht so stark ins Gewicht fallen. Insbesondere bedeutet $C = \infty$, daß jeder Fehler unendlich stark gewichtet wird und eine so gefundene Hyperebene die Klassen tatsächlich vollständig trennt. In seiner dualen Form lautet das abgeänderte Optimierungsproblem nun

$$\underset{\alpha \in \mathbb{R}^m}{\text{maximiere}} W(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2}\sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j k(x_i, x_j)$$

$$\text{unter der Bedingung, daß } 0 \leq \alpha_i \leq C \text{ für alle } i = 1, \ldots, m \text{ und } \sum_{i=1}^{m} \alpha_i y_i = 0$$

Die Einführung von Schlupfvariablen führt also zu einer oberen Schranke für die Lagrangeschen Multi-

plikatoren α_i. Die Entscheidungsfunktion lautete dann analog

$$f(\mathbf{x}) = sgn\left(\sum_{i=1}^{m} \alpha_i y_i k(x_i, x) + b\right)$$

2.1.3 Support Vektor Regression (SVR)

Bei der Regression ist die Domäne von y nun nicht mehr $\{0,1\}$, sondern $\mathbb{R}$. Gesucht wird also ein Funktion, die Ausgabewerte $y \in \mathbb{R}$ in Abhängigkeit der Eingabedaten $x \in \mathcal{X}$ beschreibt. Anstatt eine nichtlineare Regression im Eingaberaum durchzuführen, kann man nun auch eine lineare Regression im durch einen geeigneten Kernel aufgespannten Feature Raum berechnen. Dies kann z.B. durch die *ε-insensitive Fehlerfunktion* erreicht werden.

$$l(x, y, f(\mathbf{x})) := |y - f(\mathbf{x})|_\varepsilon := \max\{0, |y - f(\mathbf{x})| - \varepsilon\}$$

Der Fehler ist also solange gleich Null, wie die Datenpunkte nur ε weit von der Funktion entfernt liegen. Danach steigt er linear an[2] (siehe Abb. 2.3).

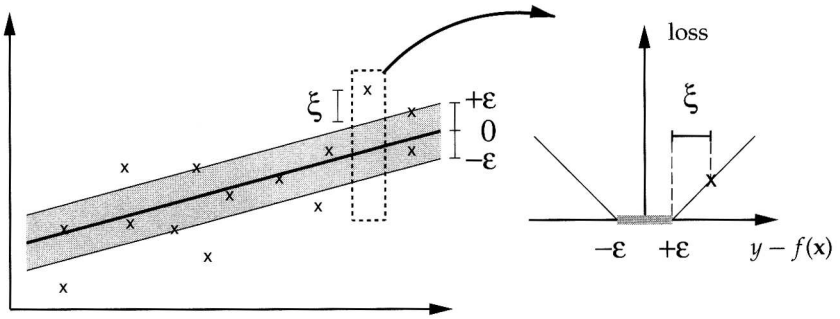

Abb. 2.3 ε-insensitive Fehlerfunktion (Abb. aus [7] S. 18)

Um die lineare Regression $f(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle + b$ zu erhalten, wird nun folgende Zielfunktion minimiert.

$$\frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{m} |y_i - f(\mathbf{x}_i)|_\varepsilon$$

Der Parameter ε muß zusätzlich zu C anfangs vom Benutzer gewählt werden.

Um nun analog zum vorhergehenden Abschnitt Fehler zuzulassen, werden hier nun wiederum Schlupfvariablen verwendet, wobei diesmal zwei Fälle, nämlich $f(\mathbf{x}_i) - y_j > \varepsilon$ und $y_j - f(\mathbf{x}_i) < \varepsilon$, abgedeckt werden müssen. Die zwei Arten von Schlupfvariablen werden hier, wie in [7], mit ξ und ξ^* bezeichnet. Das neue Optimierungsproblem stellt sich dann wie folgt dar:

$$\begin{aligned} &\texttt{minimiere} \\ &\mathbf{w} \in \mathcal{H},\ \xi, \xi^* \in \mathbb{R}^m,\ b \in \mathbb{R} \end{aligned} \quad \tau(\mathbf{w}, \xi, \xi^*) = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{m}(\xi_i + \xi_i^*)$$

$$\begin{aligned} \texttt{unter der Bedingung, daß}\ & f(\mathbf{x}_i) - y_i \leq \varepsilon + \xi_i \\ & y_i - f(\mathbf{x}_i) \leq \varepsilon + \xi_i^* \\ & \xi, \xi_i^* \geq 0 \end{aligned}$$

[2] Anmerkung: Für $\varepsilon \to 0$ ist die ε-insensitive Fehlerfunktion gerade die L_1-Norm, d.h. die Summe der Absolutbeträge der Vektorkoeffizienten, der Differenz zwischen den beiden zu evaluierenden Vektoren.

Analog zum vorhergehenden Abschnitt wird dieses Problem als Lagrange-Funktion formuliert und in seine duale Form gebracht, die dann lautet

$$\underset{\alpha,\alpha^* \in \mathbb{R}^m}{\texttt{maximiere}} W(\alpha,\alpha^*) = -\varepsilon \sum_{i=1}^{m}(\alpha_i^* + \alpha_i) + \sum_{i=1}^{m}(\alpha_i^* - \alpha_i)y_i - \frac{1}{2}\sum_{i,j=1}^{m}(\alpha_i^* - \alpha_i)(\alpha_j^* - \alpha_j)k(x_i,x_j)$$

$$\texttt{unter der Bedingung, daß } 0 \leq \alpha_i, \alpha_i^* \leq C \texttt{ für alle } i = 1,\ldots,m \texttt{ und } \sum_{i=1}^{m}\alpha_i - \alpha_i^* = 0$$

Die geschätzte Funktion hat dann die Form

$$f(x) = \sum_{i=1}^{m}(\alpha_i - \alpha_i^*)k(x_i,x) + b$$

Auch hier muß b extra berechnet werden, da es in der dualen Formulierung des Optimierungsproblems nicht vorkommt. Dies geschieht über die Nebenbedingungen $f(\mathbf{x}_i) - y_j \leq \varepsilon + \xi_i$ und $y_j - f(\mathbf{x}_i) \leq \varepsilon + \xi_i^*$, die für $\xi_i, \xi_i^* = 0$ und $0 \leq \alpha_i \leq C$ zu Gleichungen werden.

2.1.4 Vergleich von SVM und Neuronale Netze

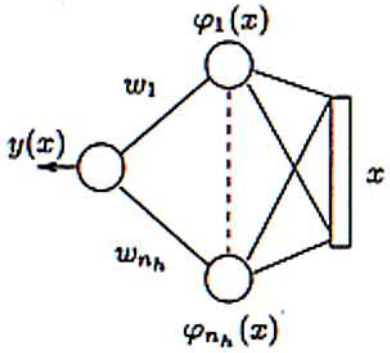

Abb. 2.4 Neuronale-Netze-Interpretation der primalen Form der SVM (Abb. aus [1] S.160)

Sowohl die primale, als auch die duale Formulierung der SVM besitzt eine Interpretation als *neuronales Netz* (*NN*). Die primale Form kommt einem *Multipayer Feedforward Perceptron* mit einer verdeckten Schicht gleich. Die Neurone der Eingabeschicht, deren Anzahl gleich der Dimension der Eingabebeispiele ist, sind vollständig mit den Neuronen der verdeckten Schicht veknüpft, welche die Abbildung Φ realisieren. Die Anzahl der Neuronen der verdeckten Schicht ist gleich der Dimension $n_{\mathcal{H}}$ des gewählten Feature-Raums. Von ihnen geht jeweils eine Verknüpfung zum Ausgabeneuron, welche mit dem entsprechenden Eintrag des Gewichtsvektors $\mathbf{w}$ gewichtet ist.

In der Interpretation der dualen Form ist die Anzahl der Neuronen in der verdeckten Schicht gleich der Anzahl der Supportvektoren. Die in diesen Neuronen realisierte Funktion ist dann nicht die explizite Abbildung in den Feature Raum, sondern die Kernelfunktion mit den jeweiligen Supportvektoren, was einer linearen Funktion im Feature Raum gleichkommt. Die Verknüpfungen zum Ausfgabeneuron ist nun nicht mehr mit den Einträgen von $\mathbf{w}$, sondern mit den jeweilgen Lagrange'schen Multiplikatoren α_i gewichtet. Da die Anzahl der Neuronen in der verdeckten Schicht nicht a priori feststeht, nennt man diese Form auch nicht-parametrisiert (siehe [1] S. 159). Bei Neuronalen Netzen wird also die Architektur vom Benutzer a priori festgelegt (im Sinne der Anzahl der Neurone in der ersten verdeckten Schicht), die SVM wählt diese automatisch.

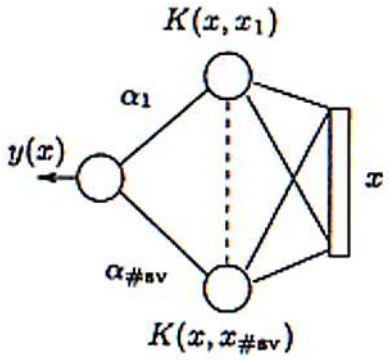

Abb. 2.5 Neuronale-Netze-Interpretation der dualen Form der SVM (Abb. aus [1] S.160)

2.1.5 Ridge Regression

Neben dem Vergleich zu modellbasierter Kalibrierung wird der Ansatz der Kalibrierung mit maschinellem Lernen noch zu einer anderen Art von Regression in Beziehung gesetzt. Dieses Verfahren wird hier nun noch kurz vorgestellt.

Bei der *least-sqare* Regressionsmethode sucht man Gewichte $\mathbf{w}$, so daß für $\hat{X} = (X|\mathbf{1})$ gilt:

$$\hat{X}\hat{\mathbf{w}} = \mathbf{y}$$

$\mathbf{1}$ bezeichnet hier einfach eine Spalte mit ausschließlich 1 als Koeffizient und $\hat{\mathbf{w}}$ die Kombination von $\mathbf{w}$ und b zu einem Vektor $\hat{\mathbf{w}} = (\mathbf{w}, b)^T$. Ist die Matrix $\hat{X}^T\hat{X}$ invertierbar, so ist die Lösung für $\hat{\mathbf{w}}$ gegeben durch

$$\hat{\mathbf{w}} = \left(\hat{X}^T\hat{X}\right)^{-1}\hat{X}^T\mathbf{y}$$

Hat die Matrix $\hat{X}^T\hat{X}$ nicht vollen Rang oder treten numerische Probleme auf, so kann man folgenden Ansatz wählen

$$\hat{\mathbf{w}} = \left(\hat{X}^T\hat{X} + \gamma\mathcal{I}\right)^{-1}\hat{X}^T\mathbf{y}$$

$\mathcal{I}$ bezeichnet hier die Identitätsmatrix, wobei aber der $\mathcal{I}_{(m+1)(m+1)} = 0$ gesetzt wird, da dies der mit b korrespondierende Koeffizient ist. Der Faktor $\gamma \in \mathbb{R}^+$ heißt *Ridge* und gibt dieser Regressionsmethode ihren Namen. Die Aufgabe von γ wird deutlich, wenn man die Eigenwertzerlegungn von $\hat{X}^T\hat{X} + \gamma\mathcal{I}$ betrachtet.

$$\hat{X}^T\hat{X} + \gamma\mathcal{I} = \mathcal{U}\left(\Lambda + \gamma\mathcal{I}\right)\mathcal{U}^T$$

wobei Λ eine Diagonalmatrix mit den Eigenwerten von $\hat{X}^T\hat{X}$ in der Diagonale und $\mathcal{U}$ die Matrix mit den Eigenvektoren von $\hat{X}^T\hat{X}$ als Spaltenvektoren ist. Bildet man nun die Inverse so gilt

$$\left(\hat{X}^T\hat{X} + \gamma\mathcal{I}\right)^{-1} = \mathcal{U}\left(\Lambda + \gamma\mathcal{I}\right)^{-1}\mathcal{U}^T = \mathcal{U}\begin{pmatrix} \frac{1}{\lambda_1+\gamma} & \cdots & 0 & 0 \\ \vdots & \ddots & \vdots & \vdots \\ 0 & \cdots & \frac{1}{\lambda_m+\gamma} & 0 \\ 0 & \cdots & 0 & \frac{1}{\lambda_{m+1}} \end{pmatrix}\mathcal{U}^T$$

Ist also $\hat{X}^T\hat{X}$ singulär, dann ist γ ein Mindestwert für den jeweiligen Eigenwert λ_k mit $1 \leq k \leq m$, der aufgrund seines geringen bzw. nicht vorhandenen Werts den Einfluß des zugeordneten Eigenvektors über alle Maße erhöhen würde bzw. der Bruch $\frac{1}{\lambda_k}$ wegen einer Division durch Null nicht definiert wäre. γ regularisiert also den Einfluß kleiner Eigenwerte.

Während die normale Regression die Quadrate der Fehler minimiert, ist mit der RIDGE REGRESSION folgende Fehlerfunktion verknüpft

$$l(\mathbf{w}, b, x, y) = \gamma||\mathbf{w}||^2 + \sum_{i=1}^{m}(\langle \mathbf{w}, \mathbf{x}_i\rangle - y_i)^2$$

Durch diese Fehlerfunktion ist die RIDGE REGRESSION die *maximal margin* Version der Methode der kleinsten Quadrate.

$$\gamma||\mathbf{w}||^2 = \gamma\langle \mathbf{w}, \mathbf{w}\rangle$$

kontrolliert den *margin* bzw. die Glattheit der Funktion und somit die Komplexität, während

$$\sum_{i=1}^{m}(\langle w, \mathbf{x}_i\rangle - y_i)^2$$

das empirische Risiko minimiert.

Auch die RIDGE REGRESSION besitzt eine duale Formulierung, in der man statt des kanonischen Skalarprodukts eine Kernelfunktion benutzt. Gesucht sind α, für die gilt

$$(K + \gamma\mathcal{I})\,\alpha = \mathbf{y}$$

wobei K die Grammatrix der jeweiligen Kernelfunktion und $\mathcal{I}$ eine normale $m \times m$ Identitätsmatrix bezeichnet. Somit ergeben sich die α zu

$$\alpha = (K + \lambda\mathcal{I})^{-1} \cdot \mathbf{y}$$

wobei die einzelnen Ausgaben y_i auch mehrdimensional sein können. Die Entscheidungsfunktion ist dann gegeben durch

$$f(x) = \sum_{i=1}^{m} \alpha_i k(x, x_i)$$

Im Gegensatz zur SVR besitzt die RIDGE REGRESSION keine Supportvektoren, es gehen also alle Trainingsbeispiele in die Ausgabe der Entscheidungsfunktion ein. Eine ausführlichere Darstellung findet sich in [6] S.22 ff.

2.2 Grundlagen zur Kameramodellierung

Im Folgenden bezeichnen Großbuchstaben Koordinaten des Objektraums, Kleinbuchstaben mit einem Strich Koordinaten des Kamerakoordinatensystems bzw. des Bildkoordinatensystems und Kleinbuchstaben mit zwei Strichen Koordinaten des Sensorkoordinantensystems.

2.2.1 Perspektivische Abbildung: Lochkameramodell

Das Lochkameramodell beschreibt eine Zentralprojektion des Objektraums[3]. Dabei werden nach den noch zu beschreibenden Gesetzen der perspektivischen Projektion Objektkoordinaten auf die Bildebene des Kamerakoordinatensystems projiziert, welche dort im Abstand c in negativer z-Richtung parallel zur x, y-Ebene liegt. Das "Loch" der Lochkamera, das sog. *Projektionszentrum*, ist dabei in der Regel der Ursprung des Kamerakoordinatensystems.

[3] In dem vorliegenden Fall ist dies das Weltkoordinatensystem des Roboters

Der Ortsvektor zu den Koordinaten eines Punkts P läßt sich danach durch eine Addition des Ortsvektors des Projektionszentrums $\mathbf{X}_0$ der Kamera mit dem Vektor $\mathbf{X}^*$ vom Projektionszentrum zum Objektpunkt beschreiben.

$$\mathbf{X} = \mathbf{X}_0 + \mathbf{X}^* \tag{1}$$

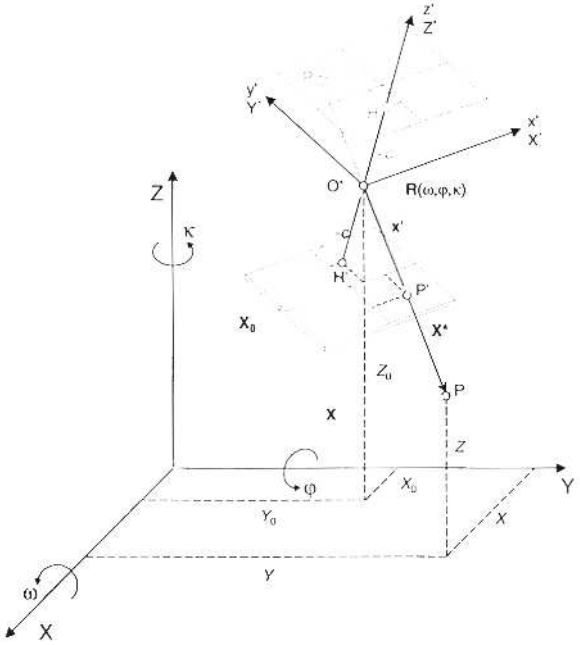

Abb. 2.6 Projektive Abbildung (aus [3] S. 224)

Da der zweite dieser Vektoren aber im übergeordneten Objektkoordinatensystem definiert und deshalb nicht verfügbar ist, greift man deshalb auf den Vektor $\mathbf{x}'$ vom Projektionszentrum zum projizierten Punkt p' zurück, nachdem man ihn mittels einer Skalierung m und Drehung $\mathcal{R}$ in den Objektraum transformiert hat.

$$\mathbf{X}^* = m\mathcal{R}\mathbf{x}' \tag{2}$$

Die Drehmatrix $\mathcal{R}$ setzt sich zusammen aus drei Einzeldrehungen κ, ω und ϕ, die den Roll-, Nick- und Gierwinkel der Kamera beschreiben.

$$\begin{aligned}\mathcal{R} &= \mathcal{R}_\kappa \cdot \mathcal{R}_\phi \cdot \mathcal{R}_\omega \\ &= \begin{pmatrix} \cos\kappa & -\sin\kappa & 0 \\ \sin\kappa & \cos\kappa & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos\phi & 0 & \sin\phi \\ 0 & 1 & 0 \\ -\sin\phi & 0 & \cos\phi \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\omega & -\sin\omega \\ 0 & \sin\omega & \sin\omega \end{pmatrix}\end{aligned}$$

Die Reihenfolge der Matrixmultiplikation und deren Orientierung (rechtshändiges Koordinatensystem), sind den Konventionen der Robotik entliehen (siehe [5]). Danach werden die Drehmatrizen in der Reihenfolge Roll-, Nick- und Gierdrehung, in der Robotik bekannt als *Roll-Pitch-Yaw*, miteinander multipliziert.

Substituiert man nun (2) in (1), so ergibt sich

$$\mathbf{X} = \mathbf{X}_0 + m\mathcal{R}\mathbf{x}' \tag{3}$$

oder ausformuliert

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \begin{pmatrix} X_0 \\ Y_0 \\ Z_0 \end{pmatrix} + m\mathcal{R} \begin{pmatrix} x' \\ y' \\ -c \end{pmatrix}$$

wobei c die Kamerakonstante oder Brennweite bezeichnet. Durch Umkehrung von (3) erhält man

$$\begin{pmatrix} x' - x'_0 \\ y' - y'_0 \\ -c \end{pmatrix} = \frac{1}{m}\mathcal{R}^T \begin{pmatrix} X - X_0 \\ Y - Y_0 \\ Z - Z_0 \end{pmatrix}$$

Werden nun die erste und zweite Gleichung durch die dritte dividiert, so verschwindet der Faktor $\frac{1}{m}$ und man erhält die sog. *Kollinearitätsgleichungen*

$$x' = x'_0 - c \cdot \frac{r_{11}(X - X_0) + r_{21}(Y - Y_0) + r_{31}(Z - Z_0)}{r_{13}(X - X_0) + r_{23}(Y - Y_0) + r_{33}(Z - Z_0)}$$
$$y' = y'_0 - c \cdot \frac{r_{12}(X - X_0) + r_{22}(Y - Y_0) + r_{32}(Z - Z_0)}{r_{13}(X - X_0) + r_{23}(Y - Y_0) + r_{33}(Z - Z_0)}$$

Sie beschreiben die Transformation von Objektkoordinaten $\mathbf{X} = \begin{pmatrix} X \\ Y \\ Z \end{pmatrix}$ in Bildkoordinaten $\mathbf{x}' = \begin{pmatrix} x' \\ y' \end{pmatrix}$. Die z-Koordinate wird hierbei außer Acht gelassen, da sie nach der Projektion konstant $-c$ beträgt. x'_0 und y'_0 sind die Koordinaten des Lotfußpunktes des Projektionszentrums in der Bildebene, dem sog *Bildhauptpunkt*. Dieser muß nicht zwingend mit dem Durchstoßpunkt der z-Achse durch die Bildebene zusammenfallen.

Da bei der Kalibration statt Bildkoordinaten Sensorkoordinaten verwendet werden, muß noch eine Translation $\mathbf{v}$ angehängt werden.

$$\mathbf{x}'' = \mathbf{x}' + \mathbf{v}$$

Der Verschiebungsfaktor $\mathbf{v}$ ergibt sich aus den Dimensionen des CCD-Sensors

$$v_x = x''_{min} + \frac{x''_{max} - x''_{min}}{2}$$
$$v_y = y''_{min} + \frac{y''_{max} - y''_{min}}{2}$$

Will man die Bildkoordinaten in Pixel statt in μm bzw. mm angeben, können die Gleichungen noch durch den Faktor

$$s_{Sensor \to Pixel} = \frac{\text{Breite bzw. Höhe des Bildes in px}}{\text{Breite- bzw. Höhe des Sensors}}$$

angepasst werden.

Da CCD-Sensoren zuweilen unterschiedliche Auflösungsvermögen (px/mm) in x- und y-Richtung besitzen, wird noch ein Skalierungsfaktor s_{xy}, die sog. *Aspect Ratio* eingeführt und ergeben somit die Gleichungen eines einfachen Kameramodells

$$x'' = v_x + x'_0 - s_{xy}c \cdot \frac{r_{11}(X - X_0) + r_{21}(Y - Y_0) + r_{31}(Z - Z_0)}{r_{13}(X - X_0) + r_{23}(Y - Y_0) + r_{33}(Z - Z_0)}$$
$$y'' = v_y + y'_0 - c \cdot \frac{r_{12}(X - X_0) + r_{22}(Y - Y_0) + r_{32}(Z - Z_0)}{r_{13}(X - X_0) + r_{23}(Y - Y_0) + r_{33}(Z - Z_0)}$$

2.2.2 Kompensation von Linsenverzeichnungen

Das eben geschilderte Kameramodell nimmt eine perfekte Linse an, d.h. es beschreibt nicht die in der Einleitung erwähnte Linsenverzeichnungen. Um diesen beizukommen, wird das Modell um zusätzliche Ausgleichsterme $\Xi_x(x', y')$ bzw. $\Xi_y(x', y')$ erweitert.

$$x'' = v_x + x'_0 - s_{xy}c \cdot \frac{r_{11}(X - X_0) + r_{21}(Y - Y_0) + r_{31}(Z - Z_0)}{r_{13}(X - X_0) + r_{23}(Y - Y_0) + r_{33}(Z - Z_0)} + \Xi_x(x', y')$$
$$y'' = v_y + y'_0 - c \cdot \frac{r_{12}(X - X_0) + r_{22}(Y - Y_0) + r_{32}(Z - Z_0)}{r_{13}(X - X_0) + r_{23}(Y - Y_0) + r_{33}(Z - Z_0)} + \Xi_y(x', y')$$

Die Funktion $\Xi : \mathbb{R}^2 \rightarrow \mathbb{R}^2$ definiert ein Vektorfeld, welches in Abhängigkeit der Bildposition einen Verschiebungsvektor angibt, der die Störung des Bildes an dieser Stelle modelliert. Das Vektorfeld sind also diejenigen Vektoren, die man auf die Bilder störungsfrei projizierter Punkte addieren muß, um ihre Position im störungsbehafteten Bild auszumachen.

Im Rahmen dieser Arbeit wird das Vektorfeld durch Summen über Produkte von *Tschebyscheffpolynomen* T_d beschrieben, wobei d den Grad des Tschebyschffepolynoms bezeichnet.

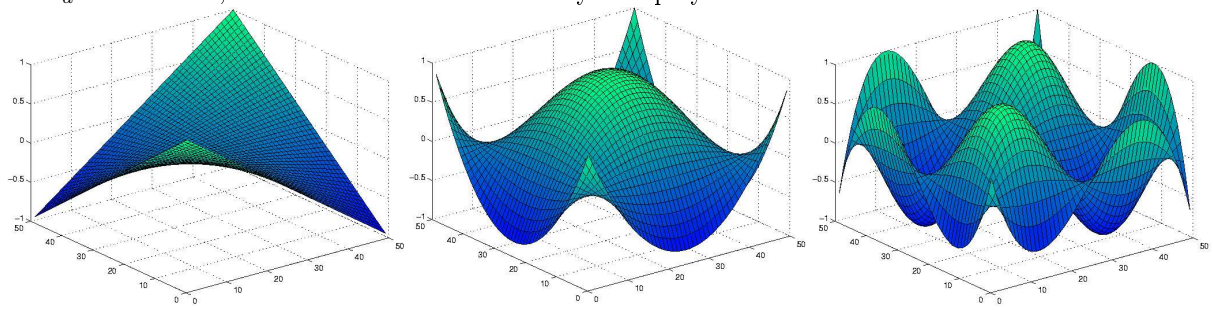

Abb. 2.7 Vektorfelder für Produkte von Tschebyscheffpolynomen ersten (links), dritten (Mitte) und vierten Grades (rechts)

Tschebyscheffpolynome sind auf dem Intervall $[-1;1]$ rekursiv definiert, in welchem sie orthogonal aufeinander stehen.

$$\begin{aligned} T_0(x) &= 1 \\ T_1(x) &= x \\ T_{n+1} &= 2xT_n(x) - T_{n-1}(x) \end{aligned}$$

Die Formeln für die Ausgleichsterme lauten nun

$$\Xi_x(x', y') = \sum_{i,j=0}^{d} a_{ij} T_i(s_x x') T_i(s_y y')$$

bzw.

$$\Xi_y(x', y') = \sum_{i,j=0}^{d} b_{ij} T_i(s_x x') T_i(s_y y')$$

wobei x', y' die von dem Modell errechneten Bildkoordinaten und s_x, s_y Skalierungfaktoren bezeichnen, die dafür Sorge tragen, daß $s_x x'$ bzw. $s_y y'$ genau im Intervall $[-1;1]$ liegen. Die Parameter a_{ij} und b_{ij} kommen zu den Parametern des Kameramodells hinzu und werden ebenfalls in dem im nächsten Abschnitt beschriebenen Regressionsverfahren geschätzt.

2.2.3 Regressionsverfahren

Der hier benutzte Ansatz, um die Parameter des Modells an die Daten zu fitten, ist unter dem Namen *Bündeltriangulation*[4] bekannt. Darunter versteht man die simultane rechnerische Einpassung vieler im Raum angeordneter Bilder (Strahlenbündel) in Bezug zu einem übergeordneten Koordinatensystem.

Das iterative Verfahren zur Berechnung der Parameter kann kurz so zusammengefasst werden:

(1) Alle zu schätzenden Parameter $\mathbf{p}_0 \in \mathbb{R}^p$ des Kameramodells werden mit geeigneten Startwerten, d.h. jenen, die hinreichend nahe am globalen Optimum der Parameter liegen, initialisiert.

(2) Die Raumpunkte des Trainingsdatensatzes werden mit dem Kameramodell und dem derzeitigen Parametersatz $\mathbf{p}$ projiziert und der Fehler $\mathbf{l}$ zu den tatsächlichen Bildpunkten berechnet.

[4] auch *Bündelblockausgleichung*, *Mehrbildtriangulation* oder *Mehrbildorientierung*

(3) An der derzeitigen Stelle im Parameterraum wird das Modell linearisiert und aus dem Fehler des Kameramodells eine Verbesserung $\hat{\theta}$ der Parameter berechnet.

$$p_{t+1} = p_t + \hat{\theta}$$

(4) Die Prozedur wird ab Schritt (2) solange wiederholt, bis die Verbesserungen der Parameter bzw. der Fehler des Kameramodells hinreichend klein ist.

Im Rest dieses Abschnitts wird nun v.a. Schritt (3) genauer betrachtet. Folgende kleine Rechnung stellt den Bezug zwischen der Verbesserung der Parameter θ und dem Fehler auf den bekannten Punkten her. $\mathbf{x}$ bezeichnet hierbei die Bildkoordinaten und $\mathbf{p}$ die Parameter. Der Index t bezeichnet den Iterationsschritt und der Stern die Werte am gesuchten Optimum.

$$\mathbf{x}'^{*} - \mathbf{x}'_t = f(\mathbf{p}^*) - f(\mathbf{p}_t)$$

Wird nun $f(\mathbf{p}^*)$ linearisiert, d.h. durch eine Taylorentwicklung an der Stelle $\mathbf{p}_t$ mit Abbruch nach dem ersten Glied ausgedrückt, dann erhält man folgende Form

$$\begin{aligned} \mathbf{x}'^{*} - \mathbf{x}'_t &= f(\mathbf{p}_t) + \mathcal{B}(p_t - p^*) + \mathbf{w} - f(\mathbf{p}_t) \\ \mathbf{l} &= \mathcal{B}(p_t - p^*) + \mathbf{w} \\ \mathbf{l} &= \mathcal{B}\theta + \mathbf{w} \end{aligned}$$

$\mathbf{w}$ bezeichnet hier das Restglied der Taylorentwicklung oder, anders interpretiert, Rauschen in den Daten. Zwischen dem Fehler $\mathbf{l}$ und den gesuchten Verbesserungen θ ergibt sich somit folgende lokale Beziehung

$$\mathbf{l} = \mathcal{B}\theta + \mathbf{w}$$

wobei θ die wahren Verbesserungen der Parameter sind und im Idealfall $\mathbf{w} \sim \mathcal{N}(0, \Sigma_{\theta\theta})$ ist. D.h. es wird angenommen, daß Fehler und Parameter in einem linearem Zusammenhang $\mathcal{B}\theta$ stehen und mit zusätzlichem Rauschen $\mathbf{w}$ belegt sind.

Die Schätzung der Parameterverbesserungen entspricht also einer Linearisierung des nichtlinearen Kameramodells am Punkt $\mathbf{p}_t$. Die eben genannte Beziehung läßt sich nun auch wie folgt formulieren ($\mathbb{E}$ bezeichnet den Erwartungswert):

$$\mathbb{E}(\mathbf{l}) = \mathcal{B}\theta$$

bzw.

$$\hat{\theta} = \mathcal{A}\mathbf{l}$$

Der Name BLUE-Schätzer kommt von *best linear unbiased estimator* , wobei *unbiased* (siehe auch unbiased) dem deutschen Fachausdruck *erwartungtreu* entspricht. Ein BLUE-Schätzer ist, wie der Name schon verrät, linear in den Daten und erwartungtreu, bei gleichzeitiger Minimierung der Varianz der zu schätzenden Parameter. Erwartungtreue bedeutet allgemein, daß

$$\mathbb{E}(\hat{\theta}) = \mathbb{E}(\mathcal{A}\mathbf{l}) = \mathcal{A}\mathbb{E}(\mathbf{l}) = \theta$$

wobei θ den wahren Wert der zu schätzdenden Parameterverbesserungen bezeichnet. Bei erwartungtreuen Schätzern ist also der Erwartungswert der geschätzten Parameter der wahre Wert des Parameters selbst.

$\mathcal{A}$ ist hier eine $p \times (2m)$-Matrix, wobei $(2m)$ die Dimension des Residuumsvektors $\mathbf{l}$ ist (ein Residuum für jede Raumrichtung des Kamerakoordinatensystems). Aus der Erwarungtreue ergibt sich nun

$$\mathbb{E}(\mathbf{l}) = \mathcal{B}\theta$$

wobei $\mathcal{B}$ eine bekannte $(2m) \times p$-Matrix ist, mit der Bedingung $\mathcal{AB} = \mathcal{I}$ oder auch $\mathbf{a}_i\mathbf{b}^j = \delta_{ij}$. $\mathbf{a}_i$ und $\mathbf{b}^j$ bezeichnen die i-te Zeile bzw. die j-te Spalte ihrer Matrix. Das sog. *Kronecker-Symbol* δ_{ij} ist gleich Eins, wenn $i = j$ und Null sonst. Die Bedingungen für Erwartungtreue lauten also

$$\mathbf{a}_i\mathbf{b}^j = \delta_{ij}$$

Im dem vorliegenden Fall ist $\mathcal{B}$ eine Matrix aus den Jacobimatrizen des Kameramodells f an allen Datenpunkten der Trainingsmenge. Das Modell wird also als eine Funktion

$$f_{X_j}(\mathbf{p}) = \begin{pmatrix} x' \\ y' \end{pmatrix}$$

der Parameter, parametrisiert durch den jeweiligen Datenpunkt X_j, aufgefasst. Ist

$$\mathcal{J}_j = \begin{pmatrix} \frac{\partial f_{x,X_j}}{\partial p_1} & \cdots & \frac{\partial f_{x,X_j}}{\partial p_p} \\ \frac{\partial f_{y,X_j}}{\partial p_1} & \cdots & \frac{\partial f_{y,X_j}}{\partial p_p} \end{pmatrix}$$

die Jacobimatrix an der Stelle des durch X_j parameterisierten Kameramodells an der Stelle $\mathbf{p}_t$, dann ist

$$\mathcal{B} = \begin{pmatrix} \mathcal{J}_1 \\ \vdots \\ \mathcal{J}_m \end{pmatrix}$$

Die Varianz der Parameterverbesserungen ergibt sich nun wie folgt

$$\begin{aligned} Var(\hat{\theta}) &= \mathbb{E}\left((\mathcal{A}\mathbf{l} - \mathcal{A}\mathbb{E}(\mathbf{l}))^2\right) \\ &= \mathbb{E}\left((\mathcal{A}(\mathbf{l} - \mathbb{E}(\mathbf{l})))^2\right) \\ &= \mathbb{E}\left((\mathcal{A}(\mathbf{l} - \mathbb{E}(\mathbf{l}))(\mathbf{l} - \mathbb{E}(\mathbf{l}))^T\mathcal{A}^T)\right) \\ &= \mathcal{A}\Sigma_{ll}\mathcal{A}^T \end{aligned}$$

Σ_{ll} bezeichnet hier die Kovarianzmatrix der Bildfehler der einzelnen Datenpunkte. Sie ist nicht gegeben und muß geschätzt werden. Siehe hierzu auch die Zusammenfassung des Algorithmus weiter unten.

Der BLUE-Schätzer, d.h. die Matrix $\mathcal{A}$ wird durch die Minimerung der eben hergeleiteten Varianz unter den Nebenbedingungen für Erwartungstreue berechnet. Dies kann wiederum mit der Methode der Lagrangeschen Multiplikatoren erreicht werden. Die analytische Lösung des Problems ergibt dabei

$$\hat{\theta} = (\mathcal{B}^T\Sigma_{ll}^{-1}\mathcal{B})^{-1}\mathcal{B}\Sigma_{ll}^{-1}\mathbf{l}$$

Gleichzeitig ist die Kovarianzmatrix der Parameter gegeben durch

$$\Sigma_{pp} = (\mathcal{B}^T\Sigma_{ll}^{-1}\mathcal{B})^{-1}$$

Für eine ausführlichere Herleitung des BLUE-Schätzers siehe [2] S.133-141, für die ausformulierte Minimierung unter Nebenbedingungen [2] S.153 ff.

Zusammenfassend hier nocheinmal der Algorithmus zur Schätzung der Kameraparameter

1. Initialisierung:
   $\mathbf{p}_0$ wird mit guten Anfangsschätzwerten belegt. Diese können aus der räumlichen Anordnung der zu kalibrierenden Kamera entnommen werden. Die Parameter zur Kompensation der Linsenverzeichnung werden mit Null initialisiert.

   Die Kovarianzmatrix der Residuen ist nicht bekannt und muß deshalb ebefalls mit einer Schätzung initialisiert werden. Im Nachhinein kann überprüft werden, ob die Schätzung gut war oder nicht.

Die initiale Schätzung von s_0 wird dann als gut angesehen, wenn ihr a posteriori Wert $\hat{s}_0$ nicht zu sehr von s_0 abweicht. Darüber hinaus werden die Fehler als unkorreliert angenommen, wodurch die Kovarianzmatrix zur Diagonalmatrix wird. Sei

$$\mathbf{v} = s_0 \begin{pmatrix} s_x \\ s_y \end{pmatrix}$$

der Vektor mit einer Anfangsschätzung der Standardabweichung der Bildfehler gewichtet mit s_0. Dann ist Σ_{ll} eine Diagonalmatrix, die m-mal $\mathbf{v}$ in der Diagonale enthält.

2. Berechnung des Residuums:
Das Residuum $\mathbf{l}$ besteht aus den konkatenierten Residuen aller Datenpunkte also

$$\mathbf{l} = \begin{pmatrix} x_1'^* - f_{x,X_1}(\mathbf{p}_t) \\ y_1'^* - f_{y,X_1}(\mathbf{p}_t) \\ \vdots \\ x_m'^* - f_{x,X_m}(\mathbf{p}_t) \\ y_m'^* - f_{y,X_m}(\mathbf{p}_t) \end{pmatrix}$$

3. Berechnung der konkatenierten Jacobimatrix:
Die konkatenierte Jacobimatrix $\mathcal{B}$ wird wie oben beschrieben berechnet. Dies kann sowohl numerisch, als auch analytisch geschehen. Steht Mathematik-Software zur Verfügung, so ist die analytische Variante wegen ihrer höheren Genauigkeit und dem später geringeren Zeitverbrauch vorzuziehen.

4. Verbesserung der Parameter:
Die Verbesserung der Parameter wird wie oben beschrieben nach der Formel

$$\hat{\theta} = (\mathcal{B}^T \Sigma_{ll}^{-1} \mathcal{B})^{-1} \mathcal{B} \Sigma_{ll}^{-1} \mathbf{l}$$

berechnet. Die verbesserten Parameter sind dann

$$\mathbf{p}_{t+1} = \mathbf{p}_t + \hat{\theta}$$

5. Abbruchbedingung:
Wenn der Fehler klein genug ist bzw. die Parameterverbesserung hinreichend klein ist, so daß sie keine merkliche Verbesserung mehr erzielt, dann wird die Schleife abgebrochen, ansonsten mit Punkt 2. fortgefahren.

2.2.4 Stereokameramodell

Durch die im vorhergehenden Abschnitt beschriebene Regressionsmethode kann sowohl die äußere als auch die innere Orientierung einer Einzelkamera bestimmt werden. Kennt man diese nun für beide Kamerapartner, so kann man sie zu einem Stereokameramodell verbinden. Dieses besteht einfach aus den beiden Einzelkameraparametrisierungen, dem dem die Tiefe eines in beiden Kamerabildern sichtbaren Punktes berechnet werden kann. Dazu stehen die jeweils zwei Kollinearitätsgleichungen der Einzelkameras zur Verfügung.

$$x_l' = -c_l \cdot s_{xy,l} \frac{r_{l,11}(X - X_{l,0}) + r_{l,21}(Y - Y_{l,0}) + r_{l,31}(Z - Z_{l,0})}{r_{l,13}(X - X_{l,0}) + r_{l,23}(Y - Y_{l,0}) + r_{l,33}(Z - Z_{l,0})} \tag{4}$$

$$y_l' = -c_l \cdot \frac{r_{l,12}(X - X_{l,0}) + r_{l,22}(Y - Y_{l,0}) + r_{l,32}(Z - Z_{l,0})}{r_{l,13}(X - X_{l,0}) + r_{l,23}(Y - Y_{l,0}) + r_{l,33}(Z - Z_{l,0})} \tag{5}$$

$$x_r' = -c_r \cdot s_{xy,r} \frac{r_{r,11}(X - X_{r,0}) + r_{r,21}(Y - Y_{r,0}) + r_{r,31}(Z - Z_{r,0})}{r_{r,13}(X - X_{r,0}) + r_{r,23}(Y - Y_{r,0}) + r_{r,33}(Z - Z_{r,0})} \tag{6}$$

$$y_r' = -c_r \cdot \frac{r_{r,12}(X - X_{r,0}) + r_{r,22}(Y - Y_{r,0}) + r_{r,32}(Z - Z_{r,0})}{r_{r,13}(X - X_{r,0}) + r_{r,23}(Y - Y_{r,0}) + r_{r,33}(Z - Z_{r,0})} \tag{7}$$

Zur Berechnung der Tiefe werden drei dieser Gleichungen nach den Variablen X, Y und Z gelöst.

Bevor die Tiefe bzw. allgemein die Raumkoordinaten eines Punktes berechnet werden können, müssen die Bildkoordinaten dieses Punktes zunächst entstört werden, d.h. die Linsenverzeichnungen, mit welchen dieser Punkt behaftet ist, werden entfernt.

2.2.5 Entfernen der Linsenverzeichnungen

Wie schon erwähnt definieren die beschriebenen Verzeichnungsparameter ein Vektorfeld auf den Bildkoordinaten. Sie geben die Translation an, die ein perfekt projizierter Raumpunkt durch die Linsenverzeichnungen erfährt.

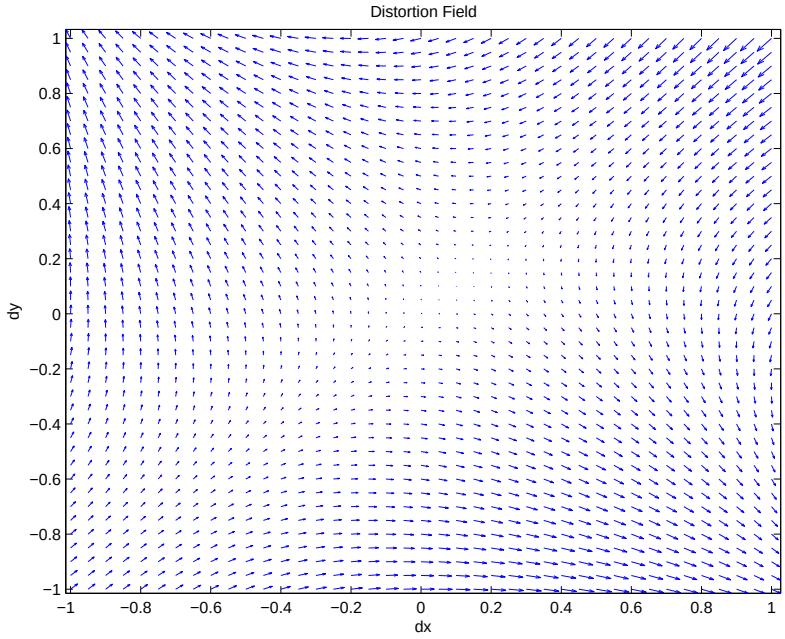

Abb. 2.8 Beispiel für ein Verzeichungsfeld einer Kamera.

Um diese aus verzeichneten Bildpunkten, also den aufgenommenen Daten, herauszurechnen, muß die Abbildung invertiert werden. Da die Funktion

$$\Xi_x(x', y') = \sum_{i,j=0}^{d} a_{ij} T_i(s_x x') T_i(s_y y')$$

bzw.

$$\Xi_y(x', y') = \sum_{i,j=0}^{d} b_{ij} T_i(s_x x') T_i(s_y y')$$

i.d.R. nicht komplett invertierbar ist, wird sie stattdessen nur lokal invertiert. Hierzu wird das mehrdimensionale Newtonverfahren eingesetzt, um mit dessen Hilfe die Differenz zwischen gestörtem Bildpunkt und entstörtem Bildpunkt plus Störung zu minimieren. Dies entspricht einer Nullstellensuche auf dieser Fehlerfunktion. Der Bildpunkt, der diese Fehlerfunktion minimiert, ist der gesuchte, entstörte Bildpunkt, falls die Funktion lokal an dieser Stelle bijektiv ist.

$$F(\mathbf{x}', \mathbf{v}') = \begin{pmatrix} v_1' \\ v_2' \end{pmatrix} - \begin{pmatrix} x_1' \\ x_2' \end{pmatrix} - \begin{pmatrix} \Xi_x(x_1', x_2') \\ \Xi_y(x_1', x_2') \end{pmatrix}$$

Der verzeichnete Bildpunkt wird hierbei mit $\mathbf{v}'$, der gesuchte Verzeichnungsfreie mit $\mathbf{x}'$ bezeichnet. F ist genau dann Null, wenn $\mathbf{x}'$ der Punkt ist, der mit den Linsenstörungen den beobachteten Punkt $\mathbf{v}'$ ergibt.

Das Newton-Verfahren wird nun iterativ durchgeführt, d.h. die Fehlerfunktion wird an der bis dahin besten Stelle linear approximiert und Verbesserungen für den entstörten Bildpunkt berechnet. Der verbesserte Bildpunkt liefert im nächsten Iterationsschritt den bis dahin besten Punkt. Dies wird solange durchgeführt bis der Fehler die Verbesserung hinreichend klein sind.

$$\mathbf{x}'_{n+1} = \mathbf{x}'_n + \mathcal{D}^{-1} F(\mathbf{x}'_n)$$

$\mathcal{D}$ bezeichnet hierbei die Jacobimatrix der Funktion F am Punkt x_n.

$$\mathcal{D} = \begin{pmatrix} \frac{\partial F_1}{\partial x'_1} & \frac{\partial F_1}{\partial x'_2} \\ \frac{\partial F_2}{\partial x'_1} & \frac{\partial F_2}{\partial x'_2} \end{pmatrix}$$

Als Startwert $\mathbf{x}'_0$ wird der verzeichnete Bildpunkt $\mathbf{v}'$ benutzt. Ist also die Funktion in einem Kreis mit Radius $||\mathbf{v}' - \mathbf{x}^*||_2$ um $\mathbf{v}'$ invertierbar, dann ist $\mathbf{x}^*$ das tatsächliche Urbild von $\mathbf{v}'$ unter dem Vektorfeld Ξ.

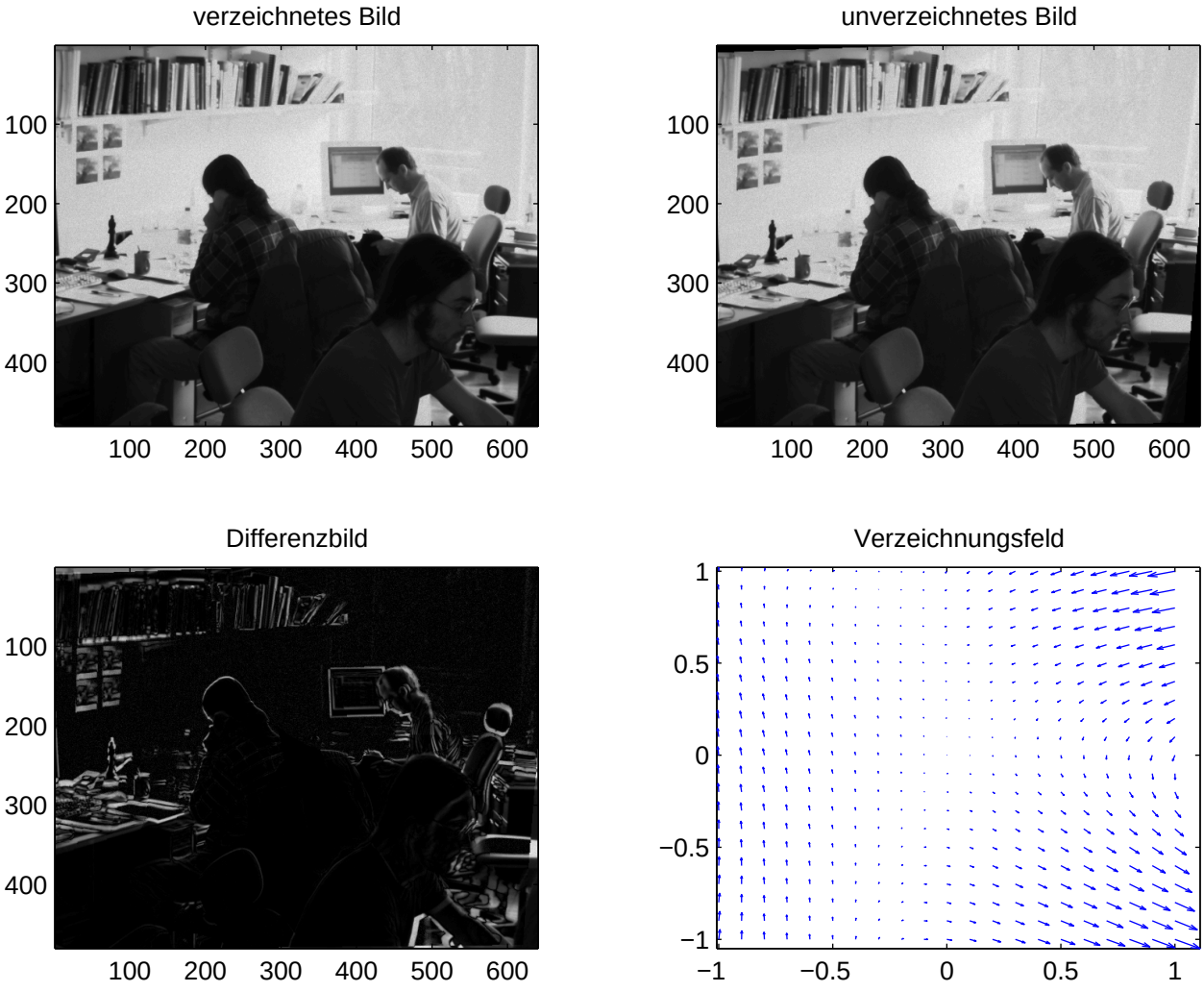

Abb. 2.9 Beispiel für die Entstörung eines gesamten Bildes. Die sehr starken Störungen in der rechten Bildhälfte müssen nicht der Realität entsprechen, da das Verzeichnungsfeld anhand von Datenpunkten in der linken Bildhälfte geschätzt wurden.

3 Lösungsmethoden und ihre Umsetzung

In diesem Kapitel werden diejenigen Methoden beschrieben, die zur Lösung der gestellten Aufgaben eingesetzt wurden. Hierbei werden die im theoretischen Teil geschilderten Verfahren konkretisiert sowie neue, denen keine zentrale Rolle zukam, kurz anhand der von ihnen zu bewältigenden Aufgabe beschrieben.

Das Kapitel ist in drei Abschnitte unterteilt. Der erste Abschnitt schildert die Aufnahme der Daten der Form

$$\mathbf{x}_i = (x_{il}, y_{il}, x_{ir}, y_{ir})$$

und

$$\mathbf{y}_i = (X_i, Y_i, Z_i)$$

die als Grundlage für die Bestimmung der Kameraparameter bzw. für den überwachten Lernprozess dienen. $(x_{il}, y_{il}, x_{ir}, y_{ir})$ bezeichnen hier die Bildkoordinaten der beiden Kamerabilder, wobei der Index i die Nummer des aufgenommenen Datums bezeichnet und die Indizes l (links) und r (rechts) die Kamera angeben, mit der die jeweiligen Datenpunkte aufgenommen wurden. (X_i, Y_i, Z_i) bezeichnen die Raumkoordinaten, gewonnen durch die Position des Roboterarms. X gibt hierbei die Breite, Y die Höhe und Z die Tiefe an.

Der erste Unterabschnitt befasst sich mit der Detektion und Mittelpunktsbestimmung der LED im Bild, der zweite mit der Bestimmung der Objektraumpunkte $\mathbf{y}_i$ und ihrer räumlichen Verteilung.

Der zweite Abschnitt konkretisiert die theoretischen Grundlagen aus Teil 2.2. Zuerst wird beschrieben, welche der Kameraparameter durch die Regression geschätzt werden sollen, welche Korrelationen zwischen ihnen bestehen und wie die unerwünschten von ihnen unterbunden werden. Daraufhin wird die genau Verschmelzung der beiden Einzelkameramodell zu einem Stereokameramodell vorgestellt.

Im dritten und letzten Abschnitt wird die Wahl des Kernels zur Kalibrierung mit maschinellem Lernen motiviert und die Suche der Parameter für die Ridge Regression bzw. SVR beschrieben. Beide bestehen im engeren Sinn aus drei einzelnen Algorithmen, welche unabhängig auf die einzelnen Ausgabedimensionen lernen. Da sich die Abbildung auf diese nur geringfügig unterscheiden, wird für alle drei derselbe Parametersatz verwandt.

3.1 Datenaquisition

3.1.1 Detektion des Kalibrationsobjekts

Die Detektion des Kalibrationsobjekts gliedert sich in zwei Abschnitte. Zuerst wird die LED mittels einer Schwellwertoperation lokalisiert. Dann wird in einem kleineren Bildfenster um die Diode herum der genaue Mittelpunkte der Ellipse bestimmt.

Um den Schwellwert zu bestimmten, mit dem entschieden werden soll, an welcher Stelle des Bildes sich die Diode befindet, wird zunächst ein Histogramm der Grauwerte des Bildes bestimmt, in dem sie lokalisiert werden soll. Der Grauwert, der von einem gewissen Prozentsatz p überschritten wird, das sog. *p-Perzentil* liefert den Schwellwert für die Detektion der LED. In der Praxis hat sich $p = 0.01\%$ bewährt. Der Median der Bildindizes, deren Grauwert den Schwellwert überschreitet, liefert die Vorschätzung des Mittelpunktes der LED.

Nun wird ein Fenster um den ermittelten Punkt gelegt. Die Fenstergröße wird hierbei so gewählt, daß möglichst alle Punkte, deren Grauwert über 80% des ermittleten Schwellwerts liegen, in dem Fenster enthalten sind. Sollte dabei in irgendeine Richtung der Bildrand erreicht werden bzw. wird das Fenster zu groß, wird die Messung nicht in die Daten mitaufgenommen, da im ersten Fall die Möglichkeit besteht, daß die Diode nicht komplett im Bild zu sehen ist und somit der Mittelpunkte nicht genau bestimmt werden kann und im zweiten Fall davon ausgegangen werden kann, daß die Prälokalisierung der Diode diese nicht erfaßt hat[5].

Um die Genauigkeit zu erhöhen, mit der letztlich der Mittelpunkt bestimmt werden kann, wird ein zweidimensionaler, kubischer Spline (siehe Definition 6.1.3) durch die Grauwerte des Fensters errechnet, so daß dieses nun aus der fünffachen Anzahl von Bildpunkten in x- und y-Richtung besteht.

[5]Es hat sich z.B. herausgestellt, daß die Reflektion von Sonnenlicht auf dem Roboterarm solche Effekte hervorrufen kann

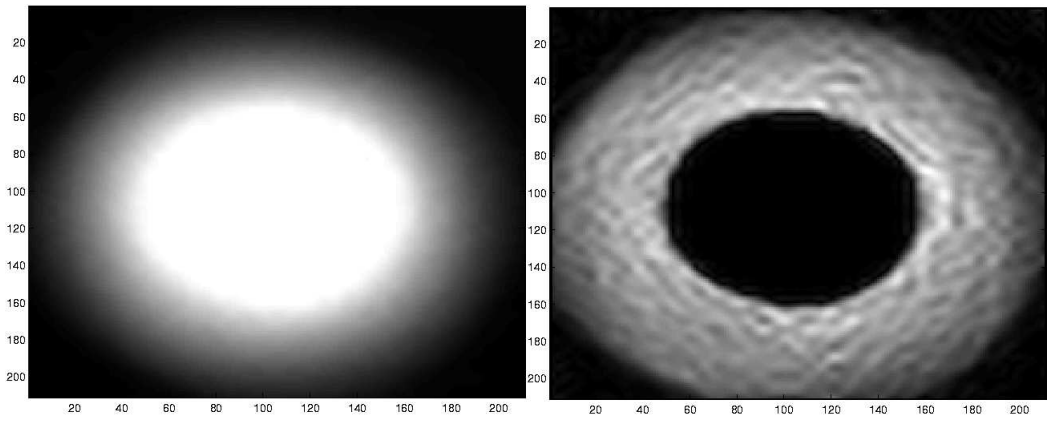

Abb. 3.0 2D-Spline durch die Grauwerte des Fensters um die Diode (links) und dessen 1. Ableitung mit einem Sobel-Operator (rechts)

Da lediglich der Rand der Diodenfläche von Interesse ist, wird nun ein Kantendetektor auf das Fenster angewandt. Der Kantendetektor, ein sog. *Sobel-Operator*, ermittelt die erste Ableitung der Grauwerte des vorliegenden Bildes. Dies wird mit Filtermasken der Form

$$H_x = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \text{ und } H_y = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}$$

erreicht. Der neue Grauwert eines Pixel errechnet sich durch die mit den Filterkoeffizienten gewichtete Summe der umliegenden Grauwerte. Dies wird sowohl in x- als auch in y-Richtung ausgeführt und die Grauwerte der Filter addiert. Genauere Darstellung des Sobel-Operators findet sich in [3] S. 372f.

Nun werden von der Vorschätzung des Mittelpunkts aus mittels einer erneute Schwellwertoperation die Ränder der LED in beide Bildrichtungen ermittelt, d.h. von der Vorschätzung aus werden vertikal und horizontal in beide Richtungen diejenigen Koordinaten ermittelt, deren Grauwert den Schwellwert als erstes überschreiten. In der folgenden Abbildung sind diese Punkte vertikal durch einen grünen Kreis gekennzeichnet. Nun kommt der *Zhou-Operator* zu Anwendung. Hierbei werden zunächst die Schnittpunkte von parallelen Sehnen durch die Ellipse mit deren Rand berechnet. Wird dies für horizontale und vertikale Sehnen ausgeführt, so ergeben die Mittelwerte von jeweils zwei auf einer Sehne liegenden Punkten eine vertikale und eine horizontale Punktwolke.

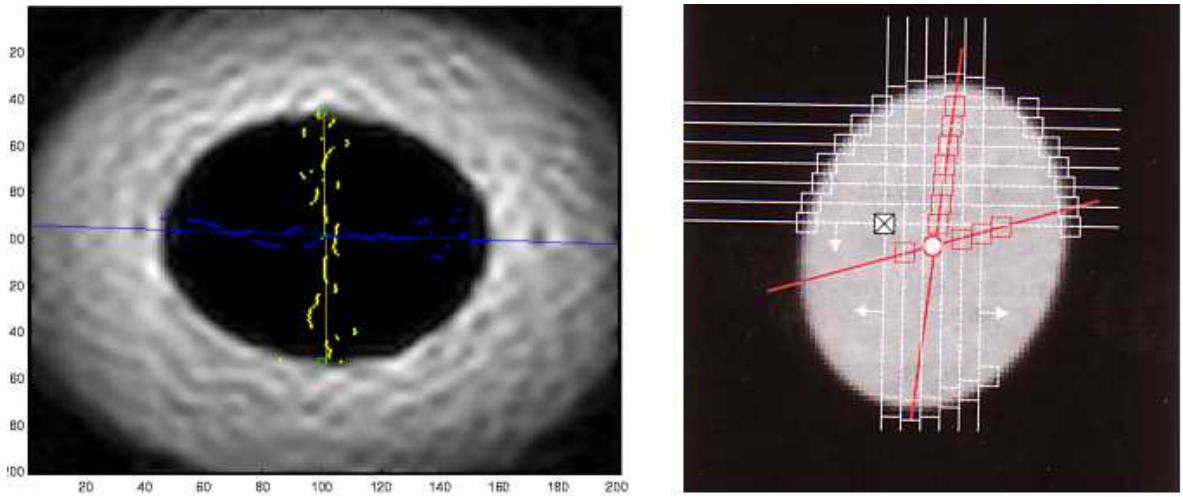

Abb. 3.1 Abbildung des Zhou-Operators wie er in dieser Arbeit verwendet wurde und Schaubild aus [3]. Der grüne Punkt ist die Stelle der Vorschätzung, das grüne Kreuz der tatsächliche Mittelpunkt der LED-Fläche. Die grünen Kreise kennzeichnen die vertikalen Ränder der LED auf der Höhe der Mittelpunktsvorschätzung.

Durch diese werden zwei *Regressionsgeraden* gelegt, deren Schnittpunkt den Mittelpunkt der Diodenfläche bestimmt. Genauere Ausführungen über den Zhou-Operator finden sich in [3] S. 409f.

3.1.2 Aufnahme der Datenpunkte

Dieser Abschnitt beschreibt die Art und Weise in der die Punkte für einen Datensatz aufgenommen werden.

Die Daten werden in Ebenen verschiedener Tiefen aufgenommen. Der Benutzer spezifiziert hierbei zunächst die Eckpunkte des Kubus, in welchem sich die Ebenen befinden sollen, die Auflösung einer einzelnen Ebene und ihre Anzahl. Die Eckpunkte werden manuell bestimmt, d.h. der Roboterarm wird an die gewünschte Position der Eckpunkte manövriert, es wird überprüft, ob sich in dieser Position die Diode noch fehlerfrei lokalisieren läßt und, ist dies der Fall, der Eckpunkt gesetzt[6]. Nun werden vier Geraden durch die Eckpunkte der Flächen in maximaler und minimaler Tiefe ermittelt. Ist $\mathbf{X}_{min}$ ein Eckpunkt der Ebene in minimaler Tiefe und $\mathbf{X}_{max}$ der gleiche Eckpunkt der Ebene in maximaler Tiefe, dann ist die Geradegleichung gegeben durch

$$g : \mathbf{Y} = \mathbf{X}_{min} + \lambda(\mathbf{X}_{max} - \mathbf{X}_{min})$$

Der Einfachheit halber wird lediglich das Inkrement berechnet, das zu einem Eckpunkt addiert wird, um den Nächsten zu erhalten.

$$\mathbf{X}_{n+1} = \mathbf{X}_n + \underbrace{\frac{\mathbf{X}_{max} - \mathbf{X}_{min}}{\#\text{Ebenen}}}_{\text{Inkrement}}$$

Da der Term $\frac{\mathbf{X}_{max} - \mathbf{X}_{min}}{\#\text{Ebenen}}$ konstant ist, braucht er für jede Gerade nur einmal berechnet werden.

[6] Die Eckpunkte sind im Roboterkoordinatensystem gegeben

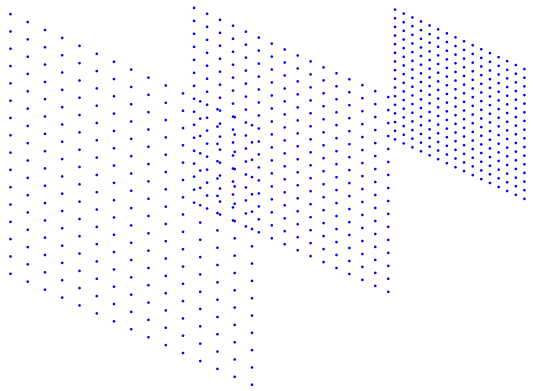

Abb. 3.2 Schema der räumlichen Anordnung der aufzunehmenden Datenpunkte

Auf diese Weise ergeben sich für jede Tiefe vier Eckpunkte eines Rechtecks, in welchem die Daten dann spaltenweise aufgenommen werden. Die Geraden, an denen sich der Arm in einer Tiefenebene entlang bewegt, werden auf die gleiche Art berechnet.
An jedem Gitterpunkt wird nun in den Bildern beider Kameras die LED lokalisiert und deren Bildkoordinaten sowie die Raumposition der LED gepeichert. Diese erhält man durch $\mathbf{X}_{LED} = \mathbf{X}_{Hand} + l \cdot \mathbf{Z}$, wobei $\mathbf{X}_{Hand}$ die räumliche Position der Roboterhand, $\mathbf{Z}$ die Richtung in der sie deutet (also die z-Achse des Handkoordinatensystems) und l die Länge des Kalibrationsobjekts bezeichnet. Die Stellgenauigkeit des Roboters von 0.01mm bildet hierbei die Grundlage für die Genauigkeit der späteren Kalibrierung durch Modell und Lernalgorithmen.

Bei der klassischen Kalibrierung wäre die Miteinbeziehung von Rauschen in den Ausgabedaten grundsätzlich möglich, wurde aufgrund der hohen Genauigkeit nicht implementiert.

Auf die eben beschriebene Weise erhält man nun einen Datensatz der Form

$$\begin{array}{cc} (x_{1l}, y_{1l}, x_{1r}, y_{1r}) & (X_1, Y_1, Z_1) \\ \vdots & \\ (x_{ml}, y_{ml}, x_{mr}, y_{mr}) & (X_m, Y_m, Z_m) \end{array}$$

3.2 Klassischer modellbasierter Ansatz

3.2.1 Korrelation zwischen Haupt- und Verzeichnungsparametern

Auf die oben beschriebene Art und Weise der Bestimmung der Einzelkameraparameter ist es nicht möglich alle Haupt- und Verzeichnungsparameter der Kamera zu schätzen, da einige Parameter lineare Abhängigkeiten bzw. hohe Korrelationen aufweisen. Deswegen muß vor der Kalibrierung festgestellt werden, welche Parameter man schätzen kann.

Zu diesem Zweck betrachtet man zunächst die durch die Regressionmethode erhaltenen Kovarianzmatrix der Parameter. Diese erhält man, indem man zunächst die a posteriori Schätzung der Gewichtseinheit s_0 bestimmt. Sodann wird der Prädikationsfehler des gelernten Modells $\mathbf{v} = \mathcal{B}\hat{\theta} + \mathbf{l}$ bestimmt. Mit der Anfangsschätzung der Kovarianzmatrix der Residuen Σ_{ll}, m als Anzahl der Trainingspunkte und u der Anzahl der Unbekannten ergibt sich die a posteriori Schätzung $\hat{s}_0$ zu

$$\hat{s}_0 = \sqrt{\frac{\mathbf{v}^T \Sigma_{ll} \mathbf{v}}{2m - u}}$$

Die Differenz $m - u$ wird hierbei mit *Redundanz* bezeichnet. Vermittels dieser Schätzung ergibt sich die Kovarianzmatrix der Parameter

$$\Sigma_{pp} = \begin{pmatrix} \sigma_{p_1}^2 & \rho_{p_1 p_2}\sigma_{p_1}\sigma_{p_2} & \cdots & \rho_{p_1 p_m}\sigma_{p_1}\sigma_{p_m} \\ \rho_{p_2 a_1}\sigma_{p_2}\sigma_{p_1} & \sigma_{p_2}^2 & \cdots & \rho_{p_2 p_m}\sigma_{p_2}\sigma_{p_m} \\ \vdots & \vdots & \ddots & \vdots \\ \rho_{p_m p_1}\sigma_{p_m}\sigma_{p_1} & \rho_{p_m p_2}\sigma_{p_m}\sigma_{p_2} & \cdots & \sigma_{p_m}^2 \end{pmatrix}$$

zu

$$\Sigma_{pp} = \hat{s}_0^2 (\mathcal{B}^T \Sigma_{ll}^{-1} \mathcal{B})^{-1}$$

Wie man an der obigen Formulierung der Kovarianzmatrix gut erkennen kann, lassen sich nun aus ihr leicht die Korrelationskoeffizienten der einzelnen Parameter berechnen

$$\rho_{ij} = \frac{(\Sigma_{pp})_{ij}}{\sqrt{(\Sigma_{pp})_{ii}}\sqrt{(\Sigma_{pp})_{jj}}}$$

Anhand dieser Korrelationskoeffizienten kann man nun stark korrelierende Parameter ausfindig machen und gegebenenfalls ausschalten. Zur Verifizierung kann man zusätzlich noch die Bewegungsfelder der Bildpunkte betrachten, die eine Veränderung je einer der Parameter hervorruft bzw. die Einträge der Jacobimatrix analytisch berechnen und so eventuelle lineare Abhängigkeiten feststellen. Beide Vorgehensweisen seien hier noch exemplarisch an jeweils einem Parameter dargestellt.

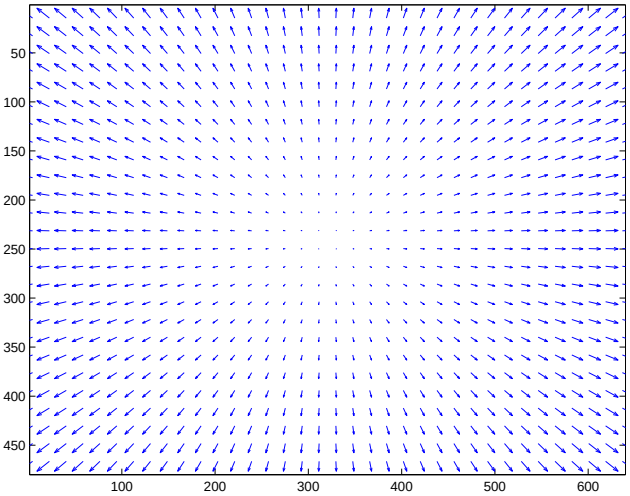

Abbildung 3.3 zeigt das durch $a_{10} = b_{01} = 1$ erzeugte Verzeichnungsfeld der Tschebyscheffpolynome. Ein derartiges Feld kann auch beobachtet werden, wenn die Brennweite der Kamera $f \approx c$ verändert wird. Dies kann man sich leicht veranschaulichen, indem man sich vorstellt, durch eine Kamera zu blicken, während deren Fokussierung verändert wird. Bildpunkte in der Mitte des Bildes werden sich kaum verändern, aber je weiter außen dieser Punkt liegt, desto größer wird seine Verschiebung sein.
Dementsprechend werden also die Parameter a_{10} und b_{01} eine hohe Korrelation mit c aufweisen und sollten ausgeschaltet werden. Wahlweise kann man auch Restriktionen auf den Verzeichnungsparameter einführen. So kann z.B. $a_{10} = -b_{01}$ gesetzt werden, da ja nur die Kombination $a_{10} = b_{01}$ das unerwünschte Flußfeld erzeugt.

Abb. 3.3 Flußfeld der Bildpunkte für $a_{10} = b_{01}$

Dabei ist darauf zu achten, daß dann entweder der Parameter b_{01} oder a_{10} geschätzt wird und sich mit dieser Restriktion auch die Jacobimatrizen und damit die $\mathcal{B}$-Matrix der Regression verändert.

Die Methode, lineare Abhängigkeiten anhand der partiellen Ableitung zu erkennen, ist nur bei Koeffizienten praktikabel, deren partielle Ableitung überschaubar ist. Ein Beispiel hierfür sind die partiellen Ableitungen der Abszisse der Bildkoordinate nach der des Hauptpunkts x_h bzw. dem Verzeichnungsparameter a_{00}.

$$\frac{\partial x'}{\partial x_h} = 1 = \frac{\partial x'}{\partial a_{00}}$$

Da zusätzlich die partiellen Ableitungen der Ordinate nach diesen Koeffizienten ebenfalls gleich sind

$$\frac{\partial y'}{\partial x_h} = 0 = \frac{\partial y'}{\partial a_{00}}$$

besitzt die Jacobimatrix dort zwei linear abhängige Spalten und diese Parameter werden einen Korrelationskoeffizienten von $\rho = 1$ aufweisen. Maximale Korrelationen und damit lineare Abhängigkeiten müssen in jedem Fall ausgeschaltet werden, da für die Invertierung der Matrix $\mathcal{B}^T\Sigma_{ll}^{-1}\mathcal{B}$ die $\mathcal{B}$-Matrix des Regressionsverfahrens vollen Spaltenrang besitzen muß.

3.2.2 Parametrisierung der Einzelkameras

Nachdem im vorhergehenden Abschnitt die möglicherweise auftretenden Probleme geschildert wurden, wird in diesem Abschnitt das konkrete Verfahren zur letztendlichen Parameterisierung einer Einzelkamera beschrieben.

Das in Matlab programmierte Kameramodell (siehe beigelegte CD) erlaubt es, durch Setzen von Flags einzelne Parameter von der Schätzung auszuschließen. Dies geschieht im Regressionverfahren einfach durch das Entfernen von Spalten aus der $\mathcal{B}$-Matrix. Die nicht geschätzten Werte müssen dann allerdings vom Benutzer auf einen bestimmten Wert gesetzt werden. Darüber hinaus erlaubt es das Kameramodell, Restriktionen der Art $a_{11} = -b_{21}$ auf den Koeffizienten der Tschebyscheffpolynome einzuführen. Da die Restriktionen beliebig vom Benutzer gesetzt werden können und sich mit diesen auch die $\mathcal{B}$-Matrix verändert, wurde folgende Heuristik benutzt: Wenn keine Restriktionen auf den Koeffizienten gesetzt sind, dann wird die analytisch berechnete Jacobimatrix benutzt, ansonsten wird diese numerisch berechnet. Dies allerdings nur als Anmerkung, um das hier geschilderte Verfahren des Auffindens einer geeignete Parameterisierung nachvollziehen bzw. für andere Kameras selbst durchführen zu können.

Zuerst wurde der Hauptpunkt (x_h, y_h) von der Schätzung ausgeschlossen, da anhand der Daten nicht zwischen einer Veränderung des Nickwinkels ω und der Ordinate des Hauptpunkt y_h bzw. zwischen einer Veränderung des Gierwinkels φ und der Abszisse des Hauptpunkts x_h unterschieden werden kann. Tatsächlich kann man die beiden Parameter anhand ihrer Verzeichnungsfelder unterscheiden, da die Flußfelder der Winkel an den Seiten leichte kissenförmige Verzerrungen aufweisen, aber die Unterschiede signifikant festzustellen war durch den gegebenen Aufbau von Kamera und Roboter nicht möglich. Die Annahme, daß der Hauptpunkt mit dem Ursprung des Kamerakoordinatensystems gleich sei, lieferte überdies gute Ergebnisse.

Alle anderen Hauptparameter, also $c, s, \omega, \varphi, \kappa, X_0, Y_0, Z_0$ wurden mittels der Regression bestimmt.

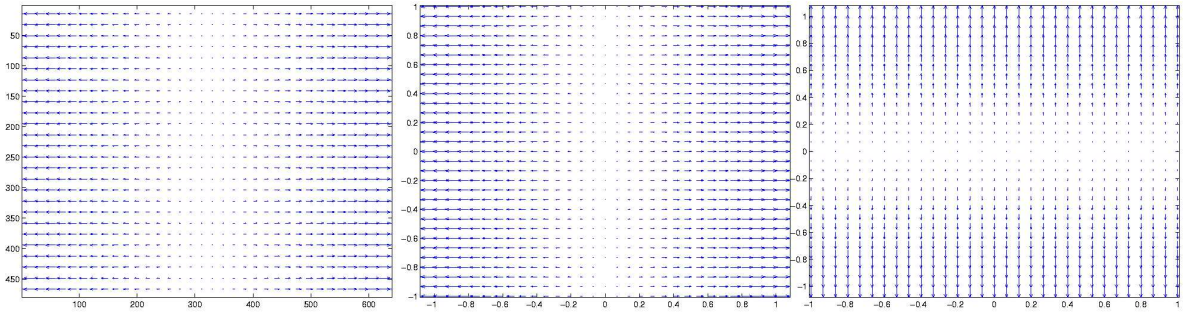

Abb. 3.4 Verzeichnungsfeld der Parameter s, a_{10} und b_{01}

Von den Verzeichnungsparametern wurde als erstes die beiden Koeffizienten a_{00}, b_{00} für die Polynome nullten Grades[7], also einer konstanten Verschiebung um den Vektor $(a_{00}, b_{00})^T$ ausgeschlossen, da es keine Linsenverzeichnung gibt, die diese Modellierung rechtferigen würde. Außerdem würden bei diesen aufgrund ihrer linearen Abhängigkeit von x_h und y_h die eben beschriebene Problematik mit den Winkeln ω und ϕ auftreten.

Darauf wurden alle Terme 1. Ordnung ins Modell mitaufgenommen. Dabei wiesen die Terme a_{10} und b_{01} eine hohe Korrelation mit dem Hauptparameter s auf. Daher wurden diese Parameter ausgeschaltet. Bei einer Überprüfung von Kombinationen verschiedener Terme ersten Grades stellte sich heraus, daß $a_{01} = -b_{10}$ ein Flußfeld erzeugte, welches dem von κ gleicht. Da diese Kombination natürlich nicht wünschenswert ist, wurde a_{01} aus dem Parametersatz entfernt und dafür die Restriktion $a_{01} = b_{10}$ gesetzt, da sie kein bekanntes Verzeichnungsfeld erzeugt.

[7] Der Grad des mit einem Koeffzienten verknüpften Tschebyscheffpolynoms ist leicht herauszufinden, indem man einfach den Zeilen- und Spaltenindex addiert.

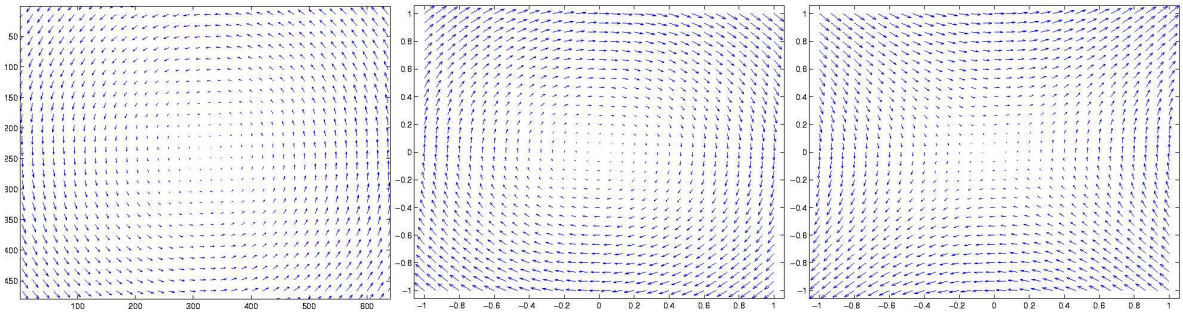

Abb. 3.5 Bewegungsfeld des Rollwinkels κ und Verzeichungsfelder der Kombinationen $a_{01} = -b_{10}$ und $a_{01} = b_{10}$

Im zweiten Schritt wurden nun alle ausgewählten Parameter beibehalten und die Terme 2. Ordnung eingeschaltet.

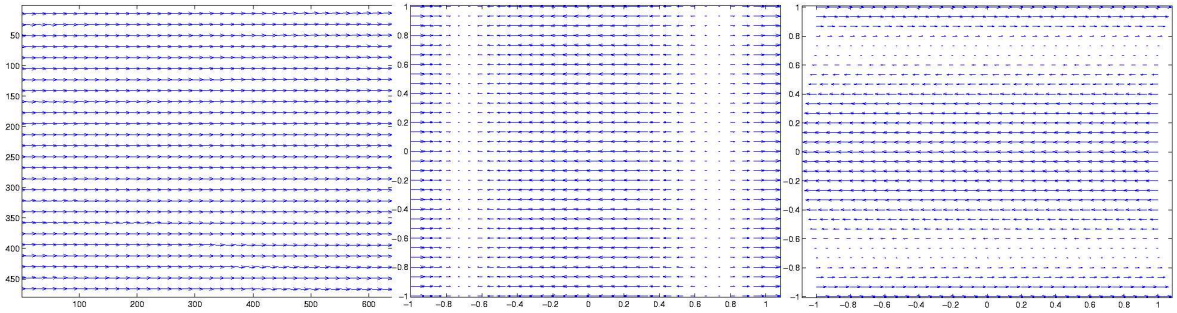

Abb. 3.6 Verzeichnungsfelder der Parameter φ, a_{20} und a_{02}

Von diesen wurden zuerst die Koeffizienten a_{20} und a_{02} ausgeschaltet, da sie eine hohe Korrelation mit dem Gierwinkel φ aufwiesen. Gleichzeitig korrelierten b_{20} und b_{02} stark mit dem Nickwinkel ω.

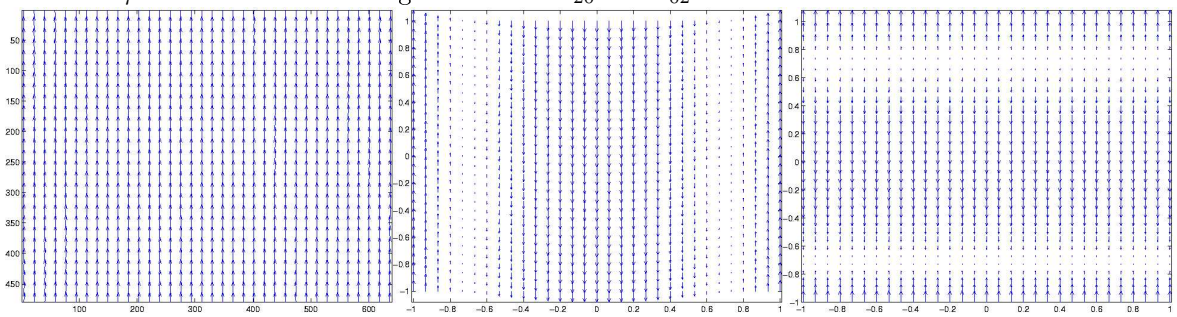

Abb. 3.7 Verzeichnungsfelder der Parameter ω, b_{20} und b_{02}

Eine Lösung dieses Problems fand sich, indem diese ausgeschaltet und die Restriktionen $a_{20} = b_{02}$ und $a_{02} = b_{20}$ eingeführt wurden, da sich durch sie die Schätzungen der Parameter nicht veränderten, der Fehler aber auf 0.5603px zusammenschrumpfte. Daraus wurde geschlossen, daß diese Kombinationen von Koeffizienten einen in den Daten enthaltenen Zusammenhang erklären, der nicht durch Haupt- bzw. die bis dato gesetzten Verzeichnungskoeffizienten erklärt werden konnte.

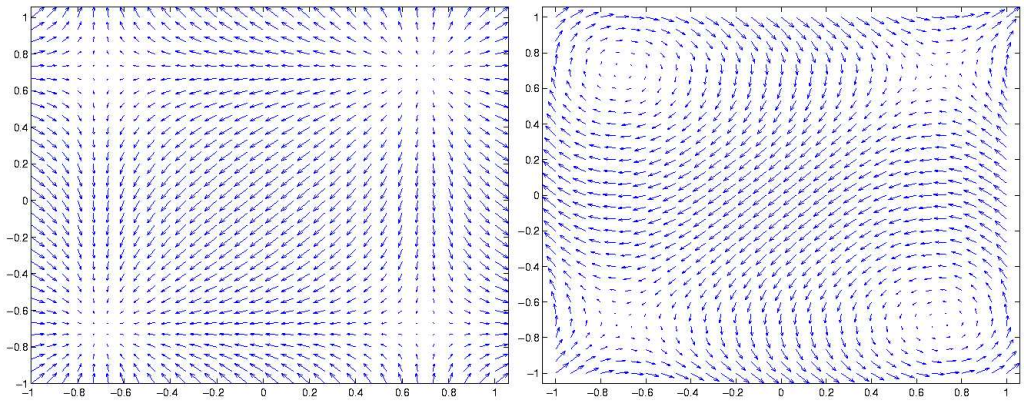

Abb. 3.8 Flußfelder der Kombinationen $a_{20} = b_{02}$ und $a_{02} = b_{20}$

Alle weiteren Koeffzienten für Terme 2. Ordnung wurde beibehalten.

Im dritten und letzten Schritt wurden die Terme 3. Ordnung zugelassen. Von diesen korrelierten a_{12} und b_{21} stark mit c und s und wurde daher ausgeschaltet.

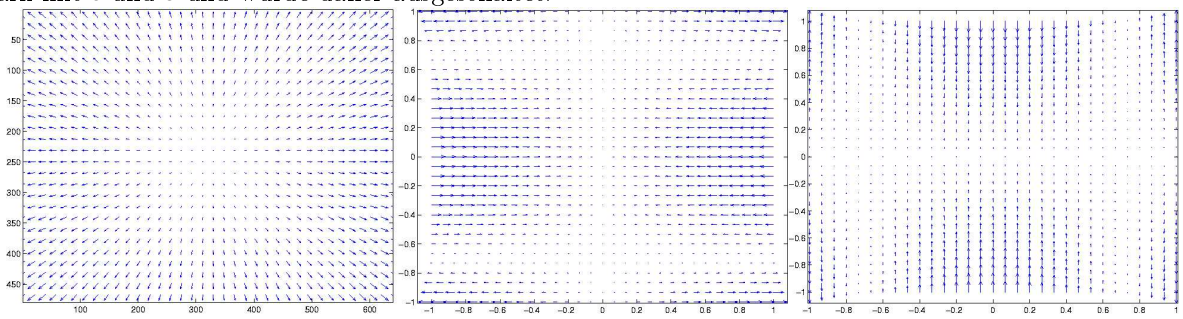

Abb. 3.9 Verzeichnungsfelder der Parameter c, a_{12} und b_{21}

Alle Terme höherer Ordnung, wurden nicht mehr zugelassen, da davon ausgegangen wurde, daß die bis hier bestimmten Parameter und Kombinationen von Parametern die Daten ausreichend beschreiben. Bei Tschebyscheffpolynomen vierten oder noch höheren Grades wäre die Gefahr gegeben, daß sie Rauschen in den Daten beschreiben und somit den Fehler auf Daten, welche nicht zur Regression herangezogen werden, erhöhen (d.h. overfitten).

3.2.3 Parametrisierung der Stereokamera

Das Modell der Stereokamera besteht aus zwei einzelnen Kameramodellen, die unabhängig voneinander kalibriert werden. Die äußere Orientierung der einzelnen Partner, d.h. die Koordinaten des Projektionszentrums $\mathbf{X}_0$ und die Nick, Gier- und Rollwinkel ω, φ, κ und die innere Orientierung, also die Parameter s und c, erlauben es mittels drei der vier Kollinearitätsgleichungen die Position eine Raumpunktes zu berechnen.

3.3 Kalibrierung mit maschinellem Lernen

3.3.1 Wahl der Kernel

Da die korrekte Wahl des Kernel nach wie vor eine ungeklärte Frage ist und dabei viel Erfahrung bzw. Intuition ins Spiel kommt wurden hier nur der polynomielle und der RBF-Kernel in Betracht gezogen, da diese

neben dem kanonischen Skalarprodukt die am häufigsten verwendeten Kernel darstellen. Natürlich gibt es trotzdem gute Gründe bzw. Motivationen gerade mit diesen Kerneln zu arbeiten. Dies soll im Folgenden noch kurz ausgeführt werden. Da die zu lernende Abbildung, wie schon im Abschnitt zur modellbasierten Kalibrierung beschrieben, nichtlinear in den Daten ist, wurden die linearen Kernel, also verschiedene Skalarprodukte, von vornherein ausgeschlossen.

Der heterogene polynomielle Kernel

$$k(x, x') = \langle x, x' + 1 \rangle^d$$

findet seine Berechtigung in der Tatsache, daß die zu lernende Funktion auf fast dem gesamten Definitionsbereich ungleich Null ist und sicherlich durch Mononome verschiedenen Grades ausgedrück werden kann. Diesen globalen Aspekt, also die Tatsache die Funktion nur endlich vielen Nullstellen besitzt, wird gut durch den polynomiellen Kernel beschrieben, da dessen Funktion $k(x, \cdot)$ ebefalls nur endlich viele Nullstellen besitzt bzw. ihr Träger der ganze durch die Datenpunkte aufgespannte Raum ist.

Der RBF-Kernel kommt in Betracht, da er lokale Eigenschaften der Daten gut beschreiben kann, d.h. derjenige Datenpunkt, welcher dem auszuwertenden am nächsten liegt, hat den größten Einfluß auf den Funktionswert. Dieser Kernel betont also mehr den lokalen Aspekt der Daten. Auch wenn die Gaußfunktion des RBF-Kernels keine Nullstelle besitzt, ist diese ab einem gewissen Abstand zum jeweiligen Datenpunkt praktisch gleich Null. Er scheint also eindeutig besser geeignet Linsenverzeichnungen auszugleichen, die ja gerade eine lokale Eigenschaft sind. Allerdings beruht die allgemeine zu lernende Abbildung nicht auf lokalen Eigenschaften, wodurch wiederum der polynomielle Kernel vorzuziehen wäre.

Beide Kernel sind auf eine gleichmäßige Verteilung der Daten angewiesen. In Regionen, die nur spärlich mit Datenpunkten besetzt sind, kann die durch Linearkombination von Kernelfunktionen des polynomiellen Kernels beschriebene Funktion, ähnlich wie bei sog. *Überschwingern* von Interpolationspolynomen, praktisch beliebige Werte annehmen, die mit der zu lernenden Funktion wenig gemein haben. Die durch RBF-Kernel ausgedrückte Funktion ist wegen der eben beschriebenen "Lokalität" dort praktisch konstant Null und liefert damit ebenfalls eine schlechte Beschreibung der zu lernenden Funktion.

Aufgrund dieser Überlegungen wurden für beide Kernel Parametersuchen durchgeführt. Für die Ergebnisse siehe Abschnitt 4.3.

3.3.2 Parametersuche

Um geeigente Parameter für einen Lernalgorithmen und den jeweiligen Kernel zu finden wurden Gittersuchen durchgeführt, wobei der Fehlerwert an jedem Knoten mit einer Kreuzvalidierung (siehe unten) ermittelt wurde. Bei einer Gittersuche oder engl. *Gridsearch* wird für jeden zu bestimmenden Parameter eine Menge $\mathcal{P}_i$ mit Werten gebildet, die das ganze Spektrum dieses Parameters abdecken sollten. Besitzt ein Algorithmus mit seinem Kernel zusammengenommen k Parameter, so wird bei der Gittersuche für jeden Parametersatz

$$p \in \mathcal{P}_1 \times \ldots \times \mathcal{P}_k$$

eine Lernmaschine trainiert und ihr Fehler auf Testdaten bestimmt. Ein solches Parametertupel bildet also einen Gitterknoten. Nach der Suche wird derjenige Parametersatz ausgewählt, der den niedrigsten Fehler erbracht hat.

Die Bestimmung des Fehlers geschieht im Fall dieser Arbeit mittels Kreuzvalidierung oder engl. *cross validation*. Hierbei wird die Trainingsmenge in n gleichgroße Partitionen aufgeteilt. Die Lernmaschine wird dann auf $n - 1$ Teilen trainiert und auf dem übriggebliebenen getestet. Dies wird für alle n Teilmengen durchgeführt und als Fehler der Durchschnitt der Fehlerwerte genommen.

Abb. 3.10 zeigt die verschiedenen Parametermengen, die für die Gittersuche betrachtet wurden.

$$C \in \mathcal{C} = \{15000, 10000, 8000, 4000, 2000, 1000, 500, 100, 10, 1\}$$
$$\varepsilon \in \mathcal{E} = \{8, 6, 4, 2, 1, 0.5\}$$
$$\sigma \in \Sigma = \{8, 4, 2, 1, 0.5, 10^{-2}, 10^{-4}, 10^{-8}\}$$
$$d \in \mathcal{D} = \{3, 5, 7, 9, 11\}$$
$$\gamma \in \Gamma = \{10^{-10}, 10^{-5}, 10^{-4}, 10^{-2}, 1, 2, 4\}$$

Abb. 3.10 In der Gittersuche verwendete Parametermengen der Lernalgorithmen und ihrer Kernel.

Die Ergebnisse finden sich in den entsprechenden Unterabschnitten des folgenden Abschnitts.

4 Ergebnisse

Dieses Kaptiel gliedert sich in drei Teile, wobei der erste die verwendete Hardware und Software beschreibt, der zweite die Ergebnisse der klassischen Kalibriation und der dritte die des Ansatzes mit maschinellem Lernen schildert.

Die Trainingsmenge, auf der die Kamera kalibriert bzw. die Algorithmen trainiert wurden, enthielt 200 Punkte, die zufällig aus insgesamt 992 aufgenommenen Punkten ausgewählt wurden. Für die klassische Kalibrierung wurden die Parameter c, s, X_0, Y_0, Z_0, ω, ϕ, κ, sowie a_{03}, a_{11}, a_{21}, a_{30}, b_{02}, b_{03}, b_{10}, b_{11}, b_{12}, b_{20}, b_{30} geschätzt, wobei auf den Verzeichnungskoeffizienten die Restriktionen $a_{01} = b_{10}$, $a_{02} = b_{20}$ und $a_{20} = b_{02}$ gesetzt waren (siehe 3.2.2).

4.1 Material

Zur Aufnahme der Daten wurde ein von den Werkstätten des Max-Planck-Institus hergestelltes Kalibrationsobjekt, bestehend aus einer dem Greifer des Roboters angepassten Metallhülle für die Beherbung der Stromversorgung und einer LED an der Spitze. Diese LED ist an ihrem Ende abgeflacht, um die subpixelgenaue Detektion als Ellipse zu ermöglichen. Zusätzlich wurden noch die Seiten der LED mit Isolierband abgeklebt, so daß nur das flache Ende der LED zu sehen war.

Der zur Bewegung des Kalibrationsobjekt verwendete Roboterarm[8] besitzt sieben Freiheitsgrade und eine Stellgenauigkeit von 0.01mm (siehe beigeheftete Roboterspezifikation). Die inverse Kinematik war gegeben und der Roboter war über einen Server durch Matlab ansteuerbar.

Die Bilder wurden mit zwei auf eine Pan-Tilt-Unit der Firma Directed PerceptionInc [9] montierten CCD-Kameras der Firma Basler [10], mit Cinegon Objektiven [11] aufgenommen.

[8] siehe auch *http://www.sdia.or.jp/mhikobee/products/mechatronics/e_index.html*
[9] siehe *http://www.directedperception.com/ptu_over.htm*
[10] siehe auch *http://www.baslerweb.com/*
[11] siehe beigeheftete Objekivdaten

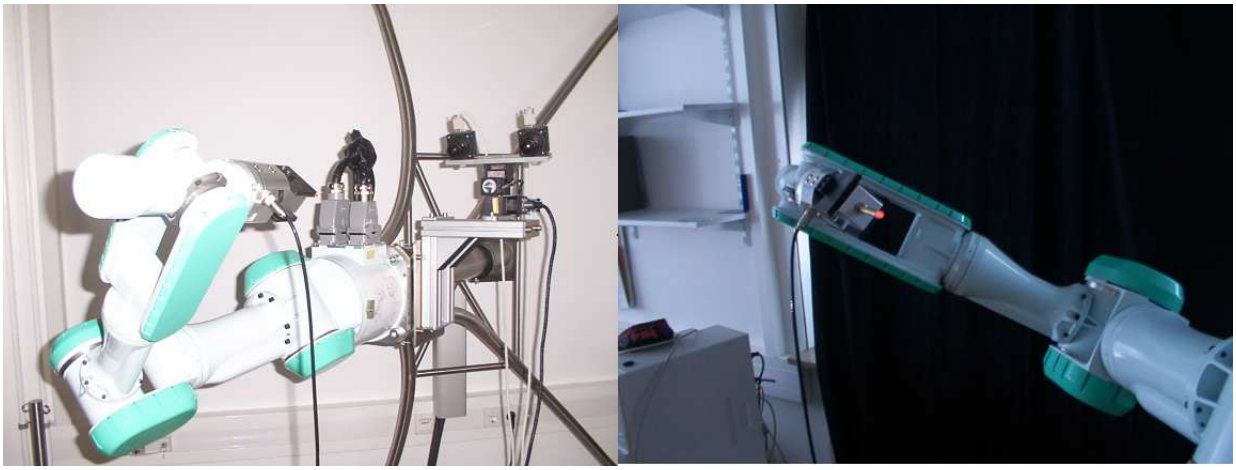

Abb. 4.0 Roboterarm mit Kalibrationsobjekt und Kameras

Als Implementierung der Lernalgorithmen wurde die frei verfügbar Machine Learning Matlab Toolbox *Spider* verwendet. Die Kameramodelle wurden selbst implementiert (siehe beigeheftete CD).

4.2 Modellbasierter Ansatz

4.2.1 Einzelkamerakalibrierung

Als Beispiel einer Einzelkamerakalibrierung werden hier die Ergebnisse der Kalibrierung der linken Kamera aufgeführt. Als Startwerte für die Winkel wurden `pan`$= -900$ und `tilt`$= -300$ verwandt, was ungefähr einem Gier- bzw. Nickwinkel von $\omega = 15.429°$ bzw. $\phi = 44.229°$ entspricht. Die Umrechnung geschieht im Kameramodellobjekt nach der Formel

$$\omega = \frac{(-\texttt{tilt}/3600) \cdot \zeta}{360} \cdot 2 \cdot \pi$$
$$\phi = -\frac{((-1760 - \texttt{pan})/3600) \cdot \zeta}{360} \cdot 2 \cdot \pi$$

wobei $\zeta = 185.1428$ die Auflösung der Pan-Tilt-Einheit bezeichnet.

Als Startwerte für das Projektionszentrum wurde die Position der Roboterhand benutzt, nachdem diese so nahe wie möglich an die Linse der linken Kamera bewegt wurde. Bei dieser Prozedur gilt es besonders vorsichtig vorzugehen, da theoretisch die Möglichkeit einer "Selbstenthauptung" des Roboters besteht.

Das Regressionsverfahren terminierte nach vier Schritten mit einem Fehler von $l_{rmse} = 0.49382\texttt{px}$, wobei

$$l_{rmse} = \sqrt{\frac{\sum_{i=1}^{m}(x'_i - \hat{x}'_i)^2 + (y'_i - \hat{y}'_i)^2}{2m}}$$

die Wurzel des mittleren quadratischen Fehler[12] bezeichnet.

```
Performing Regression ...
Computing Numerical Jacobi matrix, because some restrictions are set...
Norm dp 5.0142  Error:  73.323
 omega= 12.4614 phi=40.8138 kappa=-1.0584
Computing Numerical Jacobi matrix, because some restrictions are set...
Norm dp 1.2436  Error:  2.2428
 omega= 12.5584 phi=40.9585 kappa=-1.0581
Computing Numerical Jacobi matrix, because some restrictions are set...
Norm dp 0.013335  Error:  0.49582
```

[12] root mean squared error

```
 omega= 12.558 phi=40.9603 kappa=-1.0588
Computing Numerical Jacobi matrix, because some restrictions are set...
Norm dp 8.6254e-005  Error:  0.49382
 omega= 12.558 phi=40.9603 kappa=-1.0588
Computing Numerical Jacobi matrix, because some restrictions are set...
Norm dp 1.1552e-006  Error:  0.49382
 omega= 12.558 phi=40.9603 kappa=-1.0588

-----------------Calibration Results-----------------------
c: 12.832
s: 0.98746
X: 358.5283
Y: 334.8791
Z: -72.2024
omega: 0.21918
phi: 0.71489
kappa: -0.01848
a03: 0.0058145
a11: -1.0712
a21: 0.52611
a30: -0.66805
b02: -0.40149
b03: -0.59548
b10: -3.7424
b11: -1.4154
b12: -0.015804
b20: 0.11513
b30: 0.186
```

Abb. 4.1 Programmausgabe während der Regression und Ergebnisse. Die Winkel sind in Bogenmaß angegeben. *dp* bezeichnet die Norm des Verbesserungsvektors. Dieser kann daher für verschiedene Startwerte und Anzahl der Trainingspunkte andere Werte annehmen, wird aber ausgegeben, um den Benutzer währen der Regression die Nähe zum Minimum abschätzen zu lassen.

Im folgenden werden nun einige Teile des Ergebnisprotokolls aufgeführt und erläutert.[13]

Zunächst sollte ein Blick auf die Kovarianzmatrix geworfen werden, um sicherzustellen, daß keine der Parameter ein übermäßig hohe Korrelation aufweisen. Σ_{pp} wurde wie in Abschnitt 2.2.3 berechnet. Lediglich die Parameterpaare $\{X_0, Z_0\}$ und $\{\omega, \kappa\}$ wiesen einen Korrelationskoeffizienten > 0.97 auf, wobei die erste Korrelation von $\rho(X_0, Z_0) = 0.9765$ sehr wahrscheinlich von der Tatsache her rührt, daß die Blickrichtung der Kamera ungefähr einen Winkel von $45°$ zu sowohl der $X-$ als auch der $Z-$Achse des Roboterkoordinatensystems aufweist und somit eine Veränderung einer der beiden Koordinaten des Projektionszentrums ein ähnliches Bewegungsmuster auf der x, y-Ebene des Kamerakoordinatensystems aufweisen muß. Die Korrelation von $\rho(\omega, \kappa) = 0.9877$ ist wahrscheinlich durch die Tatsache begründet, daß die Daten nur in der anderen Kamera näheren Hälfte des Bildes aufgenommen wurden, da nur dies der Bereich des Stereosehens war. Dadurch ruft aber eine Drehung- bzw. ein Nicken der Kamera ein ähnliches Flußfeld der Bildpunkte hervor.

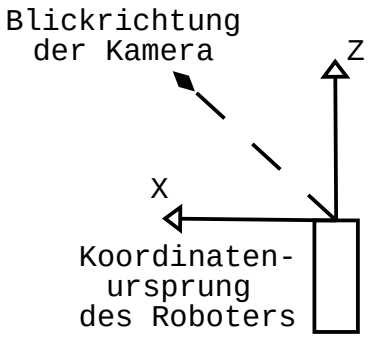

Abb. 4.2 Skizze des Verhältnis zwischen Roboterkoordinatensystem und Kamerablickrichtung

Bei den Verzeichungsparametern für die x'-Komponente des Vektorfelds konnte für die gegebenen Daten ein Parameter nicht signifikant geschätzt werden, da seine Standardabweichung höher als sein tatsächlicher

[13] Das ganze Ergebnisprotokoll ist auf der CD enthalten.

Wert war.

$$a_{03} = 0.005814511 \ \sigma_{a_{03}} = 0.06933439$$

Bei den Parameter der y'-Komponente des Verzeichnungsfeldes war dies für den Parameter b_{12} der Fall.

$$b_{12} = -0.01580437 \ \sigma_{b_{12}} = 0.2470712$$

Eine weiterer Hinweis auf die ein gute Schätzung liefert der Wert der a posteriori Gewichtseinheit[14] mit $\hat{s}_0 = 1.012$,da dieser keine allzu große Differenz zu seinem a priori Wert $s_0 = 1$ aufweist. Mittels $\hat{s}_0$ läßt sich auch die Genaugkeit, d.h. die Standardabweichung der Bildkoordinaten $\hat{s}_{x'_i} = 0.506$ und $\hat{s}_{y'_i} = 0.506$ mit

$$\hat{s}_{x'_i} = \hat{s}_0 \sqrt{(\Sigma_{ll})_{2i}}$$
$$\hat{s}_{y'_i} = \hat{s}_0 \sqrt{(\Sigma_{ll})_{2i+1}}$$

berechnen. Diese weisen aufgrund ihrer gleichen Initialisierung denselben Wert auf.

Ein weiteres und an dieser Stelle letztes Qualitätskriterium sind die Residuen der Bildkoordinaten, berechnet für Testdaten, da auf den Trainingsdaten ohnehin ein niedriger Fehler zu erwarten ist. Wenn die Parameter angemessen gesetzt wurden und die Regression erfolgreich war, sollten diese keine Vorzugsrichtung aufweisen.

[14] für dessen Berechnung siehe 3.2.1

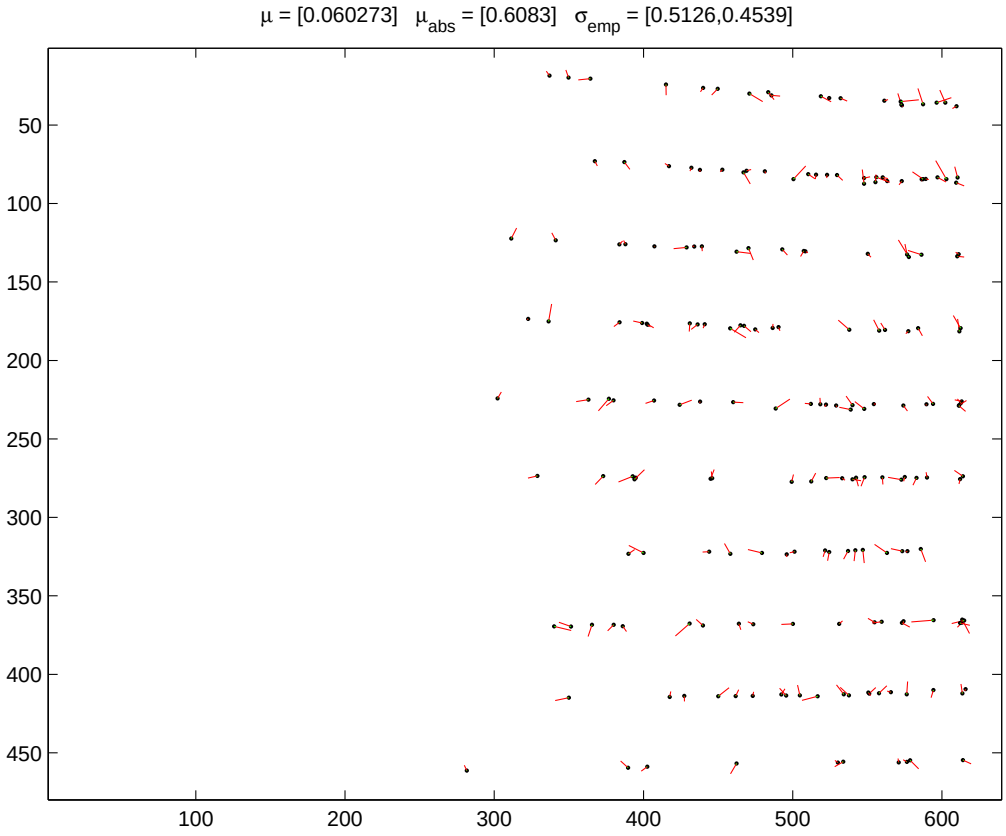

Abb. 4.3 Residuen der Bildkoordinaten: Die Roten Linien geben die Richtung des Residuums an, ihre Länge entspricht aber nicht dem tatsächlichen Residuum an dem Punkt. Dieses wird durch die grünen Linien angezeigt. Um den Plot nicht zu überladen, wurden die Residuen nur auf 200 von den 792 Testdatenpunkten berechnet.

Die in Abb. 4.3 angegbenen Mittelwerte μ und μ_{abs} beziehen sich auf die Norm der einzelnen Residuenvektoren. μ bezeichnet dabei die Norm des Mittelwerts der Residuen und μ_{abs} den Mittelwert der Normen der Residuen.

$$\mu = \left|\left| \frac{1}{m} \sum_i l_i \right|\right|_2 \quad \text{wobei } l_i \text{ das } i-\text{te Residuum bezeichne}$$

$$\mu_{abs} = \frac{1}{m} \sum_i ||l_i||_2$$

Da sich bei der Berechnung von μ einzelne Terme wegheben ist dieser Mittelwert deutlich kleiner als μ_{abs}. Es ist nun klar, daß sich die Terme bei der Berechnung von μ nur gegenseitig auslöschen können, wenn die Residuen keine Vorzugsrichtung aufweisen. Je näher als μ an μ_{abs} liegt, desto mehr Residuen besitzen eine solche. Da der Wert von $\mu = 0.060273$ aber nur etwas zehn Prozent des Werts von $\mu_{abs} = 0.6083$ aufweist, scheinen die Residuen keine Vorzugsrichtung zu besitzen, welches auch der subjektive Eindruck bestätig.

4.2.2 Stereokamerakalibrierung

Wie in Kapitel 2.2.4 beschrieben, werden die beiden kalibrierten Einzelkameras zu einer Stereokamera zusammengefügt. Dies erlaubt es, die Raumkoordinaten eines korrespondierenden Bildpaares festzustellen. Die

rechte Kamera wurde mit denselben Einstellungen wie ihr linker Gegenpart kalibriert. Da für die beiden Kameras auch dieselben Startwerte benutzt wurde, benötigte die Regression für die rechte Kamera einige Schritte mehr, konvergierte aber nach sieben Schritten, jedoch mit einem Fehler von $l_{rmse} = 1.0899\text{px}$.

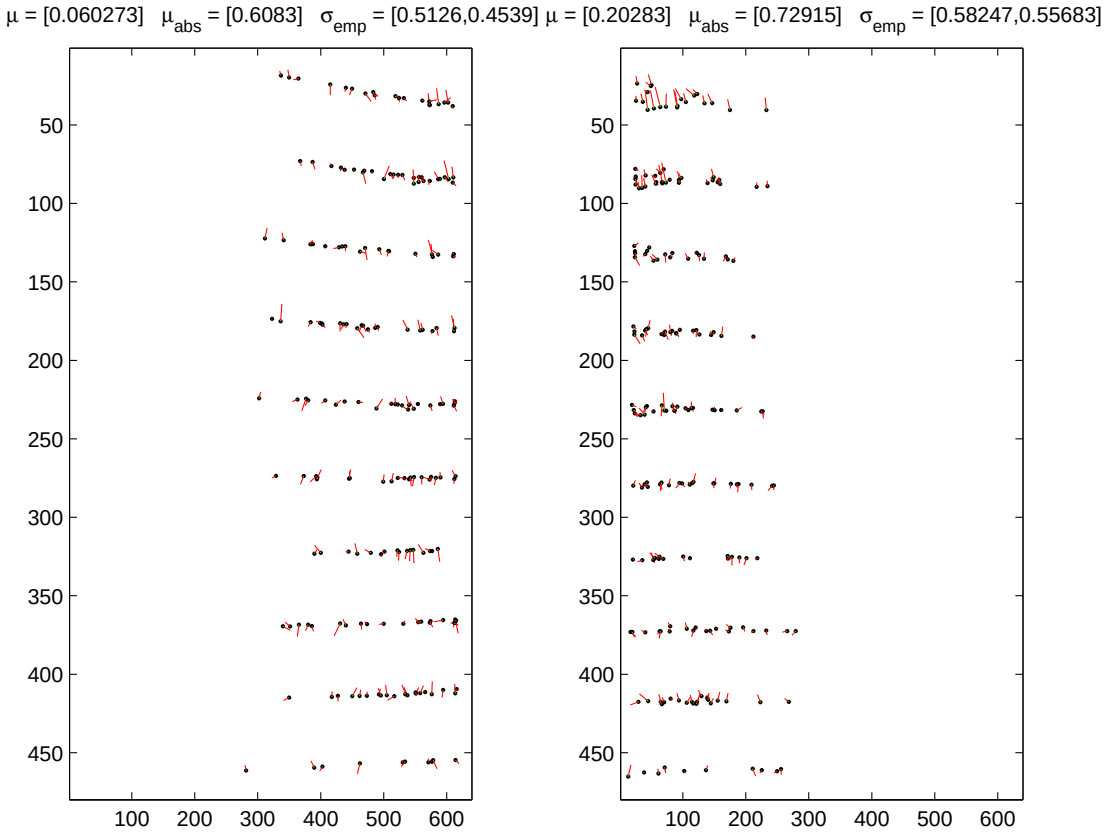

Abb. 4.4 Residuumsplot des Stereokamerapaares auf Testdaten. Auch die Fehler der rechten Kamera zeigen keine Vorzugsrichtung.

Abb. 4.4 zeigt die Residuen beider Kameras. Auch in der rechten Kamera ist keine Vorzugsrichtung der Fehler zu erkennen.

Vergleicht man die Werte der Drehwinkel beider Kameras, so scheinen die Ergebnisse plausibel, da beide Kameras auf der Schwenkeinheit befestigt sind und somit annähernd die gleichen Winkel zum Roboterkoordinatensystem aufweisen sollten.

	linke Kamera	**rechte Kamera**	Differenz
ω	12.558°	11.833°	0.725°
φ	40.960°	42.098°	-1.1380°
κ	-1.0588°	-1.7660°	0.70720°

Abb. 4.5 Vergleich der Drehwinkel beider Kameras zu den jeweiligen Achsen des Weltkoordinatensystems des Roboters. Geringe Differenzen können durchaus auftreten, da die beiden Kameras lediglich mit einer Schraube auf der Schwenkeinheit befestigt sind.

Auch die Werte der beiden Projektionszentren sind sinnvoll, da sie sich in der Tiefe und Breite, aber nicht in der Höhe unterscheiden.

	linke Kamera	**rechte Kamera**	Differenz
X_0	358.5283mm	292.5499mm	65.978mm
Y_0	334.8791mm	334.5857mm	0.29340mm
Z_0	-72.2024mm	-14.5399mm	-57.662mm

Abb. 4.6 Vergleich der Projektionszentren beider Kameras, wobei die Werte im Weltkoordinatensystems des Roboters gegeben sind. Die Tasache, daß die Projektionszentren sich in der Tiefe und Breite, aber nicht in der Höhe unterscheiden sollten, spiegelt sich gut in den Werten wieder.

Die Norm des Differenzvektors sollte ungefähr dem Abstand der Kameras entsprechen, was mit einer Länge von 87.625mm der Fall ist.

Man kann die Werte der Projektionszentren in Beziehung zu den Winkeln setzen, indem man idealisiert annimmt, daß die beiden auf einer Ebene liegen, was anhand der geringen Höhendifferenz keine allzu starke Annahme ist. Der Vektor

$$\mathbf{b} = \begin{pmatrix} 65.978 \\ -57.66 \end{pmatrix}$$

gibt nun ungefähr den Positionsunterschied zwischen der linken und der rechten Kamera an. Der zu ihm orthogonale Vektor

$$\mathbf{b}_\perp = \begin{pmatrix} 0.65805 \\ 0.75298 \end{pmatrix} \text{ mit } ||\mathbf{b}_\perp|| = 1$$

sollte nun in die Blickrichtung der Kameras zeigen und somit sollte der Winkel γ zwischen der z-Achse des Objektkoordinatensystem und $\mathbf{b}_\perp$ ungefähr dem Gierwinkel φ der Kamera entsprechen. Da

$$\gamma = \arccos\left(\left\langle \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \begin{pmatrix} 0.65805 \\ 0.75298 \end{pmatrix} \right\rangle\right) = \arccos(0.75298) = 41.151°$$

und der Mittelwert

$$\frac{\varphi_l + \varphi_r}{2} = 41.529°$$

sehr gut übereinstimmen, passen die Werte der Winkel zu denen des Projektionszentrums.

Ebenso wie die Residuen der Bildpunkte, sollte die Residuen der durch das Stereokameramodell berechneten Raumpunkte keine Vorzugrichtung aufweisen. Abb. 5.6 zeigt das 3D-Äquivalent zu den Bildresiduumsplots. Der visuelle Eindruck, daß keine Richtung ungebührlich oft auftritt, wird durch die schon erläuterten Werte μ und μ_{abs} bestätigt.

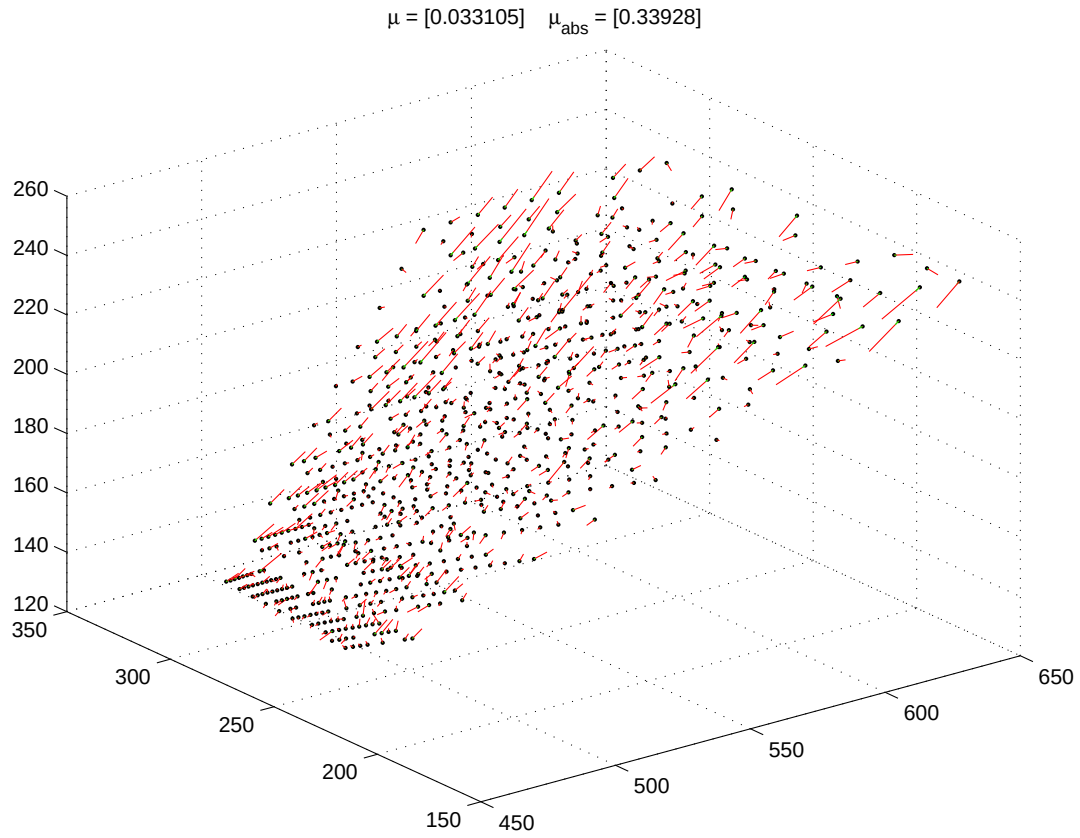

Abb. 4.7 Residuen der durch das Stereokameramodell berechneten Raumpunkte.

Wie man auch an dieser Abbildung sehen kann nimmt der Fehler mit der Tiefe zu. Dies kommt daher, daß die Disparität zwischen zwei Punkte umso geringer ist, je weiter der Punkt von der Kamera entfernt liegt, also Fehler in der Bestimmung der Bildposition der Punkte einen größeren Fehler in der Tiefenberechnung hervorrufen. Bei der Tiefenbestimmung liegt es in der Natur der Sache, daß der Fehler mit der tatsächlichen Tiefe des Punkts quadratisch zunimmt, wie die folgende kleine Rechnung zeigt: Sei $\rho := x'_l - x'_r$ die Differenz der Abszissen der Bildkoordinaten, also die Disparität. Sei weiterhin b die Länge des Vektors zwischen den Projektionszentren beider Kameras, $f \approx c$ die Brennweite und o.B.d.A $\varphi = \omega = \kappa = 0$. Dann kann die Tiefe eines Punktes durch die Formel

$$z(\rho) = \frac{fb}{\rho}$$

berechnet werden. Besitzt die Disparität einen Fehler, so daß $\rho = \rho_0 + \delta_\rho$, dann ergibt eine Taylorentwicklung um ρ_0

$$z(\rho) = \frac{fb}{\rho_0} - \frac{fb}{\rho_0^2}\delta_\rho + \mathcal{O}(\delta_\rho^2)$$

Setzt man $z = z(\rho_0) + \delta_z$, dann gilt

$$\delta_z = -\frac{fb}{\rho_0^2}\delta_\rho$$

Da die Tiefe umgekehr proportional zur Disparität ist, also $z \sim \frac{1}{\rho}$, gilt:

$$\delta_z = -fbz^2\delta_\rho$$

Somit nimmt der Tiefenfehler quadratisch mit dem Fehler in den Bildkoordinaten zu. Das gleich gilt dann natürlich auch für den Positionsfehler, was folgende Abschätzung zeigt:

$$\sqrt{\delta_x^2+\delta_y^2+\delta_z^2} = \sqrt{\delta_z^2\left(\frac{\delta_x^2}{\delta_z^2}+\frac{\delta_y^2}{\delta_z^2}+1\right)} = \delta_z \underbrace{\sqrt{\left(\frac{\delta_x^2}{\delta_z^2}+\frac{\delta_y^2}{\delta_z^2}+1\right)}}_{\geq 1} \geq \delta_z$$

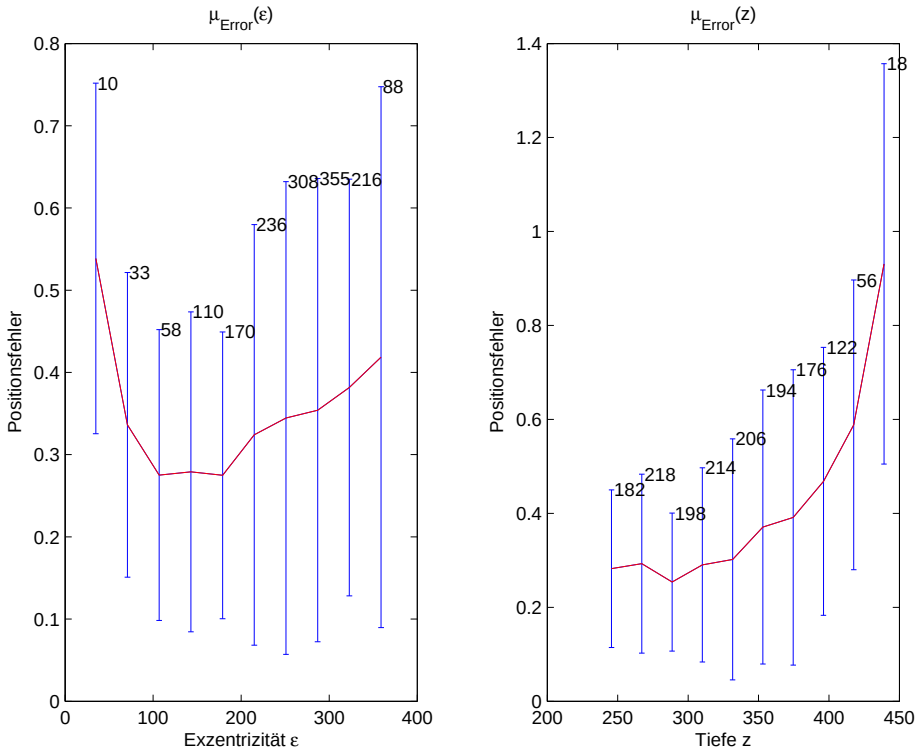

Abb. 4.8 Fehlerverteilung des Stereokameramodells in Abhängigkeit von der Exzentrizät $\mu_{Error}(\varepsilon)$ und der Tiefe $\mu_{Error}(z)$. Die Fehlerbalken geben die empirische Standardabweichung des Fehlers in dieser Exzentrizität bzw. Tiefe an. Die Zahlen an den Fehlerbalken geben die Anzahl der in diesen Wert miteingegangenen Residuen an.

Abb. 4.8 zeigt die Fehlerverteilung in Abhängigkeit von der Exzentrizität und der Tiefe. Die Exzentrizität ergibt sich durch den Abstand des Bildpunkts von Hauptpunkt, welcher in diesem Fall der Ursprung des Kamerakoordinatensystems ist. Sie wird hier mit ε bezeichnet. Als Fehler wurde die Norm der einzelnen Residuen verwendet (*Positionserror*).

Wie sehr schön zu sehen ist, entspricht die Zunahme des Positionsfehlers den Erwartungen, indem er eine parabelartige Kurve beschreibt. Auch ist die Standardabweichung für größere Z-Werte höher. Dies liegt zum einen an der geringeren Anzahl der Residuen in dieser Tiefe, zum anderen an den höheren Fehlerwerten ohne eindeutige Vorzugrichtung. Der Fehler in Abhängigkeit der Exzentrizität zeigt keinen eindeutigen Trend. Selbst die Standardabweichungen sind für die meisten Werte annähernd gleich. Dies kann wiederum als ein Indiz für die korrekte Parameterisierung der Linsenverzeichnung gewertet werden.

	X	**Y**	**Z**	**X, Y, Z**
l_{rmse}	0.2691mm	0.1467mm	0.2983mm	0.4277mm

Abb. 4.9 l_{rmse} der geschätzten Punkte in die verschiedenen Raumrichtungen.

Insgesamt weist das Modell für die gegebenen Daten einen Tiefenfehler von $l_{rmse} = 0.4277\text{mm}$ auf.

Dies zeigt, daß die Stereokamerakalibrierung, zumindest für Punkte im Arbeitsbereich des Roboters, eine geeignete Methode zur Tiefenbestimmung darstellt.

4.3 Nichtmodellbasierter Ansatz

4.3.1 Ridge Regression

Für die Kalibrierung mit RIDGE REGRESSION wurden Parametersuchen für den polynomiellen und den RBF-Kernel auf den Mengen $\Gamma \times \mathcal{D}$ bzw. $\Gamma \times \Sigma$ durchgeführt. Als Fehlermaß wurde l_{rmse} verwendet (siehe 3.3.2).

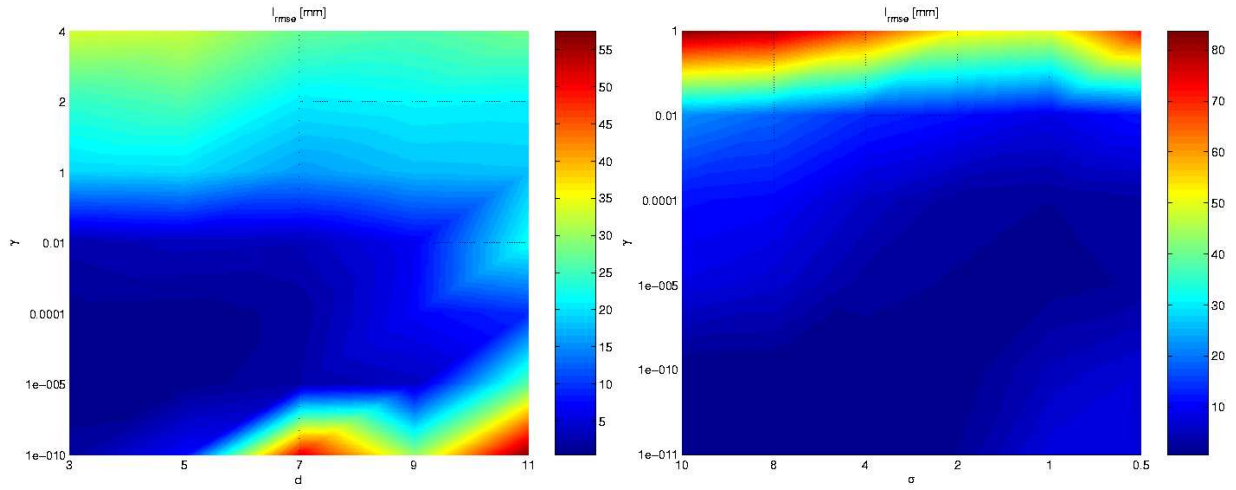

Abb. 4.10 Fehlerflächen der Parametersuchen für den polynomiellen Kernel und den RBF-Kernel. Die Parameterwerte wurden nicht äquidistant gewählt, daher ist die Fehlerfläche in beide Richtungen gestaucht.

Abb. 4.10 zeigt das Ergebnis der Suche in $\Gamma \times \mathcal{D}$. Da die Parameterwerte nicht äquidistant gewählt wurden, ist die Oberfläche in beide Richtungen gestaucht.

Es ist gut zu erkennen, daß der Algorithmus für hohe Werte von d ein starkes Ansteigen des Fehlers aufweist. Dies ist wahrscheinlich auf numerische Probleme zurückzuführen. Da einige Lernalgorithmen anfällig für Skalierungen in den Daten sind, wurden diese zuvor auf das Intervall $[-1, 1]$ skaliert. Der Wert eines Skalarprodukts zweier solcher Eingabebeispiele ist damit ebenfalls sehr klein und kann, falls der Wert kleiner als eins ist, durch hohe Potenzen numerische Probleme verursachen. In weniger starkem Ausmaß geschieht dies auch bei RIDGE REGRESSION mit RBF-Kernel für zu kleine Werte von σ. Die Glockenkurven des RBF-Kernels sind für diese Werte so schmal, daß mehr Gewicht auf die lokalen Eigenschaften der Daten fällt und die Funktion schon in geringem Abstand zu den Datenpunkten praktisch gleich Null ist.

Für zu große Werte von γ steigt der Fehler ebenfalls an, da dort der Einfluß der Eigenwerte (siehe 2.1.5) zu stark zurückgeschraubt wird und somit der Algorithmus die Daten immer weniger gut beschreiben kann. Ein ähnliches Verhalten ist auch für die RIDGE REGRESSION mit RBF-Kernel zu beobachten.

Mit einem Kreuzvalidierungsfehler von 0.44716mm stellten sich $\gamma = 10^{-5}$ und $d = 3$ als beste Parameter für RIDGE REGRESSION mit polynomiellen Kernel heraus. Bei der Verwendung eines RBF-Kernels erbrachten die Werte $\gamma = 10^{-10}$ und $\sigma = 2$ mit 0.38502mm das beste Kreuzvalidierungsergebnis.

Die Algorithmen wurden für diese Parameter nochmals auf der vollständigen Trainingsmenge angelernt und daraufhin auf einem Testdatensatz von 792 Punkten getestet. Dabei ergaben für die RIDGE REGRESSION mit polynomiellem Kernel ein Fehlerwert von 0.3681mm, für RIDGE REGRESSION mit RBF-Kernel 0.3337mm.

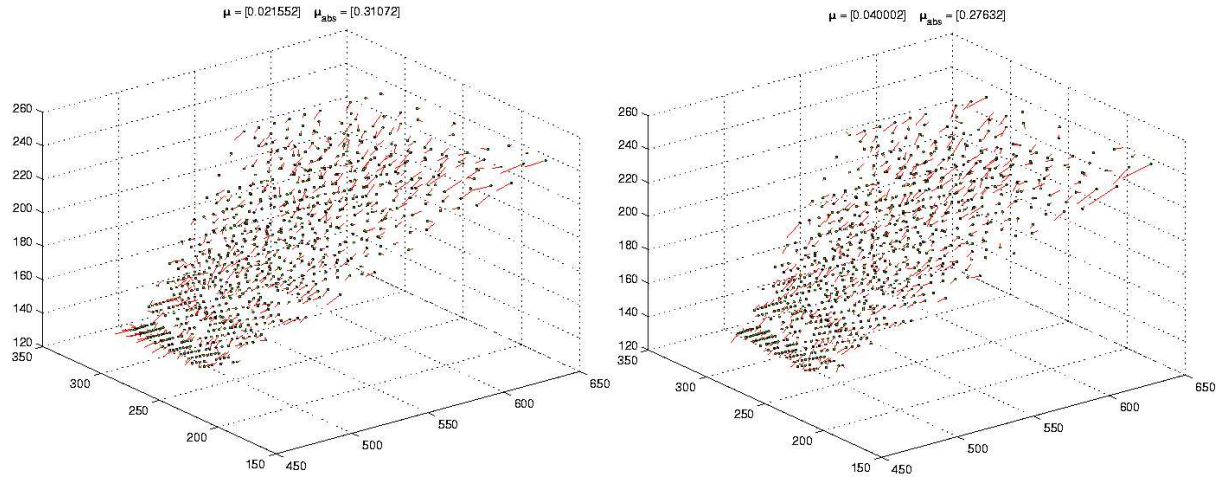

Abb. 4.11 Positionsresiduen der RIDGE REGRESSION mit polynomiellen (links) bzw. RBF-Kernel (rechts) und den Parametern $\gamma = 10^{-5}$ und $d = 3$ bzw. $\gamma = 10^{-10}$ und $\sigma = 2$

Abb. 4.11 zeigt die Positionsresiduen der RIDGE REGRESSION mit polynomiellen Kernel. Die gleiche Abbildung für den RBF-Kernel findet sich im Anhang. Wie auch bei der modellbasierten Tiefenberechnung kann hier keine eindeutige Vorzugrichtung der Residuen festgestellt werden. Die Norm des mittleren Residuums ist mit $\mu = 0.021552$mm beinahe Null, während der Mittelwert über die Norm der einzelnen Residuen $\mu_{abs} = 0.31072$mm beträgt. Bei der Tiefenberechnung mit RBF-Kernel ist ebenfalls keine Vorzugsrichtung zu erkennen. Die Werte $\mu = 0.040002$mm und 0.27632mm sind aber im Vergleich zu polynomiellem Kernel etwas schlechter, da μ^{RBF} mit ca. 14.5% einen größeren Anteil an μ_{abs}^{RBF} besitzt als μ^{poly} an μ_{abs}^{poly} mit ca. 6.9%. Dennoch sind beide Werte aufgrund ihrer geringen Größenordnung noch vertretbar.

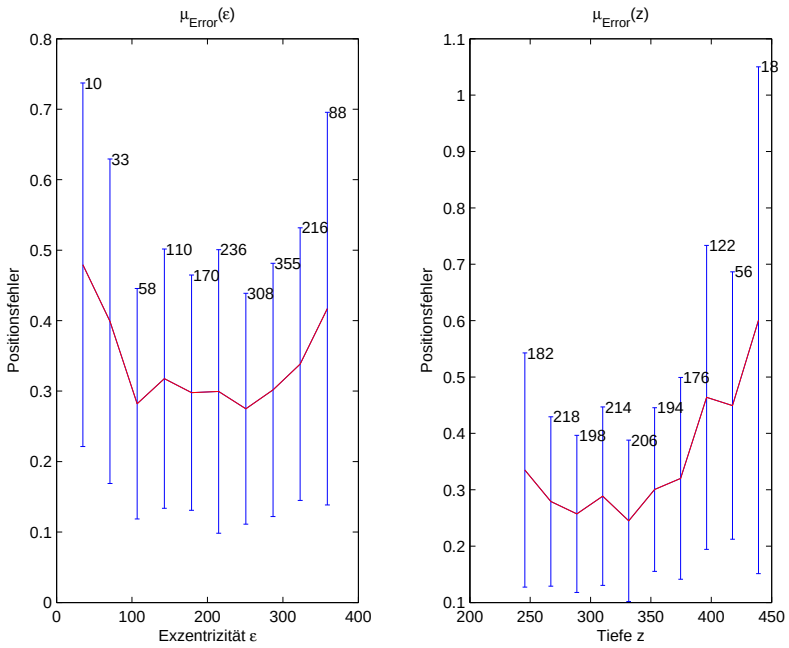

Abb. 4.12 Fehlerverteilung der RIDGE REGRESSION mit polynomiellem Kernel in Abhängigkeit der Exzentrizät $\mu_{Error}(\varepsilon)$ und der Tiefe $\mu_{Error}(z)$. Die Fehlerbalken geben die empirische Standardabweichung des Fehlers in dieser Exzentrizität bzw. Tiefe an. Die Zahlen an den Fehlerbalken geben die Anzahl der in diesen Wert miteingegangenen Residuen an.

Abb. 4.12 und 4.13 zeigen die Fehlerverteilungen in Abhängigkeit der Exzentrizität und der Tiefe. Wie erwartet zeigen beide ein nichtlineares Ansteigen des mittleren Positionsfehlers und eine zunehmende Standardabweichung mit der Tiefe, auch wenn dieses nicht so deutlich ausfällt wie beim Stereokameramodell zuvor. Auch zeigt die Fehlerverteilung in Abhängigkeit der Exzentrizität ähnlich wie zuvor keinen eindeutigen Trend und spricht somit für eine gute Modellierung der Linsenverzeichnungen durch die RIDGE REGRESSION .

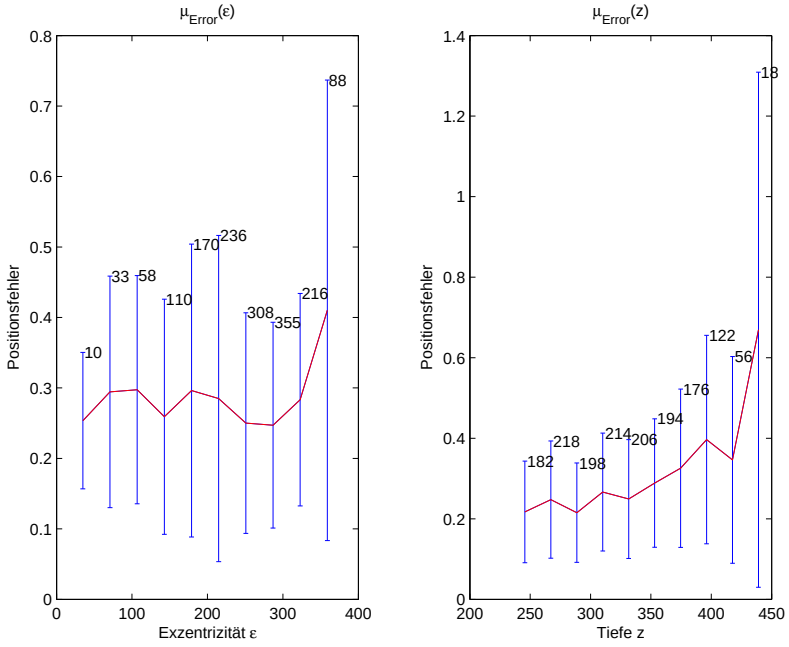

Abb. 4.13 Fehlerverteilung der RIDGE REGRESSION mit RBF-Kernel in Abhängigkeit der Exzentrizät $\mu_{Error}(\varepsilon)$ und der Tiefe $\mu_{Error}(z)$. Die Fehlerbalken geben die empirische Standardabweichung des Fehlers in dieser Exzentrizität bzw. Tiefe an. Die Zahlen an den Fehlerbalken geben die Anzahl der in diesen Wert miteingegangenen Residuen an.

Allgemein scheinen beide Kernel für das Lernen der Abbildung geeignet. Während die Residuen der RIDGE REGRESSION mit polynomiellen Kernel auf den vorliegenden Testdaten eine geringere Vorzugsrichtung aufweisen, führt die Verwendung des RBF-Kernels zu einem besseren Testfehler. Insgesamt führen beide aber zu guten Resultaten.

4.3.2 Support Vektor Regression

Für die SVR wurde eine Parametersuche ebenfalls für polynomiellen und RBF-Kernel, also auf den Mengen $\mathcal{C} \times \mathcal{E} \times \mathcal{D}$ bzw. $\mathcal{C} \times \mathcal{E} \times \Sigma$ durchgeführt. Da das ein Training einer SVR um einiges länger dauert, als das einer RIDGE REGRESSION (einige Kreuzvalidierungen waren nach fünf Tagen Rechenzeit immer noch nicht abgeschlossen), blieben die Ergebnisse der Parametersuche unvollständig. Bei der Auswertung der berechneten Kreuzvalidierungen stellte sich allerding heraus, daß für einen optimalen Kernelparameter der Wert des Fehlers nur noch von den Werten C und ϵ abhängt. D.h. um einen besseren Fehlerwert zu erhalten muß der Wert des Strafterms C erhöht und der der "Schlauchbreite" ε verringert werden, was allerdings zu längeren Trainingszeiten führt. Da die Daten ebenfalls mit Fehlern behaftet sind, hat die Verbesserung des Fehlerwertes durch Veränderung von C und ϵ sicher ihre Grenzen. Es ist allerdings vorstellbar, daß man mit $C = \infty$ und RBF-Kernel, ε bis zu einem Ansteigen des Kreuzvalidierungsfehlers verringern könnte. ε wäre dann ein Maß für den Fehler in den Ausgabedaten. Dies wurde allerdings im Rahmen dieser Arbeit nicht mehr weiter untersucht.

Die nun folgenden Ergebnisse beziehen sich auf SVR mit den Parametern $C = 15000, \varepsilon = 0.5$ und $d = 3$ bzw. $\sigma = 1$. Für die Werte ergab sich ein Kreuzvalidierungsfehler von 0.45599mm für polynomiellen und 0.7887mm für RBF-Kernel. Auf den Testdaten wiesen sie eine Fehler von 0.4304mm bzw. 0.785mm auf.

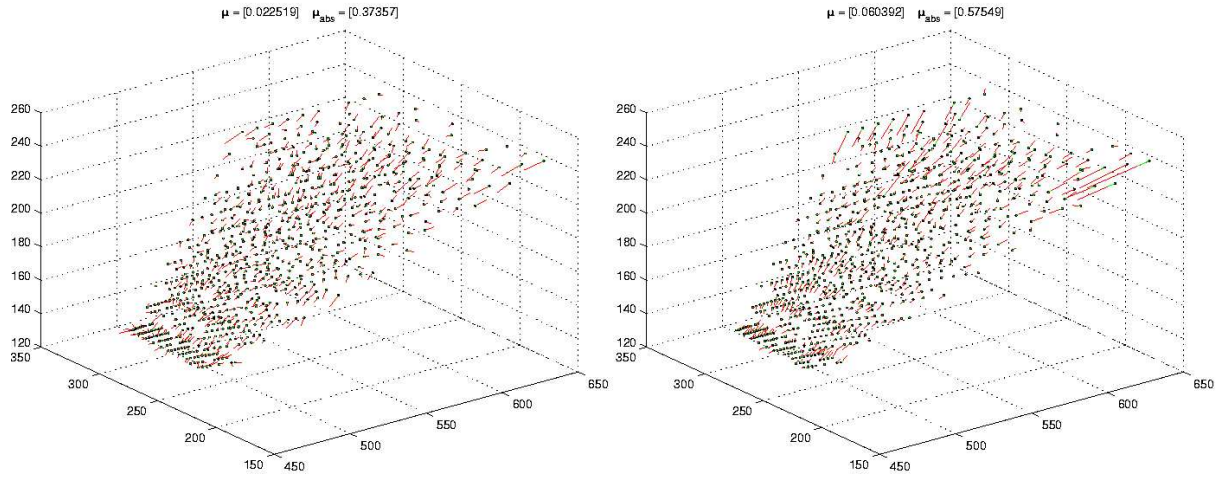

Abb. 4.14 Positionsresiduen der SVR mit polynomiellen (links) bzw. RBF-Kernel (rechts) und den Parametern $C = 15000, \varepsilon = 0.5$ und $d = 3$ bzw. $\sigma = 1$.

Ähnlich wie bei der RIDGE REGRESSION wiesen auch Positionsresiduen keine nennenswerte Vorzugsrichtung auf (die Abbildungen hierzu finden sich im Anhang). Der Mittelwert der Norm der einzelnen Residuen und die Norm des Mittelwerts der Residuen betrug für polynomiellen Kernel $\mu_{abs} = 0.37757$mm und $\mu = 0.022519$mm. Für RBF-Kernel ergaben sich Werte von $\mu_{abs} = 0.57549$ und $\mu = 0.060392$mm. Wie auch schon bei der RIDGE REGRESSION ist der Anteil von μ an μ_{abs} mit ca. 6% bei der Verwendung des polynomiellen Kernels geringer als bei der Verwendung des RBF-Kernels mit ca. 10.5% und spricht in dieser Hinsicht dafür, dem polynomiellen Kernel den Vorzug zu geben.

Im Gegensatz zu RIDGE REGRESSION benutzt die SVR bei der Berechnung der Position des Raumpunkts nicht alle Trainingsbeispiele, sondern nur die Supportvektoren (siehe hierzu 2.1.2).

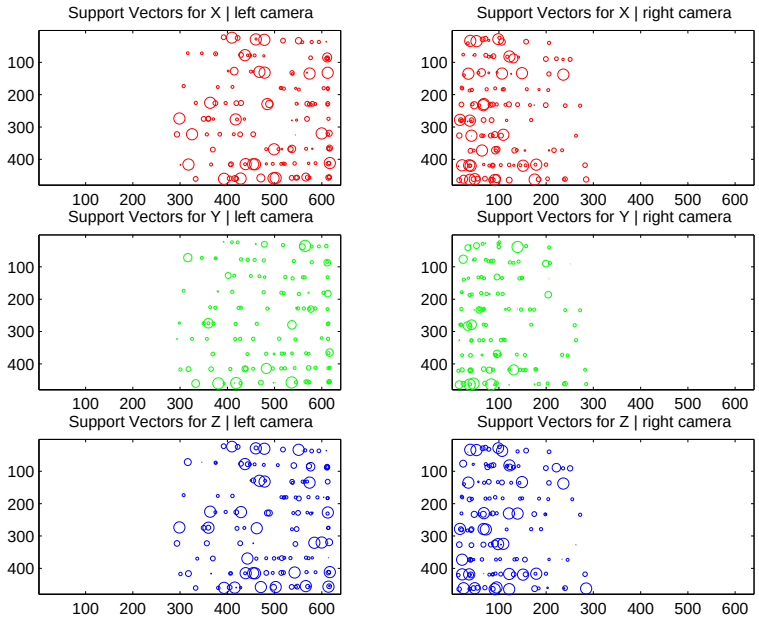

Abb. 4.15 Supportvektoren der SVR mit polynomiellen Kernel. Der Radius der einzelnen Kreise gibt die relative Größe des Lagrangeschen Multiplikators und damit den Einfluß dieses Trainingsbeispiels auf die Aussgabe an.

Abb. 4.15 zeigt die Supportvektoren für jede Raumrichtung (die Abbildung der Supportvektoren den für RBF-Kernel befindet sich im Anhang). Der Radius der einzelnen Kreise gibt die relative Größe der Lagrange-Multiplikatoren und damit den Einfluß auf den Ausgabewert an. Wie zu erwarten war, sind die Supportvektoren gleichmäßig über den gesamten Bildbereich verteilt,

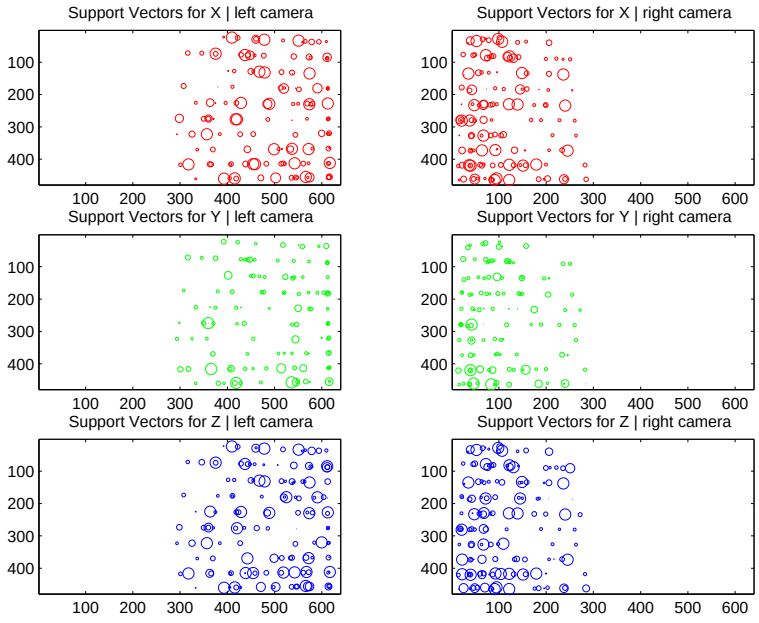

Abb. 4.16 Support Vektoren der SVR mit RBF-Kernel. Der Radius der einzelnen Kreise gibt die relative Größe des Lagrangen Multiplikators und damit den Einfluß dieses Trainingsbeispiels auf die Aussgabe an.

da es keine ausgezeichnete Position in den einzelnen Bildern gibt, die für die Berechnung der Raumpositi-

on besonders wichtig wäre. Somit sind auch fast 75% der Trainingsbeispiel Supportvektoren, wie in folgender Tabelle nachzulesen ist. Wie zu erwarten besitzen die Algorithmen für die Tiefendimension mehr Support Vektoren, da diese Abbildung schwerer zu lernen ist.

	n_{SV}^X	n_{SV}^Y	n_{SV}^Z
RBF-Kernel	142	118	154
polynomieller Kernel	140	123	144

Abb. 4.17 Anzahl der Supportvektoren für die einzelnen Raumrichtungen und verschiedene Kernel

Auch das Verhalten des Positionsfehlers mit zunehmender Tiefe und in Abhängigkeit der Exzentrizität entspricht bei der SVR den Erwartungen. Abb. 4.17 zeigt die die entsprechenden Fehlerverteilungen für polynomiellen Kernel. Der Positionfehler zeigt einen eindeutigen nichtlinearen Anstieg mit der Tiefe, wohingegen der linke Plot wiederum keinen eindeutigen Trend erkennen läßt. Die Verteilungen für RBF-Kernel verhalten sich ähnlich (siehe Anhang).

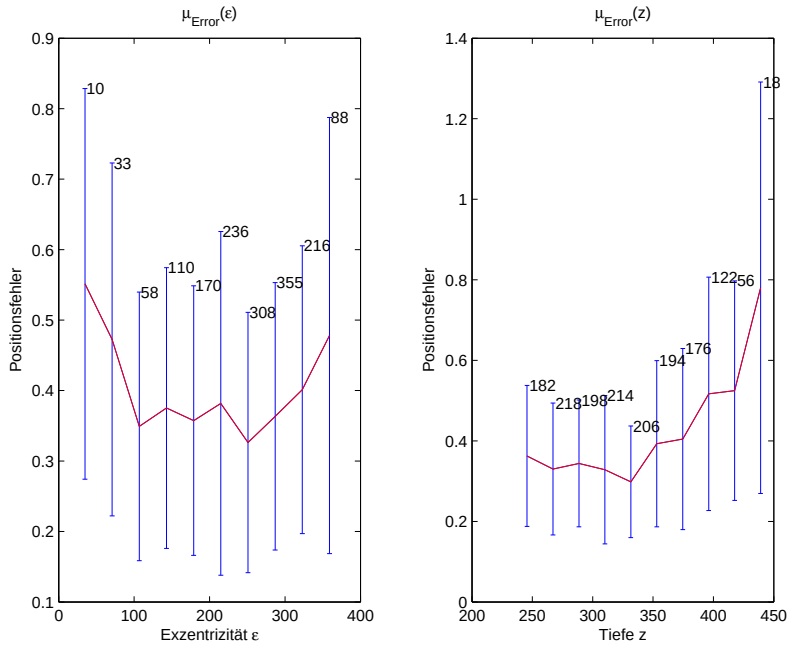

Abb. 4.18 Fehlerverteilung der SVR mit polynomiellem Kernel in Abhängigkeit der Exzentrizät $\mu_{Error}(\varepsilon)$ und der Tiefe $\mu_{Error}(z)$. Die Fehlerbalken geben die empirische Standardabweichung des Fehlers in dieser Exzentrizität bzw. Tiefe an. Die Zahlen an den Fehlerbalken geben die Anzahl der in diesen Wert miteingegangenen Residuen an.

Im Fall der SVR ist nun der Verwendung des polynomiellen Kernels eindeutig der Vorrang zu geben. Er weist nicht nur einen geringeren Testfehler als die SVR mit RBF-Kernel auf, auch die Positionsresiduen besitzen, wie schon erwähnt, eine weniger ausgeprägte Vorzugsrichtung.

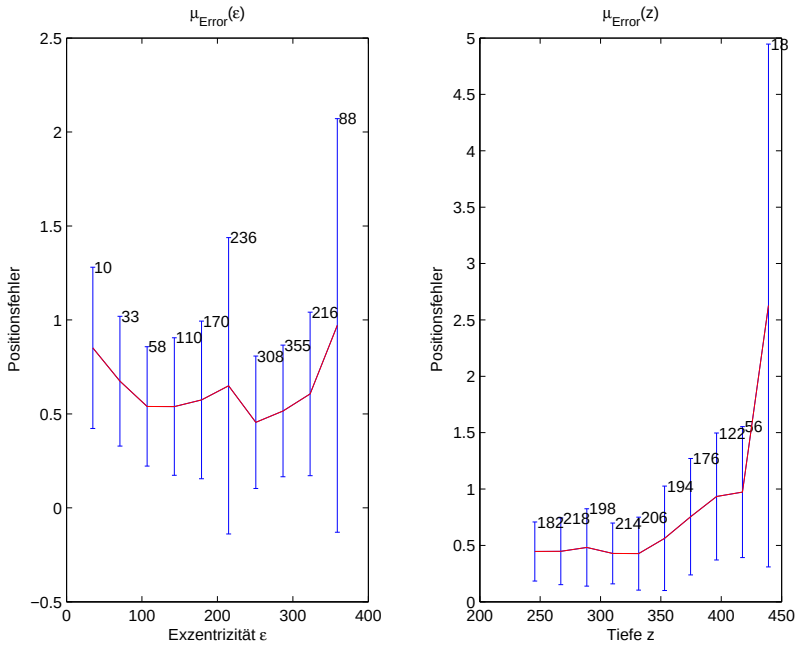

Abb. 4.19 Fehlerverteilung der SVR mit RBF-Kernel in Abhängigkeit der Exzentrizät $\mu_{Error}(\varepsilon)$ und der Tiefe $\mu_{Error}(z)$. Die Fehlerbalken geben die empirische Standardabweichung des Fehlers in dieser Exzentrizität bzw. Tiefe an. Die Zahlen an den Fehlerbalken geben die Anzahl der in diesen Wert miteingegangenen Residuen an.

4.4 Vergleich

Die drei nun vorgestellten Möglichkeiten der Kalibration einer Einzel- bzw. Stereokamera lassen sich in verschiedenen Aspekten gegenüberstellen.

Eine notwendige Vorarbeit ist die Parametersuche bei den Lernalgorithmen bzw. die Suche nach der passenden Parameterisierung bei den Einzelkameras. Kann die Suche bei der Ridge Regression und der SVR automatisiert ablaufen, muß sie bei bei der expliziten Modellierung per Hand erfolgen und unterliegt oftmals subjektiven Entscheidungen, wie z.B. welche Korrelationen noch toleriert werden kann. Vom Zeitverbrauch ist hier die Ridge Regression am schnellsten, da ihre Parametersuche auf einem herkömmlichen PC in einigen Minuten berechnet werden kann. Die meiste Zeit benötigt die SVR , bei der dieser Prozess, wie schon erwähnt, mehrere Tage benötigt und damit eigentlich den Rahmen des sinnvollen Zeitaufwands sprengt. Der zeitliche Vorteil der Ridge Regression gegenüber der SVR beruht auch auf der Tatsache, daß bei erster nur zwei Parameter bestimmt werden müssen, während es bei der SVR drei sind. D.h. einer zweidimensionalen Gittersuche steht eine dreidimensionale gegenüber. Die korrekte Parameterisierung einer Einzelkamera für die explizite Modellierung kann in wenigen Stunden gefunden werden, birgt aber immer noch die Gefahr, Daten bzw. Plots zu mißinterpretieren und sich somit sprichwörtlich auf den Holzweg zu begeben, was bei einem automatischen Verfahren, das stur nach Fehlerwert entscheidet, schwer möglich ist. Die Parameterisierung des Modells besitzt aber dann den Vorteil, daß ihre Parameter eine physikalische Interpretation besitzen, die sogar noch weitere Informationen über die Position, die Orientierung etc. der zu kalibrierenden Kamera liefert.

Was die Trainingszeit der einzelnen Algorithmen angeht, so liegt hier die gleiche Reihenfolge vor. Die meiste Zeit wird von der SVR benötigt, deren Zeitkomplexität bei $\mathcal{O}(n^3)$ liegt. Am schnellsten ist wiederum die Ridge Regression, da bei ihr der limitierende Faktor die Invertierung der Matrix $(K + \gamma\mathcal{I})$ ist, die bekanntlich auch in $\mathcal{O}(n^3)$ (für die Definition der sog. *Landauschen Symbole* siehe Definition 6.1.4) berechnet werden kann. SVR und Ridge Regression liegen zwar in derselben Komplexitätsklasse, die SVR ist jedoch

um einen konstanten Faktor langsamer, der dann den faktischen Zeitunterschied im Training ausmacht. Der limitierende Faktor bei der Regression des Kameramodells ist die Berechnung der $\mathcal{B}$-Matrix, aus den einzelnen Jacobimatrizen, wobei die analyische der numerischen Berechnung zeitlich eindeutig überlegen ist. Ein weiterer limitierender Zeitfaktor der Modellregression ist die Invertierung der Matrix $(\mathcal{B}^T \Sigma_{ll} \mathcal{B})$. Wie oft diese Invertierung geschehen muß, hängt von der Anzahl der Regressionsschritte ab. Die RIDGE REGRESSION und die Regression des Modells befinden sich also in der selben Komplexitätsklasse, wobei sich die RIDGE REGRESSION in der Praxis immer als schneller erweist.

Ein Vorteil der SVR gegenüber den beiden anderen Methode wäre die Möglichkeit des Nachbesserns. Stellt man fest, daß der Algorithmus in einer bestimmten Raumregion nicht tolerierbare Fehler begeht, dann können dort einfach neue Daten aufgenommen werden, welche mit den Support Vektoren eine neue Trainingsmenge ergeben, mit welcher eine neue SVR trainiert wird. Da diese aber so viel länger braucht als die RIDGE REGRESSION oder die explizite Modellierung, bietet dies keinen zeitlichen Vorteil gegenüber dem erneuten Training der RIDGE REGRESSION und des Modells mit zusätzliche Daten aus dieser Region, also einem insgesamt größeren Datensatz.

Ein weiterer Vorteil der SVR ist die grundsätzlich schnellere Auswertung des gelernten Modells. Da die RIDGE REGRESSION keine Support Vektoren besitzt, setzt sich das Ergebnis der gelernten Funktion immer aus einer Linearkombination der Kernelfunktion aller Trainingsbeispiele zusammen und ist somit langsamer. Die Auswertung des expliziten Modells benötigt noch mehr Zeit, da für jeden zu berechnenden Raumpunkt zunächst zwei Vektorfelder lokal invertiert werden müssen. Dies kann aber beschleunigt werden indem man einfach die Verschiebung für alle mögliche Bildpunkte berechnet und somit ein inverses Vektorfeld erschafft. Auf diese Weise kann man die Position des entstörten Punkts in $\mathcal{O}(1)$ erhalten. Die Auswertung der SVR wäre eventuell auch noch zu beschleunigen, indem man sog. *reduced set* Methoden auf die Menge der Support Vektoren anwendet. Diese verkleinern eine Menge von Daten unter Erhalt von möglichst viel Information. Dies wurde aber nicht weiter untersucht. Zum jetzigen Stand der Arbeit besteht aber kein großer zeitlicher Unterschied zwischen der Auswertung der SVR und der RIDGE REGRESSION , da im vorliegenden Fall fast alle Datenpunkte zu Supportvektoren werden und somit das eben genannte Argument stark an Gewicht verliert.

Was die Fehlerwerte angeht, so kommen aller drei Ansätz unter einen Positionsfehler von 1mm und sind somit -natürlich je nach Anwendung- für die Tiefenbestimmung geeignet. Wie schon erwähnt, bezahlt man allerdings bei der SVR einen besseren Fehlerwert mit längeren Trainingszeiten. Zu erwarten ist jedoch das die Genauigkeit des Tiefenfehlers für die Lernalgorithmen sehr schnell schlecht wird, sobald die Position der Raumpunkte außerhalb des Arbeitsbreichs des Roboters liegt. Dies dürfte bei der modellbasierten Kalibrierung auch der Fall sein, allerdings nicht in diesem Maße.

5 Zusammenfassung und Ausblick

Diese Studienarbeit stellte einen Vergleich zwischen der klassischen Tiefen- bzw. Positionsbestimmung eines Punkte aus zwei Bildern und modernen Lernalgorithmen an. Es stellte sich heraus, daß diese Algorithmen, durch den Kernel befähigt, effizient nichtlineare Funktionen zu lernen, vergleichbare und teilweise bessere Genauigkeiten erzielen als die Methode der expliziten Modellierung der beiden Stereopartner. Besonders die RIDGE REGRESSION tat sich durch hohe Genauigkeiten, effiziente Parametersuche und Training hervor. Die SVR scheint aufgrund des hohen Zeitaufwands eher weniger gut geeignet. Hier gilt es aber nochmal zu betonen, daß diese Genauigkeit überhaupt erst durch die hohe Stellgenauigkeit des Robotermanipulators erreicht werden konnte. Die klassische Methode besitzt weiterhin den Vorteil, daß die meisten Parameter des Modells ein bedeutungstagendes Gegenstück in der Realität besitzen. Die Entscheidung welcher Algorithmus zum Einsatz kommen wird, hängt letztlich vom zur Verfügung stehenden Material und dem Zeitaufwand ab, den man gewillt ist zu investieren.

Eine interessante Fortführung der Arbeit könnte eine Erweiterung der Eingaben um die Pan- und Tiltwinkel darstellen, um somit die Positionsbestimmung nicht für eine bestimmte, sondern für beliebige Stellungen der Schwenkeinheit zu lernen. Dies würde allerdings sehr wahrscheinlich die Trainingsmenge stark vergrößern, da allein für das Training der Algorithmen für hinreichend viele Orientierungen jeweils ca. 200

Datenpunkte aufgenommen werden müssten und somit die benötigte Zeit für Datenaquisition, Training und Parametersuche in die Höhe schnellen würde.

Weiterhin könnte man auch noch das Verhalten weiterer multidimensionaler Lernalgorithmen, wie z.B. *partial least squares* ergründen. Ein kleiner Test mit *Gaußschen Prozessen* zeigte bereits mit einem Positionsfehler[15] von 0.182mm, daß die Leistung der RIDGE REGRESSION noch zu überbieten ist.

Denkbar wäre auch eine Erweiterung des Kameramodells um Fehler in der Ausgabe. Da der Roboter seinen Arm nicht unendlich genau positionieren kann, könnte man ein gewisses "Wackeln" der Raumpunkte zulassen und somit eventuell die Genauigkeit des Kameramodells nochmals erhöhen. Eine gute Schätzung der Größenordnung des Fehlers könnte man aus der technischen Spezifikation des Manipulators erhalten.

6 Anhang

6.1 Definitionen

DEFINITION 6.1.1(Gram- oder Kernel-Matrix)

Gegeben ein Funktion $k : \mathcal{X} \times \mathcal{X} \to \mathbb{K}$ ($\mathbb{K}$ kann $\mathbb{C}$ oder $\mathbb{R}$ sein) und Beispiele $x_1, \ldots, x_m \in \mathcal{X}$. Die $m \times m$ Matrix

$$K_{ij} := k(x_i, x_j)$$

heißt *Gram-Matrix* oder *Kernel-Matrix* von k unter den Beispielen $x_1, \ldots, x_m$.

DEFINITION 6.1.2(positiv semidefinite Matrix)

eine komplexwertige $m \times m$ Matrix K heißt *positiv semidefinit*, wenn sie folgende Bedingung genügt

$$\sum_{i=1}^{m}\sum_{j=1}^{m} c_i \overline{c_j} K_{ij} \geq 0 \text{ für alle } c_i \in \mathbb{C}$$

Dies ist genau dann der Fall, wenn sie ausschließlich nicht-negative Eigenwerte besitzt.

Würde man den Fall der Gleichheit ausschließen, dann hieße die Matrix *positiv definit*.

DEFINITION 6.1.3(Spline)

Eine Splinefunktion der Ordnung $l \in \mathbb{N}$ zu einer Zerlegung Δ der Stützwerte ist eine Funktion $s \in C^{l-1}[a, b]$, die au jedem Intervall $[x_{k-1}, x_k]$ mit einem Polynom $l-ten$ Grades übereinstimmt. Anstelle von Splinefunktion wird oft auch die Bezeichnung *Spline* verwendet.

DEFINITION 6.1.4(Landausche Symbole oder $\mathcal{O}$-Notation)

Eine Funktion f ist in $\mathcal{O}(g)$ wenn Konstanten c_1 und c_2 existieren, so daß für alle n in $\mathbb{N}$ $f(n) \leq c_1 g(n) + c_2$ gilt, also

$$\mathcal{O}(g) = \{f | \exists c_1, c_2 \forall n \in \mathbb{N} : \ f(n) \leq c_1 \cdot g(n) + c_2\}$$

Eine dazu äquivalente Formulierung ist

$$\mathcal{O}(g) = \{f | \exists n_0 \in \mathbb{N} \exists c \forall n \in \mathbb{N} : (n_0 < n) \Rightarrow \ f(n) \leq c \cdot g(n)\}$$

[15] root mean squared error der Positionsresiduen

Literatur

[1] Sankar Basu Charles Micchelli Joos Vandewalle et. al Johan Sykens, Gabor Horvath. *Advances in Learning Theory - Methods, Models and Applications.* IOS Press Ohmsha, 1999.

[2] Steven M. Kay. *Statistical Signal Processing*, volume I. Prentice Hall, 1993.

[3] Thomas Luhmann. *Nahbereichsphotogrammetrie - Grundlagen, Methoden und Anwendungen.* Wichmann, 2000.

[4] Max Planck Institute for Biological Cybernetics, Australian National University, Institute de Recherche en Communications et en Cybernetique (Nantes). *Machine Learning Summer School Tübingen*, August 2003.

[5] P.J. McKerrow. *Introduction to Robotics.* Addison-Wesley, 1989.

[6] John Shawe-Taylor Nello Cristianini. *Support Vector Machines - and other kernel-based methods.* Cambridge University Press, 2000.

[7] Bernhard Schölkopf und Alexander J. Smola. *Learning with Kernels.* MIT Press, 2002.

Index