Kamera-Manipulator-Kalibrierung und Lokalisierung von Greifobjekten mit RBF- und HBF-Netzen

Diplomarbeit
am Lehrstuhl für Kognitive Systeme
des Instituts für Informatik und Praktische Mathematik
der Christian–Albrechts–Universität zu Kiel

vorgelegt von

Michael Benkwitz

Betreuer: Dr. Josef Pauli

August 1996

Zusammenfassung

Eine wichtige Aufgabe der visuell basierten Robotik ist die Realisierung eines Systems, das die Lokalisierung und Erkennung von Objekten und eine anschließende Annäherung durch einen Manipulator ermöglicht. Bei dieser Realisierung sollte immer der Mensch als Vorbild dienen, der bereits wenige Wochen nach der Geburt in der Lage ist, Dinge mit den Augen zu sehen, sie zu erkennen und mit den Händen zu greifen. Diese für den Menschen selbstverständliche Fähigkeit ergibt sich aus einer *Perzeptions-* (Wahrnehmen), einer *Kognitions-* (Erkennen) und einer *Aktionskomponente* (Greifen). Dabei ist für das Funktionieren der Kognitionskomponente ein bereits erlerntes Wissen über die *Erscheinung* der zu greifenden Objekte, d.h. deren Form und Struktur unter variablen Bedingungen, wie z.B. Beleuchtung, Umgebung und Orientierung, erforderlich.

In dieser Arbeit wird ein lernfähiges System vorgestellt, das mit Hilfe neuronaler Netze die Erscheinung unterschiedlicher Objekte erlernt, um sie später wiederzuerkennen. Außerdem soll dieses System anschließend einen Roboterarm (Manipulator) dazu veranlassen, sich einem erkannten Objekt anzunähern. Zur Realisierung dieses Systems wird eine Sun Workstation verwendet. Außerdem werden ein Stereokamerasystem und ein Roboterarm zur Simulation der Perzeptions- und Aktionskomponente benötigt. Ein Hauptproblem stellt dabei die Erkennung des Objektes in den beiden Bildern der Kameras sowie die anschließende Berechnung der Position des Objektes im dreidimensionalen Raum dar. Es ist hierfür einerseits eine Beschreibung der gesuchten Objekte (Zielobjekte) erforderlich und andererseits eine Beschreibung der Beziehung zwischen den Koordinatensystemen der beiden Kameras und dem des Roboters. Diese Beschreibungen werden durch das Trainieren von sogenannten *Hyperbasisfunktionen-Netzen (HBF-Netze)* und *Netzen Radialer Basisfunktionen (RBF-Netze)*, einer spezielleren Form der HBF-Netze, gelernt. Die Anwendung dieser neuronalen Netze ermöglicht die Lokalisierung der Zielobjekte sowie deren Annäherung durch den Manipulator. Durch eine lineare Kombination aus nichtlinearen gaußförmigen *Basisfunktionen* werden mit diesen Netzen Transformationen von Vektoren interpoliert, wobei die Eingabe- und Ausgabevektoren von unterschiedlich hoher Dimension sein können. Mit einer Menge von Eingaben und zugehörigen Ausgaben werden die neuronalen Netze trainiert, so daß sie in der Anwendung die gelernten Daten approximieren und bei unbekannter Eingabe aufgrund einer Generalisierung der gelernten Eingaben interpolierte Ausgaben liefern.

Insgesamt wird durch die Realisierung dieses lernfähigen Systems eine der wichtigsten Kombinationen natürlicher Prozesse beim Menschen simuliert, nämlich die Kombination von Wahrnehmung, Erkennen und Greifen.

Inhaltsverzeichnis

Kapitel 1

Einführung

In diesem Kapitel soll die Problematik bei einer visuell basierten Annäherung an ein Objekt näher erläutert werden. Es wird weiterhin ein Verfahren zur Lösung des Problems beschrieben, das wegen der Vermeidung aufwendiger mathematischer Algorithmen effizient ist.
Ein lernfähiges System zur visuell basierten Annäherung an ein Objekt erfordert zwei grundlegende Hauptbestandteile:

- ein System zur Lokalisierung und Erkennung von Objekten
- ein System zur Annäherung eines bereits erkannten Objektes durch einen Manipulator

Beide Systeme sollen robust arbeiten, d.h. sie sollen in unterschiedlichsten Situationen ihre Aufgaben mit möglichst geringem Fehler ausführen. In dieser Arbeit werden zur Realisierung dieser Systeme eine Sun Workstation mit dem Bildverarbeitungspaket KHOROS und weiterhin ein Manipulator sowie zwei auf einem Stativ befestigte Kameras verwendet, die den Manipulator beobachten. Die Workstation realisiert die beiden Systeme und stellt eine Verbindung zwischen dem Roboter und den Kameras dar. Für das Ziel der Annäherung des Objektes werden zunächst zwei Aufnahmen mit den Kameras gemacht. In diesen wird dann das Objekt lokalisiert und erkannt, damit es anschließend vom Manipulator angenähert werden kann.
Im folgenden sollen die beiden Systeme näher erläutert werden.

System zur Lokalisierung und Erkennung von Objekten

Bei der Objekterkennung ist es wichtig, daß nicht versehentlich ein Objekt falsch erkannt wird, weil es dem gesuchten Objekt ähnelt (*Diskriminierung* von Objekten). Dieses Problem wird durch das Erzeugen einer endlichen *Trainingsmenge*

von Objektbildern gelöst, die ein bestimmtes Objekt (*Zielobjekt*) bei unterschiedlichen Bedingungen zeigen. Da unendlich viele solcher Bedingungen existieren, erhält man durch die endliche Trainingsmenge nur eine komprimierte Darstellung der Mannigfaltigkeit an Situationen. Anschließend wird mit dieser Trainingsmenge eine *Erkennungsfunktion* für das Objekt gelernt, die für jede Position in einem Kamerabild einen *Zuverlässigkeitswert* berechnet, durch den entschieden wird, ob sich dort das Zielobjekt befindet. Dadurch kann das Objekt auch aus unterschiedlichen Blickwinkeln der beiden Kameras oder bei unterschiedlichen Beleuchtungen erkannt werden. Das Lernen der Erkennungsfunktion erfolgt mit HBF-Netzen, deren Aufbau in Kapitel 3 beschrieben wird.
Die meisten Arbeiten haben sich mit einer modellbasierten Erkennung von Objekten beschäftigt, wobei jeweils die *geometrischen* Eigenschaften eines Objektes untersucht wurden [Tay94]. Bei diesen Verfahren ergibt sich jedoch ein Verlust wichtiger Informationen. In dieser Arbeit werden daher ausschließlich *photometrische* Eigenschaften verwendet, d.h. die Darstellung eines Objektes erfolgt hier wesentlich abstrakter durch die Grauwerte in einem Bild [Mur94]. Der Begriff '*Erscheinung* eines Objektes' bedeutet also in dieser Arbeit nicht die geometrische Gestalt des Objektes, sondern die Struktur der Grauwerte in einem Bild an der Position des Objektes.

System zur Manipulatorsteuerung

Das Hauptproblem der Manipulatorsteuerung besteht darin, daß der Abbildungsebene einer Kamera ein $2D$-Koordinatensystem und dem Robotersystem ein $3D$-Koordinatensystem zugrundeliegen. So kann z.B. der Manipulator in eine bestimmte Position gebracht werden, die eindeutig durch einen dreidimensionalen Vektor des Basiskoordinatensystems des Roboters dargestellt wird. Ist jedoch der zweidimensionale Vektor eines Punktes im Bild bekannt, muß zunächst der entsprechende Punkt im Raum (dreidimensional) berechnet werden, um den Manipulator dorthin steuern zu können.
Die Rekonstruktion der Position eines Raumpunktes ist nicht eindeutig, weil dessen Koordinaten in der Abbildungsebene einer Kamera, dargestellt durch einen zweidimensionalen Vektor, hierfür nicht ausreichen. Hier ist z.B. der Fall möglich, daß zwei unterschiedliche Punkte auf dieselbe Position der Abbildungsebene projiziert werden, nämlich genau dann, wenn die Verbindungsgerade der Punkte durch das Linsenzentrum verläuft (Projektionsstrahl). Um dieses Problem zu vermeiden, wird ein Stereokamerasystem verwendet, das durch die Projektion eines Punktes auf zwei Abbildungsebenen einen eindeutigen vierdimensionalen *Abbildungsvektor* liefert. Es ist allerdings dabei zu beachten, daß diese Projektion zwar injektiv, aber nicht surjektiv ist, da es vierdimensionale Vektoren gibt, die keine Position im Basiskoordinatensystem des Roboters repräsentieren. Die

Berechnung von Abbildungsvektoren eines Punktes aus den zugehörigen Roboterbasiskoordinaten sowie eine Umkehrung dieser Berechnung wird durch eine *Kalibrierung* der Kameras erreicht, wobei zwischen den beiden folgenden Kalibrierungsarten unterschieden wird:

- *Stereokamera-Manipulator-Kalibrierung*
- *Manipulator-Einzelkamera-Kalibrierung*

Für diese Arbeit ist die erste Art der Kalibrierung besonders wichtig, nämlich für die Steuerung des Manipulators in Richtung des Objektes, dessen Koordinaten in beiden Kamerabildern bereits bestimmt wurden. Wird hierfür die Position im Basiskoordinatensystem des Roboters berechnet, kann der Manipulator direkt dorthin bewegt werden. Es wird in dieser Arbeit ein RBF-Netz für die Stereokamera-Manipulator-Kalibrierung verwendet, dessen Aufbau in Kapitel 3 beschrieben wird. Diese Art der Kalibrierung eignet sich jedoch nur für eine grobe Annäherung unter Vernachlässigung einer Hindernisvermeidung.
Sind die Basiskoordinaten des Manipulators bekannt, während die Bildkoordinaten bestimmt werden sollen, wird die Manipulator-Einzelkamera-Kalibrierung verwendet. Sie eignet sich insbesondere für eine *Servosteuerung* des Manipulators [Hag95] bzw. eine feine Annäherung, die z.B. wichtig ist, um beim Greifvorgang eine Kollision des Greifers mit dem Objekt oder anderen Hindernissen zu vermeiden. Nach der Lokalisierung und Erkennung des Objektes kann für jede Bewegung des Manipulators die Position der Greiffinger in den Bildern vorhergesagt werden. Dadurch können stets Beziehungen zwischen der Greifer- und der Objektposition in den Bildern hergestellt werden. Ein wesentliches Problem ergibt sich jedoch bei der feinen Annäherung durch schrittweise Bewegungen des Manipulators, wenn eine Extremstellung erreicht wird, aus der weitere Bewegungen unmöglich sind. Dieser Fall kann eintreten, wird aber in dieser Arbeit durch die Wahl der groben Annäherung vermieden.

Gesamtsystem

Die Kalibrierung der beiden Kameras sowie das Lernen von Erkennungsfunktionen für einzelne Objekte erfolgt 'offline' (Abb. 1.1). Es ist in Abb. 1.1 zu erkennen, daß für das Trainieren von RBF-/HBF-Netzen jeweils Trainingsmengen benötigt werden. Für die Erkennungsfunktion besteht diese Trainingsmenge aus Objektbildern mit ihren zugehörigen Zuverlässigkeitswerten und für die Kalibrierung aus Paaren von vierdimensionalen Computerbildkoordinaten mit den zugehörigen Weltkoordinaten. Nach dem Trainieren der Netze folgt deren Anwendung im 'Online'-Ablauf der Perzeptions-, Kognitions- und Aktionskomponenten (Abb. 1.1).

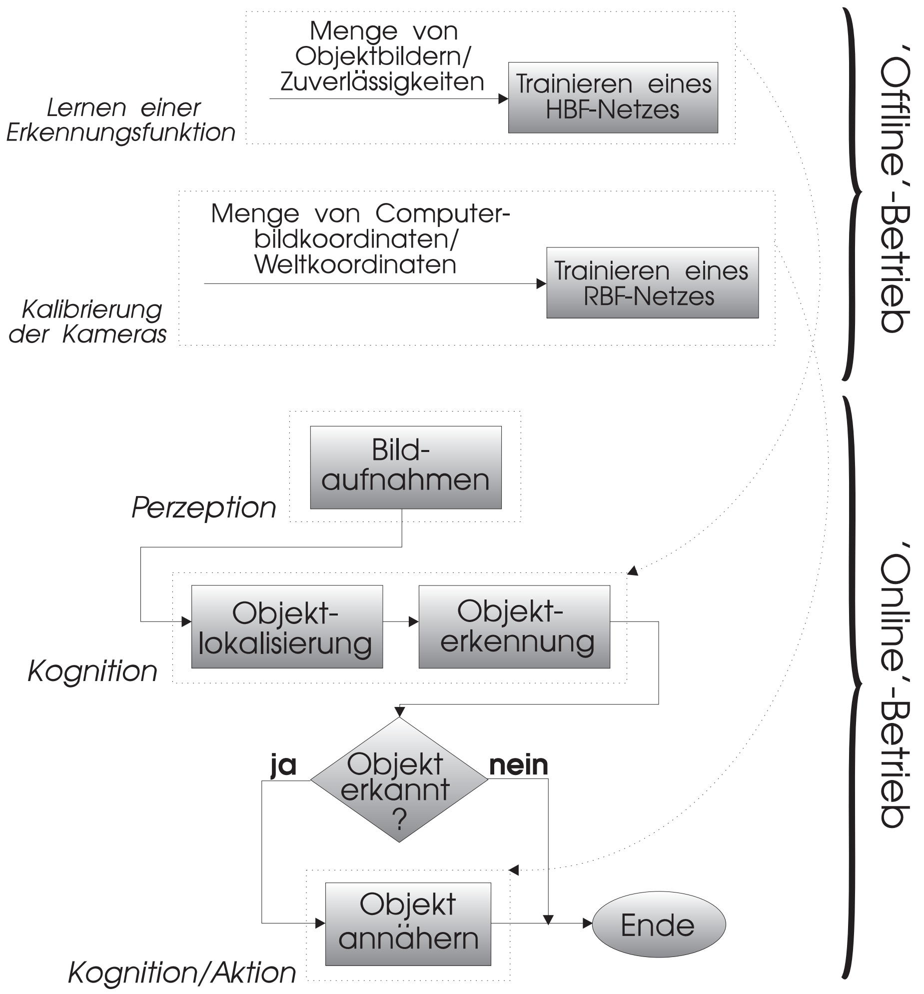


Abbildung 1.1: Der 'Offline'- und der 'Online'-Betrieb des Systems.

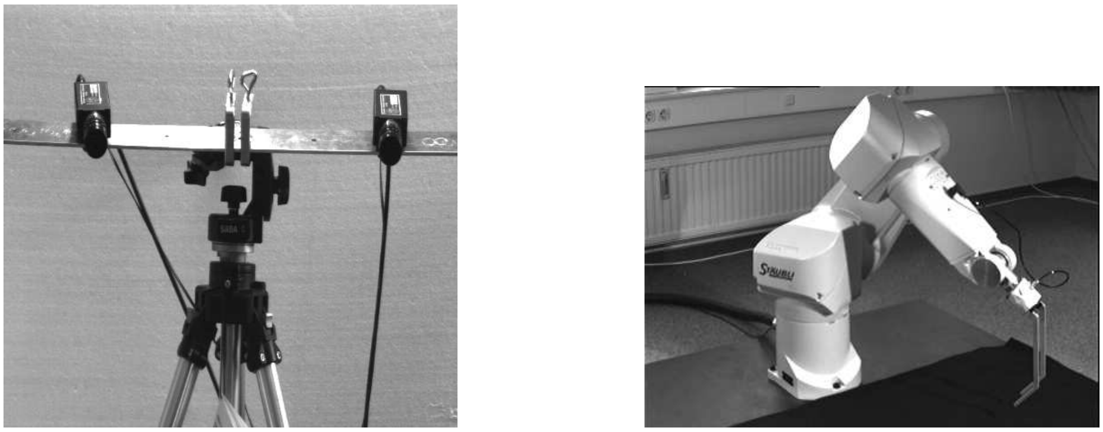
Eine Beschreibung des Kamera- und des Robotersystems sowie eine Erläuterung des Begriffs der Kalibrierung erfolgen in Kapitel 2.
Der Aufbau von HBF- und RBF-Netzen als Grundwerkzeuge für die Teilsysteme Objekterkennung und Robotersteuerung wird in Kapitel 3 dargestellt.
Kapitel 4 beschäftigt sich mit dem Problem der Stereokamera-Manipulator-Kalibrierung als Hauptteil der Robotersteuerung. Es wird zunächst der Ablauf des Trainierens eines RBF-Netzes zur Kalibrierung erläutert und dann mit einigen Versuchsreihen die Robustheit unterschiedlich konfigurierter RBF-Netze untersucht. Es soll so eine Entscheidung getroffen werden, wie das RBF-Netz zu konfigurieren ist, damit bei korrekter Lokalisierung und Erkennung eines Objektes dieses möglichst genau vom Manipulator angenähert werden kann. Mit diesen Versuchen wird gezeigt, daß viele Faktoren die Robustheit beeinflussen.

Ebenso soll für die Objektlokalisierung/-erkennung die Robustheit unterschiedlich konfigurierter HBF-Netze untersucht werden. Kapitel 5 beschreibt verschiedene Lokalisierungsverfahren sowie die Invarianz/Diskriminanz-Problematik, die sich dadurch ergibt, daß die Fähigkeiten des Netzes, das Zielobjekt von anderen Objekten zu unterscheiden (Diskriminanz) und für verschiedene Erscheinungen des Zielobjektes gleiche Zuverlässigkeitswerte zu berechnen (Invarianz), sich gegenseitig ausschließen. Dabei ist der hier verwendete Begriff 'Photometrische Invarianz' von dem häufig verwendeten Begriff 'Geometrische Invarianz' [Woo96] zu unterscheiden. In dieser Arbeit wird eine Erkennung für unterschiedliche Erscheinungen des Objektes beschrieben, also bei *Realweltänderungen* (photometrisch). In anderen Arbeiten zur Objekterkennung erfolgten bisher invariante Erkennungen nur bezüglich Änderungen der Muster in den Computerbildern (geometrisch). Ein weiterer Abschnitt in Kapitel 5 beschäftigt sich mit unterschiedlichen *Operatoren*, die zum Trainieren der HBF-Netze erforderlich sind. Durch die Anwendung dieser Operatoren werden in unterschiedlichen Bildern eines Objektes zufällige Störungen herausgefiltert, so daß zur Beschreibung des Objektes lediglich die wichtigen Informationen herangezogen werden.
In Kapitel 6 wird abschließend erläutert, wie sich das System zur Objekterkennung und Manipulatorsteuerung evtl. verbessern läßt und womit sich nachfolgende Arbeiten beschäftigen können, die auf dieser Arbeit aufbauen.

Kapitel 2

Das Kamera- und das Robotersystem

Dieses Kapitel beschreibt den Aufbau des Stereokamera- und des Robotersystems (Abb. 2.1). Weiterhin sollen die Begriffe Stereokamera-Manipulator-Kalibrierung und Manipulator-Einzelkamera-Kalibrierung näher erläutert werden. Die Kalibrierung geschieht in dieser Arbeit durch die Anwendung eines RBF-Netzes neuronal und ist von der konventionellen Kalibrierung dadurch zu unterscheiden, daß explizite Berechnungen kameraspezifischer Parameter entfallen.



(a) (b)

Abbildung 2.1: Das Stereokamera- (a) und das Robotersystem (b).

Für eine grobe Annäherung des Manipulators ist ein großes Sichtfeld erforderlich, das durch starre Kameras, die sich außerhalb des Arbeitsbereiches befinden, gewährleistet wird. Eine Alternative zu diesem Aufbau mit einem starren Kamerasystem ergibt sich, indem die Kameras am Ende des Manipulators befestigt werden [Tsai89]. Vorteilhaft bei dieser Methode ist, daß der Manipulator das zu greifende Objekt im Bild nicht verdecken kann.

2.1 Das Stereokamerasystem

Zur Lokalisierung und Erkennung von Objekten wird ein Stereokamerasystem benötigt. In dieser Arbeit werden dafür zwei s/w-Kameras verwendet, die jeweils einen CCD-Chip (Charge Coupled Device) besitzen und starr sind, d.h. sie werden auf einem Stativ befestigt und bleiben unbeweglich (Abb. 2.1 (a)). Weiterhin werden zwei Objektive, jeweils mit einer Brennweite von $f = 12$ mm, benutzt. Die Brennweiten der Objektive sind ebenso unveränderlich.
Jeder Kamera liegt ein Koordinatensystem zugrunde, dessen Ursprünge mit den Symbolen C_l (Abb. 2.2) und C_r bezeichnet werden. Die optische Achse einer Kamera verläuft senkrecht zur *Kameraebene* und durch den Ursprung. Ein Raumpunkt P wird durch eine Kamera auf die *Abbildungsebene* abgebildet, welche parallel zur Kameraebene mit einem Abstand $\hat{f}_{l/r}$ (links/rechts) liegt. $\hat{f}_{l/r}$ bezeichnet man als *effektive Brennweite* der Kamera.

Es sind nun folgende Koordinaten zu unterscheiden:

- *Weltkoordinaten*, 3-dimensional, Einheiten in Millimeter (mm)
- *Kamerakoordinaten*, 3-dimensional, Einheiten in Millimeter (mm)
- *Chipbildkoordinaten*, 2-dimensional, Einheiten in Millimeter (mm)
- *Computerbildkoordinaten*, 2-dimensional, Einheiten in Pixel

Jeder Punkt im Raum, der im Sichtbereich einer Kamera liegt, besitzt eine eindeutige Darstellung in den Koordianten einer dieser vier Typen. Im folgenden werden einige Parameter und Berechnungen beschrieben, die die Umrechnung von Koordinaten eines Typs in die Koordinaten eines anderen Typs erlauben. Bei der Bildverarbeitung werden Computerbilder verwendet, während die Koordinaten des Robotergreifers stets in Koordinaten des Basiskoordinatensystems vorliegen. Wurde also ein Objekt in einem Computerbild lokalisiert und erkannt, werden dessen Koordinaten im Basiskoordinatensystem des Roboters benötigt, um eine Annäherung des Manipulators zum Objekt auszuführen. Da das Roboterbasiskoordinatensystem ein spezielles kartesisches Weltkoordinatensystem darstellt, werden die Basiskoordinaten auch als Weltkoordinaten bezeichnet. Um eine Berechnung der zweidimensionalen Computerbildkoordinaten aus den dreidimensionalen Weltkoordinaten eines Raumpunktes und die Umkehrung dieser Berechnung auszuführen, müssen einige kameraspezifische Parameter bekannt sein.

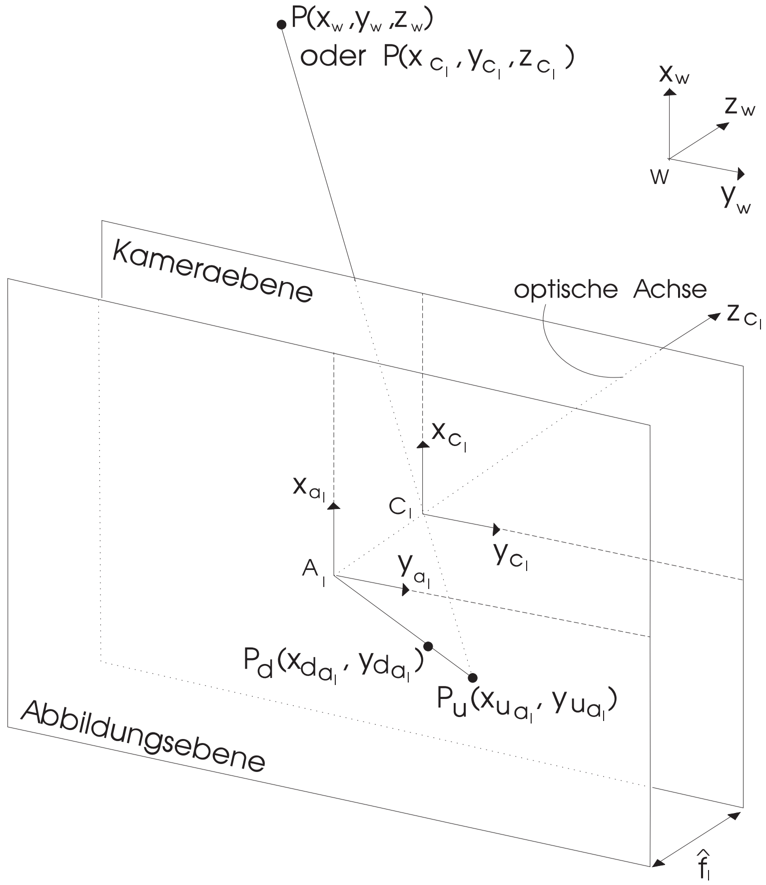


Abbildung 2.2: Projektion eines Raumpunktes mit der linken Kamera.

Man unterscheidet die *externen* von den *internen* Parametern. Zu den internen Parametern, die die optischen Eigenschaften des Linsensystems beschreiben, zählt u.a. die effektive Brennweite. Die externen Parameter beschreiben die Position und Orientierung des Kamerakoordinatensystems zum Weltkoordinatensystem. Sind nun die Kamerakoordinaten x_{c_l}, y_{c_l}, z_{c_l}, x_{c_r}, y_{c_r} und z_{c_r} eines Raumpunktes bezüglich der Kamerakoordinatensysteme bekannt, können die Chipbildkoordinaten des Punktes, der als Projektion durch die Kameras auf den Abbildungsebenen (dadurch ergibt sich der Index a) erscheint, wie folgt berechnet werden [Faug93]:

$$x_{u_{a_{l/r}}} = -x_{c_{l/r}} \frac{\hat{f}_{l/r}}{z_{c_{l/r}}}, \quad y_{u_{a_{l/r}}} = -y_{c_{l/r}} \frac{\hat{f}_{l/r}}{z_{c_{l/r}}} \tag{2.1}$$

Diese Werte ergeben sich jedoch nur unter der Annahme eines einfachen Lochkameramodells. Allerdings treten i.a. radiale Linsenverzerrungen auf, wodurch sich mit (2.1) statt der <u>u</u>nverzerrten Chipbildkoordinaten $(x_{u_{a_{l/r}}}, y_{u_{a_{l/r}}})$ verzerrte (distorted) Chipbildkoordinaten $(x_{d_{a_{l/r}}}, y_{d_{a_{l/r}}})$ ergeben. Dadurch wird die folgende Umrechnung benötigt:

$$x_{u_{a_{l/r}}} = x_{d_{a_{l/r}}} + D_{x_{l/r}}, \quad y_{u_{a_{l/r}}} = y_{d_{a_{l/r}}} + D_{y_{l/r}} \tag{2.2}$$

Dabei sind die Verzerrungen $D_{x_{l/r}}$ und $D_{y_{l/r}}$ abhängig von den Abständen der Chipbildkoordinaten zu den Ursprüngen $A_{l/r}$ und zwei (internen) *Verzerrungsfaktoren* $\kappa_{1_{l/r}}$ und $\kappa_{2_{l/r}}$ für jede Kamera. Da der CCD-Chip einer Kamera mit einer Anzahl $N_{c_{x_{l/r}}}$ von Sensorelementen in $x_{c_{l/r}}$-Richtung und einer Anzahl $N_{c_{y_{l/r}}}$ von Sensorelementen in $y_{c_{l/r}}$-Richtung bestückt ist, ergibt sich für jede Kamera ein diskretisiertes, $N_{b_{x_{l/r}}}$ Pixel breites und $N_{b_{y_{l/r}}}$ Pixel hohes Computerbild. Seien $d_{x_{l/r}}$ bzw. $d_{y_{l/r}}$ die Abstände zweier Sensorelemente in $x_{c_{l/r}}$- bzw. $y_{c_{l/r}}$-Richtung und $(C_{x_{l/r}}, C_{y_{l/r}})$ die Computerbildkoordinaten der Ursprünge $A_{l/r}$[1], dann ergeben sich mit den in (2.1) und (2.2) berechneten Koordinaten für die Computerbildkoordinaten $(x_{b_{l/r}}, y_{b_{l/r}})$ folgende Abhängigkeiten [Tsai86]:

$$\begin{aligned} x_{b_{l/r}} &= f_x\left(x_{u_{a_{l/r}}}, C_{x_{l/r}}, d_{x_{l/r}}, N_{c_{x_{l/r}}}, N_{b_{x_{l/r}}}\right) \\ y_{b_{l/r}} &= f_y\left(y_{u_{a_{l/r}}}, C_{y_{l/r}}, d_{y_{l/r}}, N_{c_{y_{l/r}}}, N_{b_{y_{l/r}}}\right) \end{aligned} \tag{2.3}$$

Dabei stellen f_x und f_y mathematische Funktionen dar, die hier nicht näher erläutert werden. Sie werden in [Tsai86] genauer beschrieben.

Mit den bisherigen Berechnungen lassen sich dreidimensionale Kamerakoordinaten in zweidimensionale Computerbildkoordinaten umrechnen. Da jedoch für einen vom Manipulator erreichbaren Raumpunkt nur dessen Weltkoordinaten bekannt sind, wird eine weitere Umrechnung von Weltkoordinaten in Kamerakoordinaten benötigt. Es sind hierfür jedoch weitere (externe) Kameraparameter erforderlich. Um die Kamerakoordinaten eines Raumpunktes aus dessen Weltkoordinaten x_w, y_w und z_w zu ermitteln, ist zunächst die Transformation innerhalb der beiden 3D-Koordinatensysteme zu betrachten [Faug93]. Sei W der Ursprung des Weltkoordinatensystems. Da man durch Rotation des Weltkoordinatensystems und anschließender Translation jeweils ein Kamerakoordinatensystem erhalten kann, ergeben sich die Kamerakoordinaten wie folgt:

$$\begin{pmatrix} x_{c_l} \\ y_{c_l} \\ z_{c_l} \end{pmatrix} = R_l \begin{pmatrix} x_w \\ y_w \\ z_w \end{pmatrix} + \vec{t}_l, \quad \begin{pmatrix} x_{c_r} \\ y_{c_r} \\ z_{c_r} \end{pmatrix} = R_r \begin{pmatrix} x_w \\ y_w \\ z_w \end{pmatrix} + \vec{t}_r \tag{2.4}$$

[1] Die Ursprünge $A_{l/r}$ erscheinen in der Regel im Zentrum des Computerbildes, d.h. es gilt $(C_{x_{l/r}}, C_{y_{l/r}}) = \left(\frac{N_{b_{x_{l/r}}}}{2}, \frac{N_{b_{y_{l/r}}}}{2}\right)$.

Dabei stellen R_l und R_r folgende Rotationsmatrizen und $\vec{t}_l$ und $\vec{t}_r$ folgende Translationsvektoren dar:

$$\begin{aligned} R_l &= \begin{pmatrix} r_{11_l} & r_{12_l} & r_{13_l} \\ r_{21_l} & r_{22_l} & r_{23_l} \\ r_{31_l} & r_{32_l} & r_{33_l} \end{pmatrix} \in \mathbb{R}^{3\times 3}, \quad \vec{t}_l = \begin{pmatrix} t_{x_l} \\ t_{y_l} \\ t_{z_l} \end{pmatrix} \in \mathbb{R}^3 \\ R_r &= \begin{pmatrix} r_{11_r} & r_{12_r} & r_{13_r} \\ r_{21_r} & r_{22_r} & r_{23_r} \\ r_{31_r} & r_{32_r} & r_{33_r} \end{pmatrix} \in \mathbb{R}^{3\times 3}, \quad \vec{t}_r = \begin{pmatrix} t_{x_r} \\ t_{y_r} \\ t_{z_r} \end{pmatrix} \in \mathbb{R}^3 \end{aligned} \tag{2.5}$$

Es folgt eine Auflistung der in diesem Abschnitt betrachteten internen und externen Parameter einer Kamera mit einer kurzen Beschreibung.

interne Kameraparameter:

Parameter	Beschreibung
$\hat{f}_{l/r}$	effektive Brennweiten
$\kappa_{1/2_{l/r}}$	Verzerrungsfaktoren
$C_{x_{l/r}}$	x-Koordinaten der Ursprünge $A_{l/r}$ im Computerbild
$C_{y_{l/r}}$	y-Koordinaten der Ursprünge $A_{l/r}$ im Computerbild
$d_{x_{l/r}}$	Abstand zweier Sensorelemente in x-Richtung der Kamerakoordinaten
$d_{y_{l/r}}$	Abstand zweier Sensorelemente in y-Richtung der Kamerakoordinaten
$N_{c_{x_{l/r}}}$	Anzahl der Sensorelemente in x-Richtung der Kamerakoordinaten
$N_{c_{y_{l/r}}}$	Anzahl der Sensorelemente in y-Richtung der Kamerakoordinaten
$N_{b_{x_{l/r}}}$	Breite des Computerbildes in Pixeln
$N_{b_{y_{l/r}}}$	Höhe des Computerbildes in Pixeln

externe Kameraparameter:

Parameter	Beschreibung
$R_{l/r}$	Rotationsmatrizen zur Umrechnung der Weltkoordinaten in Kamerakoordinaten
$\vec{t}_{l/r}$	Translationsvektoren zur Umrechnung der Weltkoordinaten in Kamerakoordinaten

Bei der *konventionellen Kamerakalibrierung* müssen diese Parameter explizit bestimmt werden, um eine Umrechnung von Weltkoordinaten in Computerbildkoordinaten oder eine Umrechnung von Computerbildkoordinaten in Weltkoordinaten zu ermöglichen.
Bei der in dieser Arbeit verwendeten *neuronalen Kamerakalibrierung* ist die Bestimmung der internen und externen Kameraparameter nicht erforderlich.
Der Abschnitt 2.3 beschreibt die Unterschiede für den konventionellen und den neuronalen Ansatz.

2.2 Das Robotersystem

Nach der Lokalisierung und Erkennung eines Zielobjektes soll sich ein Manipulator diesem mit seinem Greifer annähern. Hierfür wird ein Roboterarm der Firma STÄUBLI verwendet, der durch sechs Freiheitsgrade seine Beweglichkeit erreicht (Abb. 2.1 (b)).
Dem Roboter liegen zwei Koordinatensysteme zugrunde, das Basiskoordinatensystem mit seinem Ursprung an einer unveränderlichen Position im Inneren des Roboters und das Werkzeugkoordinatensystem mit seinem Ursprung im Werkzeugflansch (Abb. 2.3). Jede Roboterstellung ist eindeutig beschreibbar durch die Winkelstellungen $j_1, \ldots, j_6$ seiner sechs Gelenke oder durch die Parameter x_w, y_w, z_w, *yaw*, *pitch* und *roll*, wobei x_w, y_w und z_w die kartesischen Koordinaten des Werkzeugflansches und *yaw*, *pitch* und *roll* die Orientierung (Euler-Notation der Orientierung) des Werkzeugsystems angeben. Die Koordinaten werden in Millimetern und die Orientierung in Grad beschrieben. Die Parameter *yaw*, *pitch* und *roll* bedeuten dabei drei aufeinanderfolgende Rotationen des Werkzeugkoordinatensystems um zwei seiner Achsen. Die erste erfolgt mit dem Wert *yaw* um die z-Achse, die zweite mit *pitch* um die y-Achse und die dritte mit *roll* um die neue z-Achse [Cra89].

(a) (b)

Abbildung 2.3: Das Basis- (a) und das Werkzeugkoordinatensystem (b) des Roboters.

In dieser Arbeit werden die kartesischen Koordinaten eines Objektes bezüglich des Basiskoordinatensystems ermittelt, um anschließend eine Annäherung des Greifers zum Objekt durchzuführen. Hierfür werden *yaw*, *pitch* und *roll* als konstante Größen angenommen, d.h. die Orientierung des Greifers bleibt unverändert.
Damit der Manipulator die Bewegung zu den gewünschten kartesischen Koordinaten ausführen kann, ist eine Umrechnung der Koordinaten in die Winkelstellungen notwendig. Man spricht dabei von dem Problem *Inverser Kinematik*, welches nicht unbedingt zu einer Lösung führen muß und wegen nichtlinearer Gleichungen nicht immer einfach ist [Cra89]. Die Existenz/Nicht-Existenz solcher Lösungen für die kartesischen Koordinaten eines Raumpunktes bestimmen den *Arbeitsbereich* des Manipulators. Da das Problem der Inversen Kinematik durch den STÄUBLI-Roboter gut gelöst wird, kann angenommen werden, daß sich für einen bestimmten Arbeitsbereich bei Eingabe der kartesischen Koordinaten der Manipulator dorthin bewegt.

2.3 Kamerakalibrierung

Eine *Kamerakalibrierung* wurde bisher definiert als Prozeß zur Bestimmung der in Abschnitt 2.1 beschriebenen internen und externen Parameter einer Kamera [Tsai86]. Nach der Bestimmung dieser Parameter ist es möglich, die zweidimensionalen Computerbildkoordinaten eines abgebildeten Raumpunktes aus dessen dreidimensionalen Weltkoordinaten zu berechnen.
Bei der umgekehrten Berechnung von Computerbildkoordinaten in Weltkoordinaten muß für jede Kamera ein Strahl unendlich vieler Raumpunkte bestimmt werden, auf dem der gesuchte Raumpunkt liegt. Der Schnittpunkt beider Strahlen ergibt diesen Punkt. Die Berechnung der Weltkoordinaten wird *Triangulation* genannt [Har93].
Die explizite Bestimmung der Parameter durch die konventionelle Kalibrierung kann simultan [Dan96] oder sequentiell geschehen. Bei der simultanen Bestimmung ergeben sich komplizierte Gleichungssyteme. Werden die Parameter jedoch sequentiell ermittelt, treten wegen einer Fehlerfortpflanzung leicht Ungenauigkeiten auf.

In dieser Arbeit erfolgt die Kamerakalibrierung neuronal und beschreibt den Prozeß der Konfigurierung eines RBF-Netzes ohne Bestimmung der Position, Orientierung und optischen Eigenschaften der Kameras, also der internen und externen Parameter. Diese spielen nur implizit eine Rolle. Für diese Art der Kalibrierung wird lediglich eine Menge von Eingabe-/Ausgabedaten benötigt, damit das Netz nach dem Training zuverlässige Ausgaben liefert. Die Zuverlässigkeit kann durch die Anzahl dieser Daten beliebig reguliert werden und drückt sich dadurch aus, wie genau die gewünschte Ausgabe berechnet wird. Bei der Stereokamera-Manipulator-Kalibrierung sind die Eingabedaten vierdimensionale Abbildungs-

vektoren und die Ausgabedaten die Weltkoordinaten der zugehörigen Raumpunkte. Die Abweichung einer RBF-Ausgabe zur gewünschten Ausgabe kann hier als Euklidischer Abstand des berechneten Raumpunktes zum gewünschten definiert werden. Diese Abweichung wird dann in mm angegeben. Bei der Manipulator-Einzelkamera-Kalibrierung sind die Eingabedaten Weltkoordinaten von Raumpunkten und die Ausgabedaten die zugehörigen Computerbildkoordinaten. Hier wird die Abweichung als Abstand der berechneten Position im Computerbild zur gewünschten in Pixeln angegeben.
Die neuronale Kalibrierung der Kameras bietet gegenüber der konventionellen Kalibrierung den Vorteil, daß durch die Anzahl der Eingabe-/Ausgabedaten die Zuverlässigkeit beliebig erhöht werden kann. Außerdem ergeben sich weder Ungenauigkeiten noch komplizierte Berechnungen, da die explizite Bestimmung der Kameraparameter nicht benötigt wird.

Für die Alternative, bei der das Kamerasystem am Ende des Manipulators befestigt wird, ergibt sich ein deutlicher Nachteil dadurch, daß *eine* Kalibrierung nicht ausreicht. Es müssen hier mehrere Kalibrierungen erfolgen, nämlich eine Manipulator-, eine Kamera-Manipulator- und eine Kamerakalibrierung [Tsai89].

In dem folgenden Kapitel 3 erfolgt eine allgemeine Beschreibung von RBF-Netzen, die für die neuronale Kamerakalibrierung verwendet werden. In Kapitel 4 werden unterschiedlich konfigurierte RBF-Netze auf ihre Zuverlässigkeit bei ihrer Verwendung zur Kamerakalibrierung untersucht.

Kapitel 3

RBF- und HBF-Netze

In diesem Kapitel werden der Aufbau und der Vorgang des Trainierens sowie die Anwendung von Hyperbasisfunktionen-Netzen (HBF-Netze) und Netzen Radialer Basisfunktionen (RBF-Netze), einer speziellen Form der HBF-Netze, beschrieben. Diese Netze werden für die Robotersteuerung sowie die Objektlokalisierung und -erkennung in dieser Arbeit benötigt.
Ein wichtiger Aufgabenbereich von RBF- und HBF-Netzen ist die Funktionsapproximation [Pow87]. In dieser Arbeit werden die Netze zur Approximation multidimensionaler Funktionen der folgenden Form verwendet:

$$f : \mathbb{R}^m \longrightarrow \mathbb{R}^n, \quad m, n \in \mathbb{N}. \tag{3.1}$$

3.1 Regularisierung

Sei f eine Funktion wie in (3.1) und $\tilde{f}$ eine Funktion, die f approximieren soll. Sind nun von f einige Eingabe-/Ausgabewerte $(\vec{x}_1, f(\vec{x}_1))$,...,$(\vec{x}_M, f(\vec{x}_M))$ bekannt, nennt man die Balance zwischen der *Korrektheit* und der *Glattheit* von $\tilde{f}$ *Regularisierung* [Pog90]. Dabei bedeutet Korrektheit, daß die Funktionswerte von $\tilde{f}$ und f an den Stützstellen möglichst genau übereinstimmen. Die Glattheit von $\tilde{f}$ bedeutet, daß je zwei ähnliche Eingaben $\vec{x}_a$ und $\vec{x}_b$ ähnliche Ausgaben $\tilde{f}(\vec{x}_a)$ und $\tilde{f}(\vec{x}_b)$ implizieren. Das Regularisierungsproblem kann gelöst werden durch die Minimierung des folgenden Funktionals:

$$H\left[\tilde{f}\right] = \sum_{i=1}^{M} \left[\tilde{f}(\vec{x}_i) - f(\vec{x}_i)\right]^2 + \lambda\Omega\left[\tilde{f}\right], \tag{3.2}$$

wobei das Funktional Ω den Grad der Glattheit darstellt und $\lambda \in \mathbb{R}_{>0}$ als *Regularisierungsparameter* die Gewichtung zwischen Glattheit und Korrektheit steuert.

Ist über die Form der Funktion f kein oder nur wenig Wissen a priori bekannt, spricht man von einer *nichtparametrischen Approximation* [Orr96]. In dieser Arbeit sollen alternativ dazu Funktionen approximiert werden, deren mathematische Form zwar bekannt ist (s. (2.1), (2.2), (2.3) und (2.4)), deren Parameter also implizit eine Rolle spielen, die jedoch nicht explizit bestimmt werden sollen (neuronale Kamerakalibrierung). Es sollen diese Funktionen so approximiert werden, daß eine Minimierung des Funktionals (3.2) erfolgt.

3.2 Netze Radialer Basisfunktionen

Das RBF-Netz ist ein zweistufiges Netz, bestehend aus einer verdeckten Schicht von N Knoten mit nichtlinearen Basisfunktionen r_j, $j = 1, \ldots, N$, und einer Ausgabeschicht von n Knoten, die eine gewichtete Linearkombination der Basisfunktionen bilden [Hus93]. Jede Basisfunktion ist vergleichbar mit einem *rezeptiven Feld*, das die höchste *Aktivierung* für Eingaben erzeugt, die in der Nähe des Zentrums des Feldes liegen.
Das Trainieren des Netzes besteht in der 1. Phase aus der Bestimmung der Basisfunktionen und in der 2. Phase aus dem Lernen von Gewichten, die für die Linearkombination benötigt werden. Das Lernen der 1. Schicht erfolgt unüberwacht, meistens durch einen Cluster-Algorithmus, während in der 2. Schicht überwachtes Lernen stattfindet.
Da ein RBF-Netz verwendet wird, um eine Funktion zu approximieren, muß das Funktional (3.2) aus Abschnitt 3.1 minimiert werden, um ein ausgewogenes Verhältnis zwischen Glattheit und Korrektheit der durch das Netz berechneten Funktion zu erhalten. Während die Gewichtung zwischen Glattheit und Korrektheit in Abschnitt 3.1 durch den Parameter λ gesteuert werden konnte, besteht beim RBF-Netz diese Möglichkeit durch dessen Konfigurierung, u.a. durch die Anzahl der Basisfunktionen.
Die folgenden Abschnitte erläutern die Möglichkeiten der Konfigurierung von RBF-Netzen. Zur besseren Veranschaulichung wird zunächst $m = n = 1$ gewählt (Abb. 3.1). Darauf folgt eine Betrachtung allgemeinerer Netze, die multidimensionale Eingaben und Ausgaben erlauben.

3.2.1 Eindimensionale Funktionen durch RBF-Netze

Sei folgende Funktion f gegeben, die durch ein RBF-Netz approximiert werden soll:

$$f : \mathbb{R} \longrightarrow \mathbb{R}, \quad x \mapsto y$$

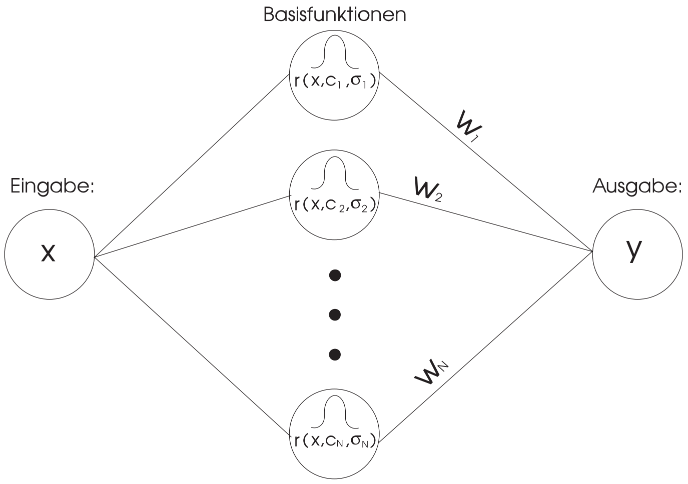


Abbildung 3.1: Schematischer Aufbau des RBF-Netzes für eindimensionale Funktionen.

Phase 1 (Bestimmung der Basisfunktionen)

Um f mit einer Funktion $\tilde{f}$ zu approximieren, die durch ein RBF-Netz berechnet wird, muß eine *Trainingsmenge* $\mathcal{T}$ von M Eingabe-/Ausgabedaten (x_i, y_i), $i = 1, \ldots, M$, vorgegeben werden. Das Netz sollte nach dem Trainieren in der Lage sein, zu jeder Eingabe x_i die zugehörige Ausgabe y_i mit geringem Fehler zu berechnen und bei unbekannten Eingaben x eine interpolierte Ausgabe y zu liefern. Erwartet wird also die Minimierung des Funktionals in (3.2).

In dieser Arbeit werden gaußförmige Basisfunktionen[1] verwendet, die eine *unimodale* Struktur besitzen, d.h. sie liefern im Zentrum den Wert 1 und mit zunehmendem Abstand vom Zentrum geringere Werte. Dadurch werden Eingaben, die einer Eingabe der Trainingsmenge ähnlich sind, durch die Basisfunktionen auf ähnliche Funktionswerte abgebildet (Korrektheit von $\tilde{f}$). Die Glattheit von $\tilde{f}$ wird hierbei durch eine Überlagerung der Basisfunktionen gesteuert. Dadurch können auch für Eingaben, die zwischen den Eingaben der Trainingsmenge liegen, die gewünschten Ausgaben interpoliert werden.

Die gaußförmigen Basisfunktionen sind wie folgt definiert:

$$r_j = r\left(x, c_j, \sigma_j\right) = \exp\left[-\frac{\left(x - c_j\right)^2}{2\sigma_j^2}\right], \quad j = 1, 2, \ldots, N \tag{3.3}$$

Dabei stellen die c_j die *Zentren* der Basisfunktionen dar und können z.B. durch eine Clusterung der Eingabedaten der Trainingsmenge nach Ähnlichkeit und anschließender Mittelwertberechnung bestimmt werden. Auf diese Weise erzielt man

[1] In [Pog90] wurde bewiesen, daß mit sogenannten *Green'schen Funktionen* als Basisfunktionen das Funktional in (3.2) minimiert wird.

eine Generalisierung der Eingabedaten. Eine Generalisierung des Eingaberaumes ist wichtig, weil bei der Verwendung vieler *Trainingsdaten* Redundanzen entstehen, die bei einer umfassenden Generalisierung entfallen. Werden jedoch zu wenig Cluster bestimmt, gehen wichtige Informationen der einzelnen Trainingsdaten verloren und es erfolgt eine *Übergeneralisierung.* Bei einem effizienten Netz muß ein ausgeglichenes Verhältnis zwischen Redundanz und Generalisierung existieren. Die Optimierung des Verhältnisses von Redundanz und Generalisierung eines effizienten Netzes ist vergleichbar mit der Optimierung des Verhältnisses zwischen der Korrektheit und der Glattheit einer Funktion durch die Minimierung von (3.2).
Die Anzahl der Basisfunktionen und die Werte der *Normierungsparameter* σ_j^2 bestimmen die Überlagerung der Basisfunktionen und damit die Glattheit der Approximierungsfunktion $\tilde{f}$. Die Steuerung der Glattheit bei der Regularisierung mit den Parametern λ und Ω (s. (3.2)) erfolgt also bei RBF-Netzen mit σ^2 und N.
Bei äquidistanten Eingabedaten der Trainingsmenge empfiehlt es sich, äquidistante Zentren mit gleichem Normierungsparameter $\sigma^2 = \sigma_1^2, \ldots, \sigma_N^2$ zu wählen. Für ungleichmäßig verteilte Daten eignet sich ein Cluster-Algorithmus, welcher die Eingabedaten nach Ähnlichkeit zu N Clustern gruppiert. Es ergeben sich damit für die Cluster Mittelwerte und Standardabweichungen. Die Quadrate der Standardabweichungen können anschließend als individuelle Normierungsparameter $\sigma_1^2, \ldots, \sigma_N^2$ gewählt werden. Zwei unterschiedliche Cluster-Algorithmen und die Wirkung der Normierungsparameter werden in Abschnitt 3.2.2 beschrieben.
Ein Knoten j der verdeckten Schicht erzeugt auf jede Eingabe x eine lokale Antwort r_j, die im Bereich $]0, 1]$ liegt und sich nur dann signifikant auf die Ausgabe auswirkt, wenn x nahe genug am Zentrum dieses Knotens liegt. Diese Antwort wird auch Aktivierung genannt.
Der Name *Radiale Basisfunktionen (RBF)* rührt daher, daß die Basisfunktionen radial symmetrisch sind, d.h. für alle x, die in gleicher radialer Distanz zum Zentrum des j-ten Knotens der verdeckten Schicht liegen, erzeugt dieser Knoten die gleiche Antwort r_j. Die Ausgabe y ergibt sich dann wie folgt:

$$y = \tilde{f}(x) = (r_1 w_1 + \ldots + r_N w_N) \approx f(x) \tag{3.4}$$

Phase 2 (Bestimmung der Gewichte der 2. Schicht)

Sei $\mathcal{T} = \{(x_1, y_1), \ldots, (x_M, y_M)\}$ die Trainingsmenge und $r_{i_j} = r(x_i, c_j, \sigma_j)$ die Antwort der Basisfunktion des j-ten Knotens der verdeckten Schicht auf x_i. Seien weiterhin der *Gewichtsvektor* $\vec{w}$, die *Aktivierungsmatrix* A und der Vektor $\vec{t}$ wie folgt definiert:

$$\vec{w} = \begin{pmatrix} w_1 \\ \vdots \\ w_N \end{pmatrix}, \quad A = \begin{pmatrix} r_{1_1} & \cdots & r_{1_N} \\ \vdots & & \vdots \\ r_{M_1} & \cdots & r_{M_N} \end{pmatrix}, \quad \vec{t} = \begin{pmatrix} y_1 \\ \vdots \\ y_M \end{pmatrix}$$

Die Interpolationsbedingung verlangt, daß bei Einsetzen der Eingabewerte der Trainingsmenge in (3.4) die entsprechenden Ausgabewerte berechnet werden, d.h. $\tilde{f}(x_i) = y_i$, $i = 1, \ldots, M$, und kann somit in folgender Matrizenform dargestellt werden:

$$\vec{t} = A\vec{w} \tag{3.5}$$

Mit der Existenz einer Inversen A^{-1} der Aktivierungsmatrix können die Gewichte des Netzes bestimmt werden:

$$\vec{w} = A^{-1}\vec{t}$$

Ist das Gleichungssystem (3.5) jedoch nicht eindeutig lösbar, kann durch die sogenannte *Pseudoinverse* [Orr96] der Aktivierungsmatrix eine optimale Lösung ermittelt werden. Zwei Verfahren zur Bestimmung der Gewichte der 2. Schicht des Netzes werden in Abschnitt 3.2.2 vorgestellt.

3.2.2 Multidimensionale Funktionen durch RBF-Netze

Sei eine multidimensionale Funktion f gemäß (3.1) gegeben, die durch $\tilde{f}$ mit einem RBF-Netz (Abb. 3.2) approximiert werden soll.

Phase 1 (Bestimmung der Basisfunktionen)

Wie in Abschnitt 3.2.1 muß eine Trainingsmenge $\mathcal{T}$ vorgegeben werden, wobei nun jedoch die Eingabedaten von $\mathcal{T}$ m-dimensionale und die Ausgabedaten n-dimensionale Vektoren sind. Sei also $\mathcal{T} = \{(\vec{x}_1, \vec{y}_1), \ldots, (\vec{x}_M, \vec{y}_M)\}$. Allgemein gilt nun für die gaußförmigen Basisfunktionen:

$$r_j = r(\vec{x}, \vec{c}_j, \sigma_j) = \Phi(\|\vec{x} - \vec{c}_j\|, \sigma_j) \tag{3.6}$$

Dabei wird Φ folgendermaßen definiert:

$$\Phi(d, \sigma) = \exp\left(-\frac{d^2}{2\sigma^2}\right) \tag{3.7}$$

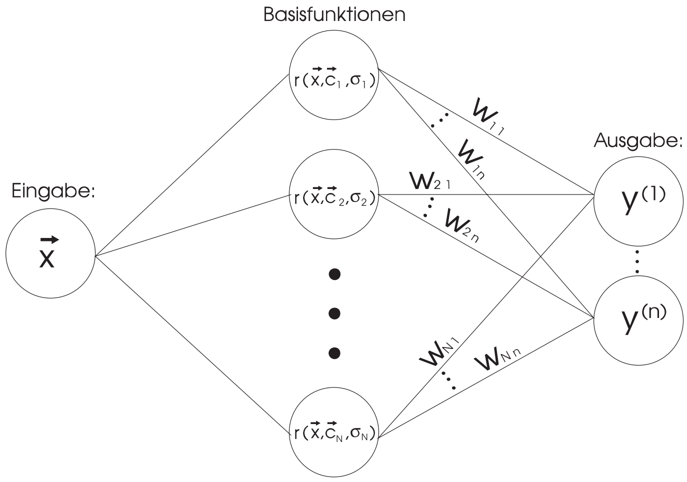


Abbildung 3.2: Schematischer Aufbau des RBF-Netzes für multidimensionale Funktionen.

Um den Abstand zwischen den Vektoren $\vec{x}$ und $\vec{c}_j$ zu ermitteln, kann eine beliebige Norm $\|\vec{x} - \vec{c}_j\|$ verwendet werden. Häufig benutzt man die *Euklidische Norm*, also $\|\vec{x} - \vec{c}_j\| = \sqrt{(\vec{x} - \vec{c}_j)^T (\vec{x} - \vec{c}_j)}$.
Die Zentren $\vec{c}_j$ der Basisfunktionen besitzen dieselbe Dimension wie die Eingabevektoren $\vec{x}_i$ und können bei ungleichmäßig verteilten Eingaben durch einen Cluster-Algorithmus bestimmt werden. Häufig wird hierfür folgender *K-MEANS-Algorithmus* verwendet:

procedure K_MEANS
  Initialisiere die Zentren $\vec{c}_j$, $j = 1, \ldots, N$;
  /* die Zentren können mit Zufallswerten aus dem Eingaberaum oder durch die ersten N Eingabevektoren belegt werden */
  repeat
    Initialisiere Indexmengen $C_j = \{\}$, $j = 1, \ldots, N$, für die Cluster ;
    /* Ordne jedem Eingabevektor das Cluster zu, dessen Zentrum ihm am nähesten ist */
    for all $\vec{x}_i$, $i = 1, \ldots, M$, **do**
      $j' := \arg_j \left\{ \min_{j \in \{1,\ldots,N\}} \{\|\vec{x}_i - \vec{c}_j\|\} \right\}$;
      $C_{j'} := C_{j'} \cup \{i\}$;
    end ; /* rof */
    /* Berechnung der Mittelwerte */
    for all j, $j = 1, \ldots, N$, **do**
      $\vec{c}_{j_{old}} := \vec{c}_j$;
      $\vec{c}_j := \frac{1}{|C_j|} \sum_{i \in C_j} \vec{x}_i$;
      $var_j := \frac{1}{|C_j|} \sum_{i \in C_j} (\vec{x}_i - \vec{c}_j)^T (\vec{x}_i - \vec{c}_j)$;
    end ; /* rof */

until $(\vec{c}_j = \vec{c}_{j_{old}})$, $j = 1, \ldots, N$;
/* von einer Iteration zur nächsten findet in den Zentren keine Änderung mehr statt */
end ; /* K_MEANS */

Bei diesem Algorithmus sollen die Eingabevektoren $\vec{x}_i$ nach Ähnlichkeit zu N Clustern gruppiert werden, wobei N vorzugeben ist. Ein *Clusterzentrum* sei definiert durch den Mittelwert der Vektoren in diesem Cluster. Zunächst werden die Zentren mit Zufallswerten aus dem Eingaberaum oder durch die ersten N Eingabevektoren belegt. Es folgt nun in einer Iteration des Algorithmus' die Zuordnung jedes Eingabevektors zu dem Cluster, dessen Zentrum ihm am ähnlichsten ist und anschließend die Bestimmung der neuen Zentren. Die Ähnlichkeit von Vektoren kann durch eine beliebige Abstandsnorm definiert werden. Der Algorithmus terminiert, wenn von einer Iteration zur nächsten keine Änderung in den Clusterzentren stattgefunden hat.
Der Algorithmus liefert weiterhin die Ausgabe var_j als Varianz in einem Cluster, also die quadratische mittlere Abweichung der Vektoren in diesem Cluster von seinem Zentrum, wobei als Distanzmaß die Euklidische Norm gewählt wird. Die Elemente der *Indexmengen* C_j, $j = 1, \ldots, N$, geben die Indizes $i \in \{1, \ldots, M\}$ der Eingabevektoren an, die den Clustern zugeordnet werden.
Alternativ ergibt sich der *ISODATA-Algorithmus* aus dem K-MEANS-Algorithmus durch Hinzufügen von zwei weiteren Funktionen. Es werden hier nach dem Ermitteln der Zentren durch K-MEANS diejenigen Cluster, die eine große Varianz besitzen, in zwei Cluster geteilt und nahe beieinander liegende Cluster zu einem vereinigt. Die Aufrufe K-MEANS, Teilung und Vereinigung werden solange wiederholt, bis weder Teilung noch Vereinigung von Clustern stattfindet. Dieser Algorithmus bietet den Vorteil, daß keine Angabe der Anzahl an Clusterzentren nötig ist, sondern lediglich die Angabe von Parametern, die beschreiben, wie hoch die Varianz in Clustern sein muß, damit Teilung oder Vereinigung stattfindet. Weiterhin wird eine maximal zulässige Varianz angegeben. Somit terminiert der Algorithmus mit einer optimalen Lösung durch möglichst wenig Cluster und eine geringe Varianz in den Clustern.
Die beiden beschriebenen Cluster-Algorithmen sind in dem Bildverarbeitungspaket KHOROS implementiert und werden in dieser Arbeit benutzt.

Die Form der Basisfunktionen eines RBF-Netzes wird bestimmt durch eine nichtlineare Funktion Φ. Bei dieser Funktion muß die Bedingung $\lim_{|x|\to\infty} \Phi(x) = 0$ gelten, damit die Aktivierung eines Knotens j bei zunehmendem Abstand der Eingabe $\vec{x}$ vom Zentrum $\vec{c}_j$ der Basisfunktion des Knotens abnimmt. Am häufigsten wird hierfür die Gaußfunktion aus (3.7) verwendet. Da σ den Wendepunkt der Gaußkurve angibt, bestimmt der Normierungsparameter σ_j^2 damit in (3.6) den *Signifikanzbereich* als Teil des Definitionsbereiches der Gaußfunktion um das Zentrum $\vec{c}_j$, der eine signifikante Aktivierung erzeugt. Dieser Parameter kann z.B. mit der Varianz des Clusters gleichgesetzt werden, d.h. $\sigma_j^2 = var_j$,

$j = 1, \ldots, N$. Bei äquidistanten Eingabevektoren der Trainingsmenge würde eine sinnvolle Clusterung äquidistante Clusterzentren mit geringen Varianzen ergeben, was dann zur Gleichheit der Normierungsparameter $\sigma_1^2, \ldots, \sigma_N^2$ führen würde. Im folgenden wird deshalb bei äquidistanten Eingabevektoren nur ein Normierungsparameter σ^2 für alle Clusterzentren verwendet.

Der Spezialfall $\sigma_j = 0$ führt zu Problemen bei der Berechnung von Φ. Um dies zu vermeiden, kann festgelegt werden, daß jedes Cluster von mindestens zwei Eingabevektoren belegt wird und keine Eingabevektoren doppelt auftreten.
Durch die Verwendung der Euklidischen Norm und der Gaußfunktion ergibt sich schließlich für die Aktivierung r_j des j-ten Knotens auf die Eingabe $\vec{x}$ (vgl. mit (3.3) für den eindimensionalen Fall):

$$r_j = r(\vec{x}, \vec{c}_j, \sigma_j) = \exp\left[-\frac{(\vec{x} - \vec{c}_j)^T (\vec{x} - \vec{c}_j)}{2\sigma_j^2}\right], \quad j = 1, 2, \ldots, N \tag{3.8}$$

Die Ausgabe $\vec{y}$ ergibt sich aus folgendem Gleichungssystem:

$$\vec{y} = \tilde{f}(\vec{x}) = \begin{pmatrix} r_1 w_{1_1} + & \ldots & + r_N w_{N_1} \\ \vdots & & \vdots \\ r_1 w_{1_n} + & \ldots & + r_N w_{N_n} \end{pmatrix} \approx f(\vec{x}) \tag{3.9}$$

Eine andere Möglichkeit der Berechnung von $\vec{y}$ erfolgt bei *normierten RBF-Netzen*, indem der Wert jeder Komponente durch die Summe der Aktivierungen aller Knoten dividiert wird:

$$\vec{y} = \tilde{f}(\vec{x}) = \begin{pmatrix} \frac{r_1 w_{1_1} + \ldots + r_N w_{N_1}}{r_1 + \ldots + r_N} \\ \vdots \\ \frac{r_1 w_{1_n} + \ldots + r_N w_{N_n}}{r_1 + \ldots + r_N} \end{pmatrix} \approx f(\vec{x}) \tag{3.10}$$

In Kapitel 4 wird durch einige Versuche gezeigt, daß sich die Normierungsparameter $\sigma_1^2, \ldots, \sigma_N^2$ signifikant auf die Güte einer durch ein RBF-Netz realisierten Approximation auswirken. Es wird weiterhin analysiert, ob durch die Verwendung von normierten RBF-Netzen geringere Approximationsfehler erreicht werden können und ob sich auch bei diesen Netzen der Normierungsparameter auf den Approximationsfehler auswirkt.

Phase 2 (Bestimmung der Gewichte der 2. Schicht)

Sei $r_{i_j} = r(\vec{x}_i, \vec{c}_j, \sigma_j)$ die Antwort der Basisfunktion des j-ten Knotens der verdeckten Schicht auf $\vec{x}_i$ und $y_i^{(k)}$, $k = 1, \ldots, n$, $i = 1, \ldots, M$, der Wert des Ausgabevektors $\vec{y}_i$ in der k-ten Komponente. Weiterhin seien die Gewichtsvektoren $\vec{w}_1, \ldots, \vec{w}_n$, die Aktivierungsmatrix A und die Vektoren $\vec{t}_1, \ldots, \vec{t}_n$ wie folgt definiert:

$$\vec{w}_k = \begin{pmatrix} w_{1_k} \\ \vdots \\ w_{N_k} \end{pmatrix}, \; A = \begin{pmatrix} r_{1_1} & \cdots & r_{1_N} \\ \vdots & & \vdots \\ r_{M_1} & \cdots & r_{M_N} \end{pmatrix}, \; \vec{t}_k = \begin{pmatrix} y_1^{(k)} \\ \vdots \\ y_M^{(k)} \end{pmatrix}, \; k = 1, \ldots, n$$

Dann kann die Interpolationsbedingung, die verlangt, daß bei Einsetzen der Eingabewerte der Trainingsmenge in (3.9) die entsprechenden Ausgabewerte berechnet werden, d.h. $\tilde{f}(\vec{x}_i) = \vec{y}_i$, $i = 1, \ldots, M$, in folgender Martrizenform dargestellt werden:

$$\vec{t}_k = A\vec{w}_k, \quad k = 1, \ldots, n \tag{3.11}$$

Für ein normiertes RBF-Netz ist jedoch die Aktivierungsmatrix A wie folgt zu wählen:

$$A = \begin{pmatrix} \frac{r_{1_1}}{r_{1_1}+\ldots+r_{1_N}} & \cdots & \frac{r_{1_N}}{r_{1_1}+\ldots+r_{1_N}} \\ \vdots & & \vdots \\ \frac{r_{M_1}}{r_{M_1}+\ldots+r_{M_N}} & \cdots & \frac{r_{M_N}}{r_{M_1}+\ldots+r_{M_N}} \end{pmatrix}$$

Die Gewichte w_{j_k}, $j = 1, \ldots, N$, $k = 1, \ldots, n$, können mit verschiedenen Verfahren ermittelt werden:

Lineare Ausgleichsrechnung

Mit der Existenz einer Inversen A^{-1} der Aktivierungsmatrix können die Gewichtsvektoren des Netzes wie folgt bestimmt werden:

$$\vec{w}_k = A^{-1}\vec{t}_k, \quad k = 1, \ldots, n$$

Im allgemeinen existiert diese Inverse jedoch nicht. In diesem Fall kann aber eine Lösung gefunden werden, die die Interpolationsbedingung (3.11) mit geringst möglichem Fehler erfüllt. Definiert man $\tilde{f}^{(k)}(\vec{x}_i)$, $k = 1, \ldots, n$, $i = 1, \ldots, M$, als

k-te Komponente der Netz-Ausgabe auf den Eingabevektor $\vec{x}_i$ und Kostenfunktionen C_k als Summe der quadratischen Fehler für jede Komponente wie folgt:

$$\tilde{f}\left(\vec{x}_i\right) = \begin{pmatrix} \tilde{f}^{(1)}(\vec{x}_i) \\ \vdots \\ \tilde{f}^{(n)}(\vec{x}_i) \end{pmatrix} = A\vec{w}_k, \quad i = 1, \ldots, M$$

$$C_k = \sum_{i=1}^{M} \left[y_i^{(k)} - \tilde{f}^{(k)}(\vec{x}_i)\right]^2, \quad i = 1, \ldots, M \tag{3.12}$$

Dann ist die Minimierung des Interpolationsfehlers gleichwertig mit der Minimierung der Kostenfunktionen, also der folgenden Funktionale:

$$\|\vec{t}_k - A\vec{w}_k\| = \sqrt{\left(\vec{t}_k - A\vec{w}_k\right)^T \left(\vec{t}_k - A\vec{w}_k\right)}, \quad k = 1, \ldots, n \tag{3.13}$$

Die Gewichtsvektoren $\vec{w}_k, \quad k = 1, \ldots, n$, werden folgendermaßen bestimmt:

$$\vec{w}_k = \arg_{\vec{w}} \left\{ \min_{\vec{w} \in \mathbb{R}^N} \left\{ \|\vec{t}_k - A\vec{w}\| \right\} \right\}, \quad k = 1, \ldots, n \tag{3.14}$$

In (3.14) wird ein sogenanntes *Lineares Ausgleichsproblem* beschrieben, das für $M \geq N$ durch die Einführung einer Pseudoinversen $\left(A^T A\right)^{-1} A^T$ der Aktivierungsmatrix [Orr96] lösbar ist. Durch Nullsetzen der partiellen Ableitungen der Funktionale aus (3.13) ergeben sich die optimalen Gewichte durch folgende Gleichungen:

$$\vec{w}_k = \left(A^T A\right)^{-1} A^T \vec{t}_k, \quad k = 1, \ldots, n$$

Für den Fall $M \geq N$ existieren mindestens soviele Trainingsdaten wie Basisfunktionen. Während das Gleichungssystem in (3.11) für diesen Fall überbestimmt ist und somit eine eindeutige optimale Lösung bei der Bestimmung der Gewichte gesucht wird, gibt es für $M < N$ wegen der Unterbestimmung unendlich viele Lösungen, bei denen die Funktionale aus (3.13) Null werden. In diesem Fall wählt man als Pseudoinverse $A^T \left(AA^T\right)^{-1}$ und als Lösung diejenige, für die die Norm der Gewichtsvektoren $\|\vec{w}_k\|$ minimal wird [Iva92]. In den folgenden Kapiteln ergibt sich jedoch durch die Wahl einer großen Trainingsmenge immer der Fall $M \geq N$.

Least-Mean-Square-Algorithmus (LMS)

Eine Alternative zur Bestimmung der optimalen Gewichte mit Linearer Ausgleichsrechnung bietet der durch einen *Lernparameter* μ steuerbare neuronale LMS-Algorithmus. Vorteilhaft ist bei diesem Algorithmus, daß er auch bei singulären Aktivierungsmatrizen und Pseudoinversen die Gewichte u.U. berechnen kann. Außerdem eignet er sich zum *Iterativen Lernen*, indem er auch im 'Online'-Betrieb die Gewichte jederzeit aufgrund neuer Trainingsdaten adaptieren kann.
Der LMS-Algorithmus wird wie folgt beschrieben:
Für jeden Eingabevektor $\vec{x}_i$, $i = 1, \ldots, M$, wird der interpolierte Ausgabevektor $\tilde{f}(\vec{x}_i)$ des Netzes berechnet. Für alle Komponenten der n-dimensionalen Ausgabe werden die Fehler berechnet, die zusammen mit dem Lernparameter μ die Veränderung in den Gewichtsvektoren $\vec{w}_1, \ldots, \vec{w}_n$ bestimmen. Dabei gibt $\mu \in [0, 1]$ die Präzision des Lernens an. Um ein divergierendes Oszillieren der Gewichte zu verhindern, darf μ nicht zu groß gewählt werden. Ein geringer Lernparameter bewirkt jedoch eine langsame Fehlerreduktion. Bei einer Erhöhung von μ wird also das Lernen unter Vernachlässigung der Präzision beschleunigt.
Die Möglichkeit einer Gewichtsjustierung besteht darin, einen zusätzlichen Lernparameter zu verwenden, der die Gewichte zu Beginn des Lernens stark verändert und die Veränderung der Gewichte im weiteren Verlauf verringert. Dadurch findet nach und nach eine feinere Justierung statt.

Vor- und Nachteile von Linearer Ausgleichsrechnung und LMS

Ein Vorteil des LMS-Algorithmus' liegt darin, daß er durch verschiedene Parameter steuerbar und im 'Online'-Betrieb einsetzbar ist. Das bedeutet, daß er für diese Arbeit vorteilhaft wäre, wenn während des 'Online'-Betriebes stets Trainingsdaten neu erzeugt oder verändert würden, wodurch eine Adaption der Gewichte erforderlich wäre (iteratives Lernen). Das Lernen von Erkennungsfunktionen für die Zielobjekte sowie die neuronale Kamerakalibrierung erfolgen jedoch 'offline' und sind somit im 'Online'-Betrieb bereits abgeschlossen.
Ein weiterer Vorteil des LMS-Algorithmus' ist seine Unabhängigkeit von der Invertierbarkeit der Aktivierungsmatrix und ihrer Pseudoinversen. Probleme ergeben sich allerdings bei höherdimensionalen Eingaben, da in diesem Fall die Gewichte erst nach vielen Lernschritten oder gar nicht gegen einen Grenzwert konvergieren.
Auch die Berechnung der Pseudoinversen der Aktivierungsmatrix bei der Linearen Ausgleichsrechnung ist aufwendig für höherdimensionale Eingabevektoren. Jedoch spielt für diese Arbeit der Zeitfaktor eine geringe Rolle, da weder das Lernen der Objekterkennungsfunktion noch die neuronale Kamerakalibrierung im 'Online'-Betrieb stattfinden.

Insgesamt ist für diese Arbeit eine Lineare Ausgleichsrechnung bei der Bestimmung der Gewichte geeigneter als der LMS-Algorithmus.

3.3 Hyperbasisfunktionen-Netze

Im Gegensatz zu den RBF-Netzen sollen bei den HBF-Netzen für verschiedene Dimensionen unterschiedliche Signifikanzbereiche gewählt werden. Dadurch verlieren die Basisfunktionen ihre Eigenschaft der Radialsymmetrie, d.h. es können sich für Eingabevektoren, die in gleicher radialer Distanz zum Zentrum einer Basisfunktion liegen, unterschiedliche Aktivierungen für diesen Knoten ergeben. Ursache dafür ist, daß es zu jeder Basisfunktion r_j, $j = 1, \ldots, N$, einen *Normierungsvektor* $\vec{\sigma}_j$ gibt, der die Dimension des Eingabevektors $\vec{x}$ und des Zentrenvektors $\vec{c}_j$ besitzt, während bei den RBF-Netzen jeweils nur ein Normierungsparameter σ_j existiert. Die weiteren Eigenschaften von HBF-Netzen sind mit denen der RBF-Netze identisch. Die Anzahl der Basisfunktionen sowie die Normierungsvektoren bestimmen auch hier die Gewichtung zwischen der Glattheit und der Korrektheit der durch das Netz berechneten Funktion.
Die Basisfunktionen für HBF-Netze werden *Hyperbasisfunktionen* [Pog90] genannt. Seien $\vec{x}$, $\vec{c}_j$ und $\vec{\sigma}_j$ wie folgt definiert:

$$\vec{x} = \begin{pmatrix} x^{(1)} \\ \vdots \\ x^{(m)} \end{pmatrix}, \quad \vec{c}_j = \begin{pmatrix} c_j^{(1)} \\ \vdots \\ c_j^{(m)} \end{pmatrix}, \quad \vec{\sigma}_j = \begin{pmatrix} \left(\sigma_j^{(1)}\right)^2 \\ \vdots \\ \left(\sigma_j^{(m)}\right)^2 \end{pmatrix}$$

Dann ergibt sich die Aktivierung r_j des j-ten Knotens, $j = 1, \ldots, N$, auf die Eingabe $\vec{x}$ im Unterschied zu (3.8) durch folgende Hyperbasisfunktion:

$$r_j = r\left(\vec{x}, \vec{c}_j, \vec{\sigma}_j\right) = \exp\left[-\left[\frac{\left(x^{(1)} - c_j^{(1)}\right)^2}{2\left(\sigma_j^{(1)}\right)^2} + \cdots + \frac{\left(x^{(m)} - c_j^{(m)}\right)^2}{2\left(\sigma_j^{(m)}\right)^2}\right]\right] \tag{3.15}$$

Ein RBF-Netz ist also ein spezielles HBF-Netz, für das alle Komponenten eines Normierungsvektors gleich sind, d.h. es gilt $\left(\sigma_j^{(1)}\right)^2 = \ldots = \left(\sigma_j^{(m)}\right)^2$ für $j = 1, \ldots, N$. Diese für eine Basisfunktion identischen Komponenten werden beim RBF-Netz Normierungsparameter σ_j^2, $j = 1, \ldots, N$, genannt.

Eine Alternative ergibt sich durch ein normiertes HBF-Netz, dessen Eigenschaften wie beim normierten RBF-Netz definiert werden (s. Abschnitt 3.2.2). In Kapitel 5 wird analysiert, ob die Verwendung eines normierten HBF-Netzes für die Objekterkennung sinnvoll ist.

3.4 Diskussion von RBF- und HBF-Netzen

Eine wichtige Fragestellung ergibt sich durch die Überlegung, ob ein RBF- oder ein HBF-Netz zur Approximation einer Funktion geeigneter ist. Bei Funktionen, für die eine Trainingsmenge mit hochdimensionalen oder nicht äquidistant verteilten Eingabedaten bekannt ist, entstehen durch Clusterung dieser Daten individuelle Varianzen in den einzelnen Komponenten der Cluster. Da diese Varianzen signifikant voneinander abweichen können, ist hier prinzipiell zur Benutzung eines HBF-Netzes zu raten. Diese Netze eignen sich z.B. bei der Stereokamera-Manipulator-Kalibrierung (nicht äquidistant verteilte Eingabedaten) in Kapitel 4, Abschnitt 4.2, und beim Lernen einer Objekterkennungsfunktion (hochdimensionale Eingabedaten) in Kapitel 5.
Bei der Manipulator-Einzelkamera-Kalibrierung (s. Kapitel 4, Abschnitt 4.1) ergeben sich durch Clusterung der Eingabedaten in den Clustern gleiche Varianzen. Deshalb werden hierfür RBF-Netze verwendet.

Kapitel 4

Neuronale Kamerakalibrierung

In diesem Kapitel wird die neuronale Kamerakalibrierung mit einem RBF-Netz als Hauptbestandteil der Robotersteuerung beschrieben und unterschiedlich konfigurierte RBF-Netze auf ihre Robustheit untersucht. Die Manipulator-Einzelkamera-Kalibrierung und die Stereokamera-Manipulator-Kalibrierung werden in den Abschnitten 4.1 und 4.2 behandelt. Die Güte der Kalibrierung kann dabei durch die Konfigurierung des Netzes beeinflußt werden. Das Ziel ist die Bestimmung der Parameter des Netzes, die zu einer optimalen Kalibrierung führen.
Die Versuchsreihen analysieren unterschiedlich konfigurierte Netze, bei denen die Clusterzentrendichte und der Normierungsparameter variieren. Außerdem wird das *Randproblem* behandelt, welches auftritt, wenn die Zentren von Basisfunktionen am Rand liegen, wo sie nur einseitig von anderen Basisfunktionen überlappt werden. Ein weiterer Versuch untersucht die Güte der Kalibrierung bei der Verwendung eines normierten RBF-Netzes.
Abschließend soll eine Entscheidung getroffen werden, wieviele Basisfunktionen man benötigt, wie der Normierungsparameter zu bestimmen ist und ob es sinnvoll ist, das Randproblem zu beheben, damit eine neuronale Kamerakalibrierung durchgeführt werden kann, die einen geringen Fehler impliziert.

Für das erste Experiment im folgenden Abschnitt wurden die Trainingsdaten nicht unter realen Bedingungen erzeugt, sondern durch eine mathematische Funktion, die sich aus der in Kapitel 2 beschriebenen konventionellen Kamerakalibrierung ergibt. Es wird dadurch eine ungenaue Bestimmung der Computerbildkoordinaten durch Bildverarbeitung vermieden.

4.1 Manipulator-Einzelkamera-Kalibrierung

Es soll eine Funktion gelernt werden, die jedem dreidimensionalen Vektor, der die Position des Greifers in Weltkoordinaten darstellt, einen zweidimensionalen

Vektor für seine Position im Computerbild einer Kamera zuordnet. Es wird somit eine Manipulator-Einzelkamera-Kalibrierung durch das Lernen der folgenden Funktion durchgeführt:

$$f : \mathbb{R}^3 \longrightarrow \mathbb{R}^2, \quad \begin{pmatrix} x_w \\ y_w \\ z_w \end{pmatrix} \mapsto \begin{pmatrix} x_b \\ y_b \end{pmatrix} \tag{4.1}$$

Diese Kalibrierung wird für eine Servosteuerung des Roboters bzw. eine feine Annäherung an ein Objekt sowie für eine Annäherung mit Hindernisvermeidung benötigt. Für jede dieser Anwendungen kann nämlich stets die Position der Greiffinger im Computerbild vorhergesagt und somit Beziehungen zwischen der Position der Finger und den Positionen von Hindernissen oder des Objektes im Bild hergestellt werden.

4.1.1 Trainieren eines RBF-Netzes

Um die Funktion in (4.1) mit einem RBF-Netz zu lernen, werden dem Netz zunächst äquidistante Punkte im Definitionsbereich als dreidimensionale Vektoren mit ihren zugehörigen Computerbildkoordinaten vorgegeben. Nach dem Trainieren mit diesen Trainingsdaten sollte das Netz in der Lage sein, für beliebige Punkte eine interpolierte Ausgabe zu liefern, wobei die Abweichung zu den erwünschten Computerbildkoordinaten möglichst gering sein muß.
Abb. 4.1 zeigt die schematische Darstellung dieses Netzes.

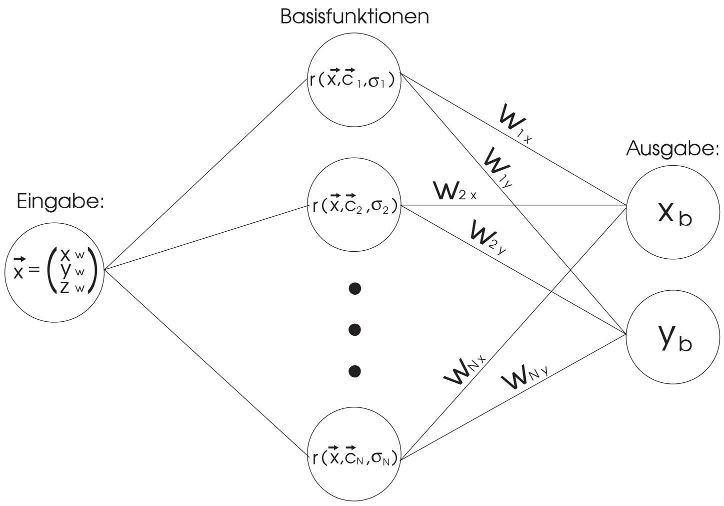


Abbildung 4.1: RBF-Netz für die Manipulator-Einzelkamera-Kalibrierung.

Auf einen Eingabevektor $\vec{x}$ soll die Basisfunktion jedes Knotens der verdeckten Schicht eine lokale Antwort erzeugen, die sich nur dann signifikant auf die Ausgabe auswirkt, wenn $\vec{x}$ nahe genug beim Zentrum dieser Basisfunktion liegt. Als

Basisfunktionen werden hierfür die folgenden radial symmetrischen Gaußfunktionen verwendet:

$$r_j = \exp\left[-\frac{(\vec{x}-\vec{c}_j)^T(\vec{x}-\vec{c}_j)}{2\sigma^2}\right], \quad j = 1, 2, \ldots, N \tag{4.2}$$

Dabei ist r_j die Antwort des j-ten Knotens der verdeckten Schicht, $\vec{c}_j$ das Zentrum der Basisfunktion dieses Knotens und σ^2 der Normierungsparameter dieses Knotens. N ist die Anzahl der Knoten der verdeckten Schicht. Der Ausgabevektor $\vec{y}$ ergibt sich dann aus folgendem Gleichungssystem:

$$\vec{y} = \begin{pmatrix} x_b \\ y_b \end{pmatrix} = \tilde{f}(\vec{x}) = \begin{pmatrix} r_1 w_{1x} + \ldots + r_N w_{Nx} \\ r_1 w_{1y} + \ldots + r_N w_{Ny} \end{pmatrix} \approx f(\vec{x}) \tag{4.3}$$

Dabei ist $\tilde{f}$ die vom Netz ausgegebene Approximation von f. Damit das Netz für jede Eingabe $\vec{x}$ eine Ausgabe $\vec{y}$ mit möglichst geringem Fehler ausgibt, müssen die Gewichte w_{jx}, w_{jy}, $j = 1, \ldots, N$, entsprechend trainiert werden. Dieses geschieht, indem eine Menge von M Trainingsdaten vorgegeben wird, die Eingabevektoren $\vec{x}$ mit ihren zugehörigen Ausgabevektoren $\vec{y}$ enthält. Im folgenden existieren stets mehr Trainingsdaten als Basisfunktionen, d.h. es gilt $M \geq N$.

Die Zentren der Basisfunktionen sind in folgenden Versuchen sinnvollerweise äquidistant zueinander und können als Clusterzentren der Trainingsdaten betrachtet werden, d.h. sie ergeben sich als Mittelwerte derjenigen Vektoren $\vec{x}$ der Trainingsdaten, die in einer Ähnlichkeitsklasse, also nahe beieinander, liegen. In den Versuchsreihen des Abschnittes 4.1.3 sind die Clusterzentren immer identisch mit einigen der Vektoren $\vec{x}$ aus der Trainingsmenge. Durch diese Wahl der Clusterzentren kann der Aufwand eines Cluster-Algorithmus' eingespart werden.

Sei $\mathcal{T} = \{(\vec{x}_1, \vec{y}_1), \ldots, (\vec{x}_M, \vec{y}_M)\}$ die Trainingsmenge und r_{i_j} die Antwort der Gaußfunktion des j-ten Knotens der verdeckten Schicht auf $\vec{x}_i$, dann müssen die Interpolationsbedingungen folgender Gleichungssysteme durch Lineare Ausgleichsrechnung optimiert werden, um die Gewichte w_{j_x}, w_{j_y}, $j = 1, \ldots, N$, so zu ermitteln, daß für jede Eingabe $\vec{x}_i$ der Trainingsmenge in (4.3) die entsprechende Ausgabe $\vec{y}_i$ mit geringem Fehler berechnet wird:

$$\begin{pmatrix} x_{b_1} \\ \vdots \\ x_{b_M} \end{pmatrix} = \begin{pmatrix} r_{1_1} & \cdots & r_{1_N} \\ \vdots & & \vdots \\ r_{M_1} & \cdots & r_{M_N} \end{pmatrix} \begin{pmatrix} w_{1_x} \\ \vdots \\ w_{N_x} \end{pmatrix} \tag{4.4}$$

$$\begin{pmatrix} y_{b_1} \\ \vdots \\ y_{b_M} \end{pmatrix} = \begin{pmatrix} r_{1_1} & \cdots & r_{1_N} \\ \vdots & & \vdots \\ r_{M_1} & \cdots & r_{M_N} \end{pmatrix} \begin{pmatrix} w_{1_y} \\ \vdots \\ w_{N_y} \end{pmatrix} \tag{4.5}$$

4.1.2 Simulieren von Trainingsdaten

Für die Versuche des folgenden Abschnittes sollen zunächst simulierte Trainingsdaten verwendet werden, die eine möglichst genaue Darstellung von realen Daten bilden.
Zur Vereinfachung wird angenommen, daß keine Linsenverzerrungen auftreten und die effektive Brennweite $\hat{f}$ dem Wert der Brennweite $f = 12\ mm$ entspricht. Das diskretisierte resultierende Computerbild der Kamera sei $N_{b_x} = 512$ Pixel breit und $N_{b_y} = 512$ Pixel hoch, und der Ursprung des Kamerakoordinatensystems befinde sich im Zentrum $(C_x, C_y) = (256, 256)$ des Computerbildes. Zur weiteren Vereinfachung wird angenommen, daß der CCD-Chip der Kamera 5.12 mm breit und hoch ist, d.h. es gilt $N_{c_x} d_x = N_{c_y} d_y = 5.12\ mm$, wobei N_{c_x} bzw. N_{c_y} die Anzahl der Sensorelemente in x- bzw. y-Richtung des Kamerakoordinatensystems und d_x bzw. d_y die Abstände zweier Sensorelemente in diese Richtungen darstellen.
Weiterhin werden noch eine Rotationsmatrix und ein Translationsvektor (dargestellt in (2.5)) benötigt. Diese werden dadurch bestimmt, daß der Manipulator in einem quaderförmigen Arbeitsbereich die acht in Weltkoordinaten beschriebenen Eckpunkte anfährt. Eine Kamera, die so aufgestellt wird, wie sie später im Realfall stehen soll, liefert jeweils ein Computerbild, aus dem die Positionen der Eckpunkte grob ermittelt werden. Mit Linearer Ausgleichsrechnung kann nun durch Vorgabe der acht Koordinatenpaare die Bestimmung der Rotationsmatrix und des Translationsvektors erfolgen[1]. Diese ergeben sich wie folgt:

$$R = \begin{pmatrix} -0.02 & -0.28 & -0.06 \\ -0.14 & 0 & 0.2 \\ 0 & 0 & 0 \end{pmatrix}, \quad \vec{t} = \begin{pmatrix} 44.17 \\ 126.02 \\ 300.0 \end{pmatrix} \tag{4.6}$$

Damit lassen sich die Computerbildkoordinaten x_b und y_b aus den Weltkoordinaten x_w, y_w und z_w wie folgt berechnen (vgl. (2.4), (2.1) und (2.3)):

$$\begin{pmatrix} x_c \\ y_c \\ z_c \end{pmatrix} = R \begin{pmatrix} x_w \\ y_w \\ z_w \end{pmatrix} + \vec{t} \tag{4.7}$$

$$x_{u_a} = -x_c \frac{\hat{f}}{z_c}, \quad y_{u_a} = -y_c \frac{\hat{f}}{z_c} \tag{4.8}$$

$$x_b = \frac{N_{b_x}}{N_{c_x} d_x} x_{u_a} + C_x, \quad y_b = \frac{N_{b_y}}{N_{c_y} d_y} y_{u_a} + C_y \tag{4.9}$$

[1] Als Abstand der Kamera zu der Szene wurden 300 cm angenommen.

Es läßt sich somit mit den ermittelten externen und internen Parametern der Kamera folgende Funktion f bestimmen, die aus den Weltkoordinaten eines Raumpunktes seine zugehörigen Computerbildkoordinaten berechnet:

$$f: \mathbb{R}^3 \longrightarrow \mathbb{R}^2, \quad \begin{pmatrix} x_w \\ y_w \\ z_w \end{pmatrix} \mapsto \begin{pmatrix} x_b \\ y_b \end{pmatrix} \tag{4.10}$$

4.1.3 Versuchsreihen mit simulierten Trainingsdaten

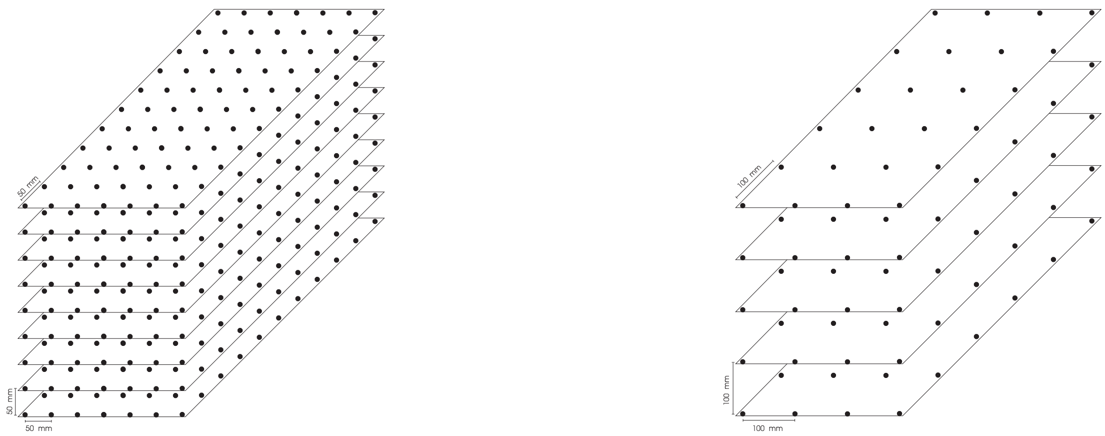
Im folgenden werden zwei Versuchsreihen durchgeführt, in denen jeweils die mathematisch darstellbare Funktion f in (4.10) durch eine Funktion $\tilde{f}$ approximiert werden soll. Die Funktion f geht dabei aus der konventionellen Kamerakalibrierung (Kapitel 2, Abschnitt 2.3) hervor. Auf diese Weise werden bei der Bestimmung der Computerbildkoordinaten für die Erzeugung einer Trainingsmenge und für die Überprüfung des RBF-Netzes auf seine Robustheit mit einer Fehlerberechnung Ungenauigkeiten vermieden, da keine Bildverarbeitung benötigt wird.
Die folgenden Faktoren wirken sich auf die Güte der Approximation aus und sollen deshalb in den Versuchsreihen analysiert werden:

- Distanz und Anzahl der Clusterzentren, also Clusterzentrendichte
- Größe des Normierungsparameters
- Auslassen von äußeren Schichten der Trainingsmenge

Durch die Zentrendichte wird der Grad der Generalisierung bestimmt. Je weniger Clusterzentren gewählt werden, desto mehr Daten der Trainingsmenge werden zusammengefaßt, d.h. es wird umfassender generalisiert. Eine sehr geringe Generalisierung durch viele Clusterzentren würde zu einer besseren Approximation führen, jedoch enstehen dadurch wegen der typischerweise großen Trainingsmenge Redundanzen. Auf der anderen Seite bewirkt eine umfassende Generalisierung einen Informationsverlust und somit schlechtere Approximationsergebnisse. Deshalb muß ein Kompromiß zwischen Redundanz und Generalisierung getroffen werden (s. Kapitel 3, Abschnitt 3.2.1).
Die Größe des Normierungsparameters bestimmt den Signifikanzbereich der zugehörigen Basisfunktion, d.h. es können durch einen hohen Normierungsparameter große Überlappungen der Cluster erzeugt werden. Eine geringe Überlappung bewirkt eine gute Approximation an den Punkten der Trainingsmenge, jedoch eine schlechte Interpolation an Zwischenpositionen. Eine hohe Überlappung führt allerdings zu einer Übergeneralisierung. Deshalb muß auch bei der Wahl des Normierungsparameters ein Kompromiß getroffen werden.

Das Auslassen von äußeren Schichten verhindert, daß Trainingsdaten zu nahe bei den Randzentren liegen, die nur einseitig von anderen Zentren überlappt werden, und bewirkt somit eine Verbesserung der Approximation.

In Versuchsreihe A wurden die Clusterzentren als äquidistant verteilte Raumpunkte aus einem 400 mm hohen, 300 mm breiten und 500 mm tiefen Quader gewählt, der als Arbeitsbereich X des Manipulators definiert ist. Die Distanz zweier benachbarter Clusterzentren betrug 50 mm (Abb. 4.2(a)). Es ergab sich somit als Anzahl an Clusterzentren $N = 9 \cdot 7 \cdot 11 = 693$. In Versuchsreihe B wurde der gleiche Definitionsbereich abgebildet, jedoch mit verdoppeltem Abstand der Zentren, also mit einer verringerten Clusterzentrendichte. Insgesamt ergaben sich hier $N = 5 \cdot 4 \cdot 6 = 120$ Clusterzentren (Abb. 4.2(b)).



(a) (b)

Abbildung 4.2: Die Clusterzentren für Versuchsreihe A (a) und nach Verdoppeln der Clusterzentrendistanz für Versuchsreihe B (b).

Versuchsreihe A:

In dieser Versuchsreihe (mit hoher Clusterzentrendichte) wurde in vier aufeinanderfolgenden Versuchen beim Trainieren eine zunehmende Zahl von äußeren Schichten der Trainingsdaten ausgelassen. Die Anordnung von Trainingsdaten und Clusterzentren dieser Reihe kann wie folgt beschrieben werden:

A0. Die Zentren der Basisfunktionen sind äquidistant mit einem Abstand von 50 mm auf den Arbeitsbereich des Manipulators verteilt (Abb. 4.2 (a)). Die Trainingsdaten sind ebenfalls äquidistant auf diesen Bereich mit einem Abstand von 25 mm verteilt (Abb. 4.3 (a)).

A1. Die Zentren der Basisfunktionen und die Trainingsdaten werden wie in A0 gewählt, jedoch wird die äußere Punkteschicht der Trainingsdaten ausgelassen (Abb. 4.3 (b)).

A2. Die Zentren der Basisfunktionen und die Trainingsdaten werden wie in A0 gewählt, jedoch werden die äußeren zwei Punkteschichten der Trainingsdaten ausgelassen (Abb. 4.3 (c)).

A3. Die Zentren der Basisfunktionen und die Trainingsdaten werden wie in A0 gewählt, jedoch werden die äußeren drei Punkteschichten der Trainingsdaten ausgelassen (Abb. 4.3 (d)).

Versuchsreihe B:

Diese Versuchsreihe (mit geringer Clusterzentrendichte) wurde analog zu Versuchsreihe A durchgeführt. Die Anordnung von Trainingsdaten und Clusterzentren dieser Reihe kann wie folgt beschrieben werden:

B0. Die Zentren der Basisfunktionen sind äquidistant mit einem Abstand von 100 mm auf den Arbeitsbereich des Manipulators verteilt (Abb. 4.2 (b)). Die Trainingsdaten sind ebenfalls äquidistant auf diesen Bereich mit einem Abstand von 25 mm verteilt (Abb. 4.3 (a)).

B1. Die Zentren der Basisfunktionen und die Trainingsdaten werden wie in B0 gewählt, jedoch wird die äußere Punkteschicht der Trainingsdaten ausgelassen (Abb. 4.3 (b)).

B2. Die Zentren der Basisfunktionen und die Trainingsdaten werden wie in B0 gewählt, jedoch werden die zwei äußeren Punkteschichten der Trainingsdaten ausgelassen (Abb. 4.3 (c)).

B3. Die Zentren der Basisfunktionen und die Trainingsdaten werden wie in B0 gewählt, jedoch werden die drei äußeren Punkteschichten der Trainingsdaten ausgelassen (Abb. 4.3 (d)).

Für beide Versuchsreihen konnte nun jeweils mit (4.10) eine Trainingsmenge $\mathcal{T} = \{(\vec{x}_1, f(\vec{x}_1)), \ldots, (\vec{x}_M, f(\vec{x}_M))\}$ erzeugt werden. Nach dem Lösen der Gleichungssysteme (4.4) und (4.5) wurde mit den berechneten Gewichten w_{j_x}, w_{j_y}, $j = 1, \ldots, N$, und der Trainingsmenge das RBF-Netz trainiert, welches die Funktion f approximiert.

Um die Güte der Approximation zu beschreiben, wurde mit einer *Testdatenmenge* $\mathcal{U} = \{(\vec{x}_{u_1}, f(\vec{x}_{u_1})), \ldots, (\vec{x}_{u_L}, f(\vec{y}_{u_L}))\}$ der *Approximationsfehler* als mittlere Abweichung der Ausgabevektoren $\tilde{f}(\vec{x}_{u_l})$ des Netzes zu den Ausgabevektoren $f(\vec{y}_{u_l})$, $l = 1, \ldots, L$, der Testdatenmenge folgendermaßen ermittelt:

$$
\begin{aligned}
error_{mean} &= \frac{1}{L} \sum_{l=1}^{L} \left\| \tilde{f}(\vec{x}_{u_l}) - f(\vec{x}_{u_l}) \right\| \\
&= \frac{1}{L} \sum_{l=1}^{L} \sqrt{\left(\tilde{f}(\vec{x}_{u_l}) - f(\vec{x}_{u_l})\right)^T \left(\tilde{f}(\vec{x}_{u_l}) - f(\vec{x}_{u_l})\right)}
\end{aligned}
\tag{4.11}
$$

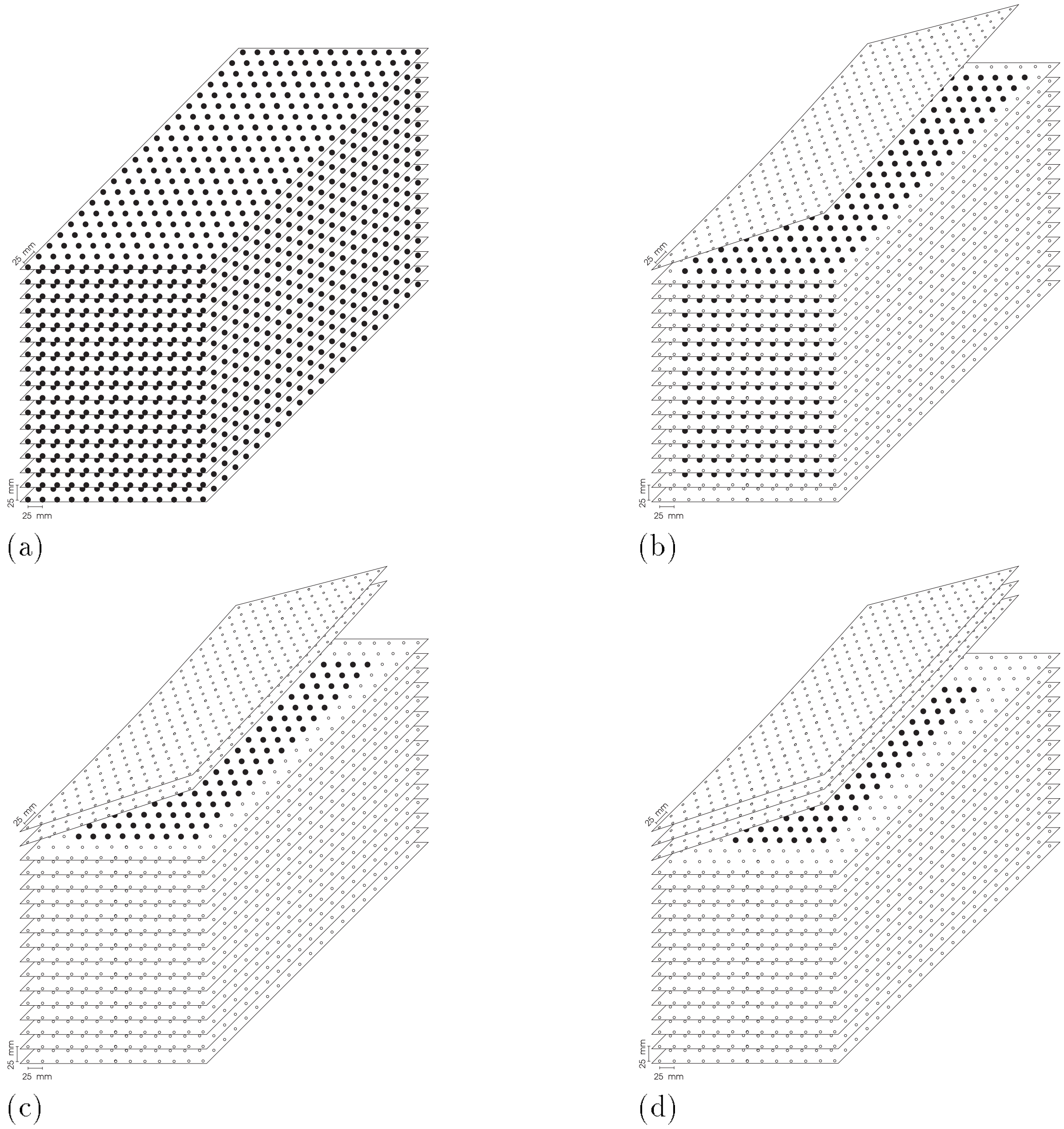


Abbildung 4.3: Die Trainingsdaten für das RBF-Netz ohne Auslassen von äußeren Schichten (a), bei Auslassen einer äußeren Schicht (b), bei Auslassen von zwei äußeren Schichten (c) und bei Auslassen von drei äußeren Schichten (d).

Da $\tilde{f}(\vec{x}_{u_l})$ und $f(\vec{x}_{u_l})$ zweidimensionale Vektoren sind, die jeweils Computerbildkoordinaten darstellen, ist die mittlere Abweichung $error_{mean}$ in Pixeln dargestellt.
Untersucht wurde nun, wie groß der Normierungsparameter σ^2 zu wählen ist, um einen möglichst geringen Fehler zu erhalten. Die Werte des mittleren Fehlers für eine Testdatenmenge $\mathcal{U}$ mit einem Testdatenabstand von 25 mm ($\mathcal{U} = \mathcal{T}$) bei unterschiedlichen σ^2 für die Versuchsreihen A und B sind in Abb. 4.4 und Abb. 4.5 zu sehen.

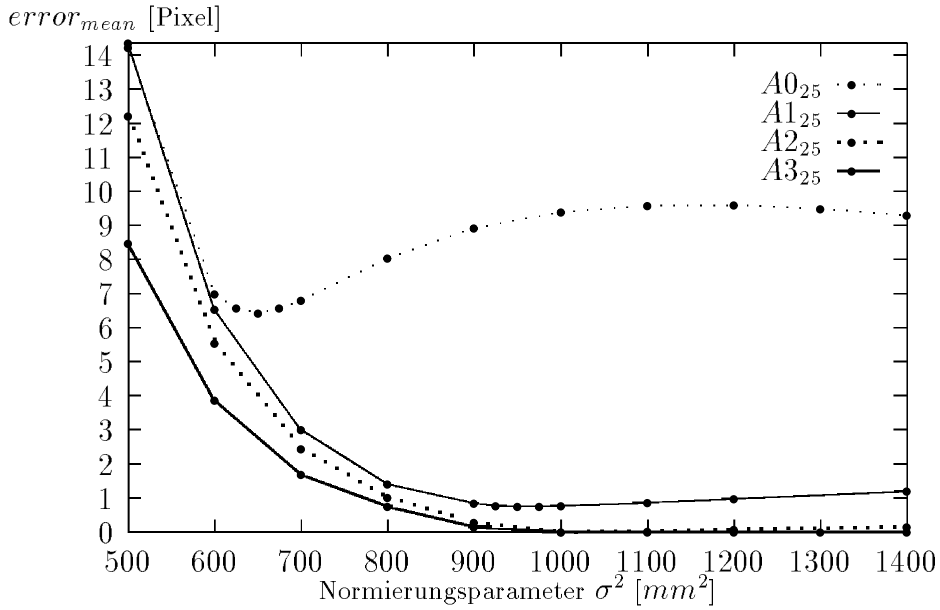


Abbildung 4.4: Approximationsfehler beim Test mit unterschiedlichen Normierungsparametern σ^2 für Versuchsreihe A.

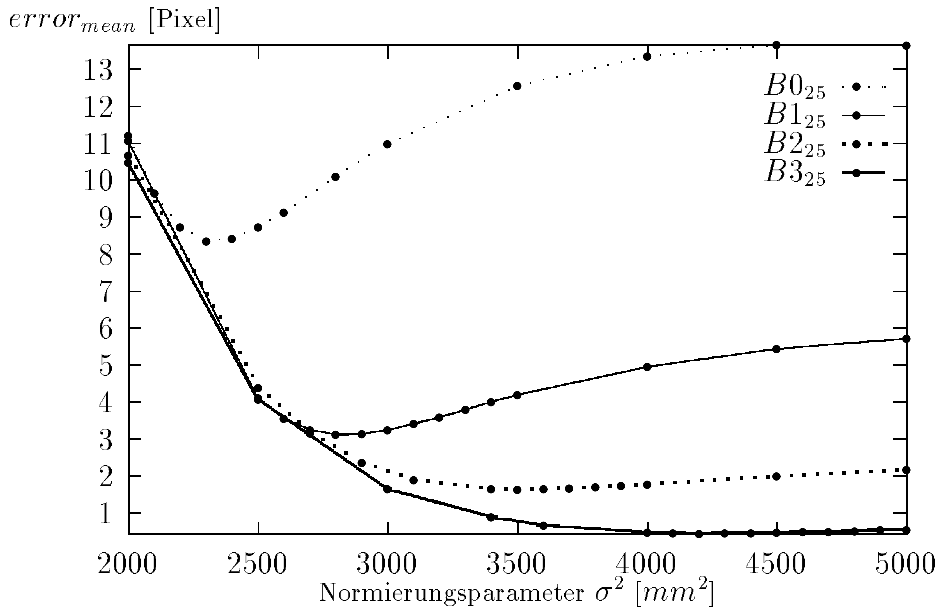


Abbildung 4.5: Approximationsfehler beim Test mit unterschiedlichen Normierungsparametern σ^2 für Versuchsreihe B.

Die Kurve $A1_{25}$ entspricht dem Versuch A1 bei einem Abstand der Testdaten von 25 mm. Die Bezeichnungen der übrigen Kurven sind analog zu deuten.
Die Verläufe der Fehlerkurven für die Versuchsreihen A bzw. B lassen erkennen, daß ein Normierungsparameter σ^2_{opt} existiert, für den der Approximationsfehler minimal wird (z.B. $\sigma^2_{opt} = 650$ bei der Kurve $A0_{25}$).

Die Fehler verringerten sich außerdem, je mehr äußere Schichten ausgelassen wurden. Es wird also die Annahme bestätigt, daß das Randproblem zu beheben ist, um einen minimalen Approximationsfehler zu erhalten. Bei zwei ausgelassenen äußeren Schichten der Trainingsdaten ergab sich in Versuchsreihe A bzw. Versuchsreihe B ein minimaler Approximationsfehler mit den Normierungsparametern ${\sigma_A}^2 = 975\ mm^2$ bzw. ${\sigma_B}^2 = 3500\ mm^2$ (vgl. $A2_{25}$ in Abb. 4.4 bzw. $B2_{25}$ in Abb. 4.5). Diese experimentell ermittelten Werte entsprechen ungefähr dem Wert $\sigma^2_{opt} = 0.37 \cdot cl^2_{dist}$, wobei cl_{dist} jeweils der Abstand zweier benachbarter Clusterzentren ist (also $cl_{dist} = 50\ mm$ bei Versuchsreihe A bzw. $cl_{dist} = 100\ mm$ bei Versuchsreihe B).

Es ist in den Verläufen der Fehlerkurven für die beiden Versuchsreihen weiterhin zu erkennen, daß sich bei ansteigender Zahl ausgelassener äußerer Schichten die Minima nach rechts verschieben. Offensichtlich hängt also das Randproblem vom Normierungsparameter ab. Um dieses Phänomen erklären zu können, ist der folgende Einschub zu beachten.

Der aus der konkreten Anwendung gewonnene optimale Normierungsparameter kann auch mathematisch bestätigt werden. Die Bestimmung des Parameters stellt ein nichtlineares Problem dar, welches durch einen Optimierungsalgorithmus von Levenberg-Marquardt [Pre92] gelöst werden kann. Der Algorithmus liefert für eine Vereinfachung durch die Reduzierung auf vier Basisfunktionen ein ähnliches Ergebnis, nämlich $\sigma^2_{opt} = 0.32 \cdot cl^2_{dist}$. Abb. 4.6 zeigt die Überlagerung von vier Basisfunktionen mit ihren Zentren bei $c_1,\ldots,c_4$ für eindimensionale Eingaben. Hier wurde $\sigma^2_{opt} = 0.32 \cdot (c_{j+1} - c_j)^2$, $j = 1, \ldots, 3$, gewählt.

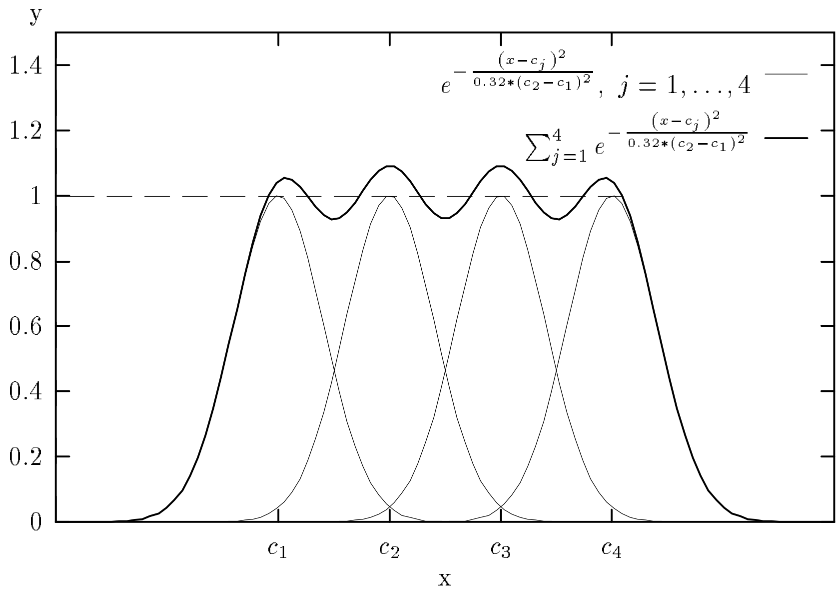


Abbildung 4.6: Vier Basisfunktionen mit ihren Zentren bei $c_1,\ldots,c_4$ und deren Überlagerung durch Addition bei einem optimalen Normierungsparameter.

Es ist zu erkennen, daß für die Überlagerung der Basisfunktionen im Bereich zwischen den beiden inneren, also im Intervall $[c_2, c_3]$, die Funktionswerte im Intervall $[0.9, 1.1]$, also eng bei 1, liegen.
Wird der Normierungsparameter geringer gewählt (z.B $\sigma^2 = 0.25 \cdot cl_{dist}^2$), tritt eine wesentlich höhere Schwankung in den Funktionswerten der Überlagerung auf. Diese Schwankung ist allerdings gleichmäßig im Bereich zwischen den Zentren der beiden äußeren Basisfunktionen, also im Intervall $[c_1, c_4]$ (Abb. 4.7). Für einen höheren Normierungsparameter (z.B. $\sigma^2 = 0.56 \cdot cl_{dist}^2$) ergibt sich eine geringere Schwankung der Funktionswerte der Überlagerung, allerdings innerhalb eines geringeren Bereiches, nämlich im Intervall $[c_2, c_3]$ zwischen den Zentren der beiden inneren Basisfunktionen (Abb. 4.8).

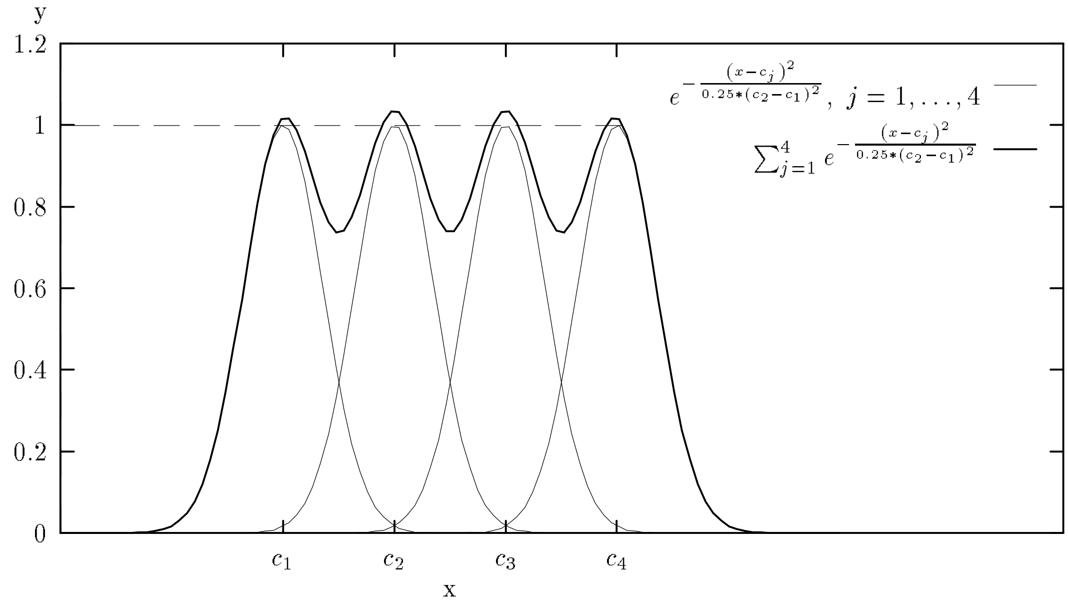


Abbildung 4.7: Vier Basisfunktionen mit ihren Zentren bei c_1,...,c_4 und deren Überlagerung durch Addition bei einem geringen Normierungsparameter.

Wünscht man sich also in einem Teilintervall $Z = [c_1 + \delta_z, c_4 - \delta_z]$ des Intervalls $[c_1, c_4]$ eine minimale und gleichmäßige Schwankung, ist bei Zunahme von δ_z ein zunehmender Normierungsparameter zu wählen.
Der Verlauf dieser drei Überlagerungskurven mit unterschiedlichen Normierungsparametern erklärt, warum in den Versuchsreihen A und B bei Anstieg der Zahl ausgelassener äußerer Schichten (Zunahme von δ_z) ein zunehmender optimaler Normierungsparameter (Verschiebung des Minimums nach rechts) ermittelt wurde.
Es wurde mit den Versuchsreihen A und B gezeigt, daß der Normierungsparameter sich signifikant auf die Güte der Approximation auswirkt. Die Bestimmung des optimalen Normierungsparameters ist wichtig, da sich ein effizientes Netz durch eine geringe Anzahl an Basisfunktionen auszeichnet. Wird z.B. eine obere Schranke $os = 1$ [Pixel] des maximal zulässigen Approximationsfehlers vorgegeben, so

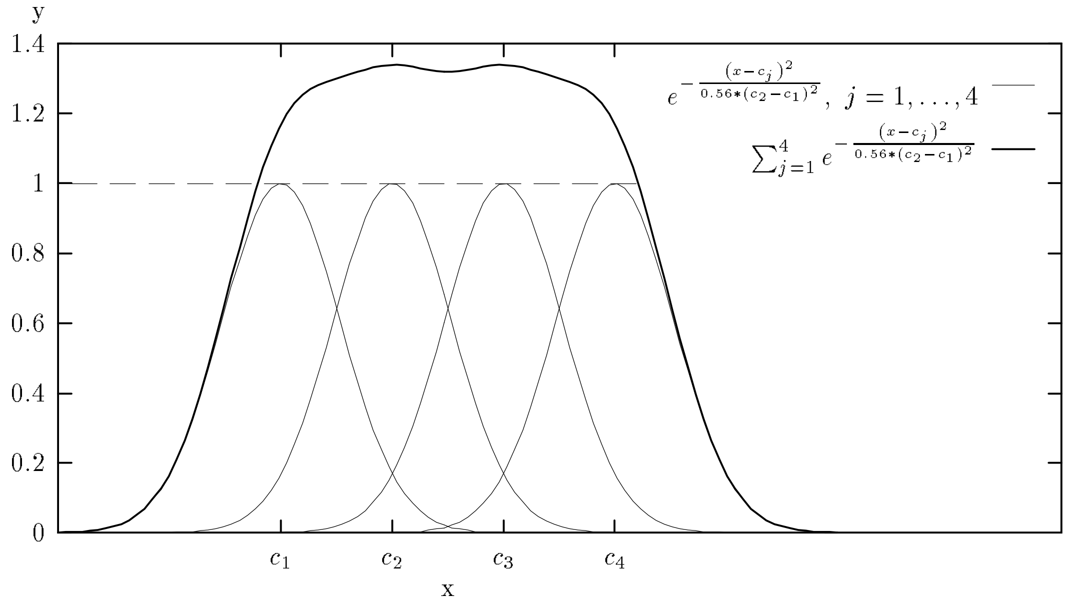


Abbildung 4.8: Vier Basisfunktionen mit ihren Zentren bei $c_1,\ldots,c_4$ und deren Überlagerung durch Addition bei einem hohen Normierungsparameter.

kann das Netz wie in Versuchsreihe A mit einer hohen Anzahl an Basisfunktionen (Clusterzentren) trainiert werden, damit der Fehler unterhalb dieser Schranke liegt. Ein wesentlich effizienteres Netz ergibt sich jedoch für den Versuch B3 aus Versuchsreihe B, da hier weniger als $\frac{1}{5}$ ($\approx \frac{120}{693}$) der Basisfunktionen aus Versuchsreihe A benötigt werden, um ebenfalls einen Approximationsfehler unterhalb von os zu erhalten. Hier ist allerdings das Auslassen von drei äußeren Schichten der Trainingsdaten beim Trainieren notwendig.

Eine weitere Möglichkeit der Manipulator-Einzelkamera-Kalibrierung ergibt sich durch die Verwendung eines normierten RBF-Netzes (s. Kapitel 3, Abschnitt 3.2.2). Die Abbildung 4.9 zeigt das Ergebnis der Versuchsreihe B für ein normiertes RBF-Netz. Ein Vergleich mit Abb. 4.5 zeigt, daß mit dem normierten Netz geringere Approximationsfehler erreicht werden können. Diese sind jedoch wie bei dem nicht normierten Netz abhängig von der Wahl des Normierungsparameters.

In Tabelle 4.1 sind die genauen Ergebnisse für die Versuchsreihen A (A0,...,A3) und B (B0,...,B3) und die Ergebnisse für die Versuchsreihe B mit normiertem RBF-Netz ($B0_n,\ldots,B3_n$) zusammengefaßt. Dabei wird wie bisher die Anzahl der Basisfunktionen mit N, der Abstand zweier benachbarter Basisfunktionen mit cl_{dist} und der optimale Normierungsparameter mit σ^2_{opt} (experimentell und rechnerisch) bezeichnet. Der Fehler $error_{min}$ gibt den minimalen Approximationsfehler an, der sich für den experimentellen optimalen Normierungsparameter gemäß (4.11) ergibt.

Die Bestimmung des Approximationsfehlers erfolgte für die Versuchsreihen A und B mit einer Testdatenmenge, die identisch mit der Trainingsmenge war ($\mathcal{U} = \mathcal{T}$).

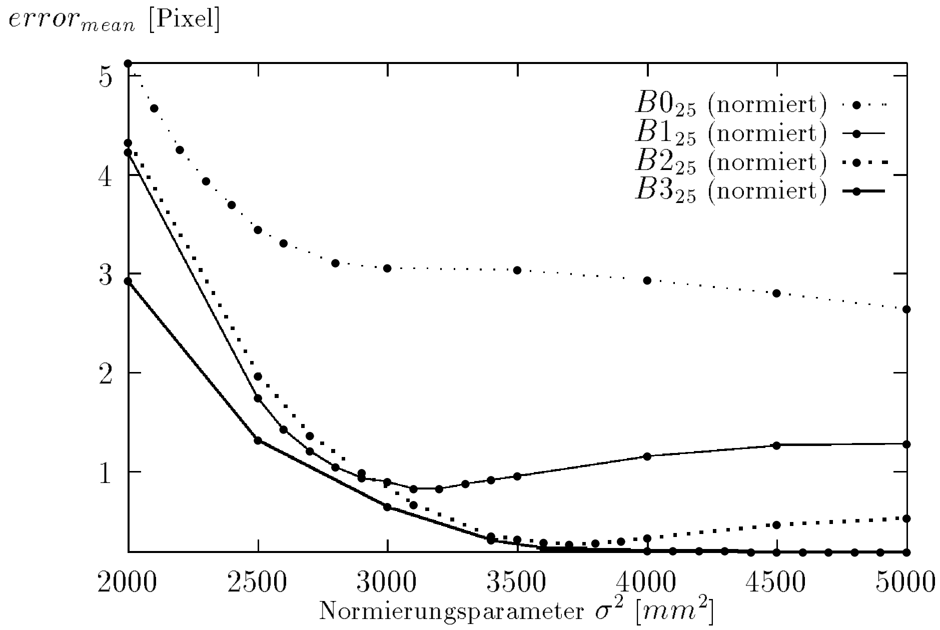


Abbildung 4.9: Approximationsfehler beim Test mit unterschiedlichen Normierungsparametern σ^2 für Versuchsreihe B mit normiertem RBF-Netz.

Versuch	N	cl_{dist} [mm]	Anzahl ausgelassener Schichten	σ^2_{opt} $[mm^2]$ (exp.)	σ^2_{opt} $[mm^2]$ (rechn.)	$error_{min}$ [mm]
A0	693	50	0	650	800	6.43
A1	693	50	1	975	800	0.75
A2	693	50	2	1000	800	0.00
A3	693	50	3	> 1000	800	0.00
B0	120	100	0	2300	3200	8.34
B1	120	100	1	2800	3200	3.12
B2	120	100	2	3500	3200	1.63
B3	120	100	3	4200	3200	0.43
$B0_n$	120	100	0	2800	3200	3.11
$B1_n$	120	100	1	3100	3200	0.83
$B2_n$	120	100	2	3700	3200	0.27
$B3_n$	120	100	3	4200	3200	0.43

Tabelle 4.1: Ergebnisse der Versuchsreihen A und B bei Verwendung des optimalen Normierungsparameters.

Da angenommen wurde, daß der Fehler unter diesen Umständen etwas verfälscht wird, wurde für den Versuch B2 die Anzahl der Testdaten erhöht, indem der Abstand von 25 mm auf 10 mm verringert wurde. Abb. 4.10 zeigt den Vergleich der Kurven für den Approximationsfehler des ursprünglichen Versuches ($B2_{25}$)

und des neuen Versuches (B2_{10}). Die Differenz der Fehler beider Versuche für den optimalen Normierungsparameter $\sigma^2_{opt} = 3500$ $[mm^2]$ beträgt $1.73 - 1.63 = 0.10$ [Pixel] und ist somit nicht gravierend.

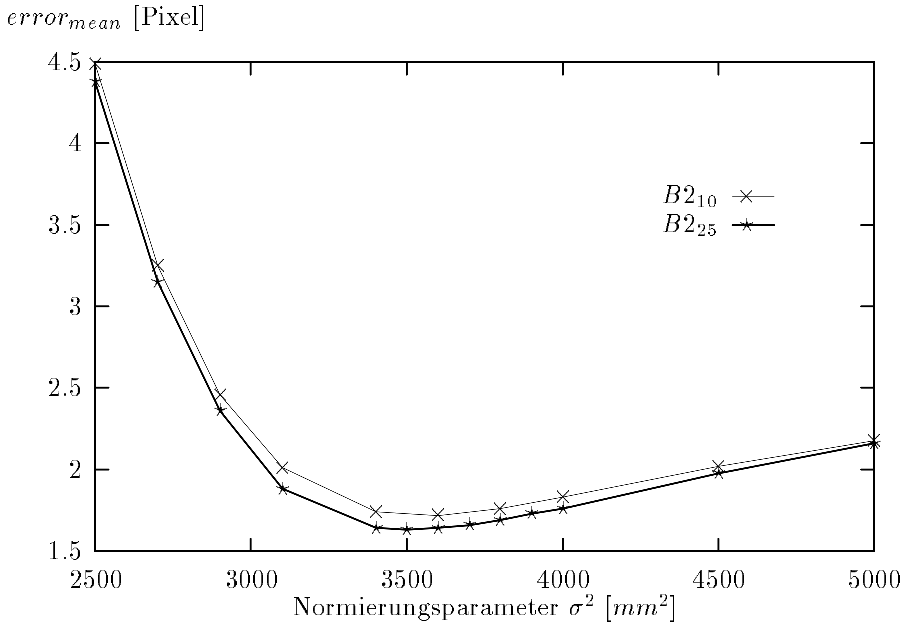


Abbildung 4.10: Approximationsfehler beim Test mit unterschiedlichen Normierungsparametern σ^2 für Versuch $B2$ bei einem Abstand der Testdaten von 25 mm ($B2_{25}$) und 10 mm ($B2_{10}$).

Die Versuche dieses Abschnittes haben gezeigt, daß bei Vorgabe eines bestimmten Arbeitsbereiches X des Manipulators und eines maximal zulässigen Approximationfehlers os [Pixel] die Konfigurierung eines effizienten RBF-Netzes bestimmt werden kann, das eine Manipulator-Einzelkamera-Kalibrierung ausführt. Die Effizienz drückt sich dadurch aus, daß einerseits möglichst wenig Basisfunktionen verwendet werden und andererseits zu den Weltkoordinaten eines Raumpunktes die zugehörigen Computerbildkoordinaten mit möglichst geringem Fehler berechnet werden. Um eine optimale Kalibrierung zu erzielen, sind somit folgende Aktionen notwendig:

1. Wähle eine geringe Anzahl an Basisfunktionen.

2. Wähle die äquidistant auf X verteilten Zentren der Basisfunktionen, und bestimme den Abstand cl_{dist} zweier benachbarter Zentren.

3. Berechne den optimalen Normierungsparameter $\sigma^2_{opt} = 0.32 \cdot cl^2_{dist}$.

4. Bestimme äquidistant auf X verteilt Trainingsdaten, wobei äußere Schichten ausgelassen werden sollten.

5. Trainiere normiertes RBF-Netz mit den Trainingsdaten.

6. Bestimme durch Anwenden des Netzes mit einer Testdatenmenge den Approximationsfehler $error_{mean}$.

7. Gilt $error_{mean} > os$, müssen die Schritte 1-6 mit einer größeren Anzahl an Basisfunktionen wiederholt werden.

4.1.4 Versuch mit realen Trainingsdaten

In dem folgenden Versuch soll bestätigt werden, daß sich auch für ein RBF-Netz, das mit realen Trainingsdaten trainiert wird, der im letzten Abschnitt ermittelte optimale Normierungsparameter ergibt. Hier ist jedoch die zu lernende Funktion mathematisch nicht gegeben. Es müssen somit die Elemente der Trainings- und der Testdatenmenge unter realen Bedingungen bestimmt werden.

Für die Erzeugung der Trainingsmenge wurde der Greifer des Roboters mit einer Lichtquelle versehen, um die Computerbildkoordinaten mit Hilfe eines einfachen Maximumoperators leicht zu ermitteln. Der Manipulator fuhr nun in dem im Abschnitt 4.1.3 bestimmten Arbeitsbereich äquidistante Punkte an, und es wurde mit der rechten Kamera jeweils eine Aufnahme gemacht. Die optische Achse der Kamera zeigte dabei auf den Schwerpunkt des Arbeitsbereiches. Die Weltkoordinaten der Lichtquelle konnten leicht bestimmt werden, weil die Position des Manipulators in Weltkoordinaten bekannt war. In die Trainingsmenge wurden dann nach jedem Schritt die Weltkoordinaten und die ermittelten Computerbildkoordinaten der Lichtquelle aufgenommen. Es sollte nun ein Netz unter den gleichen Umständen wie in Versuch B1 aus Abschnitt 4.1.3 trainiert werden. Also wurde als Abstand der Clusterzentren 100 mm und als Abstand der Trainingsdaten 25 mm gewählt, wobei die äußere Schicht für das Trainieren ausgelassen wurde. Das Training erfolgte wie in Versuch B1 mit verschiedenen Normierungsparametern. Um nach dem Trainieren die Güte der Approximation feststellen zu können, wurde eine Testdatenmenge erzeugt, indem sich der Roboter in Abständen von 25 mm im Arbeitsbereich bewegte und die jeweiligen Weltkoordinaten und Computerbildkoordinaten der Lichtquelle in die Testdatenmenge aufgenommen wurden. Durch die Eingabe der Weltkoordinaten in das trainierte Netz wurden anschließend die interpolierten Computerbildkoordinaten ermittelt.
Die Güte der Approximation läßt sich anschaulich mit einer Graphik beurteilen, die jeweils für einen Normierungsparameter die Anordnung der interpolierten Computerbildpunkte darstellt. Zur besseren Anschaulichkeit werden nur diejenigen dieser Punkte betrachtet, die sich aus den Weltkoordinaten einer einzigen (der mittleren) Ebene des Arbeitsbereiches ergeben. Die Kurve des Approximationsfehlers in Abb. 4.5 ($B1_{25}$) besitzt ihr Minimum bei dem Normierungsparameter $\sigma^2 = 2800\ mm^2$. Man erwartet, daß Testpunkte, die im Raum auf einer Geraden

lagen, auch in der Graphik für den optimalen Normierungsparameter eine Gerade bilden, wenn man von einer geringen Linsenverzerrung ausgeht. Für $\sigma^2 = 2800$ mm^2 ergab sich eine gleichförmige, fast geradlinige Anordnung der Computerbildpunkte. Weiterhin wurde für unterschiedliche Normierungsparameter der Approximationsfehler nach (4.11) ermittelt und ein Minimum bei $\sigma^2 = 2800$ mm^2 gemessen (Graphiken und Ergebnisse sind in Abb. 4.11 zu sehen). Die Fehler unterschieden sich nur geringfügig von denen des Versuches B1 für simulierte Daten. Damit kann die Aussage in Abschnitt 4.1.3 über den optimalen Normierungsparameter bestätigt werden.

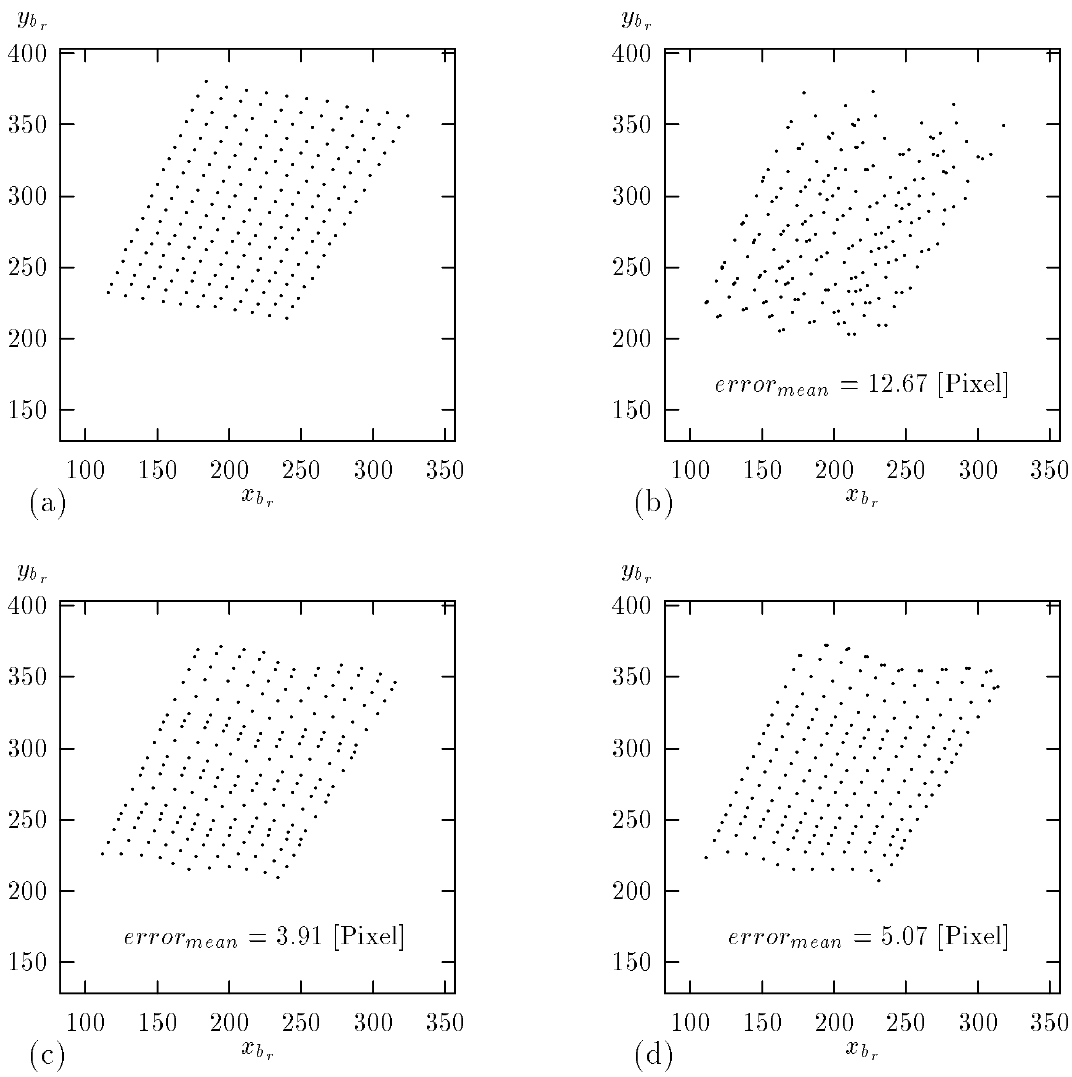


Abbildung 4.11: Darstellung der Computerbildpunkte aus der Trainingsmenge für die rechte Kamera (a) und der interpolierten Computerbildpunkte für $\sigma^2 = 2000$ mm^2 (b), $\sigma^2 = 2800$ mm^2 (c) und $\sigma^2 = 3500$ mm^2 (d).

4.2 Stereokamera-Manipulator-Kalibrierung

In diesem Abschnitt soll eine Kalibrierung beschrieben werden, nach der für einen Lichtpunkt, dargestellt in den Computerbildkoordinaten beider Kameras, seine Weltkoordinaten berechnet werden können. Es wird also folgende Funktion gelernt:

$$f : \mathbb{R}^4 \longrightarrow \mathbb{R}^3, \quad \begin{pmatrix} x_{b_l} \\ y_{b_l} \\ x_{b_r} \\ y_{b_r} \end{pmatrix} \mapsto \begin{pmatrix} x_w \\ y_w \\ z_w \end{pmatrix} \tag{4.12}$$

Dabei werden die Computerbildkoordinaten des Lichtpunktes in den Aufnahmen der beiden Kameras (links u. rechts) mit x_{b_l}, y_{b_l}, x_{b_r} und y_{b_r} und die Weltkoordinaten des Lichtpunktes mit x_w, y_w und z_w bezeichnet.
Diese Funktion kann mit Hilfe eines RBF-Netzes gelernt werden (Abb. 4.12).

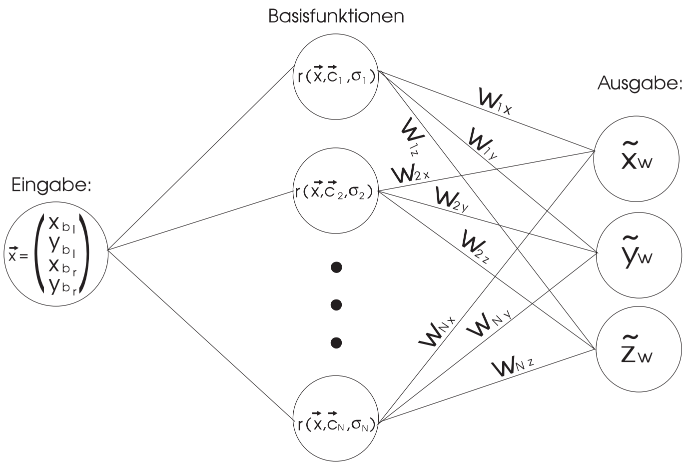


Abbildung 4.12: Schematischer Aufbau des RBF-Netzes für die Stereokamera-Manipulator-Kalibrierung.

Da die Eingabevektoren der Trainingsdaten diesmal nicht äquidistant sind, können die Clusterzentren nicht wie in Abschnitt 4.1 als Teil der Eingabevektoren der Trainingsmenge ohne einen Cluster-Algorithmus bestimmt werden. Eine Möglichkeit der Clusterung besteht darin, die Eingabevektoren mit einem geeigneten Cluster-Algorithmus nach Ähnlichkeit zu gruppieren. Alternativ dazu kann auf den äquidistanten Ausgabevektoren eine Selektion erfolgen und als Clusterzentren die zugehörigen Eingabevektoren gewählt werden. Hierbei stellt sich jedoch die Frage, ob eine Selektion der Ausgabevektoren sinnvoll ist, weil ähnliche Ausgabevektoren nicht unbedingt ähnliche Eingabevektoren implizieren. Allerdings könnte mit dieser Methode der Aufwand der Gruppierung durch einen Cluster-Algorithmus eingespart werden. Deshalb soll untersucht werden, ob sich hierdurch der Approximationsfehler signifikant verschlechtert.

4.2.1 Trainieren eines RBF-Netzes

Für das Trainieren des RBF-Netzes wird wieder eine Menge von M Trainingsdaten mit Eingabevektoren $\vec{x}$ und Ausgabevektoren $\vec{y}$ vorgegeben. Die Funktion f wird nach dem Trainieren durch eine Funktion $\tilde{f}$ wie folgt approximiert:

$$\vec{y} = \begin{pmatrix} \tilde{x}_w \\ \tilde{y}_w \\ \tilde{z}_w \end{pmatrix} = \tilde{f}(\vec{x}) = \begin{pmatrix} r_1 w_{1_x} + \ldots + r_N w_{N_x} \\ r_1 w_{1_y} + \ldots + r_N w_{N_y} \\ r_1 w_{1_z} + \ldots + r_N w_{N_z} \end{pmatrix} \approx f(\vec{x}) \tag{4.13}$$

Dabei ist r_j, $j = 1, \ldots, N$, wie in (4.2) definiert, und die Gewichte w_{j_x}, w_{j_y} und w_{j_z}, $j = 1, \ldots, N$, werden durch Optimierung der Interpolationsbedingungen der Gleichungssysteme (4.14), (4.15) und (4.16) mit Hilfe Linearer Ausgleichsrechnung berechnet.

$$\begin{pmatrix} \tilde{x}_{w_1} \\ \vdots \\ \tilde{x}_{w_M} \end{pmatrix} = \begin{pmatrix} r_{1_1} & \cdots & r_{1_N} \\ \vdots & & \vdots \\ r_{M_1} & \cdots & r_{M_N} \end{pmatrix} \begin{pmatrix} w_{1_x} \\ \vdots \\ w_{N_x} \end{pmatrix} \tag{4.14}$$

$$\begin{pmatrix} \tilde{y}_{w_1} \\ \vdots \\ \tilde{y}_{w_M} \end{pmatrix} = \begin{pmatrix} r_{1_1} & \cdots & r_{1_N} \\ \vdots & & \vdots \\ r_{M_1} & \cdots & r_{M_N} \end{pmatrix} \begin{pmatrix} w_{1_y} \\ \vdots \\ w_{N_y} \end{pmatrix} \tag{4.15}$$

$$\begin{pmatrix} \tilde{z}_{w_1} \\ \vdots \\ \tilde{z}_{w_M} \end{pmatrix} = \begin{pmatrix} r_{1_1} & \cdots & r_{1_N} \\ \vdots & & \vdots \\ r_{M_1} & \cdots & r_{M_N} \end{pmatrix} \begin{pmatrix} w_{1_z} \\ \vdots \\ w_{N_z} \end{pmatrix} \tag{4.16}$$

4.2.2 Gruppierte Eingaben durch Cluster-Algorithmus

Im folgenden Versuch soll zur Stereokamera-Manipulator-Kalibrierung ein RBF-Netz trainiert werden, dessen Clusterzentren durch Gruppierung der Eingabevektoren mit einem Cluster-Algorithmus entstehen. Bei der Anwendung soll analysiert werden, wie sich der Approximationsfehler minimieren läßt.

Die Trainingsmenge wurde dadurch erzeugt, daß der Manipulator äquidistante Punkte in einem zweidimensionalen, 300 mm breiten und 500 mm tiefen Arbeitsbereich X anfuhr und die Computerbildkoordinaten einer am Greifer befestigten Lichtquelle wie in Abschnitt 4.1.4 bestimmt wurden. Die z_w-Komponente der Ausgabevektoren der Trainingsdaten blieb also beim Trainieren konstant, der Abstand der Ausgabevektoren der Trainingsmenge betrug 50 mm, und die äußere

Schicht der Trainingsdaten wurde ausgelassen[2].
Für die Clusterung der Eingabevektoren und der Vektoren der ausgelassenen Schicht (insgesamt $7 \cdot 11 = 77$ Vektoren) wurde der in Kapitel 3 beschriebene ISODATA-Algorithmus verwendet. Es wurden hiermit $N = 24$ Cluster gebildet. Die Anordnung dieser Cluster für die vierdimensionalen Eingabevektoren, die aus den zwei zweidimensionalen Vektoren der Computerbildkoordinaten entstehen, ist schwer darzustellen. Um dennoch diese Anordnung zu veranschaulichen, sind in Abb. 4.13 die entsprechenden Gruppierungen für die äquidistanten dreidimensionalen (hier zweidimensional, da $z_w = const$) Weltkoordinaten der Ausgabevektoren zu sehen[3].

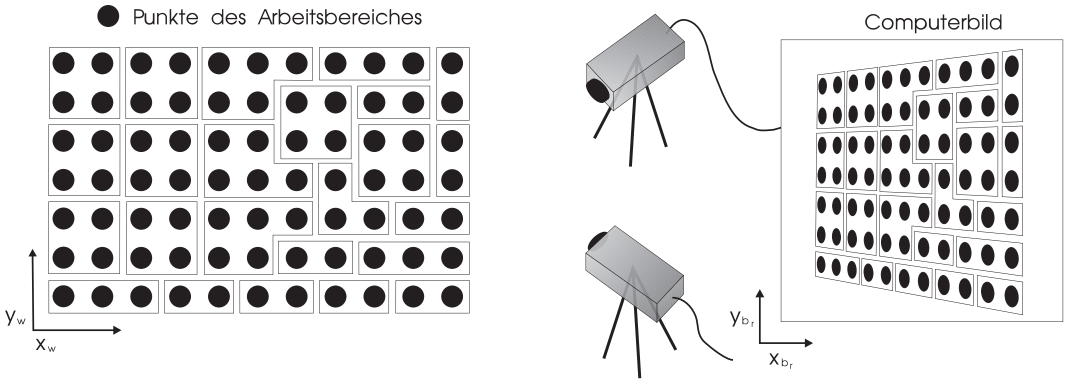


Abbildung 4.13: Anordnung der mit dem ISODATA-Algorithmus ermittelten 24 Cluster.

Es ist zu beachten, daß aufgrund der speziellen Blickrichtung der Kameras die rechteckige Arbeitsbereichebene in den Computerbildern nahezu quadratisch ist. Die Punkte in x_w-Richtung des Arbeitsbereiches wurden hierbei durch die Projektion der Kameras in den Computerbildern in x_b-Richtung zusammengeschoben (*perspektivische Verkürzung*). Die Abb. 4.14 zeigt die projizierten Clusterzentren im rechten Computerbild.

Nach dem Lösen der Gleichungssysteme (4.14), (4.15) und (4.16) mit Hilfe Linearer Ausgleichsrechnung konnte mit den berechneten Gewichten w_{j_x}, w_{j_y} und w_{j_z}, $j = 1, \ldots, N$, und der Trainingsmenge das RBF-Netz trainiert werden, welches die Funktion f durch die Funktion $\tilde{f}$ approximierte.
Um die Güte der Approximation des trainierten Netzes abschätzen zu können,

[2]Die äußere Schicht der Ausgabevektoren wurde ermittelt und auf die Eingabevektoren übertragen.

[3]In diesem speziellen Fall der Kamerablickrichtung ist die x- bzw. y-Richtung der Weltkoordinaten identisch mit der y- bzw. x-Richtung der Computerbildkoordinaten. Daher wurden die Koordinatenbezeichnungen x_{b_r} und y_{b_r} (bzw. x_b und y_b) hier und auch bei jedem weiteren Auftreten sinvollerweise vertauscht, damit die x- und y-Achsen im Computerbild und im Weltkoordinatensystem in gleicher Richtung verlaufen.

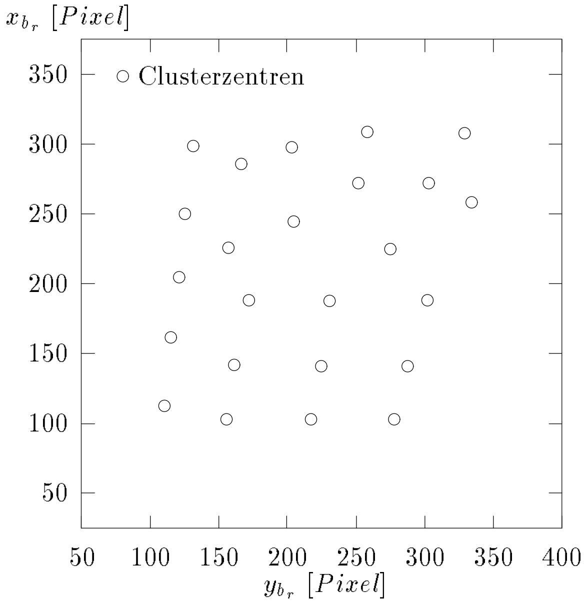


Abbildung 4.14: Projektion der Clusterzentren im Computerbild der rechten Kamera.

werden folgende Fehlerberechnungen eingeführt:

$$error_{\tilde{x}_w} = |\tilde{x}_w - x_w|, \quad error_{\tilde{y}_w} = |\tilde{y}_w - y_w| \tag{4.17}$$

Diese Fehler beschreiben für jede Position (x_w, y_w, z_w) der Lichtquelle die Abweichungen dieser Position zu der vom Netz berechneten Position $(\tilde{x}_w, \tilde{y}_w, \tilde{z}_w)$ in x_w- bzw. y_w-Richtung, gemessen in mm. Um diese Fehler zu messen, wurde der Manipulator schrittweise in Abständen von 25 mm im Arbeitsbereich bewegt. Nach jeder Bewegung wurden die ermittelten Computerbildkoordinaten in das RBF-Netz eingegeben und der Fehler der ausgegebenen Weltkoordinaten zu den Weltkoordinaten der Lichtquelle gemäß der Fehlerfunktionen in (4.17) berechnet. Weiterhin konnte jeweils ein für den Arbeitsbereich maximaler Fehler $max_{\tilde{x}_w}$ bzw. $max_{\tilde{y}_w}$, ein minimaler Fehler $min_{\tilde{x}_w}$ bzw. $min_{\tilde{y}_w}$ und ein gemittelter Fehler $avg_{\tilde{x}_w}$ bzw. $avg_{\tilde{y}_w}$ berechnet werden.
Es wurden nun auf diese Weise RBF-Netze mit unterschiedlichen Normierungsparametern konfiguriert, trainiert und auf die Güte ihrer Approximation getestet. Die resultierenden gemittelten Fehler bei der Anwendung der Netze werden in Abb. 4.15 durch Fehlerkurven dargestellt. Es ist dabei zu erkennen, daß bei Vorgabe eines maximal zulässigen Fehlers ein optimaler Normierungsparameter bestimmt werden kann (z.B. $\sigma^2_{opt} \geq 15000$ für einen maximal zulässigen Fehler von 1 Pixel in x_w- und y_w-Richtung).

Weiterhin ist zu beobachten, daß sich für den gemittelten Fehler in y_w-Richtung geringere Werte erzielen lassen. Der Grund hierfür liegt in der perspektivischen

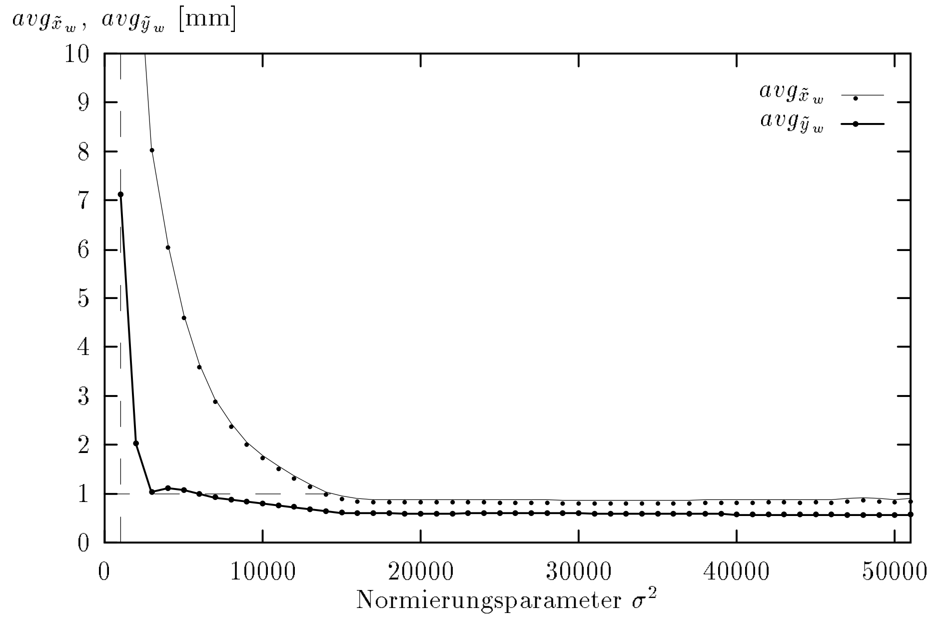


Abbildung 4.15: Die für den gesamten Arbeitsbereich gemittelten Fehler $avg_{\tilde{x}_w}$ und $avg_{\tilde{y}_w}$ bei unterschiedlichen Normierungsparametern.

Verkürzung in x_b-Richtung, bedingt durch die spezielle Blickrichtung der Kameras. Es ergibt sich nämlich wegen des Zusammenschiebens der Computerbildpunkte der Trainingsmenge in x_b-Richtung für diese Punkte eine schlechtere Auflösung. Also können Punkte in y_b-Richtung besser unterschieden werden als in x_b-Richtung, und es ergeben sich in y_w-Richtung geringere Approximationsfehler. Es wurde allerdings erwartet, daß sich ein wesentlich höherer Unterschied der Fehler für beide Richtungen ergibt und sich die Fehler beider Kurven ab einem bestimmten Normierungsparameter jeweils auf das Minimum einpendeln. Der Verlauf der Fehlerkurven zeigt jedoch, daß sich die Differenz der Fehler bei Erhöhung des Normierungsparameters verringert und somit in beide Richtungen ein annähernd gleicher Fehler erreicht werden kann[4]. Dieser Effekt tritt vermutlich wegen einer besonderen Eigenschaft von RBF-Netzen, der sogenannten *Hochgenauigkeit* (Hyperacuity) [Pog92], auf[5] und wird nicht weiter analysiert.

Für die Anwendung von drei RBF-Netzen, die mit unterschiedlichen Normierungsparametern konfiguriert wurden, werden die absoluten Fehler an Positionen des Arbeitsbereiches gemäß der Fehlerfunktionen in (4.17) durch die Graphiken der Abbildung 4.16 veranschaulicht.

---

[4]Bei $\sigma^2 = 3000$ ergibt sich für die Fehlerdifferenz $avg_{\tilde{x}_w} - avg_{\tilde{y}_w} = 8.1 - 1.0 = 7.1$, während sie bei $\sigma^2 = 20000$ bereits auf $avg_{\tilde{x}_w} - avg_{\tilde{y}_w} = 0.9 - 0.6 = 0.3$ gesunken ist.

[5]Durch einen hohen Normierungsparameter wird die Approximation in der perspektivisch verkürzten x_w-Richtung verbessert. Auch beim Menschen tritt der Effekt dadurch auf, daß er in der Lage ist, zwei dicht zusammen liegende Punkte noch zu unterscheiden, obwohl der Abstand dieser Punkte geringer ist als der Abstand zweier Zapfen auf der Retina des Auges.

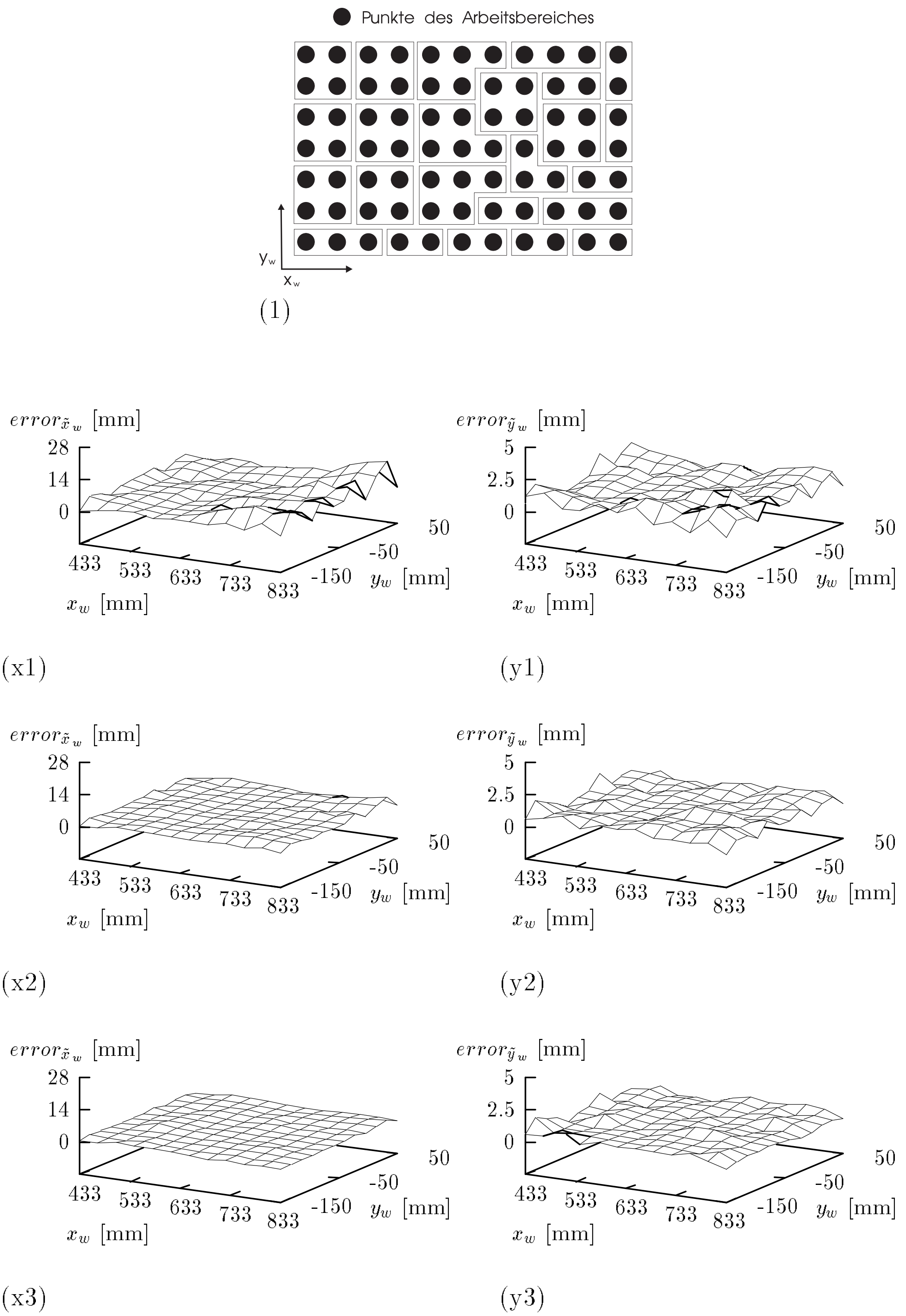


Abbildung 4.16: Anordnung der Trainingsdaten und der Cluster (1), Fehler in der Ausgabe $\tilde{x}_w$ bzw. $\tilde{y}_w$ für Normierungsparameter $\sigma^2 = 5000$ (x1) bzw. (y1), $\sigma^2 = 10000$ (x2) bzw. (y2) und $\sigma^2 = 15000$ (x3) bzw. (y3).

Die resultierenden minimalen, maximalen und gemittelten Fehler bei der Anwendung von fünf unterschiedlich konfigurierten Netzen werden in den Tabellen 4.2 und 4.3 dargestellt. Die in Abb. 4.16 veranschaulichten Fälle sind in den Tabellen markiert.

σ^2	$min_{\tilde{x}_w}$ [mm]	$max_{\tilde{x}_w}$ [mm]	$avg_{\tilde{x}_w}$ [mm]	als Graphik
5000	0.08	27.13	4.67	(x1)
10000	0.03	9.38	1.80	(x2)
15000	0.01	3.23	0.96	(x3)
20000	0.00	2.94	0.89	
25000	0.01	2.99	0.88	

Tabelle 4.2: Minimaler, maximaler und gemittelter Approximationsfehler in der Ausgabe $\tilde{x}_w$ für unterschiedliche Normierungsparameter.

σ^2	$min_{\tilde{y}_w}$ [mm]	$max_{\tilde{y}_w}$ [mm]	$avg_{\tilde{y}_w}$ [mm]	als Graphik
5000	0.00	4.37	1.07	(y1)
10000	0.00	3.04	0.80	(y2)
15000	0.00	2.35	0.62	(y3)
20000	0.00	1.98	0.59	
25000	0.03	1.94	0.60	

Tabelle 4.3: Minimaler, maximaler und gemittelter Approximationsfehler in der Ausgabe $\tilde{y}_w$ für unterschiedliche Normierungsparameter.

In Abschnitt 4.1 waren die Eingabevektoren der Trainingsmenge äquidistant, und es konnte experimentell gezeigt werden, daß sich der optimale Normierungsparameter algorithmisch berechnen läßt. Bei der Stereokamera-Manipulator-Kalibrierung sind die Vektoren jedoch ungleichmäßig verteilt, und es wurde der optimale Normierungsparameter nur experimentell bestimmt.
Wegen der unterschiedlichen Fehler in x_w- und y_w-Richtung kann vermutet werden, daß die Verwendung eines HBF-Netzes bei dieser Kalibrierung sinnvoller wäre, da für jede Dimension der Clusterzentren ein Normierungsparameter existieren würde. Der Normierungsparameter $\left(\sigma_j^{(k)}\right)^2$ der k-ten Komponente eines Clusterzentrums $\vec{c}_j$ könnte durch die Varianz für die k-ten Komponenten der Vektoren des Clusters bestimmt werden.

4.2.3 Gruppierte Eingaben ohne Cluster-Algorithmus

In Abschnitt 4.2.2 wurde eine Kalibrierung durch ein RBF-Netz vorgenommen, dessen Clusterzentren durch Gruppierung der Eingabevektoren mit einem Cluster-Algorithmus entstanden. Es soll in diesem Abschnitt nun analysiert werden, wie sich eine triviale Selektion von Ausgabevektoren und anschließender Übertragung dieser Selektion auf die Eingabevektoren auf den Approximationsfehler auswirkt. Diese Alternative erspart den Aufwand der Gruppierung mit einem Cluster-Algorithmus, da die Ausgabevektoren äquidistant sind und systematisch angeordnete Clusterzentren selektiert werden können.
Die vorhergehenden Versuche haben gezeigt, daß bei der Approximation von Funktionen durch RBF-Netze insbesondere die Anzahl der ausgelassenen äußeren Schichten der Trainingsdaten eine entscheidende Rolle spielt und sich signifikant auf den Aproximationsfehler auswirkt. Es ist zu beachten, daß bei der Eingabe der vierdimensionalen Bildvektoren nicht eindeutig die äußeren Schichten festgelegt werden können, was bei dreidimensionaler Eingabe durchaus möglich war (Abb. 4.3). Es können jedoch äußere Schichten der Ausgabevektoren festgelegt und auf die Eingabevektoren übertragen werden.

Die Zentren der Trainingsdaten sollen nun bestimmt werden, indem zunächst dreidimensionale (äquidistante) Ausgabevektoren systematisch selektiert und danach auf die entsprechenden vierdimensionalen (nicht äquidistanten) Eingabevektoren übertragen werden.

Der nächste Versuch analysiert den Approximationsfehler bei einer Stereokamera-Manipulator-Kalibrierung durch ein RBF-Netz, dessen Clusterzentren durch Selektion der Ausgabevektoren entstanden. Um die resultierenden Approximationsfehler mit denen aus Abschnitt 4.2.2 vergleichen zu können, mußten gleiche Randbedingungen geschaffen werden. Der Arbeitsbereich X und die Trainingsmenge wurden somit wie in Abschnitt 4.2.2 gewählt. Für die Clusterzentren-Bildung wurden jedoch systematisch angeordnete Ausgabevektoren der Trainingsmenge gewählt und auf die Eingabevektoren übertragen. Um die gleiche Anzahl an Basisfunktionen wie in Abschnitt 4.2.2 zu erhalten, wurden die Clusterzentren der Ausgabedaten wie in Abb. 4.17 gewählt (Anzahl an Basisfunktionen= $N = 6 \cdot 4 = 24$) und dann auf die Eingabedaten übertragen. Der Effekt der perspektivischen Verkürzung in x_b-Richtung der Computerbilder zeigt sich in Abb. 4.18.

Hierfür werden die Summen der Abstände von vier benachbarten Clusterzentren auf den Seiten des Arbeitsbereiches in x- und y-Richtung des Computerbildes miteinander verglichen. Diese Summen werden für vier Clusterzentren der vorderen und der hinteren Seite in x-Richtung des Computerbildes mit Δ_{x_v} und Δ_{x_h} bezeichnet. Die Bezeichnungen Δ_{y_l} und Δ_{y_r} stellen die Summen für vier Clusterzentren auf der linken und der rechten Seite in y-Richtung des Computerbildes dar. Durch die Projektion ergibt sich $\Delta_{y_l} \approx \Delta_{y_r} < \Delta_{x_v}, \Delta_{x_h}$. Deshalb ist die Auflösung in y_b-Richtung besser als in x_b-Richtung. Es wird somit in y_w-Richtung des Arbeitsbereiches ein geringerer Approximationsfehler erwartet als in x_w-Richtung.

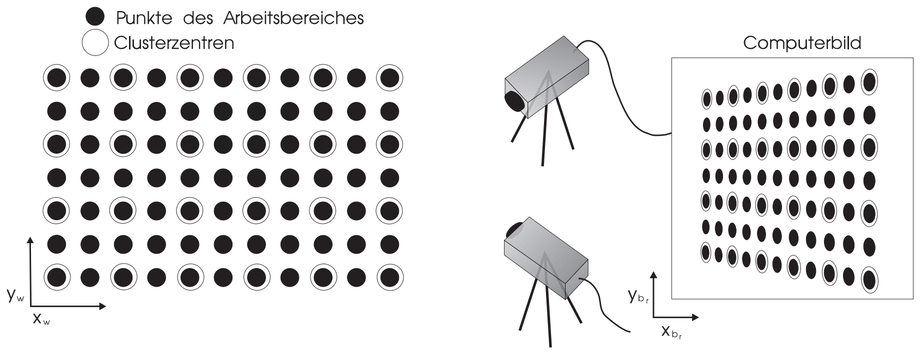


Abbildung 4.17: Systematische Anordnung der 24 Clusterzentren.

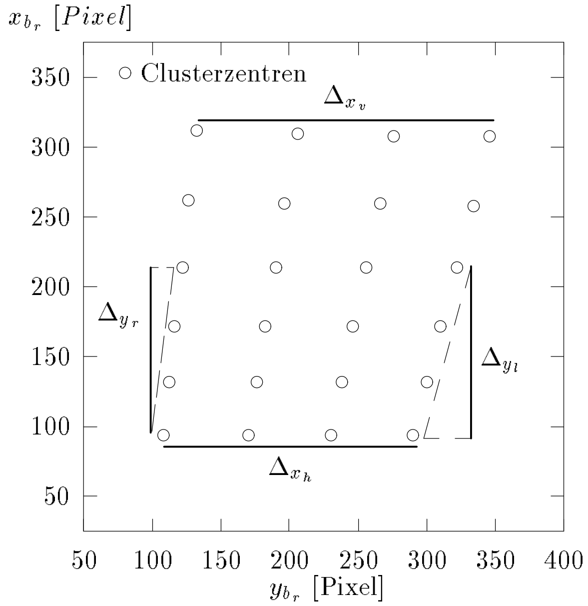


Abbildung 4.18: Projektion der Clusterzentren im Computerbild der rechten Kamera.

Wie in Abschnitt 4.2.2 konnten nun die Gewichte w_{j_x}, w_{j_y} und w_{j_z}, $j = 1, \ldots, N$, berechnet und das RBF-Netz trainiert werden.
Die Abschätzung der Güte des Netzes erfolgte schließlich ebenso wie in 4.2.2, indem der Manipulator äquidistante Punkte des Arbeitsbereiches (Abstand= 25 mm) anfuhr und die Computerbildkoordinaten der Lichtquelle ermittelt und als Eingabe für das Netz verwendet wurden. In jedem Schritt erfolgte eine Bestimmung der Approximationsfehler nach (4.17). Weiterhin wurden jeweils in x_w- und y_w-Richtung die minimalen, die maximalen und die gemittelten Fehler des gesamten Arbeitsbereiches ermittelt.
Die resultierenden gemittelten Fehler für RBF-Netze, die mit unterschiedlichen

Normierungsparametern konfiguriert wurden, sind in Abb. 4.19 dargestellt.
Aus der Abbildung geht hervor, daß in y_w-Richtung, wie erwartet, ein geringerer Fehler zu erreichen ist als in x_w-Richtung (perspektivische Verkürzung in x_b-Richtung).

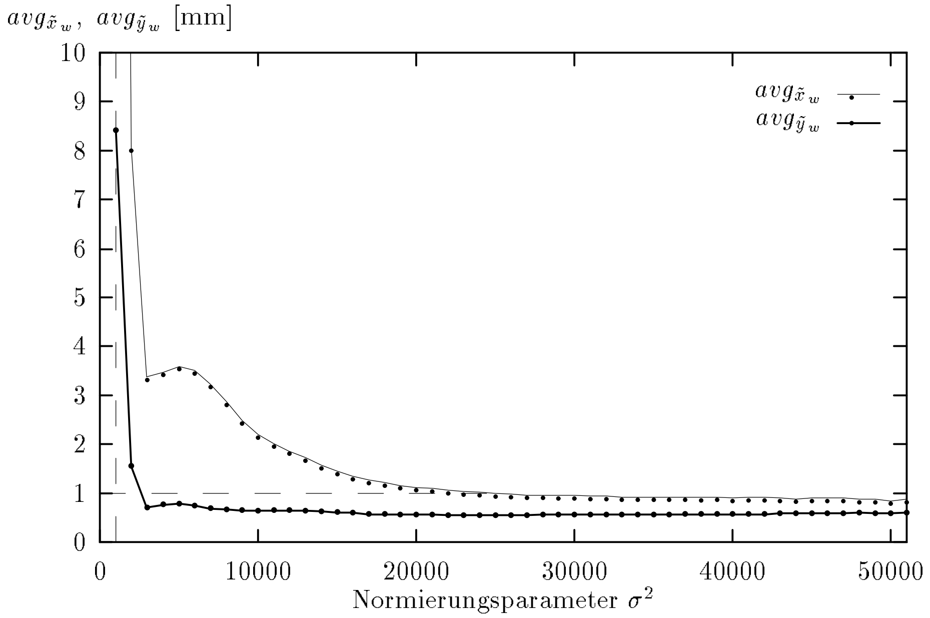


Abbildung 4.19: Die für den gesamten Arbeitsbereich gemittelten Fehler $avg_{\tilde{x}_w}$ und $avg_{\tilde{y}_w}$ bei unterschiedlichen Normierungsparametern.

Für die Anwendung von drei RBF-Netzen, die mit unterschiedlichen Normierungsparametern konfiguriert wurden, werden die absoluten Fehler an Positionen des Arbeitsbereiches gemäß der Fehlerfunktion in (4.17) durch die Graphiken der Abbildung 4.20 veranschaulicht. Die resultierenden minimalen, maximalen und gemittelten Fehler bei der Anwendung von fünf unterschiedlich konfigurierten Netzen werden in den Tabellen 4.4 und 4.5 dargestellt. Die in Abb. 4.20 veranschaulichten Fälle sind in den Tabellen markiert.
Um nun zu analysieren, ob die in diesem Abschnitt beschriebene Methode des Trainierens eines Netzes mit Selektion der Ausgabedaten sinnvoll ist, werden die Tabellen 4.4 und 4.5 und die Abb. 4.19 mit den Tabellen 4.2 und 4.3 und der Abb. 4.15 des letzten Abschnitts verglichen. Es ist zu erkennen, daß bei Vorgabe eines maximal zulässigen gemittelten Fehlers bei der Selektion der Ausgabedaten (ohne Cluster-Algorithmus) höhere Normierungsparameter erforderlich sind als bei der Gruppierung der Eingabedaten (mit Cluster-Algorithmus)[6]. Die Minima der gemittelten Fehler unterscheiden sich nur geringfügig[7].

[6] Für einen maximal zulässigen Fehler von 1 mm werden die Normierungsparameter $\sigma^2 = 25000$ bei der Selektion der Ausgabedaten und $\sigma^2 = 15000$ bei der Gruppierung der Eingabedaten benötigt.

[7] Es ergeben sich für die Differenzen der Fehler $avg_{\tilde{x}_w}$ und $avg_{\tilde{y}_w}$ in beiden Verfahren $0.88 - 1.00 = -0.12$ und $0.60 - 0.55 = 0.05$ (bei $\sigma^2 = 25000$).

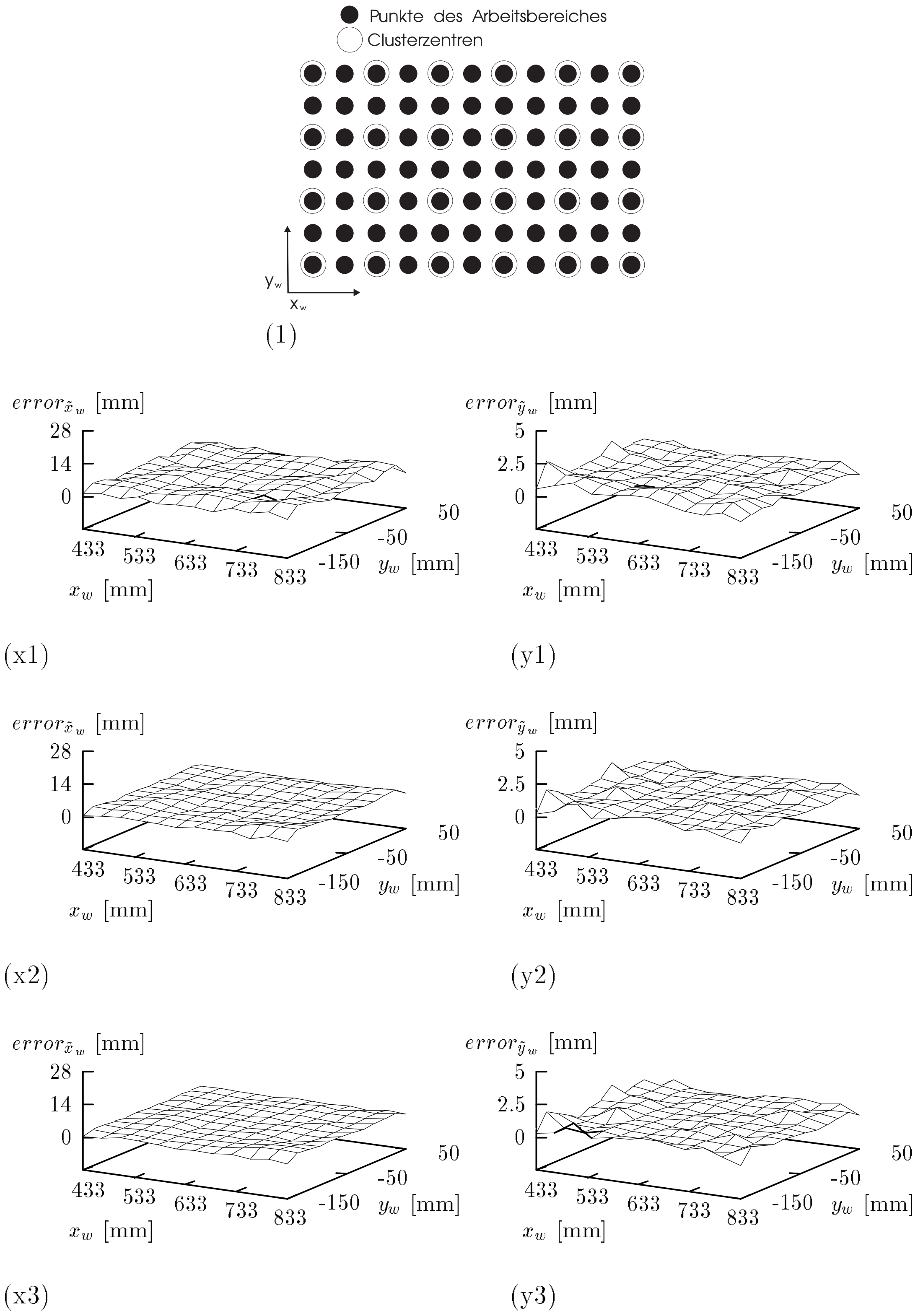


Abbildung 4.20: Systematische Anordnung der auf die Eingabevektoren zu übertragenden 24 Clusterzentren (1) und die Fehler in den Ausgaben $\tilde{x}_w$ bzw. $\tilde{y}_w$ für diese Cluster mit Normierungsparametern $\sigma^2 = 5000$ (x1) bzw. (y1), $\sigma^2 = 10000$ (x2) bzw. (y2) und $\sigma^2 = 15000$ (x3) bzw. (y3).

σ^2	$min_{\tilde{x}_w}$ [mm]	$max_{\tilde{x}_w}$ [mm]	$avg_{\tilde{x}_w}$ [mm]	als Graphik
5000	0.00	13.89	3.60	(x1)
10000	0.02	10.57	2.20	(x2)
15000	0.01	8.57	1.45	(x3)
20000	0.01	6.88	1.13	
25000	0.00	5.71	1.00	

Tabelle 4.4: Minimaler, maximaler und gemittelter Approximationsfehler in der Ausgabe $\tilde{x}_w$ für unterschiedliche Normierungsparameter.

σ^2	$min_{\tilde{y}_w}$ [mm]	$max_{\tilde{y}_w}$ [mm]	$avg_{\tilde{y}_w}$ [mm]	als Graphik
5000	0.00	3.31	0.78	(y1)
10000	0.02	2.47	0.65	(y2)
15000	0.00	2.25	0.61	(y3)
20000	0.00	2.25	0.56	
25000	0.01	2.20	0.55	

Tabelle 4.5: Minimaler, maximaler und gemittelter Approximationsfehler in der Ausgabe $\tilde{y}_w$ für unterschiedliche Normierungsparameter.

Der Grund für diese scheinbar sinnvolle Selektion der Ausgabedaten liegt darin, daß es sich bei der zu lernenden Funktion um eine annähernd lineare Funktion handelt, bei der ähnliche Ausgabedaten ähnliche Werte in den einzelnen Komponenten der Eingabedaten implizieren.
Weiterhin muß beachtet werden, daß hier die äußeren Clusterzentren weiter am Rand des Arbeitsbereiches liegen als in Abschnitt 4.2.2 (Abb. 4.17 und 4.13). Durch diese spezielle Anordnung wird das in Abschnitt 4.1.3 beschriebene Randproblem behoben. Die Güte der Approximation kann im letzten Abschnitt also noch verbessert werden, indem eine weitere Schicht der Trainingsdaten ausgelassen wird.

4.2.4 Kalibrierung durch dynamisches Template-Matching

Bisher wurde die Kalibrierung durch die Benutzung einer Lichtquelle am Ende des Robotergreifers stark vereinfacht, da somit bei nahezu geschlossener Blende die Computerbildkoordinaten mit einem Maximumoperator leicht ermittelt werden konnten. Um auf eine Verwendung von Lichtquellen verzichten zu können, müssen die Koordinaten des Greifers auf eine andere Art ermittelt werden. Eine

Lösung dieses Problems besteht darin, die Koordinaten durch *Template-Matching* zu berechnen. Hierfür fährt der Manipulator gemäß Abb. 4.21 mäanderförmig in seinem Arbeitsbereich äquidistante Punkte an, damit die Trainingsmenge erzeugt werden kann.
Ein *Kalibrierungsschritt* bestehe aus dem Anfahren eines solchen Punktes, dem Aufnehmen der Computerbilder durch die beiden Kameras und der Koordinatenbestimmung der Greiferspitze in diesen Bildern. Der Bildausschnitt des Greifers in der Startposition (0-ter Schritt) vor der Kalibrierung (Abb. 4.21) wird als Template verwendet, das beim nächsten Schritt mit dem neuen Bild gematcht wird, um dort erneut den Greifer zu lokalisieren. Der Ausschnitt um den lokalisierten Greifer wird dann als nächstes Template bestimmt. Der mäanderförmige Pfad ist wichtig, damit sich von einem Schritt zum nächsten in den Templates keine signifikanten Änderungen in der Umgebung der Greiferspitze ergeben. Andernfalls könnte die Lokalisierung nachteilig beeinträchtigt werden.

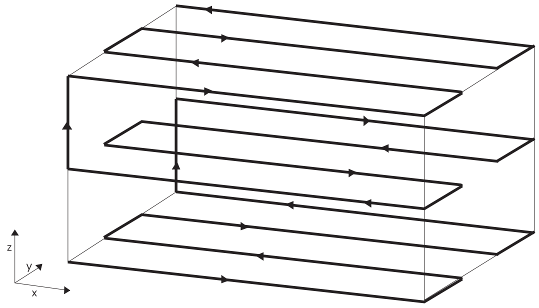


Abbildung 4.21: Der Weg des Roboters bei der Kalibrierung der Kameras.

Seien die Grauwerte eines Templates im i-ten Kalibrierungsschritt beschrieben durch folgende Funktion:

$$g_{T_i} : \{0, \ldots, B-1\} \times \{0, \ldots, H-1\} \longrightarrow \{0, \ldots, 255\}$$

Dabei gibt B die Breite, H die Höhe des Templates (jeweils in Pixeln) und $g_{T_i}(x, y)$ den Grauwert im Template an der Position (x, y) an.

Sei weiterhin die Auflösung des Computerbildes einer Kamera $I \times I$ Pixel und (x_i, y_i) die Position der linken oberen Ecke des Templates im $i-ten$ Schritt, wobei die Grauwerte des Computerbildes definiert seien durch folgende Funktion:

$$g_B : \{0, \ldots, I-1\} \times \{0, \ldots, I-1\} \longrightarrow \{0, \ldots, 255\}$$

Dann wird die Position der linken oberen Ecke (x_{i+1}, y_{i+1}) des gesuchten Templates im nächsten Schritt wie folgt bestimmt:

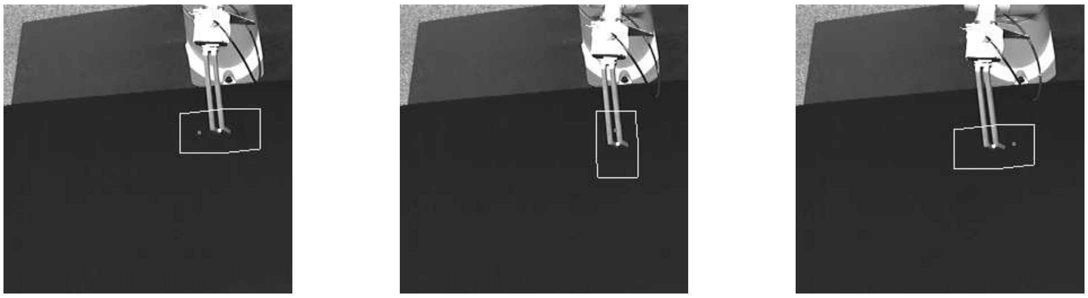
$$
(x_{i+1}, y_{i+1}) = \arg \left\{ \min_{\substack{x_B, y_B \\ \in \{0,\dots,I-1\}}} \{SSD_i\,(x_B, y_B)\} \right\}, \qquad (4.18)
$$

$$
SSD_i\,(x_B, y_B) = \sum_{x_T=0}^{B-1} \sum_{y_T=0}^{H-1} \left(g_{T_i}\,(x_T, y_T) - g_B\,(x_B + x_T, y_B + y_T)\right)^2
$$

Der Wert $SSD_i(x_B, y_B)$ (summed squared differences) beschreibt das Ergebnis des dynamischen Matchings eines Bildausschnittes an der Position (x_b, y_B) (linke obere Ecke) mit dem Template des i-ten Schrittes. Hierfür wird lediglich die Summe der quadratischen Grauwertdifferenzen in den Templates bestimmt. Die Grauwerte $g_{T_{i+1}}\,(x, y)\,,\ x \in \{0, \dots, B-1\}\,,\ y \in \{0, \dots, H-1\}$, des neu bestimmten Templates ergeben sich damit wie folgt:

$$
g_{T_{i+1}}\,(x, y) = g_B\,(x_{i+1} + x, y_{i+1} + y) \qquad (4.19)
$$

Da eine Bestimmung des neuen Templates durch Matching im gesamten Bild nach (4.18) sehr aufwendig ist, ist es sinnvoll, einen bestimmten Bereich im Bild vorherzusagen, in dem sich der Greifer beim nächsten Schritt befinden wird (Abb. 4.22).



(a) (b) (c)

Abbildung 4.22: Lokalisierung des Greifers ohne Lichtquelle bei drei aufeinanderfolgenden Schritten durch Bewegung in y- (a), x- (b) und $(-y)$-Richtung (c) des Weltkoordinatensystems. Das graue Quadrat markiert jeweils die Position der Greiferspitze im letzten Schritt, die weiße Umrandung den Matching-Bereich und das weiße Quadrat die neue Position der Greiferspitze.

Hierfür wird zunächst jeweils ein Schritt in x-, y- und z-Richtung mit *Ganzbild-Matching* durchgeführt. Aus den ermittelten Template-Positionen (x_x, y_x), (x_y, y_y) und (x_z, y_z) und der vorgegebenen Template-Position (x_0, y_0) des 0-ten Schritts (jeweils die linke obere Ecke) werden dann Richtungsvektoren $\vec{r}_x = (x_x - x_0, y_x - y_0)$, $\vec{r}_y = (x_y - x_0, y_y - y_0)$ und $\vec{r}_z = (x_z - x_0, y_z - y_0)$ berechnet, die angeben, in welcher Richtung von (x_i, y_i) beim $(i+1)$-ten Schritt der Punkt (x_{i+1}, y_{i+1}) sich ungefähr befindet, wenn sich der Greifer dabei in x-, y- oder z-Richtung des Weltkoordinatensystems bewegt. Dadurch läßt sich für jeden Schritt ein *Matching-Bereich* festlegen, so daß beim Kalibrieren ein *Teilbild-Matching* durchgeführt werden kann.

4.3 Diskussion

In diesem Kapitel wurde gezeigt, daß RBF-Netze für eine neuronale Kamerakalibrierung verwendet werden können, wobei nur sehr geringe Approximationsfehler enstehen (weniger als 1 Pixel bei der Manipulator-Einzelkamera-Kalibrierung und weniger als 1 mm bei der Stereokamera-Manipulator-Kalibrierung). Eine Verringerung des Approximationsfehlers läßt sich erreichen durch:

- Erhöhung der Clusterzentrendichte
- Erhöhung der Trainingsdatenanzahl
- Erhöhung der Anzahl ausgelassener äußerer Schichten der Trainingsdaten zur Behebung des Randproblems
- Optimale Wahl des Normierungsparameters
- Verwendung eines normierten RBF-Netzes

In den Abschnitten 4.2.2 und 4.2.3 wurde beschrieben, wie mit Hilfe neuronaler Netze die Funktionen $f_1 : \mathbb{R}^3 \longrightarrow \mathbb{R}^2$ für die Manipulator-Einzelkamera-Kalibrierung und $f_2 : \mathbb{R}^4 \longrightarrow \mathbb{R}^3$ für die Stereokamera-Manipulator-Kalibrierung approximiert werden können. Beide Funktionstypen können also zur neuronalen Kamerakalibrierung verwendet werden. Eine Möglichkeit besteht darin, mit zwei Funktionen vom Typ f_1 zu jedem Weltkoordinatenvektor der Spitze des Greifers jeweils die Computerbildkoordinaten einer Kamera zu berechnen. Die Berechnung der Weltkoordinaten der Greiferspitze aus den Computerbildkoordinaten der beiden Kameras ermöglicht die Funktion f_2.
Für beide Verfahren der Kalibrierung wurden Netze mit radialsymmetrischen Basisfunktionen verwendet. Die unter KHOROS implementierten Routinen zur Manipulator-Einzelkamera-Kalibrierung sind in Anhang A dargestellt.
Im nächsten Kapitel soll eine Objekterkennungsfunktion mit einem neuronalen Netz gelernt werden. Da jedoch die Eingabevektoren hochdimensional sind und sich in den einzelnen Komponenten der Cluster signifikante Abweichungen vom Clusterzentrum ergeben können, ist das elementare RBF-Netz mit einheitlichem Normierungsparameter hierfür ungeeignet. Ein HBF-Netz dagegen zeichnet sich dadurch aus, daß es wegen der variablen Normierungsparameter Berechnungen mit hochdimensionalen Eingabevektoren ausführen kann.

Kapitel 5

Objektlokalisierung und -erkennung

In diesem Kapitel werden verschiedene Ansätze des Trainierens eines neuronalen Netzes zur *Objektlokalisierung* und *-erkennung* untersucht. Die Begriffe Lokalisierung und Erkennung werden dabei wie folgt definiert:

Lokalisierung: Seien ein Objekt mit seiner Beschreibung[1] und ein Testbild gegeben, dann versteht man unter der Objektlokalisierung in dem Bild die Bestimmung derjenigen Bildposition, dessen umgebendes Bildmuster am besten mit der Erscheinung des Objektes übereinstimmt. Als Lokalisierungsposition kann z.B. der Schwerpunkt des Objektes bestimmt werden.

Erkennung: Seien ein Objekt mit seiner Beschreibung, ein Testbild und eine Position, an der das Objekt lokalisiert wurde, gegeben, dann versteht man unter der Objekterkennung die Entscheidung, ob es sich hierbei um das Objekt handelt oder nicht.

In der vorliegenden Arbeit wird mit einem neuronalen Netz eine Erkennungsfunktion gelernt, mit der die Lokalisierung und Erkennung eines Zielobjektes ermöglicht wird. Mit der Eingabe des Netzes kann ein beliebiger Bildbereich beschrieben werden. Die Ausgabe liefert einen *Zuverlässigkeitswert*, durch den entschieden wird, ob sich das Zielobjekt in dem eingegebenen Bildbereich befindet oder nicht. Es werden folgende aufgabenorientierte Lernstrategien unterschieden:

- Lernen des Objektes mit Umgebung (*LmU-Lernstrategie*)
- Lernen des Objektes ohne Umgebung (*LoU-Lernstrategie*)

[1] Im Gegensatz zur häufig verwendeten symbolischen Beschreibung eines Objektes durch seine geometrischen Eigenschaften [Gri90] wird hier eine subsymbolische Beschreibung durch seine sogenannten *photometrischen* Eigenschaften gelernt.

Das Lernen eines Objektes ohne seine Umgebung ist z.B. erwünscht, wenn es durch einen Manipulator lediglich angenähert werden soll. Hierfür ist die Lokalisierung des Objektes im Computerbild erforderlich, unabhängig davon, ob sich der Manipulator bereits in der Umgebung des Objektes befindet.
Soll der Manipulator jedoch in eine Greifstellung gebracht werden, muß auch die Umgebung des Zielobjektes gelernt werden. Hierbei kann bei einer schrittweisen Bewegung des Manipulators nach jedem Schritt entschieden werden, ob bereits die optimale (gelernte) Greifstellung erreicht wurde.

In den folgenden Abschnitten werden zunächst die klassischen Methoden und der eigene Ansatz dieser Arbeit für eine Objekterkennung im Überblick erläutert. Die weiteren Abschnitte beschreiben dann detailliert das Trainieren und Anwenden neuronaler Netze zur Objektlokalisierung und -erkennung.
Weiterhin werden das Diskriminanz- und das Invarianzkriterium untersucht, um zu entscheiden, ob die für ein Objekt gelernte Erkennungsfunktion dieses von anderen Objekten unterscheiden kann und ob sie für unterschiedliche Erscheinungen (s. Kapitel 1) des Zielobjektes gleiche Zuverlässigkeitswerte berechnet.

5.1 Literaturübersicht

Die in diesem Kapitel beschriebene Objekterkennung wird durch zwei Ansätze einiger bekannter Arbeiten motiviert. Ein Verfahren bei der Objekterkennung ist die sogenannte *Attention*, ein weiterer hier verwendeter Ansatz ist die *ansichtenbasierte Beschreibung eines Objektes*. Die beiden folgenden Abschnitte beschreiben grob diese Objekterkennungsmethoden, die später in ähnlicher Weise benutzt werden.

5.1.1 Ansichtenbasierte Beschreibung eines Objektes

Bei der klassischen Objekterkennung werden Objekte symbolisch durch ihre geometrischen Eigenschaften beschrieben [Gri90]. Diese Beschreibungen geben die Form des Objektes im dreidimensionalen Raum an. Eine umfassende Beschreibung eines Objektes erfordert jedoch dessen Betrachtung unter möglichst vielen variierenden Bedingungen (z.B. Änderung der Lichtverhältnisse oder der Orientierung des Objektes). Nur unter dieser Voraussetzung können Zusammenhänge zwischen geometrischen Eigenschaften des Objektes im 3D-Raum und im zweidimensionalen Computerbild einer Kamera hergestellt werden. So bewirkt z.B. die Veränderung der Orientierung eines Objektes und seiner Entfernung zu einer Kamera die geometrische Form des Objektes in dem von der Kamera aufgenommenen zweidimensionalen Computerbild.
Selbst wenn die Bestimmung der geometrischen Form eines Objektes in einem

Computerbild einfach ist, ergeben sich aufgrund der Effekte der perspektivischen Projektion durch die Kamera und aufgrund der Mannigfaltigkeit an unterschiedlichen Bedingungen Schwierigkeiten bei der Beschreibung des Objektes mit dieser Methode.

In [Mur94] wird für die Objekterkennung ein verbesserter Ansatz versucht. Hier erfolgt die Beschreibung eines Objektes durch seine photometrischen Eigenschaften, d.h. es wird die sogenannte Erscheinung eines Objektes verwendet, die sich aus den Grauwerten im Computerbild ergibt. Die Erscheinung wird zwar durch unterschiedliche Bedingungen wie Beleuchtung, Orientierung des Objektes und Abstand des Objektes zur Kamera beeinflußt, jedoch werden zur Beschreibung lediglich die zweidimensionalen Grauwertbilder benötigt. Die aufwendige Beschreibung im 3D-Raum entfällt also. Wird allerdings die Erscheinung eines Objektes durch einen Vektor seiner Grauwerte dargestellt, so ergeben sich für den Vektor sehr hohe Dimensionen. Die Beschreibung eines Objektes erfolgt hier deshalb durch die Projektion der unterschiedlichen Erscheinungen auf Punkte in einem sogenannten *Eigenraum*. Die Vektoren dieses Raumes sind verglichen mit den Grauwertvektoren des Objektes niedrig-dimensional.

Es ist zu beachten, daß wegen der Mannigfaltigkeit an unterschiedlichen Erscheinungen eine ausreichend große Auswahl für die Beschreibung des Objektes notwendig ist. Auf die Methoden der Projektion der Erscheinungen eines Objektes in den Eigenraum wird hier nicht weiter eingegangen, da ein anderer Ansatz angestrebt wird. Dieser Ansatz verwendet jedoch ebenfalls die photometrischen Eigenschaften des Objektes statt der geometrischen.

In diesem Kapitel wird ein Objekt durch Vektoren beschrieben, die sich ergeben, indem auf die Bildbereiche, die unterschiedliche Erscheinungen des Objektes darstellen, *Operatoren* angewendet werden. Mit solch einer *Filterung* (=Anwendung eines Operators) sollen zufällige Störungen in den Erscheinungen des Objektes eliminiert werden, so daß die zur Beschreibung des Objektes benötigten wichtigen Informationen übrigbleiben. Es ergibt sich dadurch für jede Komponente des Vektors eine sogenannte *Filterantwort*. Die Dimension der Vektoren bleibt also bei der Beschreibung des Objektes erhalten. Da es sich dabei in der Regel um hochdimensionale Vektoren handelt, soll versucht werden, eine Erkennungsfunktion für ein Objekt mit einem HBF-Netz (s. Kapitel 3, Abschnitt 3.3) zu lernen. Die Eingabevektoren der Trainingsmenge werden dabei aus den Filterantworten gebildet, die sich durch Anwenden eines Operators auf die Bildbereiche mit den unterschiedlichen Erscheinungen des Objektes ergeben. Eine Beschreibung der möglichen Operatoren erfolgt in Abschnitt 5.3.

Wie in Kapitel 4 zur neuronalen Kamerakalibrierung findet auch hier eine Generalisierung der Trainingsdaten statt. Dadurch erfolgt ähnlich wie in [Mur94] eine Verringerung der Dimension bei der Beschreibung des Objektes.

5.1.2 Schnelle Lokalisierung durch Attention

Eine wesentlich effizientere Objektlokalisierung aufgrund einer Verringerung des Aufwandes ergibt sich durch die Verwendung sogenannter *attentiver* Verarbeitungsstrategien. Motiviert wird diese Methode durch psychologische Untersuchungen, die bewiesen, daß das visuelle System beim Menschen durch Blickbewegungsstrategien bei einer Objektsuche nur die visuell auffälligen Bildbereiche lokalisiert [Koch85]. Grundlage dieser Attention sind visuelle Merkmale wie Bewegung, Farbe, Orientierung, Kanten etc.
In [Her94] wird ein simulierter *Attentionsmechanismus* beschrieben, der für die Aufgabe einer Gesichtserkennung in Gesichtsbildern die visuell auffälligen Bildbereiche (Mund, Nase, Augen, Ohren) lokalisiert und erkennt. Dabei erfolgt die Lokalisierung und Verarbeitung dieser Bereiche sequentiell. Die Effizienz dieser Objektsuche wird dadurch erreicht, daß irrelevante Bildbereiche nicht verarbeitet werden müssen.
In diesem Kapitel werden zur Objekterkennung zwei attentive Verarbeitungsstrategien verwendet, die unmittelbar mit der Objektlokalisierung und -erkennung zusammenhängen. Sie unterscheiden sich dadurch voneinander, daß sie einen unterschiedlichen Umfang an Wissen über die Erscheinung des Zielobjektes mit einbeziehen.
Die erste Strategie verwendet typische Merkmale des gefilterten Bildes und liefert eine sehr grobe Lokalisierung des Objektes, indem sie einen visuell auffälligen Bildbereich, den sogenannten *Aufmerksamkeitsbereich*, ermittelt. Bei der anderen Strategie wird wesentlich mehr Wissen über die Erscheinung des Zielobjektes verwendet. Hier werden typische Muster in dem gefilterten Bild betrachtet, und dadurch einige wenige *Lokalisierungspositionen* bestimmt. Es erfolgt hier eine wesentlich feinere Lokalisierung. Eine Beschreibung dieser Strategien liefert Abschnitt 5.2.2.

5.2 Eigener Ansatz im Überblick

Im letzten Abschnitt wurden Ansätze aus bisherigen Arbeiten zur Objekterkennung beschrieben. Das Diagramm in der Abb. 5.1 zeigt den groben Ablauf des Systems zur Objekterkennung mit den in dieser Arbeit vorgestellten Methoden, die durch die beschriebenen Ansätze der Literatur motiviert wurden.
Es muß beachtet werden, daß theoretisch die beiden Trainingsphasen für die Attention und für das Lernen der Objekterkennungsfunktion parallel erfolgen könnten. Für die Strategie der feinen Lokalisierung durch Attention werden jedoch Informationen benötigt, die sich erst beim Lernen der Erkennungsfunktion ergeben. Deshalb wird ein sequentieller Ablauf dieser Komponenten des 'offline'-Betriebes festgelegt.

In den folgenden Abschnitten werden alle Komponenten des Diagramms näher erläutert.

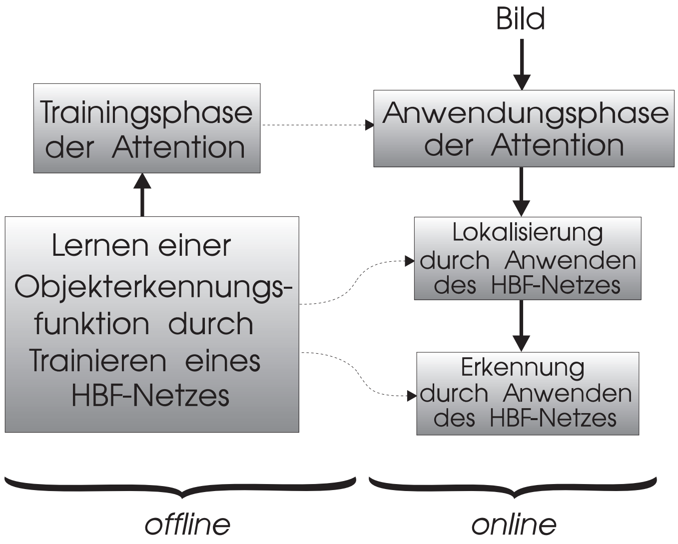


Abbildung 5.1: Ablaufdiagramm des Systems zur Objekterkennung.

5.2.1 Trainieren eines HBF-Netzes zum Lernen einer Objekterkennungsfunktion

Wie bei der neuronalen Kamerakalibrierung in Kapitel 4 soll auch für die Objektlokalisierung eine multidimensionale Funktion f durch ein Netz von Basisfunktionen approximiert werden. Die multidimensionale Eingabe stellt dabei einen Bildbereich dar, während der Zuverlässigkeitswert als eindimensionale Ausgabe die Wahrscheinlichkeit dafür angibt, ob sich das Zielobjekt in diesem Bereich befindet. Dazu ist das Erzeugen einer Trainingsmenge erforderlich, die das Zielobjekt beschreibt. Wie in Kapitel 3 werden dann in zwei Phasen die Basisfunktionen der 1. Schicht des Netzes bestimmt und anschließend die Gewichte der 2. Schicht gelernt.

Erzeugen einer Trainingsmenge

Für die Beschreibung eines Zielobjektes durch seine unterschiedlichen Erscheinungen ist es wichtig, dieses unter möglichst vielen variierenden Bedingungen zu betrachten. Z.B. beeinflussen Lichtverhältnisse die Erscheinung des Objektes sowie dessen Umgebung und Orientierung. Um diese Faktoren mit einzubeziehen, werden sogenannte *Trainingsbilder* von diesem Objekt erzeugt. Abb. 5.2 zeigt

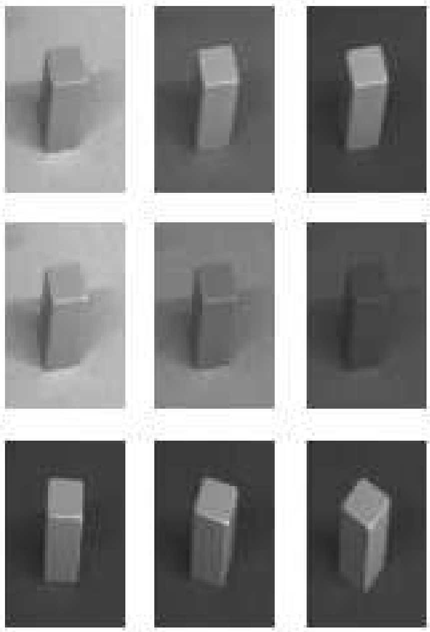
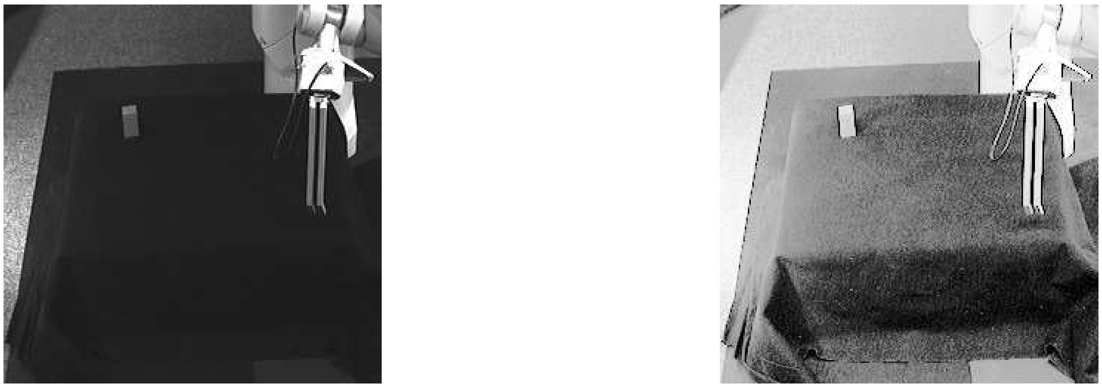


Abbildung 5.2: Objekt bei unterschiedlichen Hintergründen, Beleuchtungen und Orientierungen.

beispielhaft ein Objekt bei drei unterschiedlich hellen Hintergründen, drei verschiedenen Beleuchtungen und in drei verschiedenen Orientierungen.

Um das Objekt in jedem Bild stärker vom Hintergrund zu trennen, wird als Vorverarbeitung in allen Trainingsbildern eine Kontrasterhöhung durchgeführt (Abb. 5.3).



(a) (b)

Abbildung 5.3: Eine Aufnahme der rechten Kamera ohne (a) und mit Kontrasterhöhung (b).

Da sich für unterschiedliche Orientierungen des zu lokalisierenden Objektes verschieden große Silhouetten ergeben können, wird in den Trainingsbildern um das Objekt herum ein konstanter $(H \times B)$ Pixel großer Bereich festgelegt, in dem das Objekt für alle Erscheinungen sichtbar ist. Dieser Ausschnitt wird im folgenden als *Template* bezeichnet. Die Grauwerte in diesem Template an einer Position (x, y) ergeben sich aus folgender Funktion:

$$\begin{aligned} gw : D_x \times D_y &\longrightarrow \{0, \ldots, 255\}, \\ D_x = \{0, \ldots, B-1\}, \ D_y &= \{0, \ldots, H-1\} \end{aligned} \tag{5.1}$$

Um die unterschiedlichen Erscheinungen des Objektes auf eine einheitliche Struktur abzubilden, werden die Trainingsbilder mit einem geeigneten Operator gefiltert. Anschließend werden sogenannte *Trainings-Templates* durch den Bereich, in dem sich das Objekt befindet, bestimmt. Die Filterantworten an verschiedenen Positionen eines Trainings-Templates werden dabei durch folgende Funktion beschrieben:

$$g : D_x \times D_y \longrightarrow \mathbb{R}, \ D_x = \{0, \ldots, B-1\}, \ D_y = \{0, \ldots, H-1\} \tag{5.2}$$

Die zu lernende Funktion f lautet dann:

$$f : \mathbb{R}^{B \cdot H} \longrightarrow \mathbb{R}, \quad \begin{pmatrix} g(0,0) \\ \vdots \\ g(B-1, H-1) \end{pmatrix} \mapsto c \tag{5.3}$$

Die Eingabe des Netzes, ein $(B \cdot H)$-dimensionaler Vektor, stellt dabei die Filterantworten eines bestimmten Templates dar, während der angegebene Zuverlässigkeitswert c die Wahrscheinlichkeit angibt, mit der es sich bei dem Template um eine Darstellung des Zielobjektes handelt.
Das Trainieren geschieht nun durch Vorgabe einer Menge $\mathcal{T}$ von M Trainingsdaten, die $(B \cdot H)$-dimensionale Eingabevektoren $\vec{x}_i$ und zugehörige Ausgaben c_i, $i = 1, \ldots, M$, enthält. Die Eingabevektoren stellen dabei die Trainings-Templates des Zielobjektes dar und die Ausgaben die zugehörigen Zuverlässigkeitswerte. Da jedes Trainings-Template das Zielobjekt repräsentiert, müssen die Ausgaben $c_1 = \ldots = c_M = 1$ gesetzt werden.
Es ist zu beachten, daß die Eingabevektoren hier sehr hohe Dimensionen annehmen können[2], wodurch sich die Verwendung von HBF-Netzen besonders auszeichnet. Im Gegensatz zu anderen Netzwerkmodellen, wie z.B. Kohonenkarten [Mar92], sind HBF-Netze in der Lage, mit diesen hochdimensionalen Vektoren zu rechnen.

[2]Die Höhe des Objektes in Abb. 5.2 beträgt 70 und die Breite 50 Pixel, so daß sich für die Dimension der Vektoren hier 3500 ergibt.

Phase 1 (Bestimmung der Basisfunktionen)

Wegen der unendlichen Anzahl an unterschiedlichen Erscheinungen eines Objektes ist die Trainingsmenge typischerweise sehr groß. Deshalb ist eine Generalisierung für die Erzeugung der Basisfunktionen des HBF-Netzes notwendig, um Redundanzen zu vermeiden und damit den Rechenaufwand zu verringern. Hierfür werden die in der Trainingsmenge repräsentierten Trainings-Templates mit einem Cluster-Algorithmus aufgrund von Ähnlichkeiten zu N Clustern gruppiert. Das bedeutet, daß für die M Trainings-Templates, dargestellt durch Funktionen[3] $g_1, \ldots, g_M$, N *Clusterzentren-Templates* gebildet werden[4]. Die Clusterzentren-Templates werden durch folgende Funktionen g_{Cl_j}, $j = 1, \ldots, N$, dargestellt:

$$g_{Cl_j}(x, y) = \frac{1}{|C_j|} \sum_{i \in C_j} g_i(x, y) \tag{5.4}$$

Dabei sind die Indexmengen C_j, $j = 1, \ldots, N$, wie folgt definiert:

$$C_j \subseteq \{1, \ldots, M\}\,, \quad \dot{\bigcup_{j=1,\ldots,N}} C_j = \{1, \ldots, M\}$$

Für jede Position (x, y) eines Clusterzentren-Templates lassen sich nun die Varianz $var_{Cl}(x, y)$ und die Standardabweichung (Streuung) $\sigma_{Cl}(x, y)$ berechnen:

$$var_{Cl_j}(x, y) = \frac{1}{|C_j|} \sum_{i \in C_j} \left[g_i(x, y) - g_{Cl_j}(x, y)\right]^2, \; j = 1, \ldots, N \tag{5.5}$$

$$\sigma_{Cl_j}(x, y) = \sqrt{var_{Cl_j}(x, y)}, \; j = 1, \ldots, N \tag{5.6}$$

Hierbei wird im Gegensatz zum Cluster-Algorithmus aus Kapitel 3, Abschnitt 3.2.2, für jede Dimension r, $r = 1, \ldots, m$, eines Clusterzentrums die Varianz als quadratische mittlere Abweichung der r-ten Komponenten der Eingabevektoren von der r-ten Komponente des Clusterzentrums berechnet. Die Varianzen für alle Positionen eines Clusterzentren-Templates werden in einem sogenannten *Clustervarianzen-Template* veranschaulicht.
In Abschnitt 5.4.1 werden zwei unterschiedliche aufgabenorientierte Verfahren vorgestellt, die das Lernen der Erscheinung eines Objektes mit seiner Umgebung oder ohne Umgebung ermöglichen. Hierfür wird jeweils ein spezieller Operator bei der Filterung der Trainingsbilder verwendet. Die dort resultierenden

[3]Diese Funktionen sind vom gleichen Typ wie die Funktion g in (5.2).

[4]Die Anzahl der Cluster ist beim ISODATA-Cluster-Algorithmus aus Kapitel 3 variabel und muß nicht vorgegeben werden.

Clusterzentren- und Clustervarianzen-Templates werden in Abbildungen veranschaulicht.

Die schematische Darstellung eines HBF-Netzes zur Lokalisierung und Erkennung eines Objektes ist in Abb. 5.4 zu sehen.

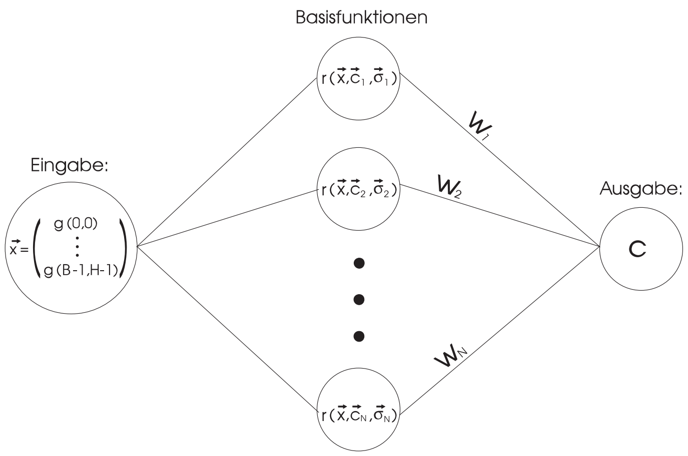


Abbildung 5.4: Schematischer Aufbau eines HBF-Netzes zum Lernen von Objekterkennungsfunktionen.

Ein Knoten der 1. Schicht soll auf jeden Eingabevektor $\vec{x}$ eine lokale Antwort erzeugen, die sich nur dann signifikant auf die Ausgabe auswirkt, wenn $\vec{x}$ nahe genug bei diesem Knoten liegt. Es werden hierfür folgende Basisfunktionen verwendet:

$$r_j = \exp\left[-\sum_{x=0}^{B-1}\sum_{y=0}^{H-1}\frac{\left[g(x,y)-g_{Cl_j}(x,y)\right]^2}{2\sigma_{Cl_j}^2(x,y)}\right], \quad j=1,\ldots,N, \tag{5.7}$$

Dabei ist r_j, $j=1,\ldots,N$, die Antwort des j-ten Knotens der verdeckten Schicht, g_{Cl_j} die Funktion, die das Zentrum der Basisfunktion dieses Knotens repräsentiert (s. (5.4)), g die Funktion für die Eingabe $\vec{x}$ und $\sigma_{Cl_j}^2$ die Funktion, die die Normierungsparameter dieses Knotens angibt (s. (5.5)).
Im Gegensatz zu den Netzen aus Kapitel 4 handelt es sich hier nicht um die spezielle Form von Radialen Basisfunktionen, sondern um Hyperbasisfunktionen [Pog90], die für unterschiedliche Dimensionen ihrer Zentren unterschiedliche Normierungsparameter verwenden. Der Grund für die Wahl von Hyperbasisfunktionen liegt darin, daß hier hochdimensionale Eingaben verwendet werden, deren Clusterung mehrere Zentren mit signifikant voneinander abweichenden Varianzen in den einzelnen Komponenten ergibt. In Abschnitt 5.4.1 veranschaulichen einige Abbildungen diese unterschiedlichen Varianzen.

Sei $\tilde{f}$ die durch das Netz berechnete Erkennungsfunktion, die f approximiert. Dann ergibt sich die Ausgabe c aus folgender Gleichung:

$$c = \tilde{f}(\vec{x}) = r_1 w_1 + \ldots + r_N w_N \approx f(\vec{x}) \tag{5.8}$$

Phase 2 (Bestimmung der Gewichte der 2. Schicht)

Sei $\mathcal{T} = \{(\vec{x}_1, 1.0), \ldots, (\vec{x}_M, 1.0)\}$ die Trainingsmenge und r_{i_j} die Antwort der Hyperbasisfunktion des j-ten Knotens der verdeckten Schicht auf $\vec{x}_i$. Dann muß die Interpolationsbedingung des folgenden Gleichungssystems mit Linearer Ausgleichsrechnung optimiert werden, um die Gewichte w_j, $j = 1, \ldots, N$, der 2. Schicht des Netzes zu berechnen.

$$\begin{pmatrix} 1.0 \\ \vdots \\ 1.0 \end{pmatrix} = \begin{pmatrix} r_{1_1} & \cdots & r_{1_N} \\ \vdots & & \vdots \\ r_{M_1} & \cdots & r_{M_N} \end{pmatrix} \begin{pmatrix} w_1 \\ \vdots \\ w_N \end{pmatrix} \tag{5.9}$$

Wie in Kapitel 4, Abschnitt 4.2.3, bereitet auch hier das Auslassen von äußeren Schichten der Trainingsdaten Probleme. Die Lösung durch Bestimmung äußerer Schichten der Ausgabedaten und anschließender Übertragung auf die Eingabedaten ist hier nicht möglich, weil die Ausgaben konstant sind ($c \equiv 1$). Es können allerdings äußere Trainings-Templates als Templates extremer Situationen bestimmt werden, indem bei der Erzeugung der Trainingsbilder die Bedingungen schrittweise verändert werden. Z.B. sind bei Änderung der Beleuchtung des Objektes durch schrittweises Erhöhen der Helligkeit um einen konstanten Wert, Trainings-Templates der extremen Situationen 'sehr hell' und 'sehr dunkel' auszulassen, weil sie scheinbar in der Nähe von Randzentren liegen.
Die verschiedenen Verfahren zum Trainieren des Netzes durch Verwendung unterschiedlicher Operatoren werden in Abschnitt 5.4.1 beschrieben.

5.2.2 Bestimmung von Aufmerksamkeitsbereichen durch Attention

Für eine schnelle Lokalisierung des Objektes werden zwei Attentionsverfahren entwickelt. In beiden Verfahren werden in einer Trainingsphase geeignete Operatoren auf diejenigen Trainingsbilder angewendet, die zum Lernen der Erkennungsfunktion verwendet wurden. Anschließend werden wie in Abschnitt 5.2.1 Trainings-Templates durch die Bereiche, in denen sich das Zielobjekt befindet, bestimmt und durch einen Cluster-Algorithmus gruppiert.

In der Anwendungsphase werden zu den in der Trainingsphase bestimmten Clustern visuell auffällige Bildbereiche (Aufmerksamkeitsbereiche) ermittelt, die für die genaue Lokalisierung verwendet werden. Da für die irrelevanten Bereiche keine weitere Verarbeitung erfolgt, wird somit der Zeitaufwand der Suche nach dem Objekt erheblich verringert[5].
Es ist zu beachten, daß zwischen der Objektlokalisierung/-erkennung und der Attention ein unmittelbarer Zusammenhang besteht. Die selektierten Aufmerksamkeitsbereiche müssen so gefunden werden, daß das Zielobjekt mit großer Sicherheit darin enthalten ist. Aus diesem Grunde basieren die Attentionsverfahren auf der hier vorgestellten Technik der Lokalisierung und Erkennung mit Hilfe von HBF-Netzen. Jeder Knoten aus der verdeckten Schicht des HBF-Netzes für die Erkennungsfunktion beschreibt eine bestimmte Menge von Mustern des Objektes. Daher ist es sinnvoll, das Clusterergebnis auf die Attention rückzukoppeln.
Nachfolgend werden die beiden Attentionsverfahren vorgestellt, von denen das erste (grobe Lokalisierung) einstufig und das zweite (feine Lokalisierung) zweistufig ist.

Einstufiges Attentionsverfahren

Bei diesem einstufigen Verfahren wird ein Aufmerksamkeitsbereich des eingegebenen Computerbildes ermittelt, in dem sich das Objekt mit hoher Wahrscheinlichkeit befindet. Die Grundlage der Attention bilden dabei die Filterantworten des in Abschnitt 5.3.1 zu beschreibenden *Gaboroperators* [Jäh93]. Dieser Operator kann so parametrisiert werden, daß er als unimodale Funktion im Schwerpunkt des Objektes entweder eine lokal maximale Antwort liefert, falls es sich um ein bezüglich des Hintergrundes helleres Objekt handelt, oder eine lokal minimale Antwort, falls das Objekt dunkler als der Hintergrund ist.
Wie in Abschnitt 5.2.1 beschrieben, können in der Trainingsphase für die mit dem Gaboroperator gefilterten Trainingsbilder der unterschiedlichen Erscheinungen des Objektes die Clusterzentren- und Clustervarianzen-Templates bestimmt werden.
Da die Lokalisierung des Objektes durch die Suche nach seinem Schwerpunkt erfolgen kann, wird bei diesem Verfahren in der Anwendungsphase für jedes Cluster ein Aufmerksamkeitsbereich ermittelt, dessen Positionen ähnliche Filterantworten wie der Mittelpunkt des Clusterzentren-Templates liefern. Die Vereinigung dieser Bereiche ergibt einen umfassenden Aufmerksamkeitsbereich, in dem das HBF-Netz angewendet wird.
Da bei der Bestimmung der relevanten Bildbereiche als typisches Merkmal lediglich die Filterantwort an der Position des Schwerpunktes mit einbezogen wird,

[5] Bei den folgenden Versuchen ergibt sich ca. eine Verringerung des Zeitaufwandes auf ein Viertel des üblichen Aufwandes.

verwendet dieses Attentionsverfahren einen geringen Umfang an Wissen über die Erscheinung des Zielobjektes. Der nächste Abschnitt beschreibt ein zweistufiges Attentionsverfahren, bei dem typische Strukturen im gefilterten Bild an der Position des Zielobjektes die Grundlage der Attention bilden.

Eine nähere Beschreibung der groben Lokalisierung mit dem einstufigen Attentionsverfahren bei Verwendung des Gaboroperators wird in Abschnitt 5.4.2 gegeben.

Zweistufiges Attentionsverfahren

Bei diesem zweistufigen Verfahren werden in einem Computerbild in der ersten Stufe mehrere Aufmerksamkeitsbereiche ermittelt, während in der zweiten Stufe eine genauere Lokalisierung durch die Bestimmung einer Lokalisierungsposition zu jedem Aufmerksamkeitsbereich stattfindet. Die Grundlage der Attention bilden dabei die Filterantworten des Bildes im Bereich des Zielobjektes, die sich durch Anwendung geeigneter Operatoren ergeben. Es ist hier also im Gegensatz zu dem einstufigen Verfahren ein wesentlich breiteres Wissen über die Erscheinungen des Zielobjektes erforderlich.
Für die 1. Stufe wird wie bei diesem Attentionsverfahren der Gaboroperator verwendet. Es resultieren in der Trainingsphase wie beim einstufigen Verfahren Clusterzentren-Templates (Clusterzentren-Templates der 1. Stufe). In der 2. Stufe werden die Trainingsbilder mit demselben Operator gefiltert, der beim Trainieren des HBF-Netzes verwendet wurde. Es werden nun die Gruppierungen der 1. Stufe übernommen und somit Clusterzentren-Templates für diese gefilterten Templates bestimmt (Clusterzentren-Templates der 2. Stufe).
In der Anwendungsphase werden zu den Clusterzentren-Templates der 1. Stufe Aufmerksamkeitsbereiche bestimmt, die hier jedoch nicht wie beim einstufigen Verfahren vereinigt werden. Stattdessen wird aus jedem Aufmerksamkeitsbereich die Position ermittelt, die den Mittelpunkt desjenigen Templates darstellt, welches dem zugehörigen Clusterzentren-Template der 2. Stufe am ähnlichsten ist.
Es ist zu beachten, daß bei Verwendung unterschiedlicher Operatoren eine Clusterung der Filterantworten in der 1. Stufe andere Gruppierungen der Templates ergeben könnte als in der 2. Stufe, wenn jeweils ein Cluster-Algorithmus verwendet werden würde. Da jedoch in der 2. Stufe die Aufmerksamkeitsbereiche der Clusterzentren-Templates aus der 1. Stufe verwendet werden, muß sinnvollerweise eine Übertragung der Gruppierungen von der 1. zur 2. Stufe stattfinden.
Die Begründung für die Wahl des Gaboroperators in der 1. Stufe und eine beispielhafte Beschreibung der attentiven Verarbeitungsstrategie erfolgt in Abschnitt 5.4.2.

Motivation für beide Verfahren

Im folgenden sollen die Effizienz und die Zuverlässigkeit beider Attentionsverfahren miteinander verglichen werden. Beide Verfahren ermöglichen eine schnelle Lokalisierung, indem sie den Aufwand der Objektsuche verringern. Ein optimales Verfahren sollte möglichst effizient sein und außerdem eine zuverlässige Lokalisierung ermöglichen.
Das zweistufige Verfahren selektiert im Vergleich zu dem einstufigen wesentlich weniger Positionen, die für den Schwerpunkt des Objektes in Frage kommen. Es muß jedoch beachtet werden, daß während der gesamten Attentionsphase das trainierte HBF-Netz, welches durch die Linearkombination der Basisfunktionen alle Clusterzentren-Templates berücksichtigt, nicht angewendet wird. Hier wird lediglich zu jedem Clusterzentren-Template das Template im Bild bestimmt, welches die größte Ähnlichkeit dazu besitzt. Es muß also bereits eine der ermittelten Lokalisierungspositionen die richtige sein. Wurden nun jedoch durch einen Cluster-Algorithmus wenige Clusterzentren-Templates bestimmt, kann sich eine falsche Lokalisierung ergeben. Wenn sich die Erscheinung des Objektes im Eingabebild nämlich erst durch die Kombination verschiedener Clusterzentren-Templates ergibt, kann sie nicht durch ein einzelnes Clusterzentren-Template repräsentiert werden[6]. Dieses Problem kann jedoch dadurch vermieden werden, daß eine genügend große Trainingsmenge erzeugt wird, wodurch viele Gruppierungen bei der Clusterung entstehen.
Bei dem einstufigen Verfahren wird ein Aufmerksamkeitsbereich ermittelt, in dem die Anwendung des trainierten HBF-Netzes stattfindet. Es werden also in diesem umfassenden Bereich alle Clusterzentren-Templates berücksichtigt. Deshalb ist das einstufige Verfahren in bezug auf Zuverlässigkeit geeigneter.

Beim Vergleich des Aufwandes beider Verfahren ist festzustellen, daß der Aufwand des zweistufigen Verfahrens im 'worst case' wesentlich geringer ist als der des einstufigen. In diesem Fall sind die einzelnen Aufmerksamkeitsbereiche disjunkt. Werden die Mengen der Positionen dieser Bereiche mit $\mathcal{A}_1,\ldots,\mathcal{A}_N$ bezeichnet[7], ergibt sich bei beiden Verfahren $(|\mathcal{A}_1| + \ldots + |\mathcal{A}_N|) \cdot N$ als Anzahl der zu berechnenden Aktivierungen nach (5.7). Da die Anzahl der zu berechnenden Zuverlässigkeitswerte jedoch unterschiedlich ist, ergibt sich für die Anzahl der auszuführenden Multiplikationen in (5.8) bei dem einstufigen Verfahren $N \cdot (|\mathcal{A}_1| + \ldots + |\mathcal{A}_N|)$ und beim zweistufigen lediglich N^2. Also ist das zweistufige Verfahren in bezug auf die Effizienz geeigneter.

[6] Z.B. ist bei zwei Clusterzentren-Templates, die das Objekt bei sehr heller und sehr dunkler Beleuchtung repräsentieren, die Kombination erforderlich, wenn das Eingabebild durch mittelstarke Beleuchtung entstanden ist.

[7] N ist die Anzahl der Clusterzentren-Templates.

5.2.3 Anwenden des HBF-Netzes zur Lokalisierung und Erkennung

Bei der Anwendung des trainierten HBF-Netzes sollen nur noch die im zweistufigen Attentionsverfahren ermittelten Lokalisierungspositionen bzw. der durch das einstufige Attentionsverfahren ermittelte Aufmerksamkeitsbereich berücksichtigt werden. Es resultieren dadurch eindeutig bestimmte Templates, deren Mittelpunkte durch die Lokalisierungspositionen bzw. durch die Positionen des Aufmerksamkeitsbereiches (im folgenden auch als Lokalisierungspositionen bezeichnet) beschrieben werden.
Die Vektoren, die diese ermittelten Templates repräsentieren, können nun als Eingabevektoren in das trainierte HBF-Netz eingegeben werden. Es werden somit nach (5.8) und (5.7) Zuverlässigkeitswerte berechnet, wobei aus der maximalen Zuverlässigkeit der Ort des lokalisierten Objektes resultiert. Es wurde hierdurch jedoch nur die Position bestimmt, an der sich das Zielobjekt mit der höchsten Wahrscheinlichkeit im Vergleich zu den übrigen Positionen befindet. Ob es sich an dieser Position wirklich um das Objekt handelt, kann durch einen Schwellwert entschieden werden. Ist der Zuverlässigkeitswert höher als der Schwellwert, wurde das Objekt an der lokalisierten Position erkannt. Andernfalls muß davon ausgegangen werden, daß sich das Objekt nicht in dem Computerbild befindet.
Eine genaue Beschreibung der Anwendung des HBF-Netzes erfordert Kenntnisse über die verwendeten Operatoren beim Trainieren und bei der Attention. Deshalb erfolgt diese in dem späteren Abschnitt 5.4.3.

5.2.4 Klassifizierung unterschiedlicher Objekte

Bisher wurde die Methode des Trainierens und Anwendens eines HBF-Netzes zur Lokalisierung *eines* Objektes vorgestellt. Soll unter mehreren Objekten unterschieden werden, kann folgende Möglichkeit der Klassifizierung verwendet werden:
Als Elemente der Trainingsmenge $\mathcal{T}$ werden $(B \cdot H)$-dimensionale Eingabevektoren $\vec{x}$, die die Trainings-Templates aller zu unterscheidenden Objekte darstellen, mit zugehörigen Ausgabevektoren $\vec{c}$ gewählt. Dabei besitzt $\vec{c}$ als Dimension die Anzahl S der zu unterscheidenden Objekte, wobei in der s-ten Komponente, $s = 1, \ldots, S$, der Wert 1 steht, falls der Eingabevektor $\vec{x}$ das s-te Objekt repräsentiert. Alle übrigen Komponenten erhalten den Wert 0.
Bei Anwendung auf einen neuen Eingabevektor $\vec{x}$ gibt ein sogenannter *Zuverlässigkeitsvektor* $\vec{c}$ in der s-ten Komponente an, mit welcher Wahrscheinlichkeit es sich um das s-te Objekt handelt. Eine Klassifizierung wird somit durch die Bestimmung der maximalen Komponente erreicht.

5.2.5 Die Kriterien Invarianz und Diskriminanz

Ein wichtiger Aspekt bei der Erkennung von Objekten durch Bildverarbeitung sind das *Invarianz-* und das *Diskriminanzkriterium*. Eine Erkennungsfunktion soll nach Möglichkeit beide Kriterien erfüllen. Einige Versuche des Abschnittes 5.5 zeigen jedoch, daß bei Erfüllung des einen Kriteriums das andere vernachlässigt wird, was einen Kompromiß erforderlich macht.

Diskriminanz

In Abschnitt 5.2.3 wurde ein Schwellwert erwähnt, der für die Erkennung eines lokalisierten Objektes durch den Zuverlässigkeitswert an dieser Position überschritten werden sollte. Eine durch ein HBF-Netz berechnete Erkennungsfunktion erfüllt das Diskriminanzkriterium genau dann, wenn die Zuverlässigkeitswerte für Templates, die das Objekt nicht darstellen, deutlich geringer als dieser Schwellwert sind und für Templates, die das Zielobjekt darstellen, deutlich größer. Damit ergibt sich die Frage, wie hoch der Schwellwert zu wählen ist, damit die Erkennungsfunktion das Kriterium erfüllt. Die Versuche in Abschnitt 5.5 zeigen, daß die Größen der Normierungsparameter $\sigma^2_{Cl_j}(x,y)$ in (5.7) dabei eine große Rolle spielen. Je geringer diese gewählt werden, desto besser kann zwischen Objekt/Nicht-Objekt unterschieden werden.

Invarianz

Ein weiteres Kriterium ist das Invarianzkriterium. Eine durch ein HBF-Netz berechnete Erkennungsfunktion erfüllt das Invarianzkriterium genau dann, wenn die Zuverlässigkeitswerte für Templates, die das Zielobjekt bei unterschiedlicher Erscheinung darstellen, ungefähr gleich sind. Auch hier beeinflussen die Normierungsparameter in (5.7) den Grad der Invarianz. Je höher diese gewählt werden, desto geringere Differenzen ergeben sich für die Zuverlässigkeitswerte.

5.3 Operatoren

In diesem Abschnitt werden einige Operatoren vorgestellt, die für das Trainieren eines HBF-Netzes zur Objektlokalisierung und in der Trainingsphase der Attention verwendet werden können. Die Verfahren für das Trainieren eines HBF-Netzes und die Attentionsverfahren bei Verwendung unterschiedlicher Operatoren werden in den Abschnitten 5.4.1 und 5.4.2 beschrieben.

5.3.1 Der Gaboroperator

Der Gaboroperator wird zum einen für den in Abschnitt 5.4.2 beschriebenen Attentionsmechanismus verwendet. Er wird außerdem zum Erzeugen der Trainings-Templates eines HBF-Netzes benutzt, wenn das Lernen der Erscheinungen eines Objektes mit seiner Umgebung erwünscht ist (LmU-Lernstrategie).

Mit einem Gaborfilter [CVI94] können in einem Computerbild bestimmte Wellenzahlbereiche selektiert werden. Er stellt somit einen *Bandpaßfilter* dar, der z.B. die Detektion von Kanten in einem Bild durch Selektion hoher Wellenzahlen ermöglicht. Durch die Faltung mit einer komplexen Gaborfunktion wird ein reelles Signal auf ein komplexes abgebildet. Man bezeichnet daher die Funktion als *Quadraturfilterpaar*.
Für die Objektlokalisierung und -erkennung wird in diesem Kapitel der reelle Teil der Gaborfunktion verwendet. Er stellt die folgende in zwei Richtungen gaußmodulierte Cosinusfunktion dar (Abb. 5.5):

$$G_{\sigma_1,\sigma_2,\lambda}(x,y) = e^{-\frac{1}{2}\left(\frac{x^2}{\sigma_1^2}+\frac{y^2}{\sigma_2^2}\right)} \cos\left(\frac{2\pi x}{\lambda}\right) \tag{5.10}$$

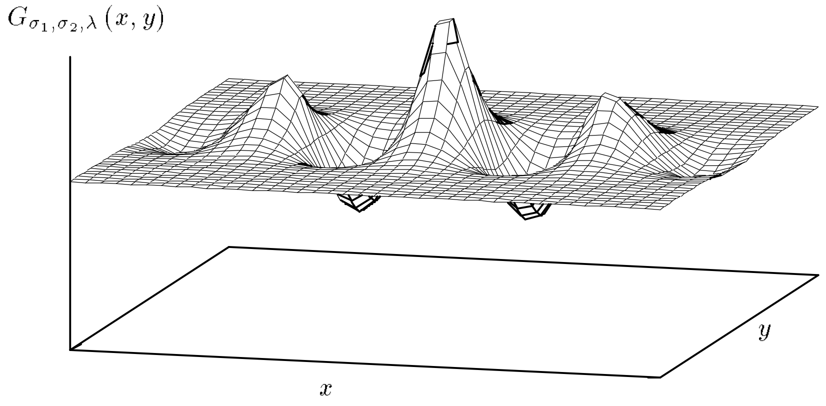


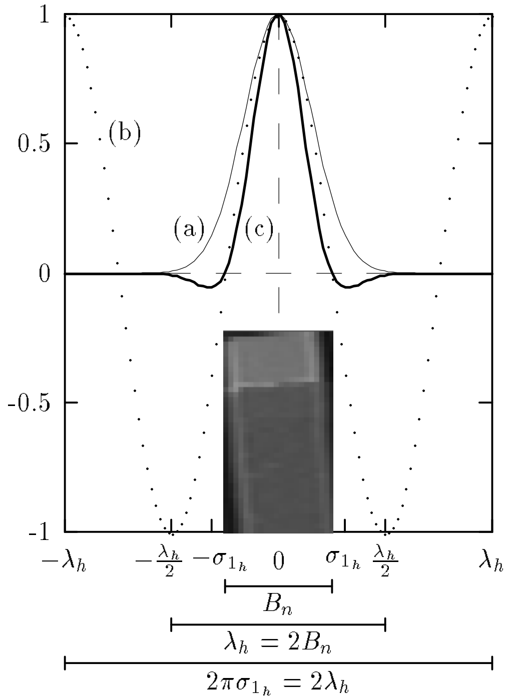
Abbildung 5.5: Der reelle Teil der komplexen Gaborfunktion.

Hierbei stellt λ die Wellenlänge der Cosinusfunktion dar, während σ_1 und σ_2 die Wendepunkte der Gaußfunktion in Richtung der x-Achse und der y-Achse angeben. Die Gaußfunktion besitzt in x-Richtung eine Ausdehnung $2\pi\sigma_1$ und in y-Richtung eine Ausdehnung $2\pi\sigma_2$.

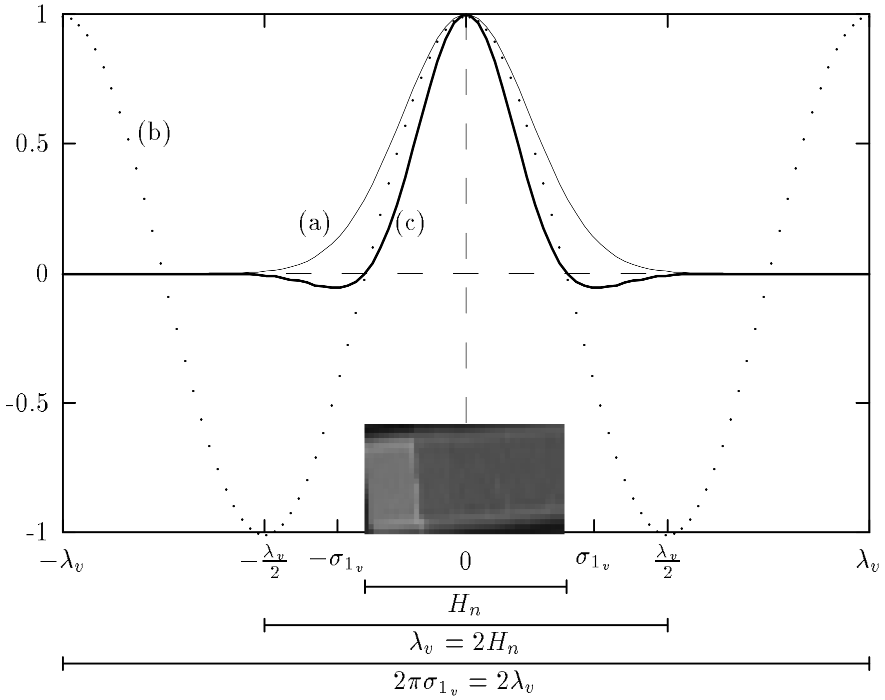
Bei der Filterung eines Computerbildes ψ mit dem Gaboroperator $\mathcal{O}_{Gab}$ ist eine unimodale Wirkung auf den Bereich des Objektes erwünscht. Das bedeutet, daß sich für den Schwerpunkt des Objektes eine maximale oder minimale Filterantwort relativ zu den Filterantworten der anderen Punkte des Objektes ergibt. Da der Gaboroperator nur in eine Richtung als Bandpaßfilter wirkt, werden zwei Gaboroperatoren $\mathcal{O}_{Gab_h}$ und $\mathcal{O}_{Gab_v}$ verwendet, von denen der eine horizontal und der andere vertikal (Laufrichtung des Cosinus) auf das Computerbild angewendet wird. Durch sequentielle Anwendung beider Operatoren und Kombination der Filterantworten durch Multiplikation ergibt sich folgendes Ergebnis der Anwendung des gesuchten Operators auf ein Computerbild ψ:

$$\mathcal{O}_{Gab}\{\psi\} = \mathcal{O}_{Gab_h}\{\psi\} \cdot \mathcal{O}_{Gab_v}\{\psi\} \tag{5.11}$$

In [Kri96] wird eine ähnliche Methode der Kombination von Operatoren, die in unterschiedlichen Richtungen wirken, betrachtet. In diesem Kapitel soll die Verwendung von zwei orthogonal wirkenden Operatoren ausreichen.
Um die unimodale Wirkung in horizontaler und vertikaler Richtung zu erzeugen, werden die zu selektierenden Wellenzahlen durch die Höhe und die Breite des Objektes bestimmt. Dadurch werden Bildbereiche, in denen die Wellenzahlen mit der Ausdehnung des Objektes übereinstimmen, durch eine einheitliche Struktur hervorgehoben.
Wird in einem Eingabebild nach dem Schwerpunkt eines Zielobjektes gesucht, können bei der Attention bereits Bereiche des gefilterten Bildes ausgeschlossen werden, in denen die Filterantworten wesentlich geringer oder höher ausfallen als die erwartete Filterantwort im Schwerpunkt des Objektes.
Die unimodale Wirkung eines Operators soll sich möglichst für alle Erscheinungen eines Objektes ergeben. Um dies zu erreichen, muß der Operator durch eine geeignete Bestimmung seiner Parameter angepaßt werden. Zunächst werden die Größen $H_n = \frac{H_S}{I}$ und $B_n = \frac{B_S}{I}$ (H_S=Höhe und B_S=Breite der Silhouette des Objektes in Pixeln) berechnet, um die Ausmaße des Objektes in einem Computerbild mit einer Auflösung von I^2 Pixeln zu normieren.
Damit ein Operator unimodal auf die Breite des Objektes reagiert, wird die Wellenlänge λ_h des Cosinus auf die doppelte normierte Breite B_n der Silhouette, die Ausdehnung der Gaußfunktion in x-Richtung auf die doppelte Wellenlänge und die Ausdehnung der Gaußfunktion in y-Richtung auf die normierte Höhe H_n der Silhouette gesetzt (Abb. 5.6 (h)). Die Ermittlung der Parameter des Operators, der auf die Höhe des Objektes reagiert, geschieht analog (Abb. 5.6 (v)). Um die Breite und Höhe des Objektes mit den Parametern der Gauß- und Cosinusfunktionen der Operatoren zu vergleichen, wurde ein Bild des Objektes in die Graphen der Abb. 5.6 eingefügt.



(h)



(v)

Abbildung 5.6: Die Gaußfunktionen $e^{-\frac{1}{2}\left(\frac{x^2}{\sigma^2_{1_{h/v}}}\right)}$ (a), die Cosinusfunktionen $\cos\left(\frac{2\pi x}{\lambda_{h/v}}\right)$ (b) und die Überlagerungen beider Funktionen (c) für die Gaboroperatoren mit den in (5.12) berechneten Parametern (h) und mit den in (5.13) berechneten (v).

Die benötigten Parameter für die Lokalisierung des Objektes in horizontaler Ausrichtung und in vertikaler Ausrichtung errechnen sich wie folgt:

$$B_n = \frac{B_S}{I}, \quad H_n = \frac{H_S}{I}, \quad \lambda_h = 2B_n, \quad \sigma_{1_h} = \frac{2B_n}{\pi}, \quad \sigma_{2_h} = \frac{H_n}{2\pi} \tag{5.12}$$

$$B_n = \frac{B_S}{I}, \quad H_n = \frac{H_S}{I}, \quad \lambda_v = 2H_n, \quad \sigma_{1_v} = \frac{2H_n}{\pi}, \quad \sigma_{2_v} = \frac{B_n}{2\pi} \tag{5.13}$$

Dadurch lassen sich nun zwei Gaboroperatoren $\mathcal{O}_{Gab_h}$ und $\mathcal{O}_{Gab_v}$ konstruieren, von denen der eine unimodal auf die Breite des Objektes (horizontaler Operator) und der andere unimodal auf die Höhe (vertikaler Operator) reagiert.

Wird ein Gaboroperator auf ein Computerbild angewendet, dessen Pixel durch Grauwerte beschrieben werden, so ergibt sich für jede Position im Bild eine *Gaborfilterantwort*. Aus den beiden Gaborfilterantworten in horizontaler und vertikaler Ausrichtung kann anschließend durch Multiplikation eine Gesamtantwort ermittelt werden. Im folgenden ist mit einer Gaborfilterantwort immer diese kombinierte Gesamtantwort gemeint. Für jede Position (x, y) des Bildes kann die Gaborfilterantwort durch folgende Funktion gab beschrieben werden:

$$gab : D_x \times D_y \longrightarrow \mathbb{R} \tag{5.14}$$

Dabei sind die Definitionsbereiche D_x und D_y abhängig von der Breite und der Höhe des gefilterten Bildes. Die Abb. 5.7 zeigt die Auswirkungen der Operatoren auf ein Grauwertbild.
Es ist zu erkennen, daß das Maximum der Filterantworten in vertikaler Richtung bei Anwendung des vertikalen Gaboroperators besser ausgeprägt ist als bei Anwendung des horizontalen Operators (seitliche Ansicht in Abb. 5.7 (a") und (b")). Bei der Bestimmung des Maximums der Filterantworten in horizontaler Richtung ist der horizontale Gaboroperator geeigneter (frontale Ansicht in Abb. 5.7 (b') und (a')).

Aus der Kombination jeweils beider Filterantworten durch ihre Multiplikation ergeben sich Gesamtantworten. Diese durch den Operator $\mathcal{O}_{Gab}$ erzeugten Filterantworten besitzen sowohl in vertikaler (seitliche Ansicht) als auch in horizontaler (frontale Ansicht) Richtung gut ausgeprägte Maxima (Abb. 5.8).

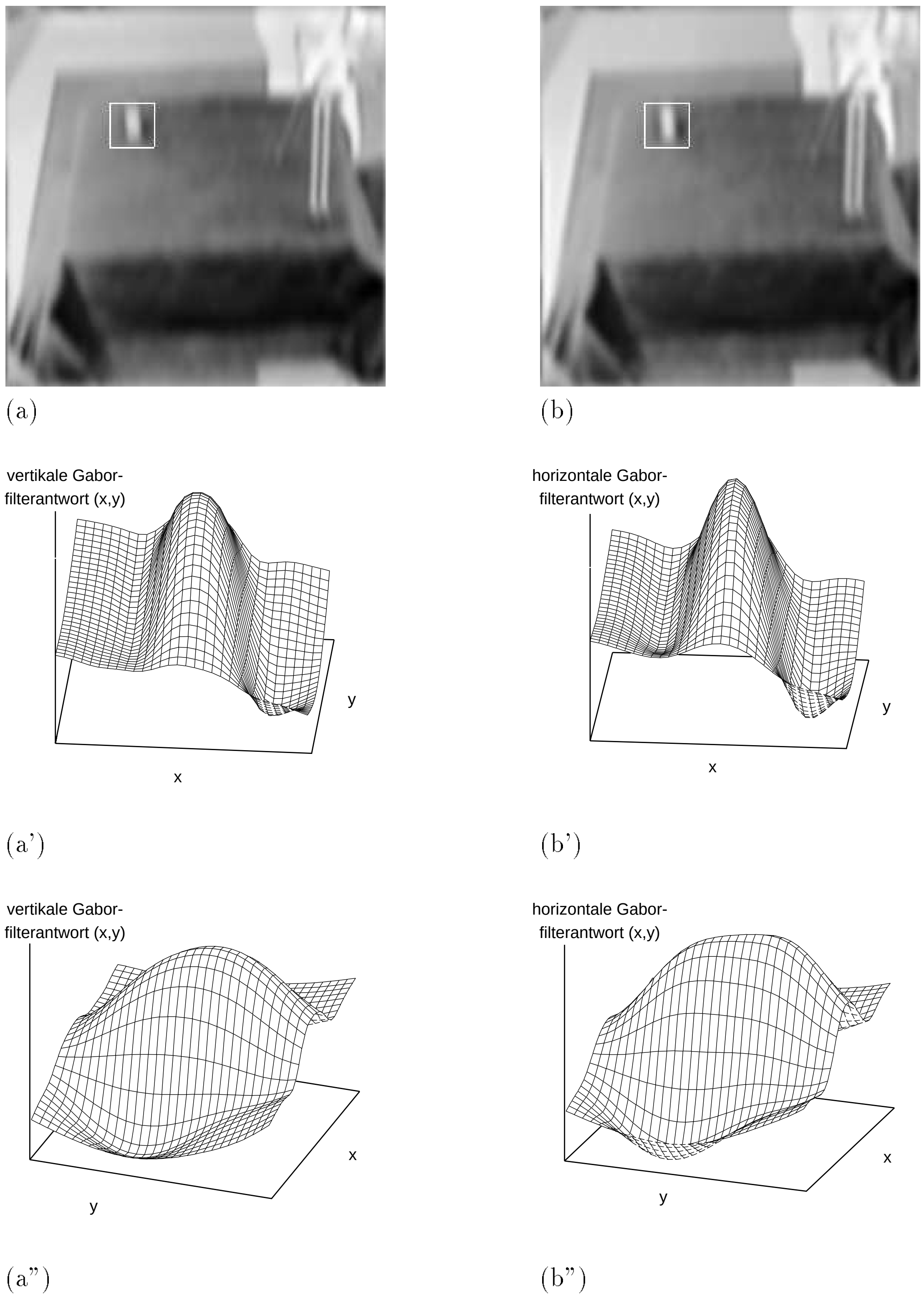
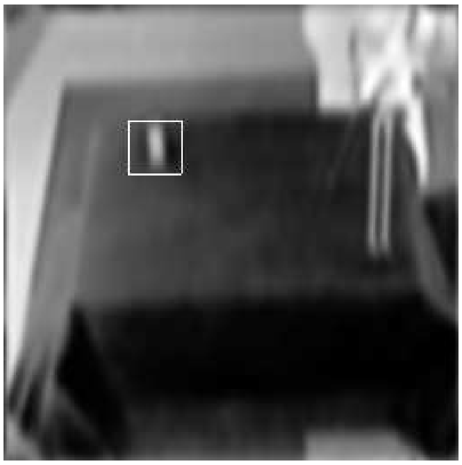
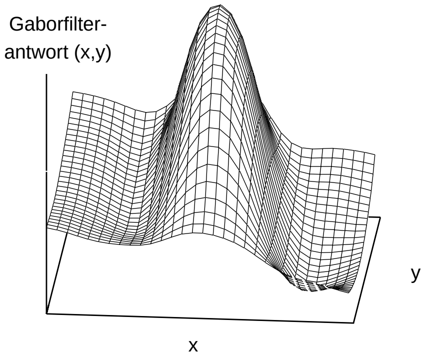
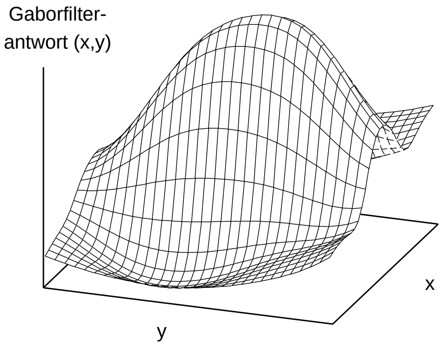


Abbildung 5.7: Wirkung des vertikalen (a) und des horizontalen Gaboroperators (b) sowie eine graphische Darstellung der Antworten in den markierten Bereichen bei einer frontalen (a'), (b') und einer seitlichen (a"), (b") Ansicht.



(c)





(c') (c")

Abbildung 5.8: Wirkung des kombinierten Gaboroperators $\mathcal{O}_{Gab}$ (c) sowie eine graphische Darstellung der Antworten in dem markierten Bereich bei einer frontalen (c') und einer seitlichen (c") Ansicht.

5.3.2 Der Gradientenoperator

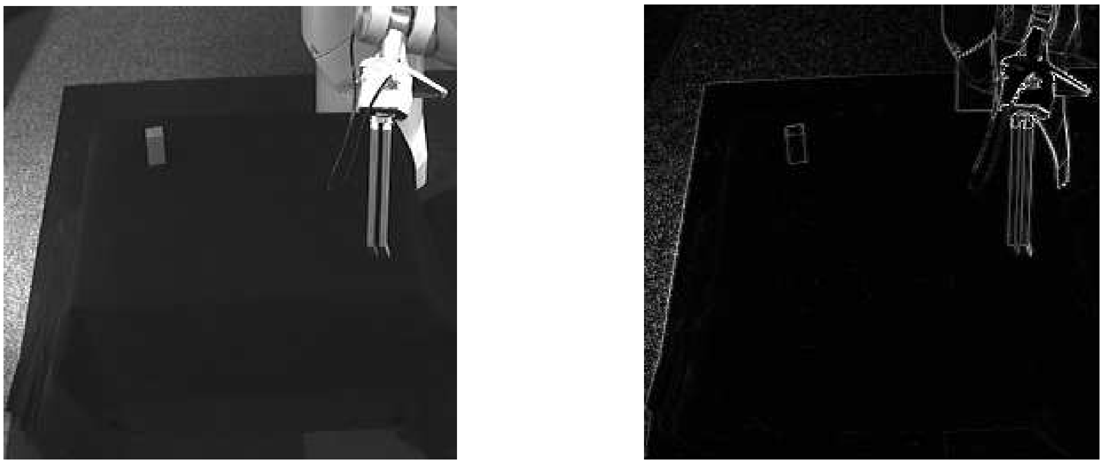
Der *Gradientenoperator* $\mathcal{O}_{Grad}$ kann zum Erzeugen der Trainings-Templates eines HBF-Netzes benutzt werden, das die Erscheinungen eines Objektes ohne seine Umgebung lernen kann (LoU-Lernstrategie). In Abschnitt 5.4.1 wird das Trainieren eines HBF-Netzes bei Verwendung dieses Operators beschrieben.

Bei Anwendung eines Gradientenoperators auf ein Computerbild erhält man ein sogenanntes *Kantenbild*, in dem die im Bild auftretenden Kanten besonders hervorgehoben werden. Dieser Operator ergibt sich aus der Summe der Beträge zweier *Ableitungsoperatoren*. Der Ableitungsoperator $D^x = (\ 1\ ,\ -1\ )$ liefert bei Fal-

tung mit einem Computerbild an Positionen mit vertikalen Kanten lokale Extrema, während $D^y = (\ 1\ ,\ -1\ )^T$ horizontale Kanten durch Extrema detektiert. Der Gradientenoperator ergibt sich damit durch die Kombination $\mathcal{O}_{Grad} = |D^x|+|D^y|$ beider Operatoren, und hebt bei Anwendung auf ein Computerbild beliebig orientierte Kanten hervor (Abb. 5.9) [Jäh93].
Wird ein Gradientenoperator auf ein Computerbild angewendet, dessen Pixel durch Grauwerte beschrieben werden, so ergibt sich für jede Position (x, y) im Bild eine *Gradientenfilterantwort*, die durch folgende Funktion *grad* beschrieben wird:

$$grad : D_x \times D_y \longrightarrow \mathbb{R} \tag{5.15}$$

Die Definitionsbereiche D_x und D_y sind abhängig von der Breite und der Höhe des gefilterten Bildes.



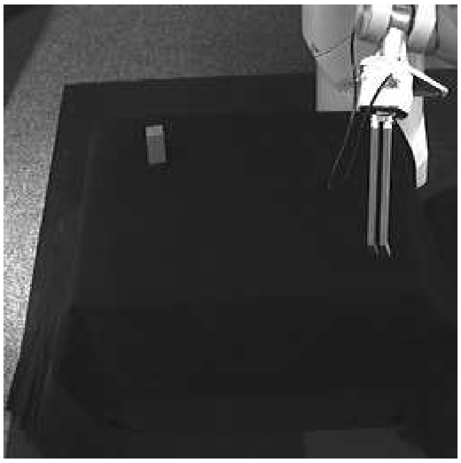
(a) (b)

Abbildung 5.9: Eine Aufnahme der rechten Kamera (a) und das zugehörige durch Gradientenfilterung erzeugte Kantenbild (b).

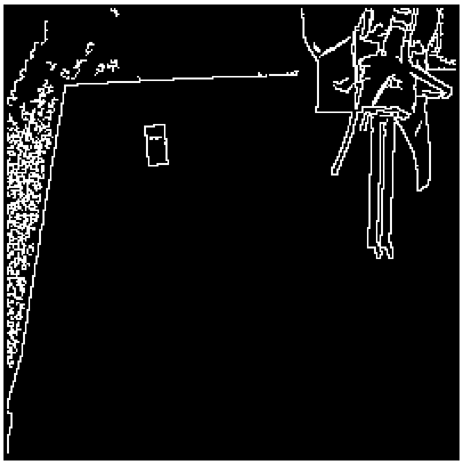
5.3.3 Der Gradienten-Schwellwert-Operator

Der Gradienten-Schwellwert-Operator $\mathcal{O}_{GS}$ (kurz: *GS-Operator*) kann ebenfalls zum Erzeugen der Trainigs-Templates eines HBF-Netzes benutzt werden, das die Erscheinungen eines Objektes ohne seine Umgebung lernen kann (LoU-Lernstrategie). In Abschnitt 5.6.1 wird alternativ zur Beschreibung eines Objektes durch seine photometrischen Eigenschaften die Beschreibung durch seine geometrischen Eigenschaften verwendet. Die Bestimmung dieser Eigenschaften erfordert die Filterung mit dem GS-Operator.

Bei der Anwendung des GS-Operators auf ein Computerbild werden zunächst die Gradientenfilterantworten berechnet. Ein Schwellwertoperator liefert anschließend für Positionen, an denen die Filterantwort größer als ein Schwellwert thr ist, den Wert 1 und für die übrigen Positionen den Wert 0. Dadurch ergibt sich hier ein binäres Kantenbild (Abb. 5.10).



(a)



(b)

Abbildung 5.10: Eine Aufnahme der rechten Kamera (a) und das zugehörige durch GS-Filterung erzeugte binäre Kantenbild (b).

Für jede Position (x, y) eines durch Grauwerte dargestellten Computerbildes liefert der Operator eine *GS-Filterantwort*, die durch folgende Funktion gs beschrieben wird:

$$gs : D_x \times D_y \longrightarrow \mathbb{R}, \; (x, y) \mapsto bin\,(grad(x, y)) \tag{5.16}$$

Dabei stellt die folgende Funktion bin den Schwellwertoperator dar, der die Gradientenfilterantworten auf binäre Werte abbildet:

$$bin : \mathbb{R} \longrightarrow \{0, 1\}\,, \quad bin\,(x) = \begin{cases} 1 & : \quad x > thr \\ 0 & : \quad x \leq thr \end{cases}$$

Die Definitionsbereiche D_x und D_y für die Funktion gs sind abhängig von der Breite und der Höhe des gefilterten Bildes.

5.4 Lokalisierung/Erkennung mit HBF-Netzen

Wie zu Beginn dieses Kapitels beschrieben, sind für das Lernen einer Objekterkennungsfunktion unterschiedliche Strategien möglich, z.B. das Lernen des Objektes mit seiner Umgebung (LmU-Lernstrategie) oder ohne (LoU-Lernstrategie).

Bei der LmU-Lernstrategie wird zum Trainieren des HBF-Netzes der in Abschnitt 5.3.1 beschriebene Gaboroperator verwendet. Um das Objekt ohne seine Umgebung zu lernen, ist die Verwendung des Gradientenoperators oder alternativ dazu die Verwendung des GS-Operators (Abschnitte 5.3.2 und 5.3.3) geeigneter.

5.4.1 Exemplarisches Lernen einer Objekterkennungsfunktion mit diversen Operatoren

Für beide Lernstrategien werden zum Trainieren des Netzes zunächst wie in Abschnitt 5.2.1 M Trainingsbilder für die unterschiedlichen Erscheinungen des Zielobjektes erzeugt. Bei der Erzeugung der Trainings-Templates sind nun die beiden Strategien voneinander zu unterscheiden.

Bei der LmU-Lernstrategie werden die Trainingsbilder mit dem (*globalen* [CVI94]) Gaboroperator $\mathcal{O}_{Gab}$ gefiltert. Dieser Operator besitzt einen großen Einzugsbereich, d.h. er bildet bei der Berechnung der Filterantwort eines Bildpunktes einen großen Bildbereich um diesen Punkt herum auf seine Filterantwort ab. Somit spielt bei der Berechnung der Filterantworten von Bildpunkten innerhalb des Objektes auch die Umgebung eine Rolle.
Bei der LoU-Lernstrategie werden die Trainingsbilder mit dem (*lokalen* [CVI94]) Gradientenoperator $\mathcal{O}_{Grad}$ gefiltert. Im Gegensatz zum Gaboroperator besitzt dieser Operator einen kleinen Einzugsbereich. Es werden hier bei der Berechnung der Filterantwort eines Bildpunktes lediglich die unmittelbar benachbarten Bildpunkte mit berücksichtigt.

Nach der Filterung mit einem der Operatoren wird ein konstanter Bildbereich bestimmt, der gerade die Silhouette des Objektes in allen Trainingsbildern enthält. Mit diesem Bereich werden nun die Trainings-Templates in den gefilterten Trainingsbildern festgelegt. Sie bilden mit ihren zugehörigen Ausgabewerten ($c = 1$) die Elemente der Trainingsmenge.
Die Filterantworten in den Trainings-Templates werden wie in Abschnitt 5.2.1 durch die Funktionen $g_1,\dots,g_M$ beschrieben. Weiterhin werden mit einem Cluster-Algorithmus die Clusterzentren-Templates und die zugehörigen Clustervarianzen-Templates bestimmt, deren Filterantworten durch die Funktionen g_{Cl_j} und var_{Cl_j}, $j = 1, \dots, N$, beschrieben werden[8] (s. (5.4) und (5.5)).
Seien beispielsweise die in Abb. 5.2 dargestellten Erscheinungen eines Objektes die Trainingsbilder. Nach einer Filterung mit dem Gabor- oder dem Gradientenoperator ergeben sich für die Trainings-Templates mit dem ISODATA-Cluster-Algorithmus aus Kapitel 3 drei Cluster ($N = 3$) mit den Indexmengen $C_1 = \{1, 4, 5, 6\}$, $C_2 = \{2\}$ und $C_3 = \{3, 7, 8, 9\}$. Die jeweiligen Trainings-,

[8] N ist die Anzahl der ermittelten Cluster.

Clusterzentren-[9] und Clustervarianzen-Templates[10], die sich aus (5.4) und (5.5) ergeben, sind in den Abbildungen 5.11 und 5.12 zu sehen, wobei die dargestellten Bildbereiche zur Anschaulichkeit zusätzlich die Umgebung der Templates enthalten. Abb. 5.13 zeigt das Ergebnis der Clusterung für die entsprechenden ungefilterten Templates.

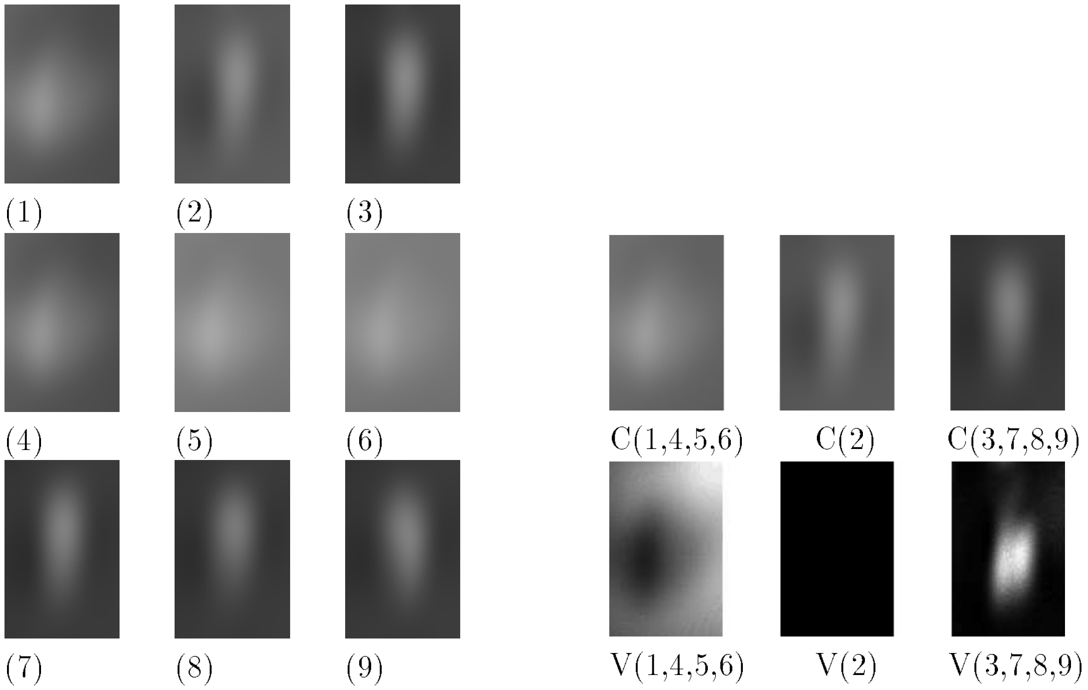


Abbildung 5.11: Trainings-Templates (1)-(9), Clusterzentren-Templates (C) und Clustervarianzen-Templates (V) bei der LmU-Lernstrategie.

In Kapitel 3, Abschnitt 3.2.2, wurde das Auftreten einer einelementigen Indexmenge bei der Clusterung als Problemfall dargestellt, weil sich damit für die Varianz eines Clusters der Wert 0 ergeben würde. Da bei diesem Beispiel die Indexmenge C_2 einelementig ist, würde eine Aktivierung nach (5.7) nicht berechnet werden können. Es soll hier jedoch nur eine Betrachtung der unterschiedlichen Filterantworten erfolgen. Zum Trainieren eines HBF-Netzes müssen wesentlich mehr Trainingsbilder verwendet werden, so daß sich für jede Gruppierung mindestens zwei Trainings-Templates ergeben.

[9] Für dunkle Objekte auf hellem Hintergrund wurde die Gaborfilterantwort invertiert, um einheitliche Strukturen zu erhalten.

[10] Die Clustervarianzen-Templates wurden normiert. Also sind nur die Helligkeiten in einem Template untereinander, nicht aber mit Helligkeiten aus anderen Templates, vergleichbar.

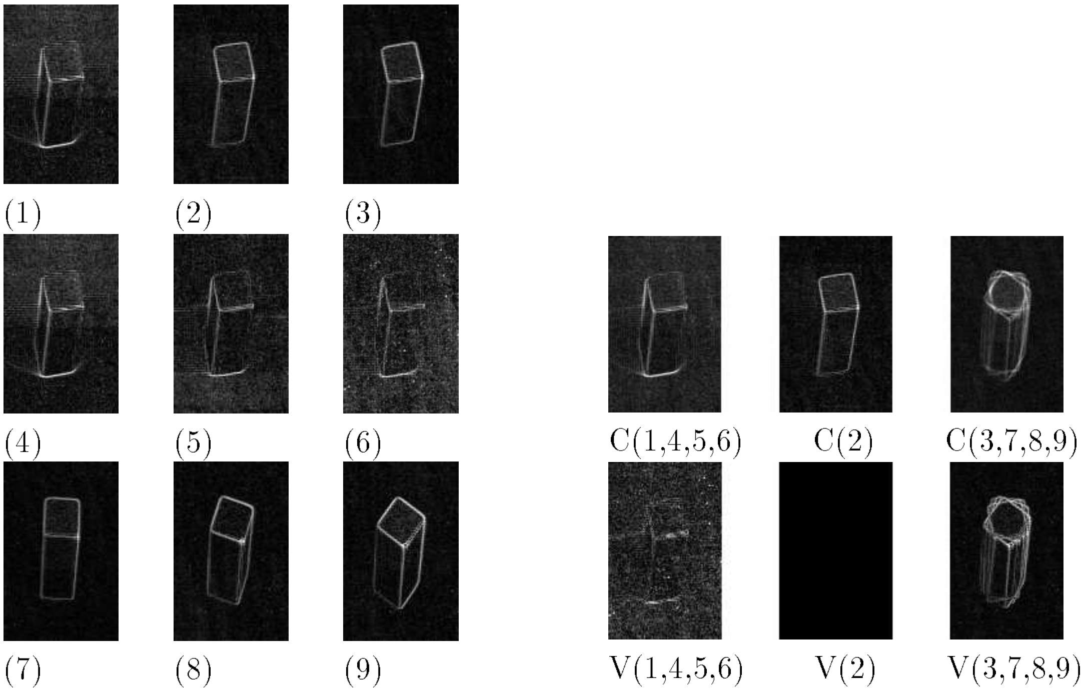


Abbildung 5.12: Trainings-Templates (1)-(9), Clusterzentren-Templates (C) und Clustervarianzen-Templates (V) bei der LoU-Lernstrategie.

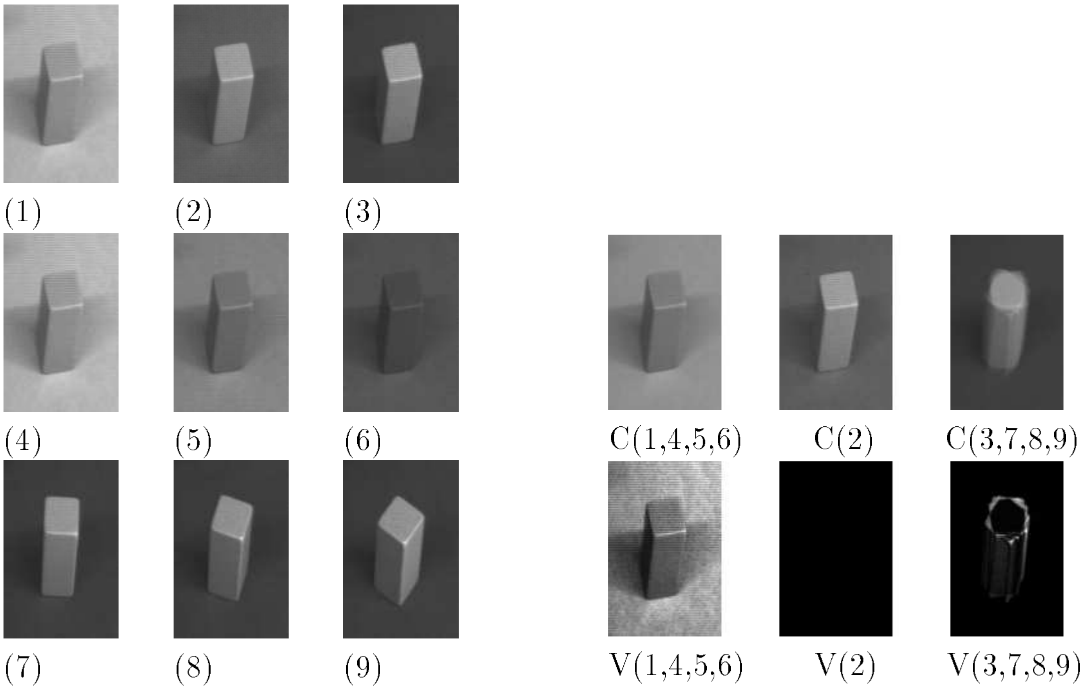


Abbildung 5.13: Ergebnis der Clusterung (C) für die ungefilterten Templates (1)-(9) mit zugehöriger Varianz (V).

5.4.2 Attention auf der Grundlage von Gabor- und Gradientenoperator

In diesem Abschnitt soll beschrieben werden, wie ein Objekt in einem Computerbild mit Hilfe der Attentionsverfahren aus Abschnitt 5.2.2 unter Verwendung des Gaboroperators grob lokalisiert werden kann. Der Ablauf der Attention setzt zunächst eine Trainingsphase voraus, in der eine Generalisierung der Gaborfilterantworten der Trainingsbilder stattfindet.

Trainingsphase der Attention

Durch die Filterung der M Trainingsbilder mit dem Gaboroperator $\mathcal{O}_{Gab}$ ergeben sich an der Position des Objektes M Templates, die durch Funktionen g'_i, $i = 1, \ldots, M$, beschrieben werden. Diese Funktionen stellen die Filterantworten in den Templates dar. Sie besitzen also denselben Definitionsbereich wie die Funktionen g_i, $i = 1, \ldots, M$, aus Abschnitt 5.2.1. Wurde schon beim Lernen der Erkennungsfunktion der Gaboroperator verwendet, gilt für alle Positionen (x, y) des Definitionsbereiches $g'_i(x, y) = g_i(x, y)$, $i = 1, \ldots, M$.
Wie in der 1. Phase beim Trainieren des Netzes werden auch hier die Templates mit einem Cluster-Algorithmus gruppiert. Da sich bei der Attention durch die Clusterung der gefilterten Trainingsbilder andere Gruppierungen ergeben können als beim Trainieren des Netzes, werden folgende Funktionen für die Darstellung der Clusterzentren- und Clustervarianzen-Templates eingeführt (vgl. mit (5.4) und (5.5)):

$$g'_{Cl_j}(x, y) = \frac{1}{|C'_j|} \sum_{i \in C'_j} g'_i(x, y), \; j = 1, \ldots, N' \tag{5.17}$$

$$g''_{Cl_j}(x, y) = \frac{1}{|C'_j|} \sum_{i \in C'_j} g_i(x, y), \; j = 1, \ldots, N' \tag{5.18}$$

$$var'_{Cl_j}(x, y) = \frac{1}{|C'_j|} \sum_{i \in C'_j} \left[g'_i(x, y) - g'_{Cl_j}(x, y) \right]^2, \; j = 1, \ldots, N' \tag{5.19}$$

$$var''_{Cl_j}(x, y) = \frac{1}{|C'_j|} \sum_{i \in C'_j} \left[g_i(x, y) - g''_{Cl_j}(x, y) \right]^2, \; j = 1, \ldots, N' \tag{5.20}$$

Die Funktionen g'_{Cl_j} und var'_{Cl_j} stellen dabei die Gaborfilterantworten in den Clusterzentren- und Clustervarianzen-Templates des einstufigen Attentionsverfahrens bzw. der 1. Stufe des zweistufigen Verfahrens dar. Für die 2. Stufe des zweistufigen Verfahrens werden die Trainings-Templates aus der Trainingsmenge des HBF-Netzes verwendet, wobei jedoch die Gruppierungen der 1. Stufe übernommen werden. Diese Methode ist daher sinnvoll, da in der 2. Stufe die Aufmerksamkeitsbereiche der Clusterzentren-Templates aus der 1. Stufe verwendet werden. Es ergeben sich hier die Funktionen g''_{Cl_j} und var''_{Cl_j}. Die Anzahl der Cluster in der Trainingsphase der Attention werde mit N' bezeichnet. Es ist zu beachten, daß sich bei der Attention Indexmengen C'_j ergeben, die sich möglicherweise von den beim Trainieren des Netzes verwendeten Indexmengen C_j unterscheiden.

Die Indexmengen C'_j, $j = 1, \ldots, N'$, sind wie folgt definiert:

$$C'_j \subseteq \{1, \ldots, M\}, \quad \dot{\bigcup_{j=1,\ldots,N'}} C'_j = \{1, \ldots, M\}$$

Anwendungsphase der Attention

Geht man davon aus, daß im Zentrum eines Templates durch den Schwerpunkt eines hellen Objektes auf dunklem Hintergrund eine maximale Gaborfilterantwort und bei einem dunklen Objekt auf hellem Hintergrund eine minimale Gaborfilterantwort[11] erzeugt wird, so kann im Eingabebild zunächst nach Punkten gesucht werden, deren Gaborfilterantworten in einem gewissen *Toleranzbereich* um die Maxima bzw. Minima der Clusterzentren-Templates herum liegen. Befinden sich diese Maxima bzw. Minima an den Positionen $\left(x'_j, y'_j\right)$, $j = 1, \ldots, N'$, so werden die Toleranzbereiche $[min_{Tol_j}, max_{Tol_j}]$, $j = 1, \ldots, N'$, durch die Streuung[12] $\sigma'_{Cl_j}(x'_j, y'_j)$ in den Clusterzentren-Templates an diesen Positionen und einen *Streuungsfaktor* F wie folgt bestimmt:

$$\begin{aligned} min_{Tol_j} &= g'_{Cl_j}\left(x'_j, y'_j\right) - F\sigma'_{Cl_j}\left(x'_j, y'_j\right) \\ max_{Tol_j} &= g'_{Cl_j}\left(x'_j, y'_j\right) + F\sigma'_{Cl_j}\left(x'_j, y'_j\right) \end{aligned} \qquad (5.21)$$

Ist also der Schwerpunkt des Objektes im Computerbild zu lokalisieren, kann die Suche auf Positionen beschränkt werden, die ähnliche Filterantworten wie die Positionen (x'_j, y'_j) in den Clusterzentren-Templates erzeugen. Durch den Streuungsfaktor wird bestimmt, welche Abweichung für die Filterantwort einer Position im Computerbild von der Filterantwort im Schwerpunkt des Objektes eines

[11] Für die Erzeugung der Trainingsmenge wurden solche Gabor-Templates durch Bildverarbeitung lediglich invertiert. Es muß also der folgende Ablauf der Lokalisierung für das gefilterte Eingabebild mit dem invertierten Bild wiederholt werden.

[12] Die Streuungen (Standardabweichungen) können durch die Clustervarianzen-Templates bestimmt werden: $\sigma'_{Cl_j}(x, y) = \sqrt{var'_{Cl_j}(x, y)}$, $\sigma''_{Cl_j}(x, y) = \sqrt{var''_{Cl_j}(x, y)}$.

Clusterzentren-Templates zulässig ist, damit die Poisition als möglicher Schwerpunkt des Objektes in Frage kommt. Abb. 5.14 (oben) zeigt die Gaborfilterantworten in der horizontalen (links) und der vertikalen (rechts) Mittellinie der Trainings-Templates aus Abb. 5.11. Da der Gaboroperator unimodal auf die Höhe und Breite eines Objektes reagiert, wird erwartet, daß die Maxima bzw. Minima der Filterantworten horizontal und vertikal in der Mitte der Clusterzentren-Templates liegen, also $x'_j = \frac{B}{2}$, $y'_j = \frac{H}{2}$, $j = 1, \ldots, N'$. Es ist zu erkennen, daß die Maxima *nicht* exakt in der Mitte liegen. Um dennoch eine genaue Lokalisierung des Objektes zu erzielen, wird später bei der Anwendung des trainierten HBF-Netzes für jede Position, an der die Templates verglichen werden sollen, ein *erweiterter Matching-Bereich* verwendet.
Die weiteren Graphiken dieser Abbildung zeigen die Gaborfilterantworten in den Clusterzentren-Templates in der horizontalen Linie (links) für $y = y'_j$, $j = 1, \ldots, 3$, und in der vertikalen Linie (rechts) für $x = x'_j$, $j = 1, \ldots, 3$, sowie die daraus resultierenden Toleranzbereiche bei einem Streuungsfaktor $F = 1$.

Seien nun die Gaborfilterantworten eines Eingabebildes mit I^2 Pixeln, in dem das Zielobjekt zu lokalisieren ist, durch folgende Funktion h gegeben:

$$h : \{0, \ldots, I-1\} \times \{0, \ldots, I-1\} \longrightarrow \mathbb{R}$$

Hierbei gebe $h(x, y)$ die Gaborfilterantwort der Position (x, y) des Bildes an. Mit dem Ziel der schnellen Lokalisierung durch Attention wird nun für jedes Clusterzentren-Template j, $j = 1, \ldots, N'$, ein Aufmerksamkeitsbereich durch die Positionen (x, y) festgelegt, für die $h(x, y) \in [min_{Tol_j}, max_{Tol_j}]$ gilt. Der Aufmerksamkeitsbereich eines Clusterzentren-Templates enthält somit alle Positionen, deren Gaborfilterantworten in dem Toleranzbereich dieses Templates liegen, also die möglichen Kandidaten für den Schwerpunkt des Objektes. Abb. 5.15 zeigt ein Gabor-gefiltertes Eingabebild mit den Aufmerksamkeitsbereichen der Clusterzentren-Templates aus Abb. 5.11 für einen Streuungsfaktor $F = 3$.
Es ist weiterhin noch eine Beschränkung der Aufmerksamkeitsbereiche erforderlich. Es können nämlich Positionen dieser Bereiche am Bildrand liegen. Da diese Positionen jedoch die Mittelpunkte von Templates darstellen und ein Matching mit einem Template, dessen Positionen außerhalb des Bildbereiches liegen, nicht möglich ist, werden die Aufmerksamkeitsbereiche auf Positionen (x, y) beschränkt, die folgende Bedingungen erfüllen müssen:

$$\begin{aligned} \left\lfloor \frac{B}{2} \right\rfloor \leq x \leq \left\lfloor (I-1) - \frac{B}{2} \right\rfloor \\ \left\lfloor \frac{H}{2} \right\rfloor \leq y \leq \left\lfloor (I-1) - \frac{H}{2} \right\rfloor \end{aligned} \tag{5.22}$$

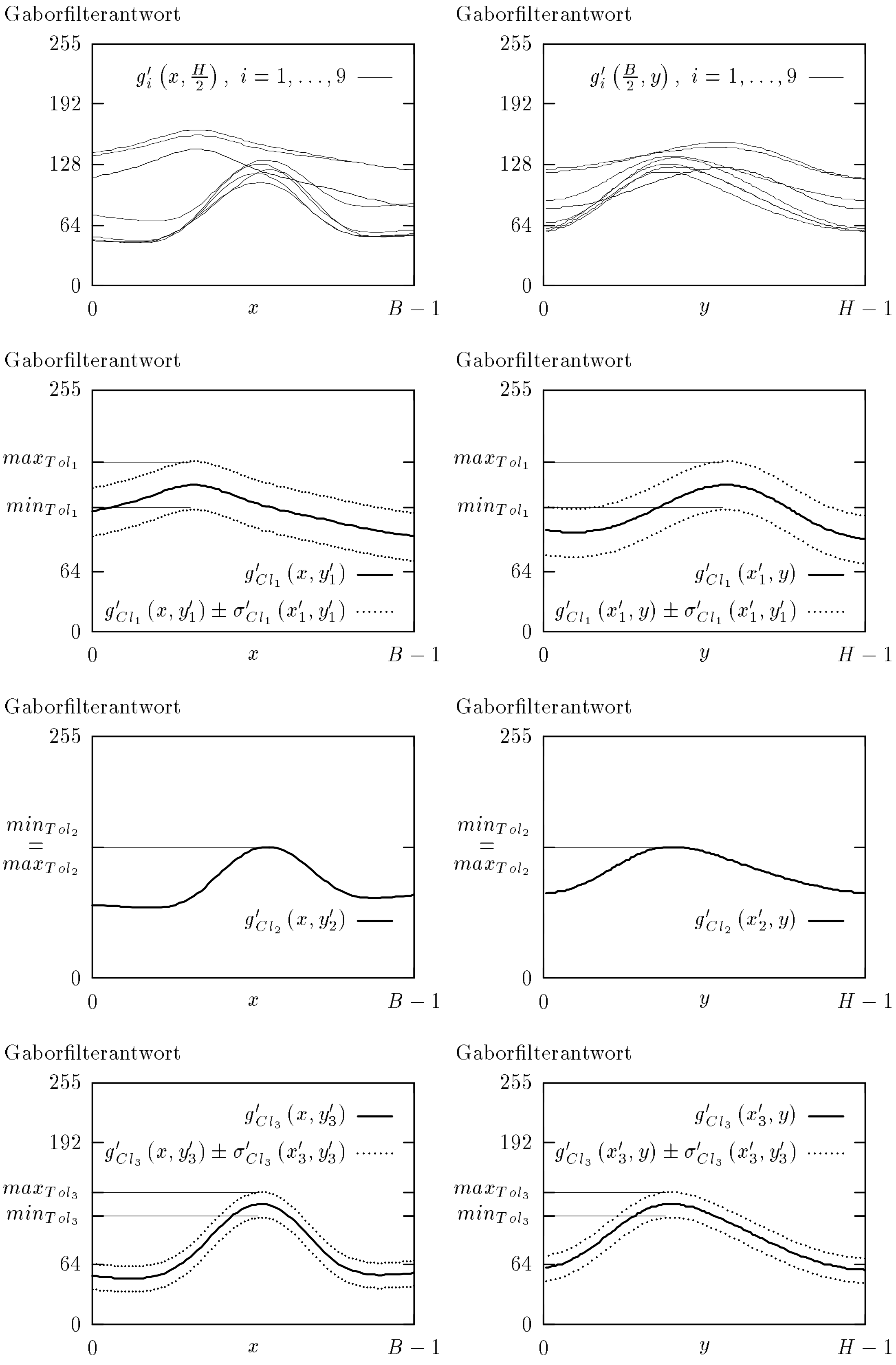


Abbildung 5.14: Die Gaborfilterantworten in der horizontalen (oben, links) und der vertikalen (oben, rechts) Mittellinie der Templates aus Abb. 5.11 und in der horizontalen (links) und vertikalen (rechts) Linie der Maxima der Clusterzentren-Templates.

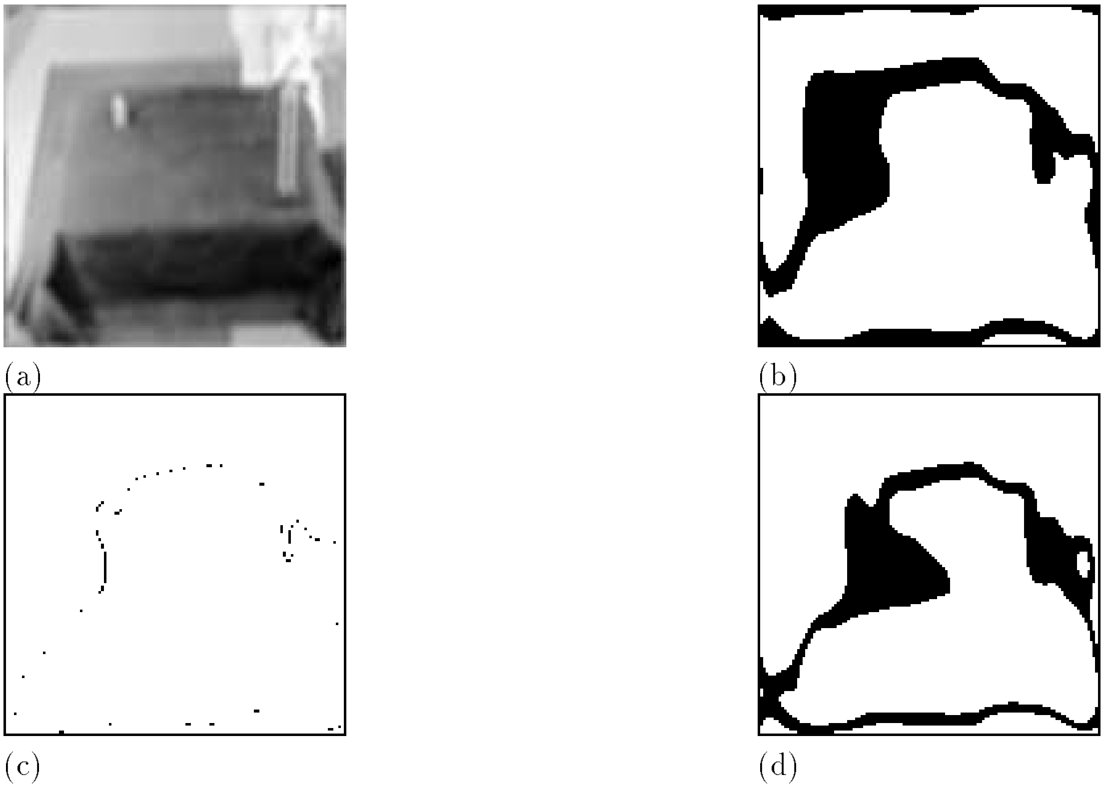


Abbildung 5.15: Ein Gabor-gefiltertes Eingabebild (a) mit seinen durch drei Clusterzentren-Templates bestimmten, schwarz gekennzeichneten Aufmerksamkeitsbereichen (b), (c), (d).

Dadurch wird gewährleistet, daß das B Pixel breite und H Pixel hohe Template, das mit einem Clusterzentren-Template gematcht werden soll, innerhalb des I^2 Pixel großen Eingabebildes liegt. Es muß dann außerdem für eine korrekte Lokalisierung vorausgesetzt werden, daß sich das Objekt vollständig im Eingabebild befindet.

Die Mengen der Positionen dieser Bereiche werden wie in Abschnitt 5.2.2 durch $\mathcal{A}_1,\ldots,\mathcal{A}_{N'}$ dargestellt. Die Positionen selbst werden mit (x_{q_p}, y_{q_p}), $q = 1, \ldots, N'$, $p = 1, \ldots, |\mathcal{A}_q|$, bezeichnet. Damit endet die 1. Stufe des zweistufigen Verfahrens. Der weitere Ablauf der Attention ist abhängig von der Wahl des Verfahrens.

Bei dem einstufigen Attentionsverfahren (s. Abschnitt 5.2.2) werden zunächst die ermittelten Aufmerksamkeitsbereiche zu einem umfassenden Bereich vereinigt, dessen Positionen die Elemente einer Menge $\mathcal{A}$ bilden ($\mathcal{A} = \mathcal{A}_1 \cup \ldots \cup \mathcal{A}_{N'}$). Für die Menge $\mathcal{L}$ der Lokalisierungspositionen beim einstufigen Attentionsverfahren gilt schließlich $\mathcal{L} = \mathcal{A}$.

In der 2. Stufe des zweistufigen Attentionsverfahrens wird für jeden Aufmerksamkeitsbereich genau eine Lokalisierungsposition ermittelt. Zu jeder Position (x_{q_p}, y_{q_p}) existiert im Eingabebild ein Template mit seinem Mittelpunkt an dieser

Position. Wird das Eingabebild mit dem Operator gefiltert, der beim Trainieren des Netzes verwendet wurde, können die Filterantworten der durch die Positionen (x_{q_p}, y_{q_p}) bestimmten Templates mit folgenden Funktionen h_{q_p}, $q = 1, \ldots, N'$, $p = 1, \ldots, |\mathcal{A}_q|$, beschrieben werden:

$$
\begin{aligned}
&h_{q_p} : \{0, \ldots, B-1\} \times \{0, \ldots, H-1\} \longrightarrow \mathbb{R} \\
&h_{q_p}(x, y) = h\left(x_{q_p} - \left\lfloor \tfrac{B}{2} \right\rfloor + x, y_{q_p} - \left\lfloor \tfrac{H}{2} \right\rfloor + y\right)
\end{aligned}
\tag{5.23}
$$

Da $\left(x_{q_p} - \left\lfloor \frac{B}{2} \right\rfloor, y_{q_p} - \left\lfloor \frac{H}{2} \right\rfloor\right)$ die Position der linken oberen Ecke eines Templates im Eingabebild darstellt, gibt $h_{q_p}(x, y)$ die Filterantwort in diesem Template an der Position (x, y) an. Gesucht wird für jeden Aufmerksamkeitsbereich das Template, welches am besten mit dem Clusterzentren-Template der 2. Stufe matcht. Es ergibt sich eine Lokalisierungsposition $(x_q, y_q) \in \left\{(x_{q_p}, y_{q_p}) \mid p = 1, \ldots, |\mathcal{A}_q|\right\}$ für jedes Cluster q, $q = 1, \ldots, N'$, die bei Einsetzung von $g(x,y) := h_{q_p}(x,y)$, $g_{Cl_j}(x,y) := g''_{Cl_q}(x,y)$ und $\sigma^2_{Cl_j}(x,y) := \sigma''^2_{Cl_q}(x,y)$ in (5.7) eine maximale Aktivierung r_q impliziert. D.h. es gilt für $q = 1, \ldots, N'$:

$$
\begin{aligned}
&(x_q, y_q) = (x_{q_{p'}}, y_{q_{p'}}), \\
&p' = \arg\left\{\max_{p \in \{1, \ldots, |\mathcal{A}_q|\}} \left\{\exp\left[-\sum_{x=0}^{B-1}\sum_{y=0}^{H-1} \frac{\left[h_{q_p}(x,y) - g''_{Cl_q}(x,y)\right]^2}{2\sigma''^2_{Cl_q}(x,y)}\right]\right\}\right\}
\end{aligned}
\tag{5.24}
$$

Für die Menge $\mathcal{L}$ der Lokalisierungspositionen beim zweistufigen Attentionsverfahren gilt schließlich $\mathcal{L} = \{(x_1, y_1), \ldots, (x_{N'}, y_{N'})\}$.

5.4.3 Exemplarisches Anwenden der Objekterkennungsfunktion

Bei der Anwendung des trainierten HBF-Netzes wird für jede der in Abschnitt 5.4.2 ermittelten Lokalisierungspositionen ein Zuverlässigkeitswert berechnet, der ausdrückt, mit welcher Wahrscheinlichkeit es sich an dieser Position um das Zielobjekt handelt. Es findet somit eine genaue Lokalisierung statt.
Weiterhin wird mit einem Schwellwert sw bestimmt, ob es sich an dieser Position wirklich um das Objekt handelt. Liegt der Zuverlässigkeitswert unterhalb der Schwelle, muß angenommen werden, daß sich das Objekt nicht im Bild befindet, da sich für alle anderen Positionen geringere Wahrscheinlichkeiten ergeben.

Um das in Abschnitt 5.4.2 beschriebene Problem der ungenauen Bestimmung von Lokalisierungspositionen zu beheben, wird für jede Lokalisierungsposition (x, y) folgender erweiterter Matching-Bereich $\mathcal{M}(x, y)$ festgelegt (Abb. 5.16):

$$\begin{aligned} &\mathcal{M}(x,y) \\ &= \left\{ (\lfloor \bar{x} \rfloor, \lfloor \bar{y} \rfloor) \,\middle|\, \bar{x} \in \left[x - \left\lfloor \tfrac{B}{2} \right\rfloor, x + \left\lfloor \tfrac{B}{2} \right\rfloor \right],\ \bar{y} \in \left[y - \left\lfloor \tfrac{H}{2} \right\rfloor, y + \left\lfloor \tfrac{H}{2} \right\rfloor \right] \right\} \end{aligned} \tag{5.25}$$

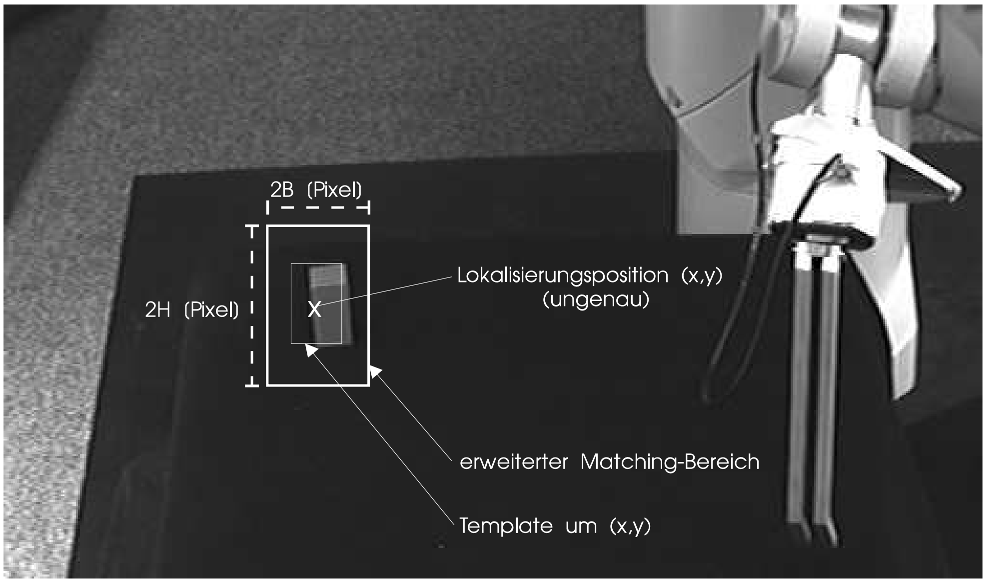


Abbildung 5.16: Die Erweiterung des Matching-Bereiches bei der Anwendung eines HBF-Netzes.

Es werden nun alle Positionen der Matching-Bereiche in einer Menge $\mathcal{M}'$ zusammengefaßt:

$$\mathcal{M}' = \bigcup_{(x,y)\in\mathcal{L}} \mathcal{M}(x,y) = \left\{ (x'_1, y'_1), \ldots, (x'_{|\mathcal{M}'|}, y'_{|\mathcal{M}'|}) \right\} \tag{5.26}$$

Eine Filterung des Computerbildes mit dem beim Trainieren des Netzes verwendeten Operator liefert nun Templates, deren Mittelpunkte mit den Positionen aus $\mathcal{M}'$ identisch sind. Sie werden durch folgende Funktionen $h_{q'}$, $q' = 1, \ldots, |\mathcal{M}'|$, beschrieben:

$$\begin{aligned} &h_{q'} : \{0, \ldots, B-1\} \times \{0, \ldots, H-1\} \longrightarrow \mathbb{R} \\ &h_{q'}(x,y) = h\left(x'_{q'} - \left\lfloor \tfrac{B}{2} \right\rfloor + x, y'_{q'} - \left\lfloor \tfrac{H}{2} \right\rfloor + y \right) \end{aligned} \tag{5.27}$$

Durch die Einsetzung von $g := h_{q'}$ in (5.7) können nach (5.8) für alle Positionen der Menge $\mathcal{M}'$ mit dem trainierten HBF-Netz Zuverlässigkeiten $c_{q'}$, $q' = 1, \ldots, |\mathcal{M}'|$, berechnet werden, die eine Wahrscheinlichkeit dafür angeben, daß sich das Zielobjekt in den durch die Positionen bestimmten Templates befindet.
Die Position $(x_{lok}, y_{lok}) \in \mathcal{M}'$, die eine maximale Zuverlässigkeit c_{max} impliziert, gibt den Ort des lokalisierten Objektes an. Der Wert c_{max} bestimmt weiterhin, ob das Objekt wirklich erkannt wurde, da das HBF-Netz mit Ausgaben $c = 1$ für das zu lokalisierende Objekt trainiert wurde. Es kann also mit Hilfe eines Schwellwertes $sw < 1$ entschieden werden, ob es sich um das Objekt handelt, falls $c_{max} \geq sw$ gilt, oder nicht für $c_{max} < sw$. Bei Benutzung des Stereokamerasystems ergibt sich damit für jedes der beiden Computerbilder solch eine Position (x_{lok}, y_{lok}) mit einem Zuverlässigkeitswert c_{max}. Der Ort des lokalisierten Objektes wird dann mit den Koordinaten $(x_{b_{l/r}}, y_{b_{l/r}})$ beschrieben, die die Zuverlässigkeitswerte $c_{max_{l/r}}$ implizieren.

5.4.4 Vor- und Nachteile bei der Verwendung unterschiedlicher Operatoren

Obwohl die beiden beschriebenen Lernstrategien jeweils für ihre aufgabenorientierte Verwendung gerechtfertigt sind, ergeben sich dadurch Vor- und Nachteile. Ein Nachteil der LmU-Strategie ist darin zu sehen, daß die Basisfunktionen lediglich ein Matching zwischen den Gaborfilterantworten in einem Bildbereich und den Gaborfilterantworten in einem Clusterzentren-Template beinhalten, während die Kantenstruktur des Objektes weitgehend vernachlässigt wird. So kann z.B. die Basisfunktion eines Clusterzentren-Templates eine hohe Aktivierung für ein Eingabe-Template liefern, obwohl dieses das Zielobjekt nicht enthält. Dieser unerwünschte Fall tritt auf, wenn die Bildpunkte der ungefilterten Eingabe-Templates ähnliche Grauwerte besitzen wie die Bildpunkte der ungefilterten Clusterzentren-Templates. Dagegen würde ein Eingabe-Template, welches das Objekt enthält, eine geringe Aktivierung bewirken, wenn die Grauwerte der ungefilterten Templates signifikant voneinander abweichen. Ein zusätzlicher Vergleich der Kanten in den Templates würde dieses Problem also beheben.
Bei der LoU-Lernstrategie werden durch die Verwendung des Gradientenoperators Informationen über die Kanten in den zu vergleichenden Templates berücksichtigt. Hier ergibt sich jedoch ein Nachteil dadurch, daß bei Verwendung der zweistufigen Attention die gesuchte Position des Objektes genau eine der Lokalisierungspositionen sein muß. Durch eine ungenaue Lokalisierungsposition würden beim Matching die Kanten im Clusterzentren-Template nicht mit denen im Eingabe-Template übereinstimmen, wodurch eine geringe Aktivierung resultieren würde. Die in Abschnitt 5.4.3 beschriebene Methode der Erweiterung des Matching-Bereiches versucht, dieses Problem zu beheben.

5.5 Versuche

In diesem Abschnitt sollen die Diskriminanz- und Invarianzeigenschaften unterschiedlich konfigurierter HBF-Netze untersucht werden. In einem weiteren Versuch wird gezeigt, daß bei der LmU-Lernstrategie der Gaboroperator zum Trainieren des Netzes geeigneter ist als der Gradientenoperator.

5.5.1 Diskriminanz- und Invarianzeigenschaften

In diesem Versuch werden zum Trainieren des HBF-Netzes Templates eines Objektes in unterschiedlichen Orientierungen benutzt. Diese Orientierungen ergeben sich durch Drehen des Objektes von 0° bis 337.5° um jeweils 22.5°. Abb. 5.17 zeigt die 16 resultierenden Trainingsbilder.

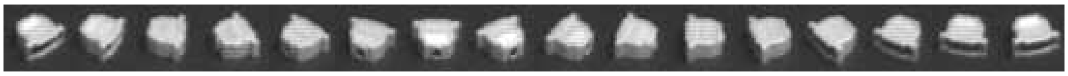
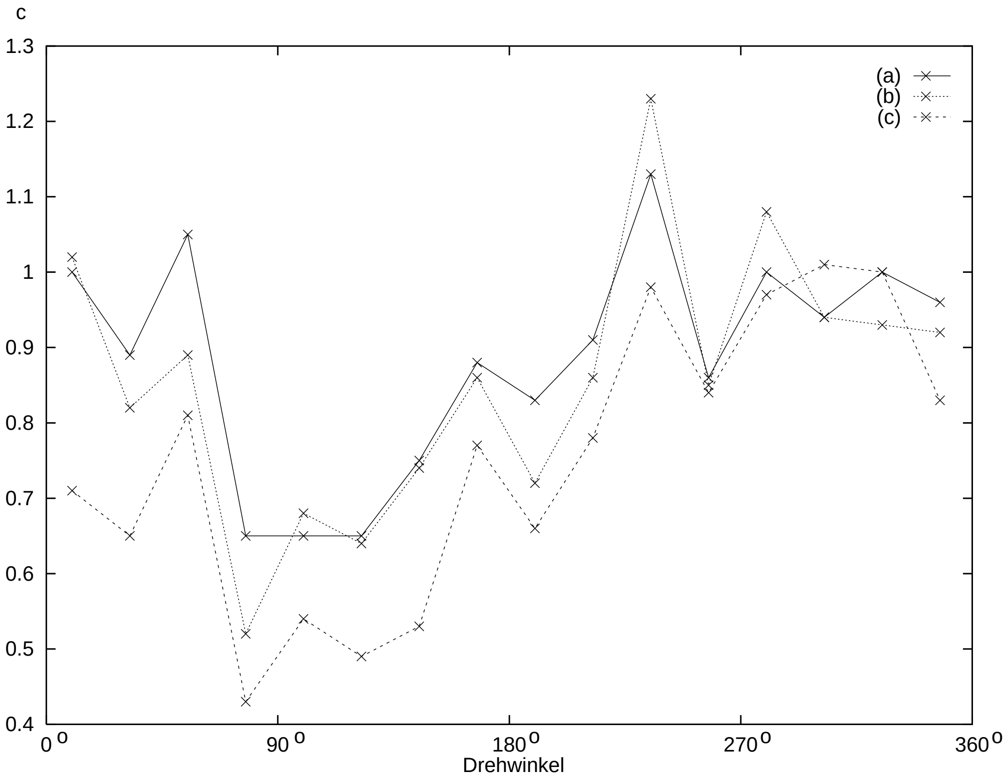
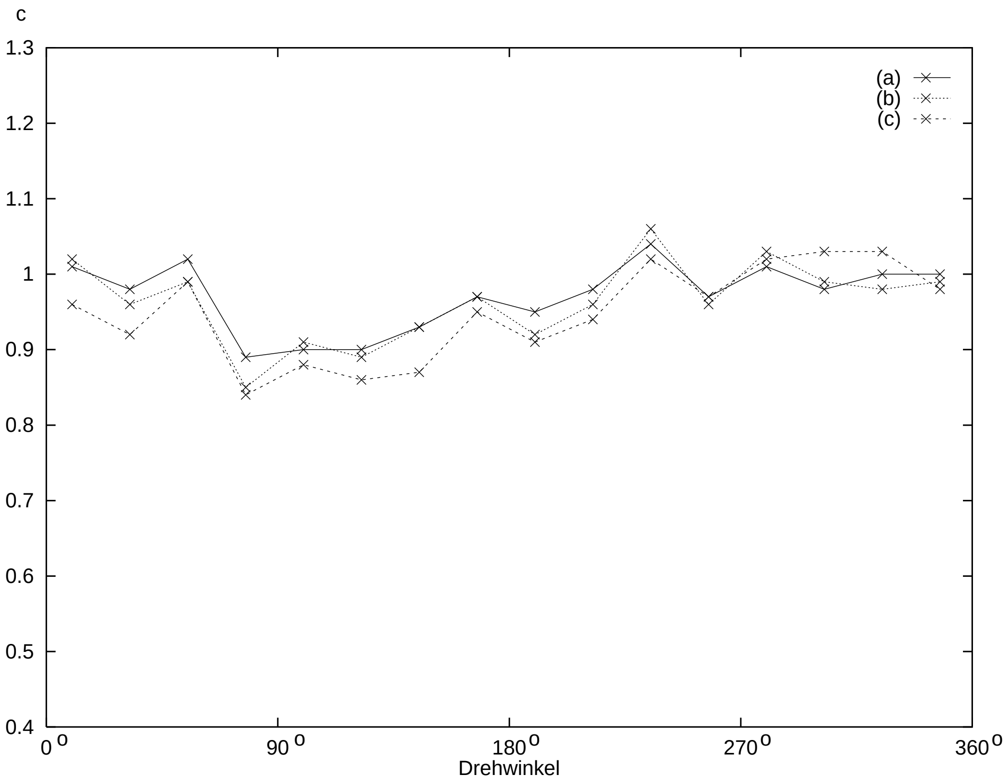


Abbildung 5.17: Die Trainingsbilder eines Zielobjektes.

Um das Invarianzkriterium zu überprüfen, wurde nun untersucht, wie signifikant sich unterschiedlich hohe Normierungsparameter aus (5.7) und die Anzahl der Basisfunktionen auf die Zuverlässigkeitswerte in 16 Testbildern an der Position des Objektes auswirkten.
Die Anzahl der Basisfunktionen betrug 3, 4 und 8. Um geringe bzw. hohe Normierungsparameter zu erhalten, wurden diese lediglich mit einem *Varianzfaktor* $f_{var} = 5$ bzw. $f_{var} = 80$ multipliziert.
Das Lokalisieren und Erkennen des Objektes geschah mit der LoU-Lernstrategie aus Abschnitt 5.4. Hierfür wurden Testbilder gewählt, die das Objekt bei einem *Drehwinkel* von 10° bis 347.5° mit einer Schrittweite von 22.5° enthielten. In diesen Bildern wurde an der Position des Objektes jeweils durch Anwendung des trainierten HBF-Netzes der Zuverlässigkeitswert ermittelt. Abb. 5.18 zeigt das Ergebnis dieses Versuches. Es ist deutlich zu erkennen, daß die Zuverlässigkeitswerte beim hohen Varianzfaktor (B) nicht stark voneinander abweichen und ungefähr bei 1 liegen, während beim kleinen Varianzfaktor (A) die Werte schwanken, wodurch das Invarianzkriterium nicht mehr erfüllt wird. Die teilweise besseren Werte bei einer geringeren Anzahl an Basisfunktionen läßt sich dadurch begründen, daß die in (5.5) berechneten Normierungsparameter bei wenigen Basisfunktionen größer ausfallen als bei vielen. Jedoch darf das Problem der Übergeneralisierung (s. Kapitel 3, Abschnitt 3.2.1) nicht mißachtet werden und deshalb die Anzahl der Basisfunktionen nicht zu niedrig sein.



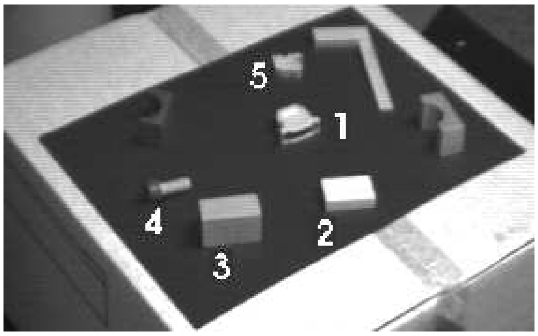
(A)



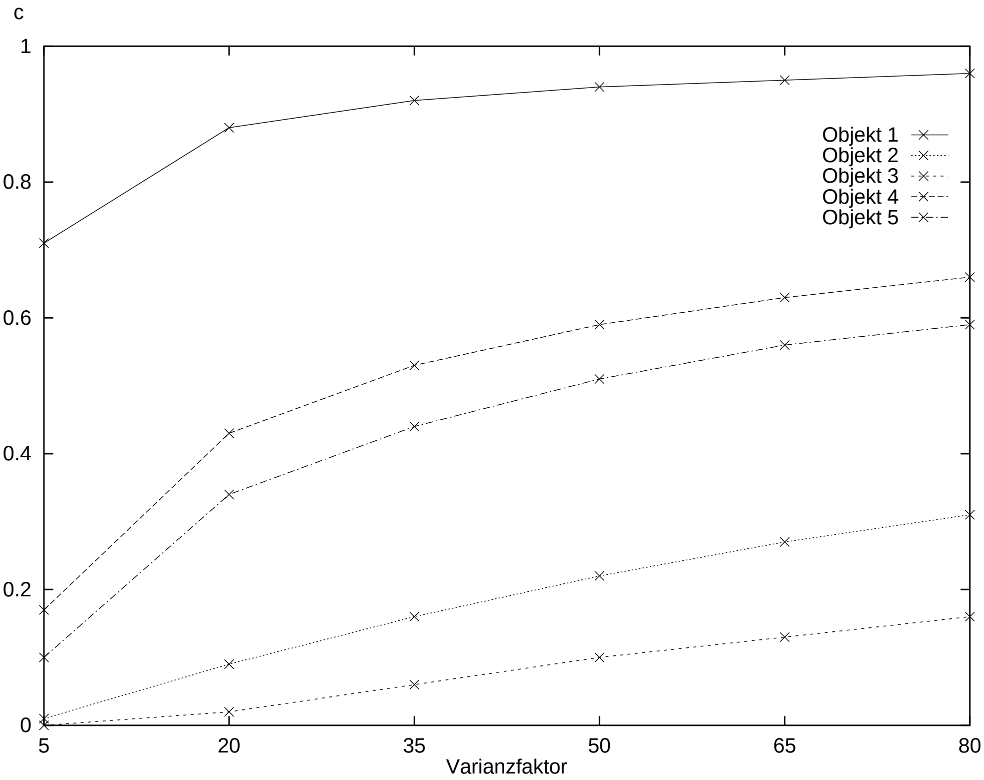
(B)

Abbildung 5.18: Die berechneten Zuverlässigkeiten c für das Zielobjekt in den Testbildern an der richtigen Position bei 3 (a), 4 (b) und 8 (c) Basisfunktionen und einem Varianzfaktor $f_{var} = 5$ (A) und $f_{var} = 80$ (B).

Um das Diskriminanzkriterium zu überprüfen, wurde untersucht, wie signifikant die Zuverlässigkeitswerte in einem Testbild (Abb. 5.19 (A)) an Positionen von vier unterschiedlichen Objekten vom Zuverlässigkeitswert an der Position des Zielobjektes abweichen (Abb. 5.19 (B)). Hierfür wurden 8 Basisfunktionen gewählt.



(A)



(B)

Abbildung 5.19: Ein Testbild (A) und die berechneten Zuverlässigkeiten c für das Zielobjekt in diesem Testbild an den Positionen $P_1, \ldots, P_5$, die jeweils die Mittelpunkte der fünf Objektes aus (A) darstellen, für unterschiedlich große Varianzfaktoren und 8 Basisfunktionen (B).

Aus der Abbildung geht hervor, daß bei einem niedrigen Varianzfaktor das Zielobjekt von den anderen Objekten besser unterschieden werden konnte als bei einem hohen[13].

5.5.2 Lernen mit oder ohne Umgebung

In dem folgenden Versuch soll demonstriert werden, daß für das Lernen eines Objektes mit seiner Umgebung (LmU-Lernstrategie) beim Trainieren des HBF-Netzes der Gaboroperator geeigneter als der Gradientenoperator ist. Die LmU-Lernstrategie wird z.B. verwendet, wenn eine typische Greifstellung des Manipulators gelernt werden soll, in der zum Greifen des Objektes lediglich das Schließen der Greiffinger erforderlich ist.
Um eine typische Greifstellung zu lernen, wird eine Trainingsmenge mit positiven Beispielen benötigt. Es werden also M Trainingsbilder erzeugt, die den Greifer des Manipulators und ein Objekt in mehreren möglichen Greifstellungen zeigen. Nach der in Abschnitt 5.4.1 beschriebenen Methode werden nun die Bilder mit einem der Operatoren $\mathcal{O}_{Gab}$ oder $\mathcal{O}_{Grad}$ gefiltert und die Templates, die das Objekt enthalten, mit den zugehörigen Ausgabewerten ($c_1 = \ldots = c_M = 1.0$) in die Trainingsmenge aufgenommen.
Nach dem Trainieren eines HBF-Netzes mit der Trainingsmenge wird anhand von Testbildern überprüft, welcher Operator für das Lernen der Greifstellung geeigneter ist. Hierfür wird jeweils an der Position des Objektes durch Anwendung des Netzes der Zuverlässigkeitswert berechnet. Wenn sich für Testbilder, in denen sich der Manipulator in einer Greifstellung befindet, hohe Zuverlässigkeitswerte und für die übrigen Testbilder niedrige Zuverlässigkeitswerte ergeben, wurde mit einem geeigneten Operator trainiert.

Für die Erzeugung von 14 Testbildern (Abb. 5.20) wurde der Manipulator so gesteuert, daß sich der Greifer vertikal im Computerbild bewegte, wobei in den Bildern (6)-(9) eine optimale, zu lernende Greifstellung erreicht wurde.
Die Bilder (6)-(9) wurden jeweils mit einem der Operatoren $\mathcal{O}_{Gab}$ (LmU-Lernstrategie) oder $\mathcal{O}_{Grad}$ (LoU-Lernstrategie) gefiltert und zum Trainieren des HBF-Netzes verwendet. Bei der Clusterung der vier Trainings-Templates wurden jeweils zwei Gruppierungen gebildet.
Die bei der Anwendung des Netzes resultierenden Zuverlässigkeitswerte sind in Abb. 5.21 zu sehen. Es ist zu erkennen, daß bei der Verwendung des Gaboroperators (LmU-Lernstrategie) die Kurve der Zuverlässigkeiten für die Testbilder (6)-(9) ihren maximalen Wert erreicht und bei Erhöhung bzw. Verringerung der Testbildnummer gleichförmig abfällt. Mit zunehmender Entfernung des Manipulators von seiner Greifstellung verringert sich also der Zuverlässigkeitswert. Es kann somit eine optimale Greifstellung des Manipulators erkannt werden.

[13] Für $f_{var} = 5$ ergibt sich z.B. c(Objekt1)$-c$(Objekt4)$= 0.71 - 0.17 = 0.54$ und für $f_{var} = 80$ c(Objekt1)$-c$(Objekt4)$= 0.96 - 0.66 = 0.30$.

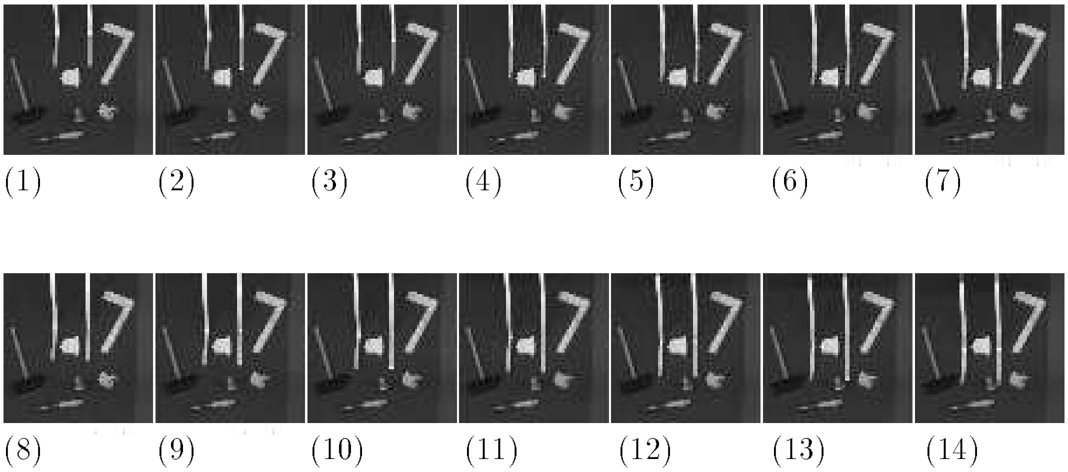


Abbildung 5.20: Testbilder zur Analyse der unterschiedlichen Lernstrategien. Die Bilder (6)-(9) wurden zum Trainieren des HBF-Netzes verwendet.

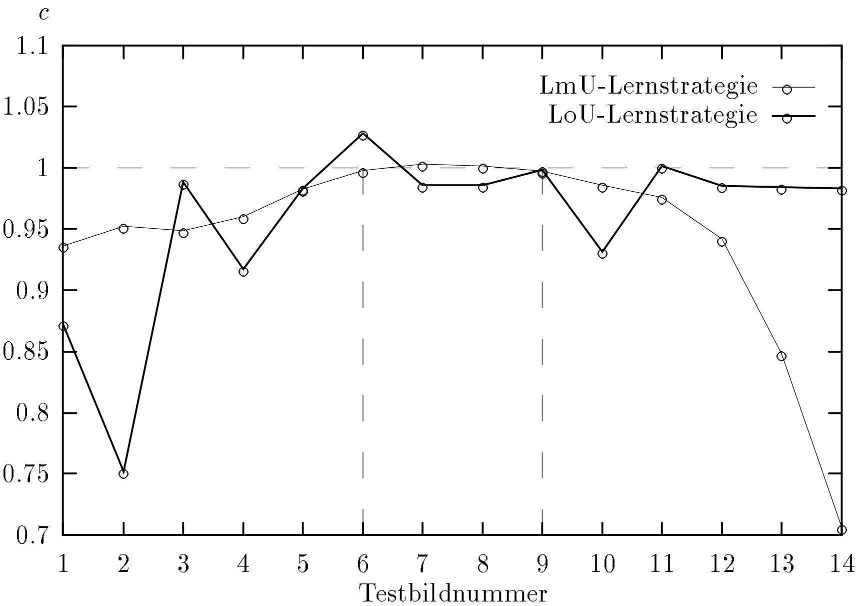


Abbildung 5.21: Die resultierenden Zuverlässigkeitswerte c an der Position des Objektes in den Testbildern bei unterschiedlichen Lernstrategien.

Bei der Verwendung des Gradientenoperators (LoU-Lernstrategie) ergibt sich dagegen *kein* gleichförmiger Kurvenverlauf der Zuverlässigkeitswerte mit ausgeprägtem Maximum. Da hier die Trainings-Templates annähernd identisch[14] sind, resultieren bei den Testbildern ähnliche Zuverlässigkeitswerte.
Zum Lernen einer typischen Greifstellung des Manipulators ist somit der Gabor-

[14]Identische Templates ergeben sich nur, wenn die Bewegung des Manipulators keine Lichtreflexionen am Objekt bewirkt. Da diese Erscheinungen nicht auszuschließen sind, ergeben sich die Schwankungen im Kurvenverlauf.

operator geeigneter als der Gradientenoperator.

Mit dem in Kapitel 4 erläuterten Verfahren zur Kalibrierung zweier Kameras durch ein RBF-Netz und dem in diesem Kapitel beschriebenen Verfahren zur Objektlokalisierung und -erkennung durch ein HBF-Netz können nun die Bestandteile der Komponenten des 'Online'-Betriebes aus Abb. 1.1 durch ein Schaubild genauer dargestellt werden (Abb 5.22).

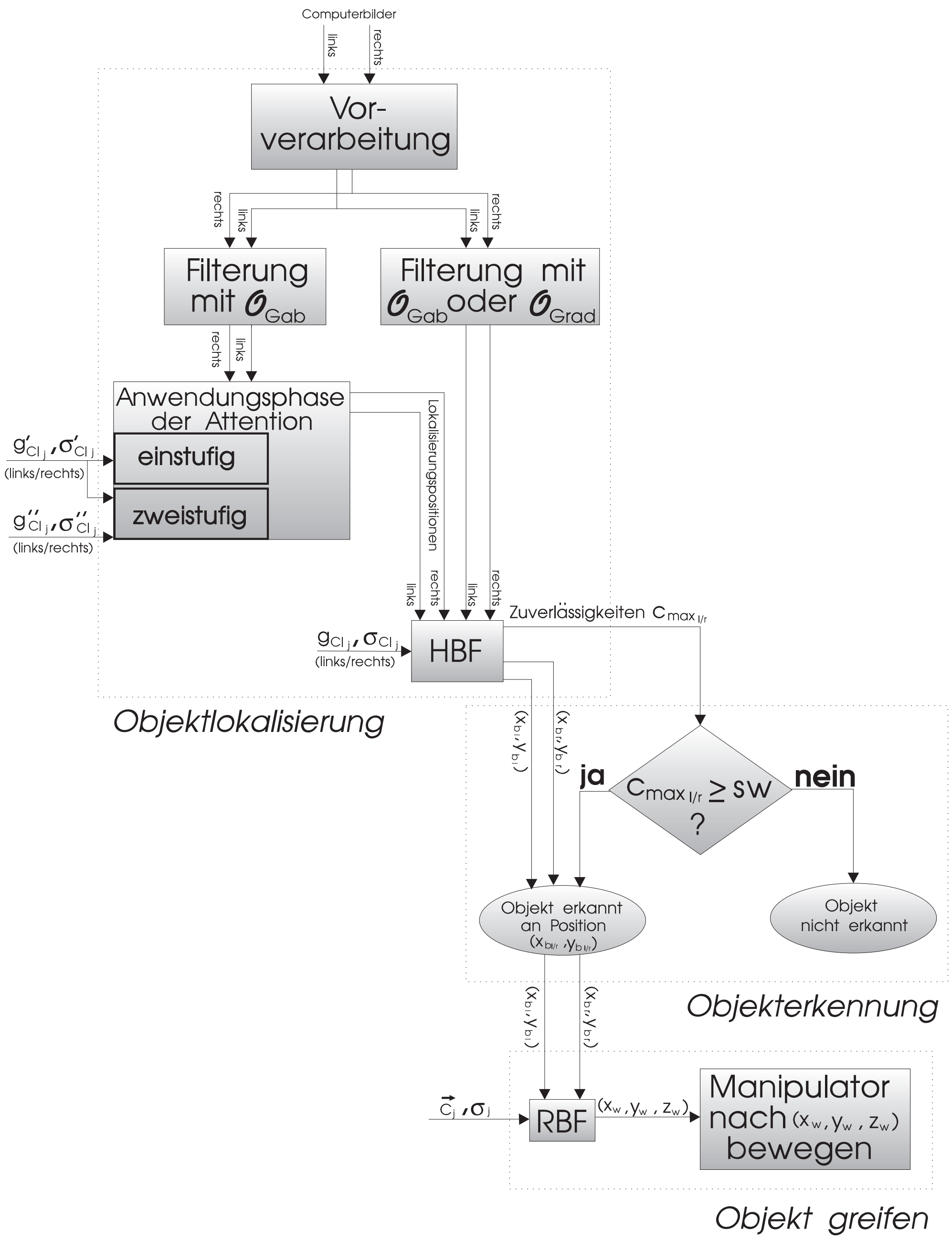


Abbildung 5.22: Komponenten des 'Online'-Betriebes.

5.6 Diskussion

In diesem Abschnitt soll diskutiert werden, ob es mögliche Alternativen zu der beschriebenen Methode der Objektlokalisierung und -erkennung gibt, die effizienter oder zuverlässiger sind.
In Kapitel 4 wurde z.B. gezeigt, daß durch die Verwendung normierter RBF-Netze eine zuverlässigere Manipulator-Einzelkamera-Kalibrierung erzielt werden kann. Es muß daher überlegt werden, ob auch bei der Objektlokalisierung eine Normierung der HBF-Netze möglich ist.
Eine weitere Alternative ergibt sich, indem das Zielobjekt nicht durch seine photometrischen, sondern durch seine geometrischen Eigenschaften beschrieben wird. Hierfür kann die im Anhang B beschriebene Hough-Transformation verwendet werden.

5.6.1 Alternative durch Hough-Transformation

Bei der LoU-Lernstrategie kann alternativ zur Beschreibung eines Objektes durch seine photometrischen Eigenschaften auch eine Beschreibung durch seine geometrischen Eigenschaften erfolgen (vgl. Abschnitt 5.1). Hierfür werden wie bisher Trainings-Templates durch Anwenden eines Operators auf die Trainingsbilder erzeugt. Es wird dabei jedoch der GS-Operator $\mathcal{O}_{GS}$ verwendet. Abb. 5.23 zeigt die Trainings-Templates für die gefilterten Trainingsbilder aus Abb. 5.2.

Die Eingabevektoren des Netzes werden bei diesem Ansatz nicht durch die Filterantworten gebildet, sondern durch die Orientierungen und Längen von Kanten in den Templates. Für jedes Template können wie in Anhang B Orientierungshistogramme erzeugt werden, die die Längen und Orientierungen von auftretenden Kanten charakterisieren. Dadurch ergibt sich zu jedem Template ein Vektor $\vec{x}$, der wie in (B.10) definiert ist.
Sei wie bisher M die Anzahl der Trainings-Templates. Dann können nach (B.10) jeweils die charakteristischen Vektoren $\vec{x}_i$, $i = 1, \ldots, M$, gebildet werden. Sie stellen die Eingabevektoren der Trainingsmenge dar. Ein Vektor $\vec{x}$ kann wie in Abschnitt 5.2.1 (vgl. mit (5.2)) durch folgende Funktionen g beschrieben werden, wobei sich hier jedoch wegen der Verwendung der geometrischen Eigenschaften die Definitionsbereiche ändern:

$$g : D_x \times D_y \longrightarrow \mathbb{R},\ D_x = \{0, 1\},\ D_y = \{0, \ldots, 179\}$$

$$g(x, y) = \begin{cases} max(y) & : \quad x = 0 \\ mw(y) & : \quad x = 1 \end{cases} \tag{5.28}$$

Bisher wurde ein Objekt durch die photometrischen Eigenschaften beschrieben. Deshalb ergab sich für die Definitionsbereiche $D_x = \{0, \ldots, B-1\}$ und

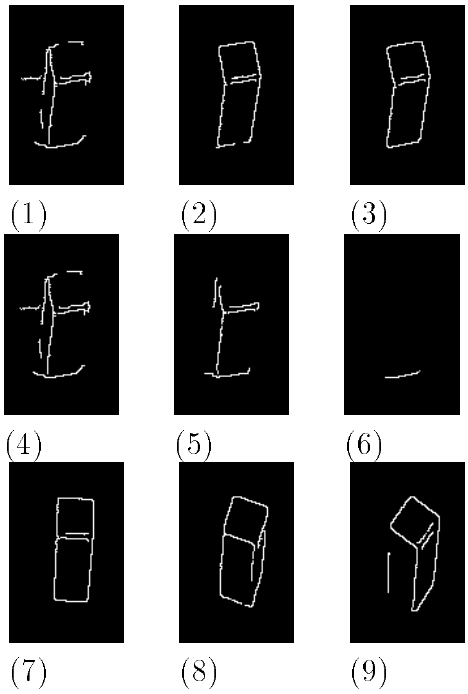


Abbildung 5.23: Trainings-Templates (1)-(9) bei der LoU-Lernstrategie durch Filterung mit dem GS-Operator.

$D_y = \{0, \ldots, H-1\}$, wobei H und B die Höhe und Breite eines Templates in Pixeln darstellten. Bei der Beschreibung eines Templates durch die Funktion g in (5.28) gibt der Funktionswert $g(x, y)$ den Wert eines Orientierungshistogramms für einen Winkel[15] y an. Dabei stellt x einen Indikator dar, der bestimmt, ob es sich um das Orientierungshistogramm der maximalen Grauwerte ($x = 0$) oder der gemittelten Grauwerte ($x = 1$) handelt.
Der Ablauf des Trainierens kann somit wie in Abschnitt 5.2.1 erfolgen, indem $B = 2$ und $H = 180$ gesetzt wird. Die Funktionen gemäß (5.28) für die einzelnen Trainingsvektoren werden mit $g_1, \ldots, g_M$ bezeichnet. Weiterhin werden zum Trainieren des HBF-Netzes die Ausgabewerte $c_i \equiv 1$, $i = 1, \ldots, M$, und die Clusterzentren benötigt. Die Clusterzentren können durch Gruppierung der Eingabevektoren $\vec{x}_i$, $i = 1, \ldots, M$, mit einem Cluster-Algorithmus bestimmt werden. Dadurch ergeben sich außerdem die Varianzen in einem Cluster.
Wie in Abschnitt 5.2.1 werden die N resultierenden Clusterzentren und -varianzen durch Funktionen g_{Cl_j} und var_{Cl_j}, $j = 1, \ldots, N$, gemäß (5.4) und (5.5) beschrieben. Der ISODATA-Algorithmus lieferte für die charakteristischen Vektoren der Trainings-Templates aus Abb. 5.23 drei Cluster mit den Indexmengen

[15] Um eine analoge Formel wie in Abschnitt 5.2.1 zu erhalten, wird der Winkel hier mit y statt mit α bezeichnet und seine Maßeinheit (°) ausgelassen.

$C_1 = \{1, 4, 7\}$, $C_2 = \{2, 3, 5, 8, 9\}$ und $C_3 = \{6\}$. Abb. 5.24 zeigt die zugehörigen Orientierungshistogramme.

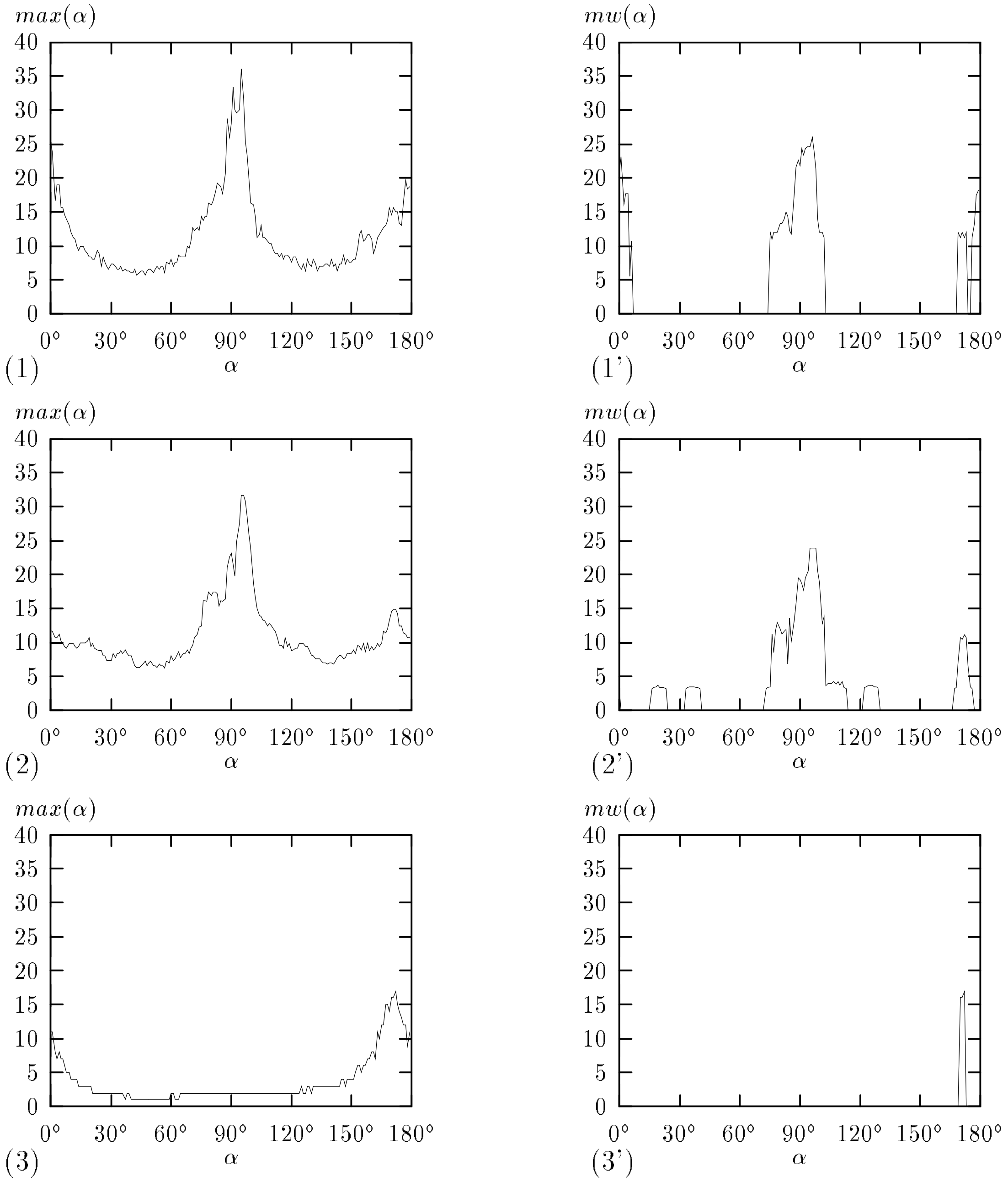


Abbildung 5.24: Orientierungshistogramme der drei Clusterzentren für die Maxima (1), (2), (3) und für die Mittelwerte (1'), (2'), (3').

Der Ablauf des Trainierens des HBF-Netzes sowie die Lokalisierung und Erkennung durch Attention und Anwendung des Netzes erfolgt unter Berücksichtigung der beschriebenen Änderungen analog wie in Abschnitt 5.4.

Versuch

In diesem Versuch soll die Robustheit eines HBF-Netzes analysiert werden, das mit dem beschriebenen Verfahren trainiert wurde. Hierfür wurde bei der Erzeugung von 12 Trainings- und 12 Testbildern eine kontinuierliche Beleuchtungsänderung vorgenommen. Das Zielobjekt entsprach dabei dem aus Abb. 5.2. Die Mengen wurden so angelegt, daß bei zunehmender Beleuchtung abwechselnd ein Bild als Trainings- und eines als Testbild verwendet wurde.
Für das Trainieren des Netzes wurden die Werte der Orientierungshistogramme der Templates mit dem Zielobjekt und die Zuverlässigkeitswerte in die Trainingsmenge aufgenommen. Eine Gruppierung mit dem ISODATA-Clusteralgorithmus ergab drei Cluster.
Da es sich um das gleiche Objekt wie in Abb. 5.2 handelt, ergaben sich für die Trainings-Templates und Clusterzentren-Templates ähnliche Orientierungshistogramme wie in Abb. 5.24. Wie bisher wurden anschließend die Gewichte der 2. Schicht des Netzes mit der Trainingsmenge trainiert.
Bei Anwendung des Netzes auf die Position des Objektes in den Trainings- und Testbildern ergaben sich Zuverlässigkeitswerte, die in Abb. 5.25 zu sehen sind.

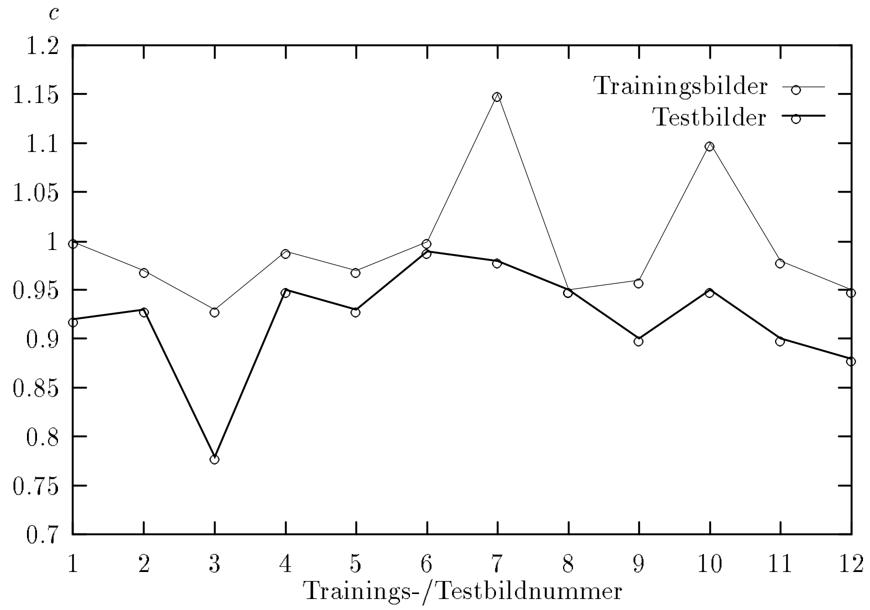
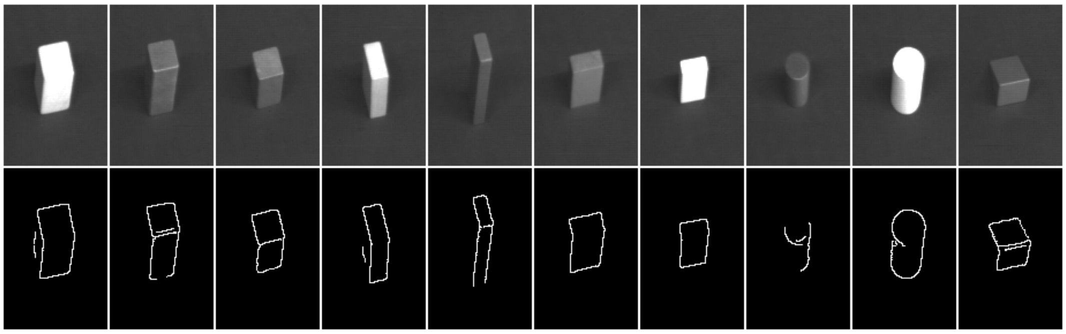


Abbildung 5.25: Die Zuverlässigkeitswerte an der Position des Objektes in den Trainings- und Testbildern.

Es ist zu erkennen, daß sich in den Testbildern zwar geringere Zuverlässigkeitswerte als in den Trainingsbildern ergeben, alle Werte jedoch über dem Wert 0.7 liegen. Es kann also für die Erkennung des Objektes ein Schwellwert $sw = 0.7$ bestimmt werden, mit dem entschieden wird, ob es sich an einer Position um das Objekt handelt (bei $c \geq sw$) oder nicht (bei $c < sw$).

Mit einer weiteren Testbildmenge sollte nun analysiert werden, ob mit der gelernten Erkennungsfunktion das Zielobjekt von anderen Objekten unterschieden werden konnte. Abb. 5.26 zeigt zehn unterschiedliche Objekte, an deren Position das HBF-Netz angewendet wurde. In Tabelle 5.1 sind die daraus resultierenden Zuverlässigkeitswerte zu sehen.



(1) (2) (3) (4) (5) (6) (7) (8) (9) (10)

Abbildung 5.26: Bilder mit dem Zielobjekt (2) und weiteren Objekten sowie die zugehörigen Kantenbilder.

Objektnummer	Zuverlässigkeitswert
1	0.87
2	1.02
3	0.73
4	0.70
5	0.71
6	0.65
7	0.67
8	0.64
9	0.61
10	0.25

Tabelle 5.1: Zuverlässigkeitswerte für die unterschiedlichen Objekte.

Aus der Tatsache, daß lediglich die Objekte 6-10 einen Zuverlässigkeitswert $c < sw$ implizieren, wird gefolgert, daß sich diese Methode der Beschreibung eines Objektes durch seine geometrischen Eigenschaften nicht besonders zur Unterscheidung unterschiedlicher Objekte eignet. Es ist allerdings zu beachten, daß sich wegen der ähnlichen Form und somit ähnlicher Kantenorientierungen der Objekte 1-5 nicht signifikant voneinander abweichende Zuverlässigkeitswerte ergeben.

Außerdem ist festzustellen, daß die Objektfarbe keinen Einfluß auf den Zuverlässigkeitswert nimmt. Z.B. impliziert Objekt 1 einen sehr hohen Wert ($c = 0.87$), obwohl es viel heller ist als das Objekt 2, mit dem das Netz trainiert wurde.

Es kann abschließend behauptet werden, daß sich die Verwendung der Hough-Transformation zum Lernen einer Objekterkennungsfunktion eignet, wenn davon ausgegangen werden kann, daß sich nur das Zielobjekt in den zu untersuchenden Bildern befindet. Sind in keinem anderen Bildbereich ähnliche Kantenstrukturen wie beim Objekt vorhanden, kann eine sichere Aussage getroffen werden, wo sich das Objekt im Bild befindet.
Es ist allerdings zu beachten, daß die Hough-Transformation sehr zeitaufwendig ist. Eine sinnvolle Anwendung wäre also eine Kombination mit den in Abschnitt 5.3 beschriebenen Operatoren, wobei lediglich die Bildbereiche einiger weniger Lokalisierungspositionen Hough-transformiert werden müssen.

5.6.2 Normierte HBF-Netze

In Kapitel 4, Abschnitt 4.1.3, wurde gezeigt, daß die Verwendung normierter RBF-Netze den Approximationsfehler bei der Manipulator-Einzelkamera-Kalibrierung reduzierten. Es soll nun entschieden werden, ob auch bei der Objekterkennung eine Normierung der HBF-Netze vorteilhaft ist.
Wird ein normiertes HBF-Netz wie in Abschnitt 5.2.1 trainiert, ergeben sich folgende Änderungen der Gleichung (5.8) zur Bestimmung des Zuverlässigkeitswertes c und der Interpolationsbedingung (5.9):

$$c = \tilde{f}(\vec{x}) = \frac{r_1 w_1 + \ldots + r_N w_N}{r_1 + \ldots + r_N} \approx f(\vec{x}) \tag{5.29}$$

$$\begin{pmatrix} 1.0 \\ \vdots \\ 1.0 \end{pmatrix} = \begin{pmatrix} \frac{r_{1_1} w_1 + \ldots + r_{1_N} w_N}{r_{1_1} + \ldots + r_{1_N}} \\ \vdots \\ \frac{r_{M_1} w_1 + \ldots + r_{M_N} w_N}{r_{M_1} + \ldots + r_{M_N}} \end{pmatrix} \tag{5.30}$$

Da die Ausgaben der Trainingsmenge $\mathcal{T}$ alle den Wert 1.0 erhalten, liefert eine Lineare Ausgleichsrechnung mit der Interpolationsbedingung (5.30) die optimalen Gewichte $w_1 = \ldots = w_N = 1.0$. Dadurch würde sich jedoch bei der Anwendung des Netzes für jedes Template aus (5.29) der Zuverlässigkeitswert $c = 1.0$ ergeben, also auch für Templates, die das Zielobjekt nicht enthalten. Dieses Problem könnte dadurch vermieden werden, daß zum Trainieren der Gewichte der 2. Schicht positive und negative Beispiele in der Trainingsmenge benutzt werden, wobei als Zuverlässigkeitswerte für positive Beispiele 1.0 und für negative 0.0 zu wählen ist. Die Menge der möglichen negativen Beispiele ist jedoch zu mannigfaltig, um sie mit einer endlichen Menge repräsentativ zu erfassen. Deshalb ist die Alternative der Verwendung normierter HBF-Netze zur Objekterkennung nicht geeignet.

Kapitel 6

Ausblick

In dieser Arbeit wurde beschrieben, wie sich mit Hilfe neuronaler Netze ein visuell basiertes System entwickeln läßt, das einige Fähigkeiten des Menschen simuliert. Ein Mensch lernt im Laufe seiner Entwicklung die Erscheinungen unzählig vieler Objekte und ist in der Lage, sie zu jedem beliebigen Zeitpunkt zu greifen. Er besitzt diese Fähigkeit dadurch, daß er zum einen ein ihm bekanntes Objekt wiedererkennt und zum anderen nach der Erkennung seinen Arm so steuern kann, daß das Greifen des Objektes ermöglicht wird.
Das in dieser Arbeit vorgestellte System simuliert die natürlichen Prozesse Wahrnehmen, Erkennen und Greifen. Es ermöglicht nämlich das Lernen einer Erkennungsfunktion für ein beliebiges Objekt und die Steuerung eines Roboterarmes zu diesem Objekt. Zur Realisierung dieses Systems wurden die neuronalen HBF- und RBF-Netze verwendet. Es wurde in mehreren Versuchsreihen analysiert, wie diese Netze zu konfigurieren sind, damit sie einerseits effizient sind, also einen geringen Zeitaufwand benötigen, und andererseits zuverlässige Ausgaben liefern. Es ist zu beachten, daß die Bewegung des Manipulators zu dem lokalisierten und erkannten Objekt nur eine grobe Annäherung darstellt. Das eigentliche Greifen kann erst ermöglicht werden, wenn die Greiffinger des Manipulators in eine optimale Greifstellung gebracht wurden. Es wurde zwar in Kapitel 5, Abschnitt 5.5.2, ein Verfahren zum Lernen einer optimalen Greifstellung beschrieben, jedoch wurden keine Experimente durchgeführt, die eine Kombination von Annäherung und Erreichen der Greifstellung verwendeten. Nachfolgende Arbeiten könnten sich mit dieser wichtigen Problemstellung auseinandersetzen.
Die in Kapitel 4, Abschnitt 4.1, beschriebene Manipulator-Einzelkamera-Kalibrierung ermöglicht eine sogenannte Servosteuerung des Manipulators, die nach der groben Annäherung des Objektes sinnvoll erscheint. Dadurch könnten geringe Bewegungsschritte ausgeführt werden und somit nach jedem Schritt eine Überprüfung der optimalen Greifstellung stattfinden. Eine nachfolgende Arbeit kann möglicherweise die Verfahren dieser Arbeit verwenden und eine Servosteuerung des Manipulators realisieren.

Ein weiteres Problem, das in dieser Arbeit nicht berücksichtigt wurde, ist die Hindernisvermeidung. Wird das lokalisierte und erkannte Objekt vom Manipulator grob angenähert, bleiben Hindernisse, die auf dem Bewegungspfad des Roboters liegen, unbeachtet. Der Roboterarm wird also evtl. mit anderen Objekten, die sich in der Szene befinden, kollidieren. Um dieses Problem zu vermeiden, ist die schrittweise Bewegung des Manipulators erforderlich, wobei nach jedem Schritt überprüft werden muß, ob sich der Roboterarm auf ein Hindernis zubewegt hat. Hierfür kann evtl. die in dieser Arbeit vorgestellte Manipulator-Einzelkamera-Kalibrierung verwendet werden, um nach einem Bewegungsschritt Beziehungen zwischen Manipulator und Hindernis im Computerbild herzustellen. Weiterhin könnten mit einer Objekterkennungsfunktion durch Anwenden der LmU-Lernstrategie *Gefahrensituationen* gelernt werden, in denen sich Manipulator und Hindernis kurz vor einer Kollision befinden.

Durch die Realisierung und Kombination dieser beschriebenen Systeme können ansatzweise einige wenige der komplexen natürlichen Prozesse des Menschen simuliert werden. Jedoch wird es immer Verbesserungsmöglichkeiten geben, solange nicht alle Vorgänge im menschlichen Gehirn verstanden wurden.

Gott schuf uns Menschen,
und doch werden wir niemals wie Gott sein.
Die Menschen schufen Computer,
diese aber werden niemals wie die Menschen sein.

Anhang A

KHOROS-Workspaces

Die folgenden Abbildungen zeigen einige Workspaces der unter KHOROS implementierten Routinen für die neuronale Manipulator-Einzelkamera-Kalibrierung aus Kapitel 4, Abschnitt 4.1, und für die Objektlokalisierung und -erkennung aus Kapitel 5.

Manipulator-Einzelkamera-Kalibrierung

Für die Bestimmung der Gewichte eines RBF-Netzes zur Manipulator-Einzelkamera-Kalibrierung werden die Routinen in Abb. A.1 verwendet.

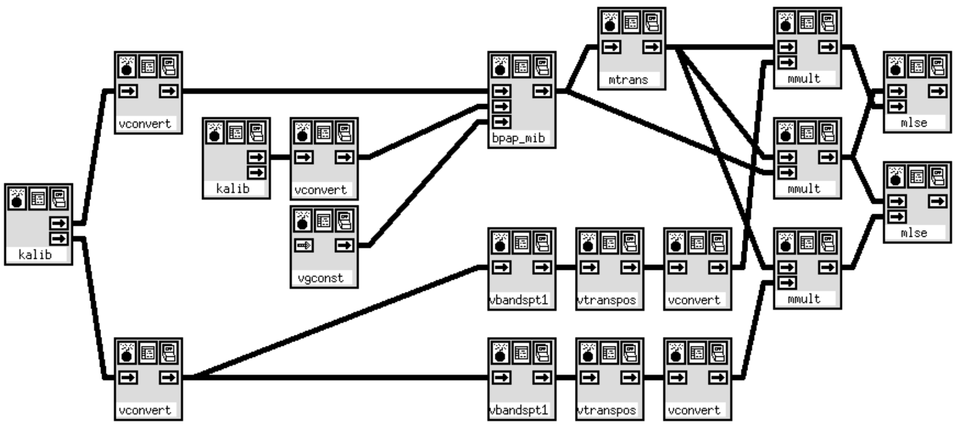


Abbildung A.1: 2. Phase des Trainierens zur Manipulator-Einzelkamera-Kalibrierung.

Die Routine **kalib** (links) steuert den Manipulator in einem vorgegebenen Arbeitsbereich schrittweise und ermittelt die Weltkoordinaten einer am Manipulator befestigten Lichtquelle sowie die zugehörigen Computerbildkoordinaten beider Kameras. Diese Koordinaten (obere Ausgabe = Weltkoordinaten, untere Ausgabe = Computerbildkoordinaten) bilden die Trainingsmenge des RBF-Netzes. Die Routine **kalib** (rechts) ermittelt auf gleiche Weise die Koordinaten der Zentren der Basisfunktionen. Für die Optimierung der Interpolationsbedingungen der Gleichungssysteme (4.4) und (4.5) mit Linearer Ausgleichsrechnung wird folgende Aktivierungsmatrix A benötigt:

$$A = \begin{pmatrix} r_{1_1} & \cdots & r_{1_N} \\ \vdots & & \vdots \\ r_{M_1} & \cdots & r_{M_N} \end{pmatrix}$$

Diese wird durch die Routine **bpap_mib** erzeugt. Dabei wird mit **vgconst** ein Normierungsparameter für die Basisfunktionen vorgegeben. Mit den Routinen **vbandspt1** werden die x- bzw. die y-Koordinaten (oben bzw. unten) des Computerbildes einer Kamera extrahiert. Die Routine **mtrans** transponiert die Aktivierungsmatrix, so daß die drei Routinen **mmult** jeweils $A^T\vec{t}_1$ (oben) für die Ausgabe der x-Koordinaten, A^TA (Mitte) und $A^T\vec{t}_2$ (unten) für die Ausgabe der y-Koordianten berechnen (vgl. mit Abschnitt 3.2.2 in Kapitel 3). Eine Optimierung der Gleichungssysteme $A^TA\vec{w}_1 = A^T\vec{t}_1$ und $A^TA\vec{w}_2 = A^T\vec{t}_2$ erfolgt durch die Routinen **mlse**. Dadurch ergeben sich schließlich die Gewichtsvektoren $\vec{w}_1$ und $\vec{w}_2$.

Die Anwendung des trainierten RBF-Netzes erfolgt mit den Routinen in Abb. A.2. Hier werden mit **kalib** wie beim Training die Koordinaten für eine Testdatenmenge erzeugt. Das linke **input** liefert die beim Training ermittelten Weltkoordinaten der Zentren der Basisfunktionen. **bpap_mib** berechnet mit diesen Koordinaten und den Weltkoordinaten aus der Testdatenmenge die Aktivierungsmatrix A. Durch Multiplikation von A mit den berechneten Gewichten (diese werden durch das rechte **input** geliefert) erfolgt die Ausgabe des Netzes.

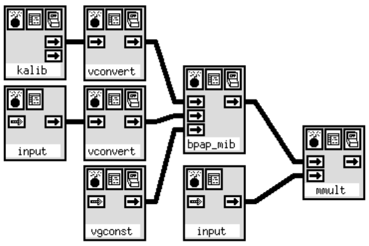


Abbildung A.2: Anwendung des RBF-Netzes zur Manipulator-Einzelkamera-Kalibrierung.

Objektlokalisierung und -erkennung

Bei der LoU-Lernstrategie erfordert das Trainieren eines HBF-Netzes zur Objektlokalisierung und -erkennung Trainingsmengen mit *Gabor-* und *Gradienten-Templates* (s. Kapitel 5, Abschnitte 5.4.1 und 5.4.2).
Die Gabor-Templates werden mit den Routinen der Abb. A.3 erzeugt.

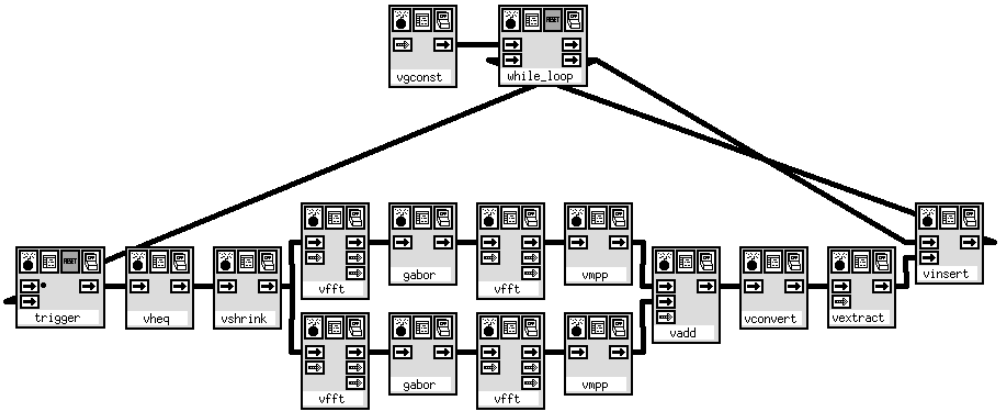


Abbildung A.3: Erzeugung von Gabor-Templates für die Trainingsmenge.

Hier werden in einer while-Schleife Trainingsbilder, die das zu lernende Objekt alle an der gleichen Position enthalten, nacheinander wie folgt verarbeitet:
Zunächst wird der Kontrast in dem Bild mit **vheq** erhöht. Dann wird das Bild mit **vshrink** verkleinert, um den Aufwand der Bildverarbeitung zu verringern. Die Gaborfilterung findet im Frequenzraum statt, d.h. das Bild wird Fourier-transformiert, Gabor-gefiltert und zurück transformiert. Diese Verarbeitungen erfolgen sequentiell durch die Routinen **vfft**, **gabor** und **vfft**. Dabei wird durch die oberen Routinen mit einem Gaboroperator gefiltert, der auf die vertikale Ausdehnung des Objektes reagiert, und durch die unteren mit einem Gaboroperator, der auf die horizontale Ausdehnung reagiert. Mit **vadd** werden die Antworten zu einer Gesamtantwort überlagert. Durch die Routine **vextract** wird der Bildbereich, der das Objekt enthält, extrahiert. Dieses Template wird anschließend mit **vinsert** in die Trainingsmenge aufgenommen.

Die Gradienten-Templates werden mit den Routinen der Abb. A.4 erzeugt.
Der Ablauf dieser while-Schleife erfolgt analog zu der vorherigen, jedoch wird hier statt der Gaborfilterung eine Gradientenfilterung mit der Routine **vgrad** durchgeführt.

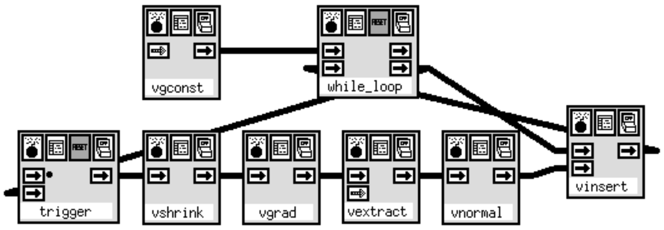


Abbildung A.4: Erzeugung von Gradienten-Templates für die Trainingsmenge.

Für die Bestimmung der Gewichte und für die Anwendung eines HBF-Netzes zur Objektlokalisierung und -erkennung werden die Routinen in Abb. A.5 verwendet.

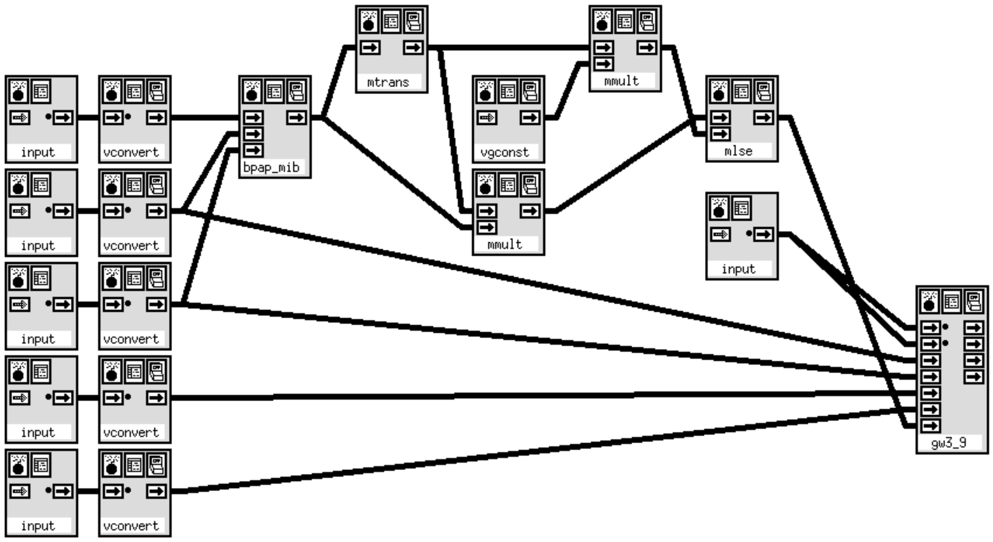


Abbildung A.5: 2. Phase des Trainierens und Anwendung des HBF-Netzes zur Objekterkennung.

Hier werden mit den drei oberen **input**s (links) die Gradientenfilterantworten in den Trainings-Templates, in den Clusterzentren-Templates (ermittelt durch den ISODATA-Algorithmus) und in den zugehörigen Clustervarianzen-Templates an die Routine **bpab_mib** übergeben. Diese berechnet die Aktivierungsmatrix. Wie bei der Manipulator-Einzelkamera-Kalibrierung werden nun mit **mtrans**, **mmult**, **vgconst** und **mlse** die Gewichte des HBF-Netzes berechnet, wobei **vgconst** den Ausgabevektor der Zuverlässigkeitswerte liefert. Diese werden hier alle

gleich 1 gesetzt, da die Templates der Trainingsmenge das zu lernende Objekt enthalten.
Die beiden unteren **input**s (links) liefern die Gaborfilterantworten in den Clusterzentren- und Clustervarianzen-Templates. Zu einem beliebigen Eingabebild, welches das rechte **input** liefert, kann nun mit der Routine **gw3_9** eine Position und ein Zuverlässigkeitswert ermittelt werden. Die Position gibt an, wo das Objekt lokalisiert wurde, und der Zuverlässigkeitswert bestimmt, mit welcher Wahrscheinlichkeit es sich dort um das gelernte Objekt handelt.

Anhang B

Die Hough-Transformation

Die Hough-Transformation [Kit88] hilft bei der Bestimmung der Orientierungen von Kanten. Hierfür wird jedes Pixel im Ortsraum zu einer Geraden (*Hough-Gerade*) im sogenannten *Parameterraum* transformiert. Treten nun mehrere Pixel im Ortsraum in einer Orientierung auf, drückt sich dies im Parameterraum durch ein Gebiet G aus, das mehrere Geraden durchlaufen. Im Spezialfall des Auftretens mehrerer Pixel auf genau einer Geraden im Ortsraum ergibt sich im Parameterraum als Gebiet G ein eindeutiger Schnittpunkt (x_s, y_s). Nach einer Suche der Schnittpunkte in dem resultierenden *Hough-Bild*, lassen sich Steigungen und Ordinatenabschnitte von Kanten im Originalbild ermitteln.
Die Hough-Transformation von einem Pixel an der Position (a, b) eines Eingabebildes erzeugt eine Gerade g im Parameterraum, die durch folgende Geradengleichung definiert wird:

$$g: \; y\left(x\right) = ax + b \tag{B.1}$$

Liegen im Spezialfall N Pixel (a_1, b_1),...,(a_N, b_N) auf einer Geraden h mit der Steigung m und einem Ordinatenabschnitt n, ergibt sich folglich für jedes Pixelpaar $\left[\left(a_i, b_i\right), \left(a_j, b_j\right)\right], \; i, j \in \{1, \ldots, N\}, \; i \neq j$, die Beziehung $m = \frac{b_i - b_j}{a_i - a_j}$.
Im Parameterraum ergeben sich für die transformierten Pixel folgende Geraden $g_1, \ldots, g_N$:

$$\begin{array}{lcccccc} g_1: & y_{g_1}\left(x\right) & = & a_1 x & + & b_1 \\ \vdots & & \vdots & & \vdots & \\ g_N: & y_{g_N}\left(x\right) & = & a_N x & + & b_N \end{array} \tag{B.2}$$

Aus der Annahme eines gemeinsamen Schnittpunktes (x_s, y_s) aller Geraden ergibt sich für diesen mit $y_s = y_{g_1}\left(x_s\right) =, \ldots, = y_{g_N}\left(x_s\right)$:

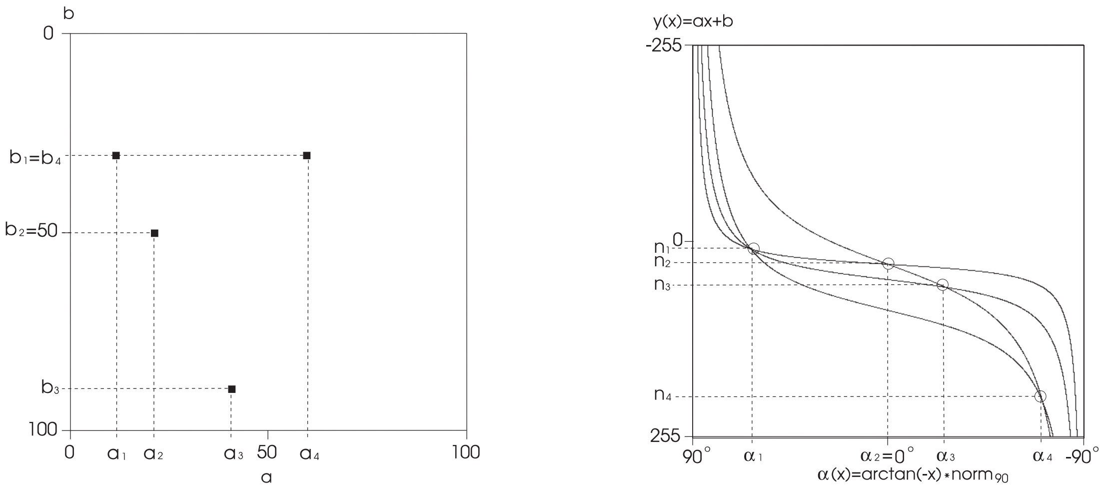
$$x_s = \frac{b_i - b_j}{a_j - a_i} = -m,\ i,j \in \{1,\ldots,N\}\,,\ i \neq j \qquad \text{(B.3)}$$

$$y_s = -ma_i + b_i,\ i \in \{1,\ldots,N\} \qquad \text{(B.4)}$$

Für den Ordinatenabschnitt $n = h(0)$ der Geraden h gilt mit (B.4):

$$\begin{aligned} n &= b_i - m\,(a_i - 0)\,,\ i \in \{1,\ldots,N\} \qquad \text{(B.5)} \\ &= y_s \end{aligned}$$

Somit können aus dem Schnittpunkt (x_s, y_s) der Hough-Geraden $g_1,\ldots,g_N$ mit $m = -x_s$ die Steigung und mit $n = y_s$ der Ordinatenabschnitt der Geraden h bestimmt werden. Auf der x-Achse des Hough-Bildes sollen jedoch nicht die Steigungen m der Geraden des Ortsraumes dargestellt werden, sondern die Steigungswinkel $\alpha = \arctan(m)$. Dafür ist bei der Darstellung eines Pixels (x, y) im Hough-Bild die Transformation $\alpha\,(x) = \arctan(-x) \cdot norm_{90}$ notwendig, wobei mit $norm_{90}$ als *Normierungsfaktor* der Bereich $[-90°, +90°]$ äquidistant auf der x-Achse dargestellt wird. Durch diese Darstellung der Steigungswinkel auf der x-Achse werden die Hough-Geraden durch Kurven dargestellt (sogenannte *Hough-Kurven*). Abb. B.1 zeigt ein Beispiel für vier Pixel $(a_1, b_1) = (10, 30)$, $(a_2, b_2) = (20, 50)$, $(a_3, b_3) = (40, 90)$ und $(a_4, b_4) = (60, 30)$ im Ortsraum.



(a) (b)

Abbildung B.1: Ein Eingabebild mit vier Pixeln (a) und das zugehörige Bild im Parameterraum (b).

Im Parameterraum ergeben sich vier Kurven, die sich in vier Punkten (α_1, n_1), (α_2, n_2), (α_3, n_3) und (α_4, n_4) schneiden. Dabei geben $\alpha_1,\ldots,\alpha_4$ die Steigungswinkel und $n_1,\ldots,n_4$ die Ordinatenabschnitte der Verbindungsgeraden der vier Punkte im Ortsraum an.
Für senkrechte Geraden im Ortsraum lassen sich im Parameterraum keine Schnittpunkte ermitteln, da kein Ordinatenabschnitt existiert. Um die Probleme bei der Schnittpunktsuche im Hough-Bild für Geraden mit einem Steigungswinkel bei $\pm 90°$ zu vermeiden, kann als Normierungsfaktor $norm_{45}$ so gewählt werden, daß nur der Bereich $[-45°, +45°]$ äquidistant auf der x-Achse dargestellt wird. Um auf diese Weise alle Schnittpunkte zu bestimmen, müssen zwei Hough-Bilder erzeugt werden, das Hough-Bild des Originalbildes (s. Abb. B.2 (a) und (a')) und das Hough-Bild des um 90° rotierten Originalbildes (s. Abb. B.2 (b) und (b')). Durch die Rotation werden die Pixel (a_i, b_i), $i = 1, \ldots, 4$, des Ortsraumes auf die Pixel $\left(\tilde{a}_i, \tilde{b}_i\right)$ übertragen. In jedem Hough-Bild werden nun die Schnittpunkte (α_j, n_j) bzw. $(\tilde{\alpha}_j, \tilde{n}_j)$, $j = 1, \ldots, 4$, ermittelt, wobei die $\tilde{\alpha}_j$ die Steigungswinkel und die $\tilde{n}_j$ die Ordinatenabschnitte der Geraden $\tilde{h}_j$ im rotierten Originalbild angeben. Damit lassen sich nun zwei Bilder mit den Geraden des Originalbildes und des rotierten Originalbildes erzeugen (Abb. B.2 (a") und (b")), die nach Rückrotation des zweiten Bildes um $-90°$ und anschließender Überlagerung das *Orientierungsbild* ergeben (Abb. B.3). Diejenigen α_j, $j \in \{1, \ldots, 4\}$, die im Bereich $[\pm 45°, \pm 90°]$ liegen, können direkt durch die entsprechenden $\tilde{\alpha}_j$ des rotierten Bildes mit $\alpha_j = \tilde{\alpha}_j - 90°$ berechnet werden.

Bei Bildern, in denen Pixel nicht auf eindeutig bestimmten Geraden liegen, entstehen im Parameterraum keine eindeutigen Schnittpunkte der Hough-Kurven, sondern Gebiete G_j, $j = 1, \ldots, M$, die von mehreren Hough-Kurven durchlaufen werden, wobei M die Anzahl der verschiedenen Orientierungen im Originalbild angibt. Um diejenigen Gebiete des Hough-Bildes, durch die am meisten Kurven laufen, zu finden, wird beim Eintragen einer Kurve in das Bild an jedem Schnittpunkt mit anderen Kurven der Pixelgrauwert erhöht. Anschließend können die lokalen Grauwertmaxima bestimmt werden, die jeweils für eine Orientierung den Winkel und den Ordinatenabschnitt angeben. Abb. B.4 zeigt die Bestimmung von Orientierungen in dem Kantenbild eines Objektes.

Probleme gibt es bei Hough-Transformation durch die Diskretisierung des Parameterraumes. Dadurch entstehen Ungenauigkeiten, die die Suche der lokalen Grauwertmaxima erschweren. Weiterhin ergibt sich bei der Maximasuche das Problem, ob zwei gefundene, nahe beieinander liegende Maxima dieselbe Orientierung unscharf repräsentieren oder ob sie zwei verschiedene Orientierungen darstellen.

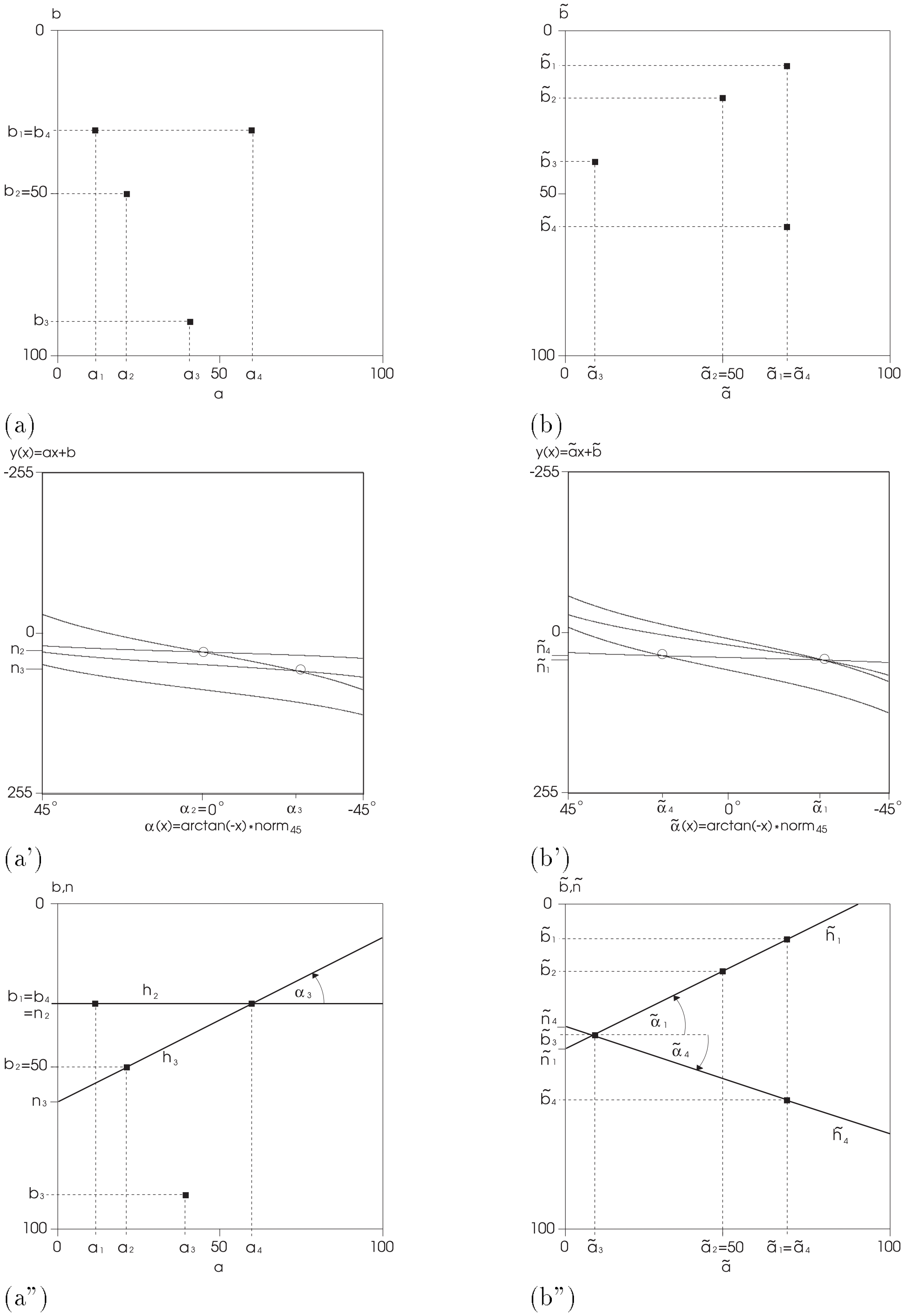


Abbildung B.2: Ein Eingabebild (a), das zugehörige Hough-Bild (a') und das resultierende Orientierungsbild (a") sowie das rotierte Eingabebild (b), dessen Hough-Bild (b') und Orientierungsbild (b").

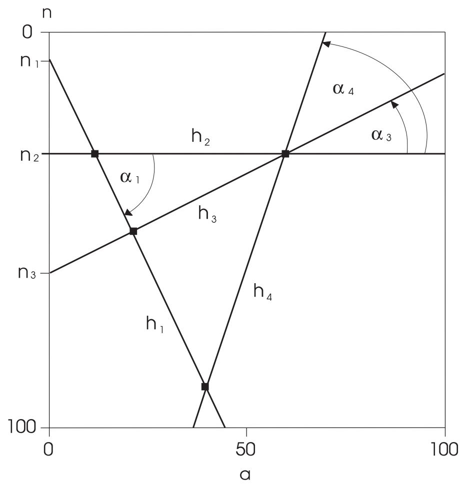
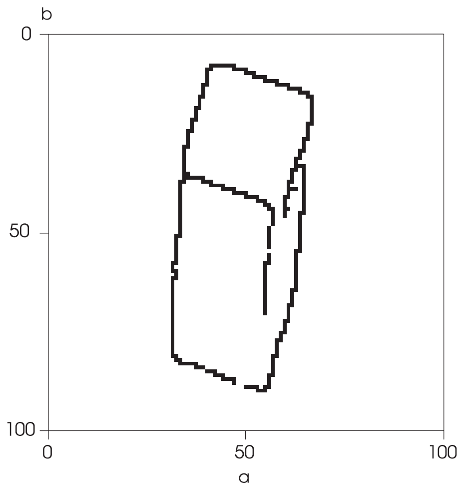


Abbildung B.3: Das Orientierungsbild zu dem Eingabebild mit vier Pixeln.

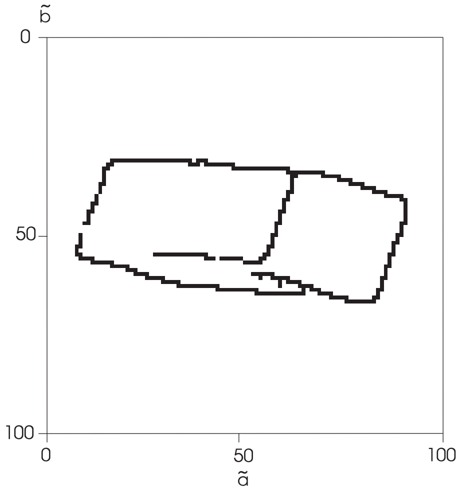
In Kapitel 5 wird die Hough-Transformation verwendet, um ein Objekt durch seine geometrischen Eigenschaften zu beschreiben. Wenn in einem Template nun nach den verschiedenen Orientierungen der Kanten gesucht werden soll, muß die Anzahl M dieser Orientierungen vorgegeben werden. Das stellt insofern ein Problem dar, als für unterschiedliche Erscheinungen des Objektes verschieden viele Kanten auftreten können. Außerdem würde die Diskretisierung des Parameterraumes zu einer ungenauen Lokalisierung des Objektes führen. Deshalb wird ein Ansatz verwendet, der das Objekt durch seine sogenannten Orientierungshistogramme beschreibt.

Werden für jeden diskreten Winkel[1] $\alpha \in \{0°, \ldots, 179°\}$ (sogenannter *Orientierungswinkel*) in den beiden Hough-Bildern eines Templates die zugehörigen Grauwerte betrachtet, so kann jeweils ein maximaler Grauwert $max(\alpha)$ und ein mittlerer Grauwert $mw(\alpha)$ berechnet werden. Durch die Spektren dieser Werte ergibt sich jeweils ein Orientierungshistogramm.
Das Ziel der Verwendung dieser Histogramme ist eine charakteristische Beschreibung der Kantenorientierungen und -längen in einem Template.
Der Wert max(α) gibt für den Winkel α die maximale Anzahl an Hough-Kurven an, die sich alle in einem Punkt schneiden. Durch diesen Wert wird somit die Länge der längsten Kante mit dem Orientierungswinkel α bestimmt. Allein durch dieses Histogramm wäre bereits eine Beschreibung der jeweils längsten Kanten möglich. Es ist aber zu bedenken, daß bei vielen Objekten parallele Kanten auftreten, von denen jeweils nur die längste dargestellt werden würde. Deshalb wird versucht, die mittlere Länge der Kanten eines Orientierungswinkels α durch den Wert $mw(\alpha)$ zu bestimmen.

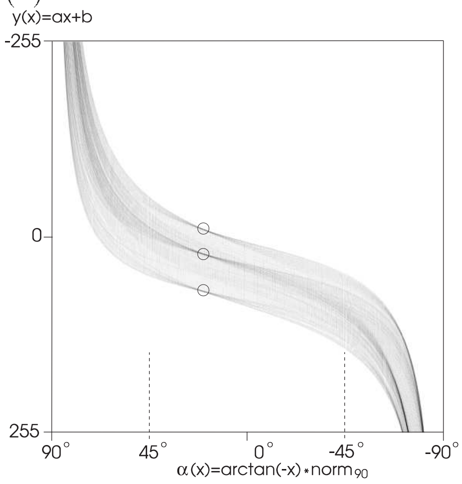
[1]Für die Winkel $\tilde{\alpha}$ in dem Hough-Bild des rotierten Bildes ist die Umrechnung $\alpha = \tilde{\alpha} - 90°$ erforderlich.



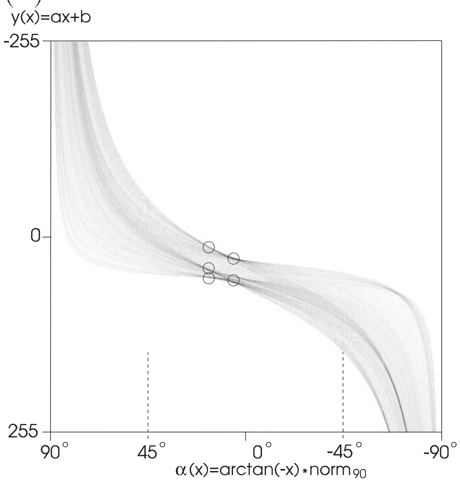
(a)



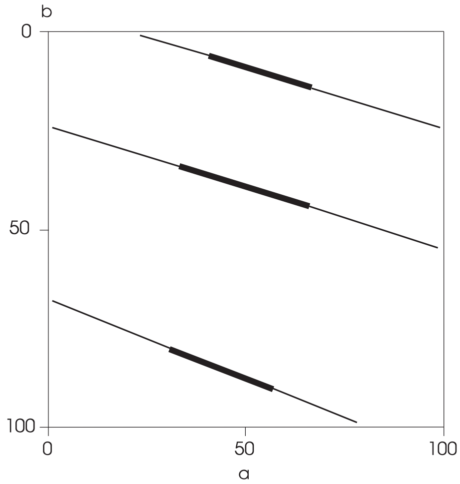
(b)



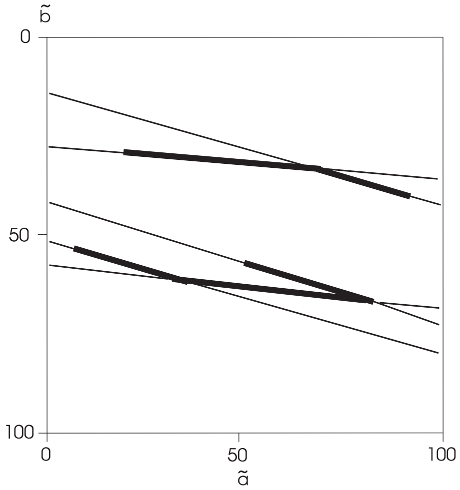
(a')



(b')



(a")



(b")

Abbildung B.4: Kantenbild eines Objektes bzw. das rotierte Kantenbild vor (a) bzw. (b) und nach (a') bzw. (b') der Hough-Transformation und das resultierende Orientierungsbild (a") bzw. (b").

Bei der Berechnung von $mw(\alpha)$ ist jedoch zu beachten, daß *jedes* Pixel des Originalbildes eine Hough-Kurve im Hough-Bild erzeugt. Da nur die lokalen Maxima des Hough-Bildes auf Kanten im Originalbild schließen lassen, muß eine geeignete *Segmentierung* erfolgen, so daß möglichst nur die Grauwerte im Bereich der Maxima übrig bleiben. Es ergibt sich also auch hier das Problem der Maximasuche.

Im folgenden wird der Einfachheit halber davon ausgegangen, daß vor der Bestimmung der Mittelwerte $mw(\alpha)$ das Hough-Bild mit Hilfe eines intuitiv ermittelten Schwellwertes THR segmentiert werden kann.
Seien die Grauwerte in den Hough-Bildern jeweils für einen Winkel α durch folgende Funktion $hough_\alpha$, $\alpha \in \{0°, \ldots, 179°\}$, beschrieben:

$$hough_\alpha : I \longrightarrow \mathbb{R} \tag{B.6}$$

Dabei gibt I die Höhe eines Hough-Bildes in Pixeln an. Sei $\mathcal{S}_\alpha \subseteq \{0, \ldots, I-1\}$ die Menge der segmentierten Positionen für eine Orientierung α. Dann lassen sich die Werte $max(\alpha)$ und $mw(\alpha)$ wie folgt berechnen:

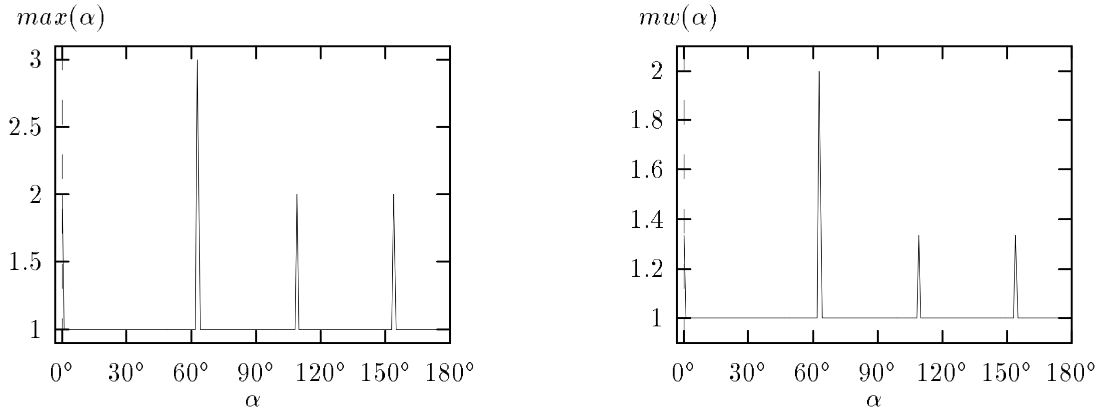
$$max(\alpha) = \max_{y \in \{0,\ldots,I-1\}} \{hough_\alpha(y)\} \tag{B.7}$$

$$\mathcal{S}_\alpha = \{y \in \{0, \ldots, I-1\} | \; hough_\alpha(y) > THR\} \tag{B.8}$$

$$mw(\alpha) = \frac{1}{|\mathcal{S}|} \sum_{y \in \mathcal{S}} hough_\alpha(y) \tag{B.9}$$

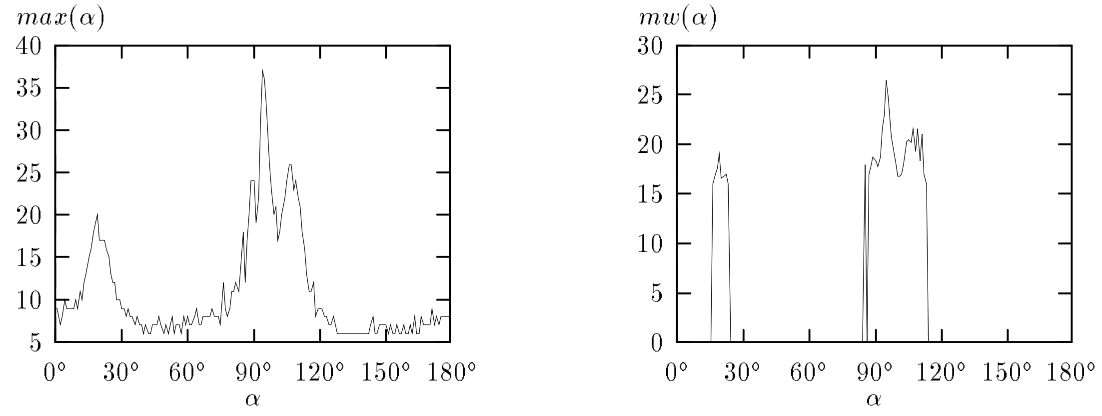
Die resultierenden Orientierungshistogramme zu dem Eingabebild mit vier Pixeln aus Abb. B.2 und zu dem Kantenbild eines Objektes aus Abb. B.4 sind in den Abbildungen B.5 und B.6 zu sehen.

Es ist zu beachten, daß die Histogramme in Abb. B.5 *keine* Kanten charakterisieren, da diese in dem Eingabebild nicht existieren. Es sollen lediglich für ein einfaches Beispiel die berechneten Werte $max(\alpha)$ und $mw(\alpha)$ veranschaulicht werden.
In Kapitel 5, Abschnitt 5.6.1, erfolgt die Darstellung jeder Erscheinung eines Objektes durch einen Vektor $\vec{x}$, der die maximalen und mittleren Grauwerte für alle Winkel enthält. Das Objekt wird damit durch seine geometrischen Eigenschaften beschrieben.



(a) (b)

Abbildung B.5: Die Orientierungshistogramme für das Maximum der Grauwerte (a) und für den mittleren Grauwert (b) der Hough-Bilder aus Abb. B.2.



(a) (b)

Abbildung B.6: Die Orientierungshistogramme für das Maximum der Grauwerte (a) und für den mittleren Grauwert (b) der Hough-Bilder aus Abb. B.4.

Ein Vektor $\vec{x}$ stellt also bei der LoU-Lernstrategie eine mögliche Eingabe für ein HBF-Netz dar und kann wie folgt beschrieben werden:

$$\vec{x} = \begin{pmatrix} max(0^\circ) \\ \vdots \\ max(179^\circ) \\ mw(0^\circ) \\ \vdots \\ mw(179^\circ) \end{pmatrix} \tag{B.10}$$

Abbildungsverzeichnis

Symbolverzeichnis

f	Brennweite der Kameras	7
$\hat{f}_{l/r}$	effektive Brennweite der linken bzw. rechten Kamera	7
$C_{l/r}$	Ursprünge im im linken bzw. rechten Kamerakoordinatensystem	7
$x_{c_{l/r}}$, $y_{c_{l/r}}$, $z_{c_{l/r}}$	Kamerakoordinaten links bzw. rechts	8
$x_{u_{a_{l/r}}}$, $y_{u_{a_{l/r}}}$	unverzerrte Bildkoordinaten	8
W	Ursprung im Weltkoordinatensystem	9
x_w, y_w, z_w	Weltkoordinaten	9
$A_{l/r}$	Ursprünge der linken bzw. rechten Abbildungsebene	9
$x_{d_{a_{l/r}}}$, $y_{d_{a_{l/r}}}$	verzerrte Bildkoordinaten	9
$D_{x_{l/r}}$	Verzerrung in x-Richtung der linken bzw. rechten Abbildungsebene	9
$D_{y_{l/r}}$	Verzerrung in x-Richtung der linken bzw. rechten Abbildungsebene	9
$\kappa_{1/2_{l/r}}$	Verzerrungsfaktoren der linken bzw. rechten Abbildungsebene	9
$N_{c_{x_{l/r}}}$	Anzahl der Sensor-Elemente in x-Richtung der linken bzw. rechten Kamerakoordinaten	9
$N_{c_{y_{l/r}}}$	Anzahl der Sensor-Elemente in y-Richtung der linken bzw. rechten Kamerakoordinaten	9
$N_{b_{x_{l/r}}}$	Breite des linken/rechten Computerbildes in Pixeln	9
$N_{b_{y_{l/r}}}$	Höhe des linken/rechten Computerbildes in Pixeln	9
$d_{x_{l/r}}$	Abstand zweier Sensor-Elemente in x-Richtung der linken bzw. rechten Kamerakoordinaten	9
$d_{y_{l/r}}$	Abstand zweier Sensor-Elemente in y-Richtung der linken bzw. rechten Kamerakoordinaten	9
$C_{x_{l/r}}$	x-Koordinate des Ursprunges A_l bzw. A_r im linken bzw. rechten Computerbild	9
$C_{y_{l/r}}$	y-Koordinate des Ursprunges A_l bzw. A_r im linken bzw. rechten Computerbild	9
$x_{b_{l/r}}$, $y_{b_{l/r}}$, x_b, y_b	Computerbildkoordinaten	9

$R_{l/r}$	Rotationsmatrizen	10
$\vec{t}_{l/r}$	Translationsvektoren	10
$j_1, \ldots, j_6$	Winkelstellungen des Manipulators	11
yaw, $pitch$, $roll$	Winkel für die Orientierung des Werkzeugkoordinatensystems	11
f	Funktion, die von RBF-/HBF-Netz approximiert wird	14
m	Dimension der Eingabevektoren von f	14
n	Dimension der Ausgabevektoren von f	14
$\tilde{f}$	Funktion, die f approximieren soll	14
$\vec{x}_i$	Eingabevektoren	14
Ω	Glattheitsfunktional bei Regularisierung	14
λ	Regularisierungsparameter	14
N	Anzahl der Basisfunktionen	15
$j = 1, \ldots, N$	Laufvariable für Basisfunktionen	15
$i = 1, \ldots, M$	Laufvariable für Eingabedaten	16
M	Anzahl Eingabevektoren/eindimensionaler Eingaben	16
r_j	Basisfunktionen	16
x	allgemeine eindimensionale Eingabe	16
y	allgemeine eindimensionale Ausgabe	16
x_i	eindimensionale Eingaben	16
y_i	eindimensionale Ausgaben	16
$\mathcal{T}$	Trainingsmenge	16
c_j	Zentren der Basisfunktionen (Clusterzentren)	16
σ^2, σ_j^2	Normierungsparameter	17
w_j	Gewichte der 2. Schicht (eindimensionale Funktion)	17
r_{i_j}	Antwort der j-ten Basisfunktion auf i-te Eingabe	17
$\vec{w}$	Gewichtsvektor	17
A	Aktivierungsmatrix	17
$\vec{t}$	Vektor der Ausgaben	17
$\vec{y}_i$	multidimensionale Ausgaben	18
Φ	allgemeine Basisfunktion	18
C_j	Indexmengen der Cluster	20
var_j	Varianzen in den Clustern	20
$\vec{x}$	allgemeine multidimensionale Eingabe	21
$\vec{y}$	allgemeine multidimensionale Ausgabe	21
k	Laufvariable für die Dimension der Ausgabe	22
w_{j_k}	Gewichte der 2. Schicht (multidimensionale Funktion)	22
$\vec{w}_k$	Gewichtsvektor für k-te Komponente	22
$\vec{t}_k$	Vektor der k-ten Komponenten der Ausgaben	22
$y_i^{(k)}$	k-te Komponente des Ausgabevektors $\vec{y}_i$	22
$\tilde{f}^{(k)}(\vec{x}_i) = \tilde{y}_i^{(k)}$	k-te Komponente der approximierten Ausgabe $\tilde{f}(\vec{x}_i)$	22
C_k	Summe quadratischer Fehler in k-ter Komponente	23

μ	Lernparameter	24
$\vec{\sigma}_j$	Normierungsvektor	25
$x^{(r)}$, $r = 1, \ldots, m$	r-te Komponente eines Eingabevektors $\vec{x}$	25
$c_j^{(r)}$, $r = 1, \ldots, m$	r-te Komponente des j-ten Zentrenvektors $\vec{c}_j$	25
$\left(\sigma_j^{(r)}\right)^2$, $r = 1, \ldots, m$	r-te Komponente des Normierungsvektors $\vec{\sigma}_j$	25
w_{j_x}, w_{j_y}, w_{j_z}	Gewichte der 2. Schicht	29
(x_{b_i}, y_{b_i})	Computerbildkoordinaten	29
X	Arbeitsbereich des Manipulators	32
$\mathcal{U}$	Testdatenmenge	33
L	Anzahl der Testdaten	33
l	Laufvariable für Testdaten	33
$\vec{x}_{u_l}$	Eingabevektoren der Testdatenmenge	33
$error_{mean}$	gemittelter Approximationsfehler	34
σ^2_{opt}	optimaler Normierungsparameter	35
σ^2_A, σ^2_B	Normierungsparameter für Versuchsreihen A und B	36
cl_{dist}	Abstand zweier benachbarter Clusterzentren	36
os	obere Schranke	37
$error_{min}$	minimaler Approximationsfehler	38
$\tilde{x}_w$, $\tilde{y}_w$, $\tilde{z}_w$	interpolierte Weltkoordinaten	44
$\tilde{x}_{w_i}$, $\tilde{y}_{w_i}$, $\tilde{z}_{w_i}$	interpolierte Weltkoordinaten	44
$error_{\tilde{x}_w}$	Approximationsfehler in x_w-Richtung	46
$error_{\tilde{y}_w}$	Approximationsfehler in y_w-Richtung	46
$min_{\tilde{x}_w}$, $max_{\tilde{x}_w}$, $avg_{\tilde{x}_w}$	minimaler, maximaler und gemittelter Fehler in x_w-Richtung	46
$min_{\tilde{y}_w}$, $max_{\tilde{y}_w}$, $avg_{\tilde{y}_w}$	minimaler, maximaler und gemittelter Fehler in y_w-Richtung	46
$\Delta_{x_{v/h}}$, $\Delta_{y_{l/r}}$	Ausdehnungen der Seiten der Arbeitsbereichebene (<u>v</u>orne, <u>h</u>inten, <u>l</u>inks, <u>r</u>echts) im Computerbild	50
g_{T_i}	Grauwertfunktion für Template im i-ten Kalibrierungsschritt	55
H, B	Höhe und Breite eines Templates in Pixeln	55
g_B	Grauwertfunktion für Computerbild	55
I	Breite und Höhe eines Computerbildes in Pixeln	55
SSD	Differenz eines Templates zu einem Bildbereich (<u>s</u>ummed <u>s</u>quared <u>d</u>ifferences)	56
$\vec{r}_x$, $\vec{r}_y$, $\vec{r}_z$	Richtungsvektoren	56
gw	Grauwertfunktion für ein Template	64
$D_x \times D_y$	Definitionsbereich für Funktionen im Template	64
g, g_i	Funktionen der Trainings-Templates	64
c, c_i	Zuverlässigkeitswerte	64
g_{Cl_j}	Funktionen der Clusterzentren-Templates	65
var_{Cl_j}	Funktionen der Clustervarianzen-Templates	65

σ_{Cl_j}	Funktionen der Streuung in den Clusterzentren-Templates	65
r	Laufvariable für die Dimension der Eingabe	65
$\mathcal{A}$, $\mathcal{A}_j$	Mengen von Positionen in Aufmerksamkeitsbereichen	70
S	Anzahl zu unterscheidender Objekte	71
s	Laufvariable für die Objekte	71
$\vec{c}$	Zuverlässigkeitsvektor	71
$G_{\sigma_1,\sigma_2,\lambda}$	Gaborfunktion	73
λ, $\lambda_{h/v}$	Wellenlänge der Cosinusfunktion	73
σ_1, σ_2, $\sigma_{1_{h/v}}$, $\sigma_{2_{h/v}}$	Wendepunkte der Gaußfunktion	73
ψ	Computerbild	73
$\mathcal{O}_{Gab}$	Gaboroperator	73
$\mathcal{O}_{Gab_{h/v}}$	horizontaler bzw. vertikaler Gaboroperator	74
H_S, B_S	Höhe und Breite einer Objektsilhouette in Pixeln	74
H_n, B_n	normierte Höhe und Breite der Silhouette eines Objektes in Pixeln	74
gab	Funktion der Gaborfilterantworten in einem Bild	76
$\mathcal{O}_{Grad}$	Gradientenoperator	78
D^x, D^y	horizontaler und vertikaler Ableitungsoperator	78
$grad$	Funktion der Gradientenfilterantworten in einem Bild	79
$\mathcal{O}_{GS}$	GS-Operator	79
thr	Schwellwert bei GS-Filterung	80
bin	Schwellwertoperator	80
g'_i	Funktionen der Templates in der 1. Stufe der Attention	84
g'_{Cl_j}, g''_{Cl_j}	Funktionen der Clusterzentren-Templates in der 1. und 2. Stufe der Attention	84
var'_{Cl_j}, var''_{Cl_j}	Funktionen der Clustervarianzen-Templates in der 1. und 2. Stufe der Attention	84
N'	Anzahl der Cluster bei der Attention	85
C'_j	Indexmengen der Cluster bei der Attention	85
$\left(x'_j, y'_j\right)$	Maxima/Minima in den Clusterzentren-Templates	85
$[min_{Tol_j}, max_{Tol_j}]$	Toleranzbereiche für die Cluster	85
σ'_{Cl_j}, σ''_{Cl_j}	Funktionen der Streuung in den Clusterzentren-Templates in der 1. und 2. Stufe der Attention	85
F	Streuungsfaktor	85
h	Funktion der Gaborfilterantworten in einem Eingabebild	86
(x_{q_p}, y_{q_p})	Positionen der Aufmerksamkeitsbereiche	88
$\mathcal{L}$	Menge der Lokalisierungspositionen	88
h_{q_p}	Funktion der Templates an Lokalisierungspositionen	89
sw	Schwellwert zur Erkennung des Objektes	89

$\mathcal{M}(x,y)$	erweiterter Matching-Bereich zu einer Position (x,y)	90
$\mathcal{M}'$	Vereinigung der erweiterten Matching-Bereiche	90
$h_{q'}$, $q' = 1,\ldots,\lvert\mathcal{M}'\rvert$	Funktion der Templates an den Positionen von $\mathcal{M}'$	90
$c_{q'}$, $q' = 1,\ldots,\lvert\mathcal{M}'\rvert$	Zuverlässigkeitswerte für die Positionen von $\mathcal{M}'$	91
(x_{lok}, y_{lok})	Position des lokalisierten Objektes	91
c_{max}	Zuverlässigkeitswert für die Position des lokalisierten Objektes	91
f_{var}	Varianzfaktor	92

Literaturverzeichnis

[Bis95] C. M. Bishop. *Neural Networks for Pattern Recognition.* Clarendon Press, Oxford, pp. 164-191, 1995.

[Cas94] A. Castano, S. Hutchinson. *Visual Compliance: Task-Directed Visual Servo Control.* IEEE Transactions on Robotics and Automation, Vol. 10, No. 3, pp. 334-342, June 1994.

[Cra89] J. J. Craig. *Introduction to Robotics: mechanics and control.* Addison Wesley, 1989.

[CVI94] G. Sommer, K. Daniilidis, U. Mahlmeister, M. Hansen. *Skript zur Vorlesung Computer Vision I.* 1994.

[Dan96] K. Daniilidis, E. Bayro-Corrochano. *The Dual Quaternion Approach to Hand-Eye Calibration.* International Conference on Pattern Recognition, Wien, 1996.

[Faug93] O. Faugeras. *Three-Dimensional Computer Vision.* The MIT Press, Cambridge, pp. 33-68, 1993.

[Gir95] F. Girosi, M. Jones, T. Poggio. *Regularization Theory and Neural Networks Architectures.* Neural Computation, Vol. 7, pp. 219-269, 1995.

[Gol65] G. Golub, W. Kahan. *Calculating the Singular Values and Pseudoinverse of a Matrix.* SIAM Numerical Analysis, B2 (2), pp. 205-224, 1965.

[Gri90] W. E. L. Grimson. *Object Recognition by Computer: The Role of Geometric Constraints.* The MIT Press, Cambridge, MA, 1990.

[Hag95] G. D. Hager, W. Chang, A. S. Morse. *Robot Hand-Eye Coordination Based on Stereo Vision.* IEEE Control Systems, pp. 30-39, February 1995.

[Har93] R. M. Haralick, L. G. Shapiro. *Computer and Robot Vision.* Addison-Wesley, pp. 156-160, 1993.

[Her94] R. Herpers, H. Kattner, G. Sommer. *GAZE: Eine attentive Verarbeitungsstrategie zum Erkennen und Lernen der visuell auffälligen Gesichtsregionen.* Mustererkennung 1994, PRODUserve Berlin, Serie Informatik Xpress, Vol. 5, pp. 341-349, 1994.

[Hus93] D. Hush, B. Horne. *Progress in Supervised Neural Networks.* IEEE Signal Processing Magazine 10, pp. 8-39, January 1993.

[Iva92] F. A. Mussa-Ivaldi. *From basis functions to basis fields: vector field approximation from sparse data.* Biological Cybernetics, Vol. 67, pp. 479-489, 1992.

[Jäh93] B. Jähne. *Digitale Bildverarbeitung.* Springer Verlag, 1993.

[Kit88] J. Illingworth, J. Kittler. *A Survey of the Hough Transform.* Computer Vision, Graphics and Image Processing 44, Academic Press, pp. 87-116, 1988.

[Koch85] C. Koch, S. Ullman. *Shifts in Selective Visual Attention: Towards the Underlying Neural Circuitry.* Human Neurobiol (1985) 4, pp. 219-227, 1985.

[Kri96] G. Krieger, C. Zetzsche. *Nonlinear Image Operators for the Evaluation of Local Intrinsic Dimensionality.* IEEE Transactions on Image Processing, Vol. 5, No. 6, pp. 1026-1042, June 1996.

[Mar92] T. Martinetz. *Selbstorganisierende neuronale Netzwerke zur Bewegungssteuerung.* Dissertation, Infix-Verlag, 1992.

[Mur94] H. Murase, S. K. Nayar. *Visual Learning and Recognition of 3-D Objects from Appearance.* International Journal of Computer Vision 14, pp. 5-24 (1995).

[Nag95] K. Nagao. *Recognizing 3D Objects Using Photometric Invariant.* International Conference on Computer Vision, pp. 480-487, 1995.

[Orr96] M. J. L. Orr. *Introduction to Radial Basis Function Networks.* http://www.cns.ed.ac.uk/people/mark.html, 1996.

[Pog90] T. Poggio, F. Girosi. *Networks for Approximation and Learning.* Proceedings of the IEEE, VOL. 78, No. 9, pp. 1481-1497, 1990.

[Pog92] T. Poggio, M. Fahle, S. Edelman. *Fast Perceptual Learning in Visual Hyperacuity.* Science, Vol. 256, pp.1018-1021, May 1992.

[Pow87] M. J. D. Powell. *Radial Basis Functions for Multivariable Interpolation.* a rewiew. In J. C. Mason and M. G. Cox (Eds.), *Algorithms for Approximation*, Clarendon Press, Oxford, pp. 143-167, 1987.

[Pre92] W. Press, S. Teukolsky, W. Vetterling. *Numerical Recipes in C.* Cambridge University Press, Massachusetts, pp. 393, 683 ff., 825, 1992.

[Sto89] J. Stoer. *Numerische Mathematik 1.* Springer-Verlag, 1989.

[Tay94] M. Taylor, A.Blake, A. Cox. *Visually guided grasping in 3D.* IEEE International Conference on Robotics and Automation, pp. 761-766, 1994.

[Tsai86] R. Y. Tsai. *An Efficient and Accurate Camera Calibration Technique for 3D Machine Vision.* IEEE Conference on Computer Vision and Pattern Recognition, pp. 364-374, 1986.

[Tsai89] R. Y. Tsai, R. K. Lenz. *A New Technique for Fully Autonomous and Efficient 3D Robotics Hand/Eye Calibration.* IEEE Transactions on Robotics and Automation, Vol. 5, No. 3, pp. 345-358, June 1989.

[Woo96] J. Wood. *Invariant Pattern Recognition: A REVIEW.* Pattern Recognition, Vol. 29, No. 1, pp. 1-17, 1996.

Index

Erklärung:

Hiermit versichere ich, daß ich die vorliegende Arbeit selbständig verfaßt und ausschließlich die angegebenen Hilfsmittel und Quellen verwendet habe.

Kiel, den