RBF - basierte Neuronale Netze zum automatischen Erwerb hindernisfreier Manipulator - Trajektorien, basierend auf monokularen Bildfolgen

vorgelegt von

Wolfram Blase

Kiel, im August 1998

Diplomarbeit

am Lehrstuhl für Kognitive Systeme
des Instituts für Informatik und Praktische Mathematik
der Christian–Albrechts–Universität zu Kiel
Prof. Dr. G. Sommer
Betreuer:
Dr. Josef Pauli

Zusammenfassung

In unserem verhaltensbasierten Robotersystem wurde ein 6-DOF Manipulator mit einer auf der Roboterhand befestigten Schwarzweißkamera ausgestattet. Durch die fortwährende Bildaufnahme vollzieht das System in einer mit Hindernissen besetzten Umwelt Wahrnehmungs–Aktions-Zyklen, um hindernisfreie Trajektorien des Roboters zu einem definierten Zielpunkt zu finden. Die Wahrnehmung des Systems basiert auf Netzwerksystemen auf der Grundlage radialer Basisfunktionen, die die Rekonstruktion der Umwelt aus dem optischen Fluß der durch die Kamera generierten Bildfolge erlernen. Parallele Algorithmen bzgl. der Bildverarbeitung und des Netztrainings sowie die Anwendung des divide-and-conquer Verfahrens auf die Netzwerke begegnen dem Problem der hohen Dimensionalität. Die auf die Wahrnehmungsphase folgenden Aktionen des Systems werden in einem Vektorfeld modelliert, das Informationen über das Ziel sowie die Hindernisse integriert. Das von einem Potentialfeld abgeleitete Vektorfeld wird dynamisch durch neue Wahrnehmungen aktualisiert. Eine Kontrollinstanz detektiert die bei Potentialfeldalgorithmen typischen lokalen Minima und modifiziert das Vektorfeld zur Umgehung dieser Minima.

Inhaltsverzeichnis

1 Einführung **1**

1.1 Gegenstand der Arbeit 1

1.2 Verwandte Arbeiten 2

1.3 Aufbau der Arbeit 4

2 Struktur des Manipulatornavigationssystems **5**

2.1 Die Hardware des Systems 5

2.2 Definition der Teilaufgaben 6

2.2.1 Rekonstruktion dreidimensionaler Positionen aus zweidimensionalen Strukturen 9

2.2.2 Überführung relativer zu absoluten Positionen 10

2.2.3 Konstruktion und Aktualisierung des Besetzungsraumes 10

2.2.4 Generierung eines Trajektorieteilstücks 11

3 Strukturdetektion und Strukturzuordnung **13**

3.1 Strukturdetektion 14

3.1.1 Der Interest Operator (IOp) 14

3.1.2 Der Eckendetektor SUSAN 17

3.1.3 Interest Operator versus SUSAN 23

3.1.4 Parallele Implementierung der Operatoren 28

3.2 Strukturzuordnung 29

3.2.1 Beschleunigung der Strukturzuordnung 31

3.2.2 Parallele Ausführung der Strukturzuordnung 35

4 Neuronale Rekonstruktion von Besetzungspositionen 37

4.1 RBF-Netze 38

4.1.1 Paralleles Training eines RBF-Netzes 40

4.2 Neuronale Rekonstruktion 41

4.2.1 Die Rekonstruktion mit einem RBF-Netz 41

4.2.2 Splitting des RBF-Netzes 44

4.2.3 Training der einzelnen Netze 48

4.2.4 Vergleich des Gesamtnetzes mit dem gesplitteten Netz 61

4.2.5 Anwendung des geplitteten Netzes 61

4.2.6 Genauigkeit des gesplitteten Netzes 63

5 Begriffsdefinition Hindernis, Trajektorie und Ziel 67

5.1 Erweiterung der Hindernisdefinition 68

5.1.1 Berechnung der Gelenkpositionen 70

5.1.2 Berechnung der Gelenkwinkel 71

5.2 Ziel und Zugriffsposition 75

5.3 Definition der Trajektorie 77

6 Vektorfeldgeneration durch Potentialfelder 79

6.1 Die Potentialfunktion und ihr Gradient 79

6.2 Hindernisvermeidung und Zielansteuerung 80

6.3 Skalierung der Repelloren 82

6.4 Statische und dynamische Szenen 83

6.5 Probleme der Potentialfeldalgorithmen 83

6.6 Monitoring 86

6.7 Monitorgesteuerte Aktionen 87

7 Erprobung des Systems **89**

7.1 Testszene und Aufgabe 89

7.2 Erlernen der Rekonstruktion der Szene 90

7.3 Rekonstruktion der Szene 91

7.4 Berechnung der Zugriffsposition 91

7.5 Berechnung der Trajektorie 91

7.6 Verwendete Parameter 92

8 Diskussion und Ausblick **97**

8.1 Diskussion 97

8.2 Ausblick 99

Literaturverzeichnis **103**

Kapitel 1

Einführung

Das Gros der heute eingesetzten Robotersysteme in der Industrie nimmt Aufgaben wahr, deren Abläufe fest vorgegeben sind. Die zur Lösung der Aufgabe benötigten Handlungen werden nicht vom Robotersystem erstellt, sondern von externer Seite in das System eingegeben. Liegt jedoch eine Situation vor, die variable Parameter aufweist, ist ein System gefordert, das innerhalb des vorgegebenen Parameterraumes eine Lösungsstrategie erarbeitet und ausführt. Dabei kann sich die Variabilität der Situation auf die Eingangsparameter beschränken. In diesem Fall hat das Robotersystem die Aufgabe, entweder in einer Wahrnehmungs-Aktionsfolge eine Lösung zu finden oder, falls nicht alle benötigten Situationsparameter zu Anfang bekannt sind, in einem Wahrnehmungs-Aktions-Zyklus das Wissen um die Parameterwerte zu vervollständigen und gleichzeitig eine Lösungsstrategie zu erarbeiten. Bei dynamischen Situationen ist der fortwährende Wahrnehmungs-Aktions-Zyklus unverzichtbar. In dieser Arbeit liegt eine Situation mit variablen Eingangsparametern vor. Die vorgestellten Verfahren ermöglichen eine Folge als auch einen Zyklus von Wahrnehmung oder Aktion.

1.1 Gegenstand der Arbeit

Das von uns verwendete Robotersystem besteht aus einem 6-DOF-Roboter, der mit einer an der Roboterhand befestigten Schwarzweißkamera ausgestattet ist. Da das System mit im Nahbereich des Roboters plazierten Hindernissen konfrontiert ist, ist die Kamera mit einem 85°-Weitwinkelobjektiv ausgerüstet. Die Systemumgebung ist durch einen Tisch oder einen (offenen) Schrank gegeben, der mit Objekten bestückt ist, die normalerweise Gegenstand menschlicher Interaktion sind. Beispiele für solche Objekte sind Flaschen oder Dosen, die in der Regel eine mit einer Textur versehene Oberfläche aufweisen. Das aus Roboter, Kamera und Computer bestehende *Manipulatornavigationssystem* hat nun die Aufgabe, bezüglich einer gegebenen Zielposition (z.B. der Mittelpunkt eines Objektes) eine Trajektorie zu generieren, die in jedem dem Zielpunkt näherkommenden Schritt den Roboter kollisionsfrei hinsichtlich der Hindernisse positioniert.

Neben der zu lösenden Aufgabe der Generation einer Trajektorie, die die hindernisfreie Bewegung zum Zielpunkt beschreibt, beeinflussen einige Randbedingungen den Entwurf des Manipulatornavigationssystems. Zum einen wird das System unter dem Gesichtspunkt der Effizienz entworfen. Die im Manipulatornavigationssystem vorkommenden Zeitspannen der Wahrnehmungs-Aktions-Zyklen werden minimiert. In diesem Zusammenhang wird der Teil der laufzeitintensiven Algorithmen parallelisiert, und die Implementation des Manipulatornavigationssystems wird ein Mehrprozessorsystem unterstützen. Ein weiterer Gesichtspunkt betrifft die Integration veränderbarer Systemparameter: Bei Neuinstallation des Systems sollen veränderte Parameter wie z.B. die Orientierung der Kamera zur Roboterhand automatisch und schnell in das System integriert werden. Der Einsatz des Weitwinkelobjektivs ist durch die im Entfernungsbereich von Dezimetern plazierten Hindernisobjekte notwendig, um ein ausreichendes Blickfeld abdecken zu können. Die resultierenden Bildverzerrungen (Fischaugen-Effekt) muß das Manipulatornavigationssystem automatisch kompensieren.

1.2 Verwandte Arbeiten

Die erste Stufe der Wahrnehmung des Manipulatornavigationssystems ist die Detektion von Grauwertstrukturen, die die Grundlage zur Korrespondenzbestimmung von Strukturen in Bildfolgen liefert. Einer der uns interessierenden Operatoren, der *Interest Operator*, findet auch bei Moravec Verwendung, der Stereobilder auf Grauwertstrukturen zur Besetzung eines *Evidence Grids* abtastet [1]. Die parallele Detektion von Ecken auf verschiedenen Rechnerarchitekturen mit einem *coarse to fine level of resolution* Algorithmus beschreiben Zhang und Deng in [2]. Der von Szeliski und Kang benutzte *Monotonicity Operator* [3] steht in enger Verbindung zum SUSAN Operator. In diesem Report wird das *structure from motion*-Problem addressiert, wobei neben Punkten auch Linien gematcht werden. In unserem Fall können wir jedoch die Bewegung als bekannt voraussetzen. Eine monokulare, rekursive Bewegungs- und Strukturschätzung erläutern Bouguet und Perona in [4]. Cui, Weng und Cohen geben zum selben Thema ein Verfahren an, das Strukturinformationen der Szene, die aus vorangegangenen Bildern einer monokularen Bildfolge gewonnen wurden, in die weitere Berechnung einbezieht [5]. Auch Mendelsohn, et al. beschäftigen sich mit der Bestimmung des optischen Flußes, allerdings bei großen Kamerabewegungen und entsprechenden Strukturversätzen unter der Voraussetzung einer statischen Szene [6]. Ein Maß für die Eignung eines Bildpaares zur Bestimmung des optischen Flusses durch das Matching von Strukturen gibt Kanatani [7]. Dabei ist ein Mittel für die notwendige Kamerabewegung zu finden, die einen Bildversatz zur Folge hat, der einerseits groß genug ist, eine robuste 3D-Rekonstruktion zu ermöglichen, andererseits nicht zu groß ist, da ein zu großer Bildversatz das Matching von Strukturpaaren verhindert. Dem Spezialfall der translatorischen Bewegung widmet sich Naeve mit der Ausnutzung von hinsichtlich der Projektion invarianter Tangenteninformationen. Der optische Fluß in einer monokularen Bildfolge ist auch Basis für die Bestimmung von Kollisionspunkten innerhalb eines geplanten Pfades in dem Artikel von Dev et al. [8]. Die *Time to Collision* berechnet Meyer in [9], ohne die Position des Hindernisses in der Szene zu rekonstruieren. Verschiedene Verfahren, die Hindernisse relativ zur Bodenebene charakterisieren, sind nicht nur für fahrbare Roboter, sondern bei entsprechender Kamerabewegung auch für feststehene Roboter geeignet. Ein solcher Algorithmus ist z.B. bei Zhang zu finden [10]. Unter der Restriktion der Kamerabewegung als reine Translation ist das Verfahren der *inversen, perspektivischen Projektion* geeignet, Hindernisse zu detektieren, die einen anderen optischen Fluß aufweisen als ein Punkt des Bodens [11]. Die Szenenrekonstruktion aus Punkten und Geraden auf der Basis von vier Referenzpunkten, die in einer monokularen Bildfolge in jedem Bild bekannt sind, ist Thema von Seitz und Dyer in [12]. Sandini et al. geben einen allgemeinen Überblick der Bildverarbeitung bei gleichzeitigen Aktionen des Systems mit Themen wie z.B. Navigation, Bewegung und optischer Fluß, reflexives Verhalten unter Verwendung des optischen Flusses, kontinuierliche Bildverarbeitung und Manipulation [13].

Die Rekonstruktion von Punkten der Szene aus korrespondierenden Grauwertstrukturen wird durch *RBF-Netze* vorgenommen. Diese werden zur Steigerung der Effizienz beim Training in ein System von Netzen überführt. Dieses Prinzip des *divide-and-conquer*-Verfahrens fand auch in den von Jordan und Jacobs beschriebenen *Hierarchical Mixtures of Expters* Eingang [14]. Eine weitere Art der Rekonstruktion auf der Basis monokularer Bilder ist mit dem *axialem Motion Stereo* gegeben. Cord und Pallmer beschreiben die Abstandsmessung auf der Basis von mit dem *Plessey-Eckendetektor* detektierten Ecken, die Lösung des Korrespondenzproblems und die optische Triangulation [15].

Nachdem in der Vergangenheit das sequentielle Modell der Informationsverarbeitung dominierte, brach diese starre Struktur in den letzten Jahren u.a. mit dem Beitrag von Brooks auf, der mit der *Subsumptionsarchitektur* ein hierarchisches und paralleles System von Verhaltensweisen schuf [16]. Dieser Ansatz findet einen Bezug in der durch Vektorfelder gegebenen Navigation. Das Vektorfeld, das die Zielansteuerung durch einen *Attraktor* implementiert, wird nach diesem Prinzip an Orten von Hindernissen durch die übergeordnete Instanz der die Hindernisvermeidung repräsentierenden *Repelloren* bestimmt. Die Subsumptionsarchitektur findet ebenfalls Verwendung bei Connell, der die Steuerung eines Roboterarmes ohne Repräsentation der Umwelt vornimmt [17], d.h. es findet keine Speicherung eines Umweltmodells statt. Dies setzt jedoch Sensoren an allen Stellen des Systems voraus, die mit der Umwelt interagieren. Im Falle unseres Roboterarms, der auch an Stellen mit Hindernissen kollidieren kann, die nicht über Sensoren verfügen (der einzige Sensor ist die Kamera an der Roboterhand), ist dieser Ansatz jedoch unzureichend. Einen kritischen Überblick über die Subsumptionsarchitektur geben Hartley and Pipitone in [18]. Eine ähnliche Integration von Verhaltensweisen durch Attraktoren und Repelloren in einem System dynamischer neuronaler Felder wie in den von uns benutzten Vektorfeldern geben Schöner et al. in [19] und [20] an. Das von uns benutzte Vektorfeld ist der Gradient eines Potentialfeldes. Auf Potentialfeldern basierende Algorithmen sind die Grundlage mehrerer Artikel wie z.B. der Beitrag von Borenstein, der das Thema der Hindernisvermeidung in Echtzeit hat [21]. Basis dieser Arbeit ist das *Virtual Force Field*, das als mehrdimensionales Histogramm eine Kombination aus *Certainty Grid* und Potentialfeld ist. Die Vektorfeldapproximation auf der Basis von Gradienten radialer Basisfunktionen auf der Grundlage weniger Beispielvektoren zur Pfadplanung beschreibt Mussa-Ivaldi in [22] und [23]. Eine kritische Betrachtung von Potentialfeldalgorithmen geben Koren und Borenstein in [24]. Ein weiterer Artikel zur Pfadplanung ist von Quinlan und Khatib, die sich mit sogenannten *elastic bands*, d.h. deformierbaren, kollisionsfreien Pfaden beschäftigen [25]. Die Deformation der Pfade wird dabei von den Orten der Hindernisse bestimmt. Die direkte neuronale Kontrolle eines Roboterarms bei Anwesenheit von Hindernissen beschreiben W.E. Kelly et al. [26]. Johnson et al. stellen die neuronale Pfadplanung in den Vordergrund [27]. Einen sehr breiten Überblick in die Pfadplanung und Kontrolle von Robotern gibt Torras in [28]. Es wird die Bewegungsplanung bezüglich algebraischer sowie heuristischer Ansätze sowie der Einsatz neuronaler Netze in der Robotik besprochen.

1.3 Aufbau der Arbeit

Der Aufbau der Arbeit orientiert sich an dem Top-Down-Verfahren. Die Darstellung des Aufbaus des Manipulatornavigationssystems findet in Kapitel 2 Platz. Die dort beschriebenen Komponenten implementieren teilweise die Wahrnehmung und teilweise die Aktionen des Systems. Zur ersten Gruppe gehören die Komponenten der Strukturdetektion, welche im Kapitel 3 beschrieben werden. Dazu werden der *Interest Operator* und der *SUSAN-Operator* verglichen. Auch die Zurordnung der detektierten Strukturen sowie Betrachtungen der parallelen Bildverarbeitung sind Teil dieses Kapitels. Das folgende Kapitel 4 behandelt mit der neuronalen Rekonstruktion von Punkten der Szene eine Komponente, die zwischen Wahrnehmung und Verarbeitung von Szeneninformationen anzusiedeln ist. Den Betrachtungen zur Effizienz des Trainings von RBF-Netzen folgt die Beschreibung des Netzsplittings, das durch Anwendung des *divide-and-conquer*-Verfahrens auf das RBF-Netz eine Verkürzung der Trainingszeiten auf einen Bruchteil der Trainingszeit des ungeteilten Netzes ermöglicht. Auch die Parallelisierung des Netztrainings ist eines der beschriebenen Mittel zur Steigerung der Effizienz. In Kapitel 5 werden Definitionen zu den Begriffen Hindernis, Trajektorie und Ziel gegeben. Diese sind Grundlage zur Navigation unter Verwendung von Vektorfeldern, die Thema des Kapitels 6 sind. Die in diesem und den folgenden Kapiteln beschriebenen Komponenten implementieren die Aktionen des Manipulatornavigationssystems. In Kapitel 6 wird außerdem die Herleitung des Vektorfeldes als Gradient eines Potentialfeldes dargestellt. Der Einführung eines universellen Funktionstypen zur Darstellung von *Repelloren*, die den Teil der Hindernisvermeidung im Vektorfeld repräsentieren, folgt eine Betrachtung der typischen Probleme von auf Potentialfeldern basierenden Verfahren. Eine Methode zur automatischen Detektion und Lösung dieser Probleme wird mit dem sogenannten *Monitor* gegeben, dessen Beschreibung das Kapitel 6 beendet. Die Erprobung des Manipulatornavigationssystems wird in Kapitel 7 protokolliert. Mit der Diskussion und einem Ausblick auf Erweiterungsmöglichkeiten des Systems schließt die Arbeit in Kapitel 8.

Kapitel 2

Struktur des Manipulatornavigationssystems

Nachdem in Kapitel 1.1 die Aufgabe des Manipulatornavigationssystems beschrieben wurde, wird in diesem Kapitel ein Überblick über die Strukturierung der Aufgabe in Teilprobleme gegeben, aus denen die zur Lösung der Aufgabe notwendigen Verfahren und korrespondierenden Komponenten resultieren. Zunächst wird jedoch die zur Verfügung stehende Hardware dargestellt, die neben der Aufgabenstellung ebenfalls Einfluß auf das zu erstellende System hat.

2.1 Die Hardware des Systems

Das Manipulatornavigationssystem basiert auf den drei Komponenten Roboter, Computer und Kamera. Im einzelnen sind dies

- ein Roboter der Firma Stäubli, bezeichnet mit RX90, sechs Freiheitsgrade und eine Wiederholgenauigkeit von $\pm 0.02mm$ aufweisend,
- eine Schwarzweißkamera, an der Roboterhand befestigt, ausgestattet mit einem Weitwinkelobjektiv von $85°$,
- ein Mehrprozessorrechner der Firma SUN, Enterprise 4000, ausgestattet mit vier ULTRASPARC 167 MHz Prozessoren (zum Vergleich von Laufzeiten wurden teilweise ein Rechner der Fa. SUN mit zwei ULTRASPARC Prozessoren mit 300 MHz sowie ein mit einem Prozessor AMD K5 100 MHz ausgestatteter Rechner herangezogen).

2.2 Definition der Teilaufgaben

Das Manipulatornavigationssystem bildet grundsätzlich visuelle Daten sowie Positionsdaten des Roboters auf eine Trajektorie des Roboterarms ab. Wir betrachten das Manipulatornavigationssystem zunächst als „Black Box“, das als Eingabe eine Bildfolge erhält sowie über die aktuelle Position vom Robotersystem informiert wird. Der anzusteuernde Zielpunkt liegt als Konstante des Verfahrens vor. Die Ausgabe des Systems bildet die durch Trajektorieteilstücke definierte Bewegung des Roboters (Abbildung 2.1).

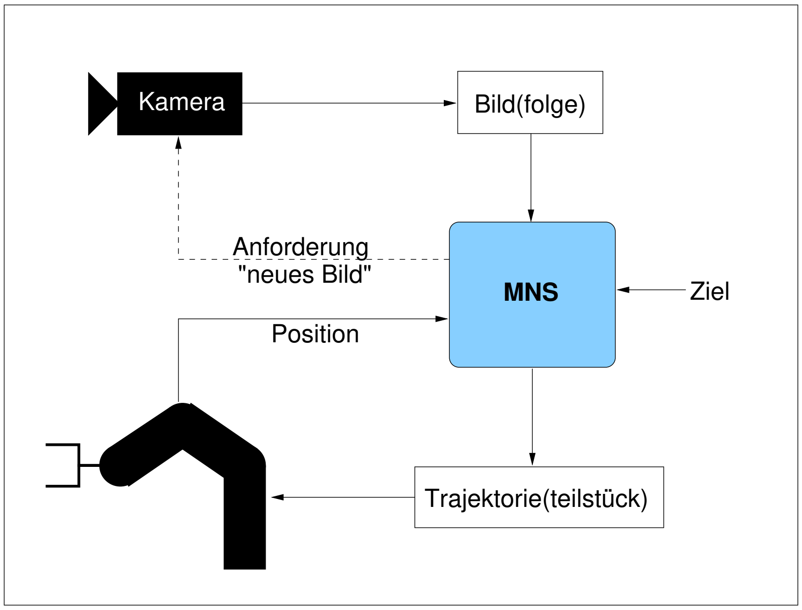

Abbildung 2.1: Ein- und Ausgaben der "Black Box" MNS, die die Modi Zyklus oder Folge von Wahrnehmung der Aktion unterstützt.

Um eine Trajektorie berechnen zu können, ist zu Beginn der Berechnung ein Anfangswissen über die den Roboter umgebende Szene notwendig. Ist diese Information bereitgestellt, hängt es von der Komplexität der Aufgabe als auch von der Szene ab, ob die Generation der Trajektorie eine weitere Wahrnehmung der Umwelt voraussetzt oder mit dem anfangs erworbenen Wissen ausgeführt werden kann. Im ersten Fall findet ein Zyklus von Bildaufnahme und Berechnung des nächsten Trajektorieteilstücks statt, während im letzten Fall der Aufnahme einer Bildfolge die Phase der Berechnung der (gesamten) Trajektorie folgt. Der Unterschied liegt also in der zeitlichen Abfolge der Komponenten Wahrnehmung und Aktion, ohne daß die Verfahren selber unterschiedlich beschaffen sein müssen. Dementsprechend werden die einzelnen Komponenten des Systems derart konstruiert, daß sowohl ein Zyklus als auch eine Folge von Wahrnehmung und Aktion unterstützt wird, obwohl sich für die von uns gestellte Aufgabe bezüglich einer statischen Szene herausstellen wird, daß mit dem Modell der Folge eine Lösung gegeben ist (Kapitel 7). Doch auch in diesem Fall ist die Generation der Trajektorie einem dynamischen Prozeß unterworfen, wenn z.B. die Feststellung, daß eine angefahrene Position nicht zum Ziel führt und daher zu vermeiden ist, zur Notwendigkeit der Aktualisierung der Trajektorie führt.

Nachdem die grundsätzlichen Anspüche an die Komponenten des Systems festgelegt wurden, werden diese nun definiert. Die Eingaben Bild und Position werden durch die Komponenten des Manipulatornavigationssystems (Abbildung 2.2) mehreren Transformationen unterworfen, wobei Abbildungen zwischen Weltkoordinatensystem, (Roboter-) Handkoordinatensystem und Kamerakoordinatensystem (Abbildung 2.3) auftreten. Die Gliederung der Verarbeitungsstufen gestaltet sich wie folgt:

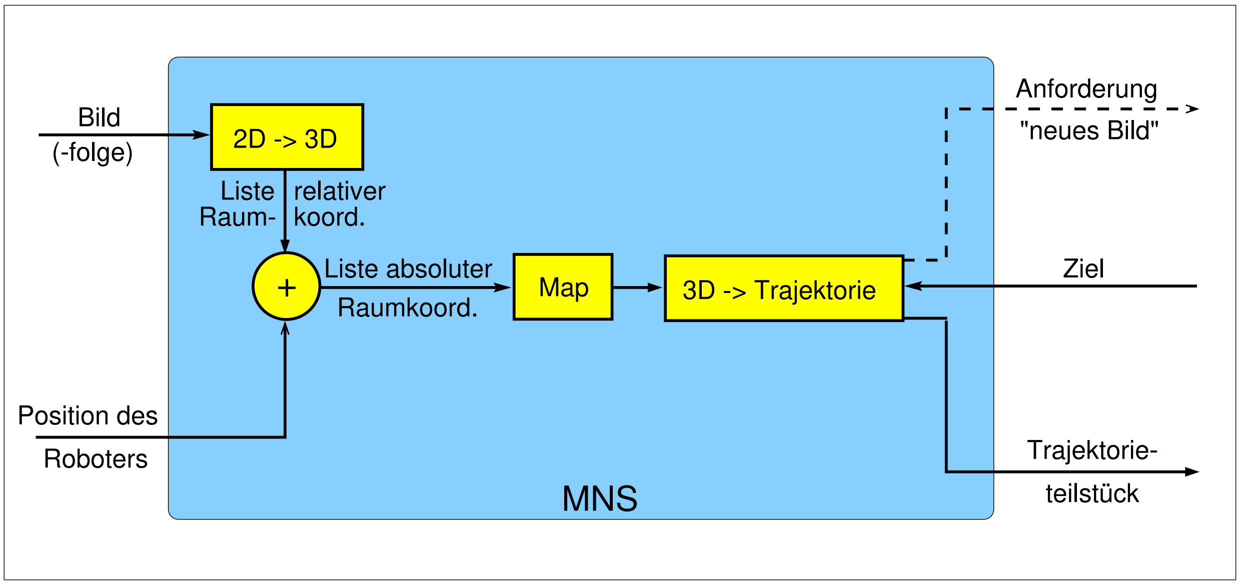

Abbildung 2.2: Komponenten des Manipulatornavigationssystems MNS

1. *Rekonstruktion dreidimensionaler Positionen aus zweidimensionalen Strukturen*

   ($\boxed{\text{2D} \rightarrow \text{3D}}$ in (Abbildung 2.2)):

   Ein Grauwertbild (Größe 512×512 Bildpunkte) wird nach bestimmten Grauwertstrukturen durchsucht. Werden diese Strukturen in aufeinanderfolgenden Bildern detektiert und zugeordnet, ist es möglich, ihre dreidimensionalen Korrespondenzen im Roboterhandkoordinatensystem zu bestimmen. Die so entstehende Liste von Punkten, die sich relativ zur Roboterhand befinden, ist, zusammen mit der Roboterhandposition, Eingabe des nächsten Schrittes.

2. *Überführung relativer zu absoluten Positionen*

($\boxed{\ +\ }$ in (Abbildung 2.2)):

Die Liste mit den relativen, dreidimensionalen Strukturpositionen aus Schritt 1. wird mit der aktuellen Roboterhandposition in absolute Positionen überführt. Die Liste absoluter Positionen im Weltkoordinatensystem ist Eingabe des Schrittes 3.

3. *Konstruktion und Aktualisierung des Besetzungsraumes*

   ($\boxed{\text{Map}}$ in (Abbildung 2.2)):

   Die Elemente der Liste absoluter Strukturpositionen im Weltkoordinatensystem werden in einen dreidimensionalen Besetzungsraum überführt, in dem die gesuchte Trajektorie zum Zielpunkt generiert wird. Dazu wird ein Raum definiert, der für jeden Punkt die Roboterhandposition sowie das Vorkommen eines Hindernisses (detektierte und ins Roboterhandkoordinatensystem übertragene Strukturposition) im Volumen des diese Position einnehmenden Roboters spezifiziert. Dieser Raum ist Basis für die Generierung eines Trajektorieteilstücks in Schritt 4.

4. *Generierung eines Trajektorieteilstücks*

   ($\boxed{\text{3D} \rightarrow \text{Trajektorie}}$ in (Abbildung 2.2)):

   Die gespeicherten Positionen des Besetzungsraumes aus Schritt 3. stellen Hindernisse (Repelloren) dar, die zusammen mit dem Zielpunkt (Attraktor) ein Vektorfeld definieren. Für eine Position in diesem Vektorfeld ist eine Trajektorie definiert, die unter Umgehung der Hindernisse in den Zielpunkt läuft. Diese wird durch neu hinzugewonnene Hindernisinformationen aktualisiert. Das nächste, auszuführende Trajektorieteilstück wird aus dem Vektorfeld für die aktuelle Postion sowie eine vorgegebene Länge des Trajektorieteilstücks ermittelt.

Ein Wahrnehmungs-Aktions-Zyklus des Manipulatornavigationssystems besteht also aus den oben geschilderten Schritten 1.-4., in denen die bisher im Besetzungsraum gespeicherten Hindernisinformationen mit neu hinzugewonnenen kombiniert werden, und somit eine aktuelle Trajektorie die folgende Roboterbewegung steuert. Nachdem der grundsätzliche Ablauf des Manipulatornavigationssystems dargestellt worden ist, werden die in den Schritten 1.-4. geschilderten Verfahren, soweit es für den Überblick des Manipulatornavigationssystems notwendig ist, näher erläutert werden, ohne jedoch den eingehenden Darstellungen der einzelnen Komponenten in den folgenden Kapiteln vorzugreifen.

2.2.1 Rekonstruktion dreidimensionaler Positionen aus zweidimensionalen Strukturen

Zur Rekonstruktion von dreidimensionalen Punkten des Roboterhandkoordinatensystems werden zwei aufeinanderfolgende Videobilder als Eingabe benötigt. Grundlage dieser Rekonstruktion sind die folgenden Komponenten:

1. *Strukturdetektion*

   Zum Punkt der Strukturdetektion ist die Frage der geeigneten Grauwertstruktur und der Eigenschaften ihres Detektors zu klären. In Kapitel 3.1 werden zwei Detektoren untersucht: Mit dem Interest Operator [29] liegt ein Detektor für Strukturen hoher Grauwertvarianz vor. Der SUSAN-Operator dagegen detektiert mit Ecken eine speziellere Kategorie der zweidimensionalen Grauwertstrukturen. Mit diesen Detektoren werden Eingabebilder auf Strukturlisten abgebildet, die die Orte der gesuchten Strukturen im Bild auflisten. Die Strukturlisten sind Eingabe des Schrittes 2.

2. *Strukturzuordnung zwischen aufeinanderfolgenden Bildern*

   Die Strukturzuordnung ist Thema des Kapitels 3.2. Sie erfolgt durch Kreuzkorrelation der Strukturumgebungen, die in der Stufe 1. der Strukturdetektion ermittelt wurden. Dabei wird zur Beschleunigung des Verfahrens eine Vorauswahl von in Frage kommenden Strukturpaaren getroffen, deren Kriterien vom System automatisch erlernt werden. Die Strukturpaare bilden die Eingabe sind Basis der zu berechnenden Besetzungspositionen.

3. *Berechnung von Besetzungspositionen*

   Mit Hilfe mehrerer Neuronaler Netze auf der Grundlage radialer Basisfunktionen können nun die Koordinaten der Strukturpaare in die drei Raumkoordinaten des Roboterhandkoordinatensystems überführt werden, wobei die Koordinaten der Tiefe (Z) getrennt von den verbleibenden (X,Y) ermittelt werden. Für die Berechnung der Tiefeninformationen wird die in den einander zugeordneten Strukturpaaren enthaltene Bildtranslation neuronal ausgewertet, während die restlichen Koordinaten der XY-Ebene in Abhängigkeit vom ermittelten Tiefenwert, ebenfalls neuronal, ermittelt werden. Die durch das System Neuronaler Netze vorgenommene Abbildung enthält gleichzeitig eine Skalierung, Translation und Entzerrung, um die nicht genau parallele Ausrichtung von Kamera- und Handkoordinatensystem (siehe Abbildung 2.3) sowie die durch das Weitwinkelobjektiv bedingten Verzerrungen zu kompensieren. Es ergeben sich dreidimensionale Punkte im Roboterhandkoordinatensystem, die als Besetzungspositionen Hindernisse im Raum markieren. Die eingehende Betrachtung dieser Verfahren findet in Kapitel 4 statt.

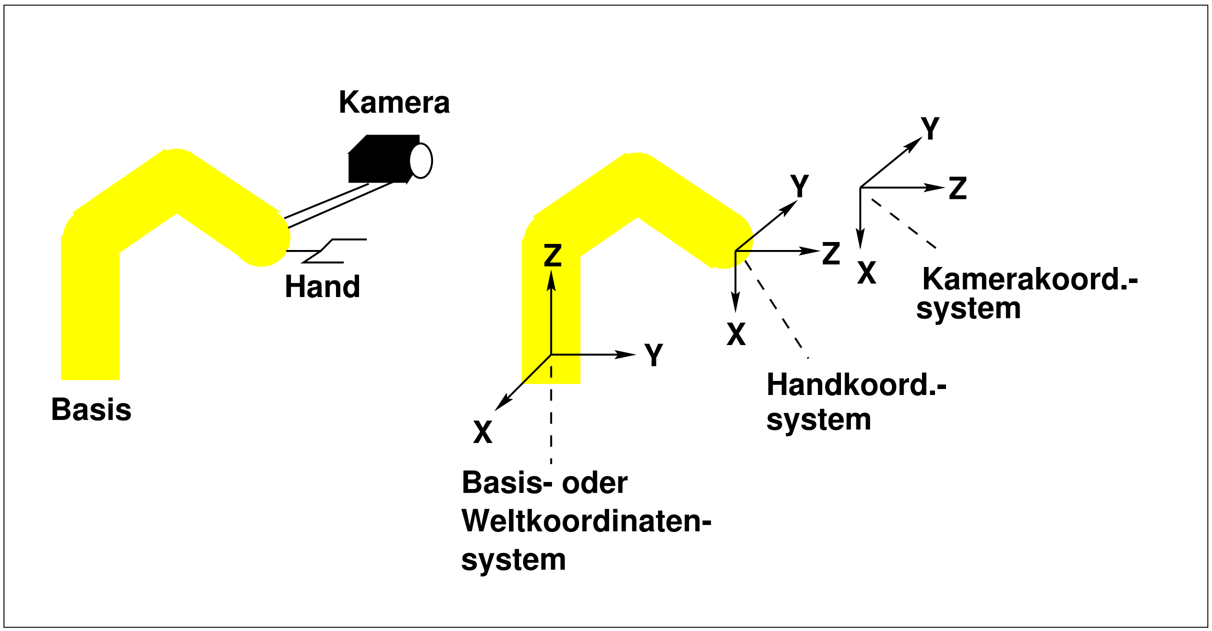

Abbildung 2.3: Aus dem Kamera- und (Roboter-)Handkoordinatensystem werden Transformationen in das Roboterbasiskoordinatensystem vorgenommen, das das Weltkoordinatensystem darstellt.

Das Zusammenwirken der in den Schritten 1.-3. geschilderten Verfahren ist in Abbildung 2.4 dargestellt.

2.2.2 Überführung relativer zu absoluten Positionen

Die im Handkoordinatensystem dargestellten Besetzungspositionen sind als relative Angaben bezüglich der Roboterhand (bzw. des Handkoordinatenursprungs) zu deuten. Um diese in das Weltkoordinatensystem, das mit dem Roboterbasiskoordinatensystem identisch ist, zu überführen, wird die zwischen den Koordinatensystemem vorhandene Rotation und Translation auf alle Daten des Handkoordinatensystems angewendet.

2.2.3 Konstruktion und Aktualisierung des Besetzungsraumes

Die Realisierung des Besetzungsraumes erfährt keine weitere Untergliederung. Die Aktualisierung der Besetzungspositionen kann in Abhängigkeit der Dynamik der Szene auf zwei unterschiedliche Weisen erfolgen:

- *Statische Szene*

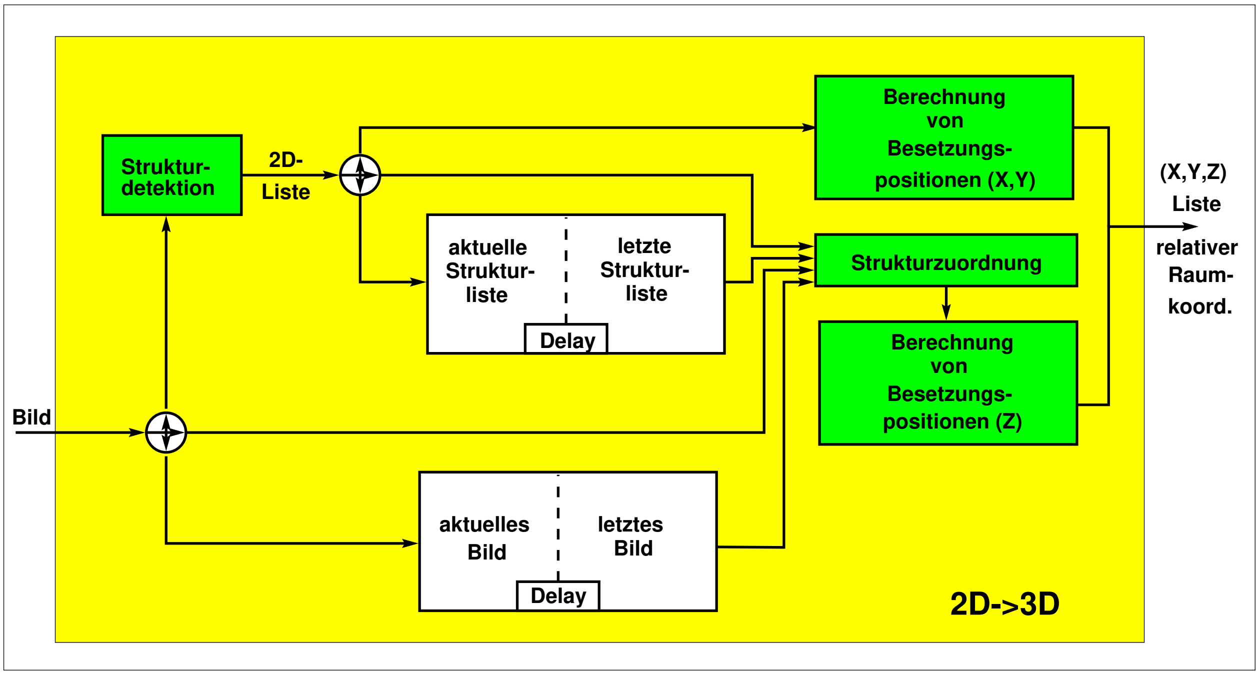

Abbildung 2.4: Berechnung von Raumkoordinaten des Handkoordinatensystems aus zweidimensionalen Bilddaten zweier aufeinanderfolgender Bilder

Für eine Szene mit unbeweglichen Objekten ist ein Hinzufügen der in jedem Schritt neu hinzugewonnenen Besetzungspunkte zur Karte der bisher gespeicherten ausreichend.

- *Dynamische Szene*

  Im Falle von beweglichen Objekten ist eine Neubewertung aller Besetzungspositionen im Bereich des aktuellen visuellen Fokus notwendig. Dies wird durch die Löschung aller im aktuellen Sichtbereich gespeicherten Besetzungspositionen und nachfolgender Neubesetzung der aktuellen Besetzungspositionen erreicht.

Die Definition und eingehende Betrachtung des Besetzungsraums wird in Kapitel 5 behandelt.

2.2.4 Generierung eines Trajektorieteilstücks

Auf der Grundlage der vorliegenden, dreidimensionalen Besetzungspositionen der Szene wird mit den folgenden Verfahren eine Trajektorie unter der Anforderung der Hindernisvermeidung berechnet:

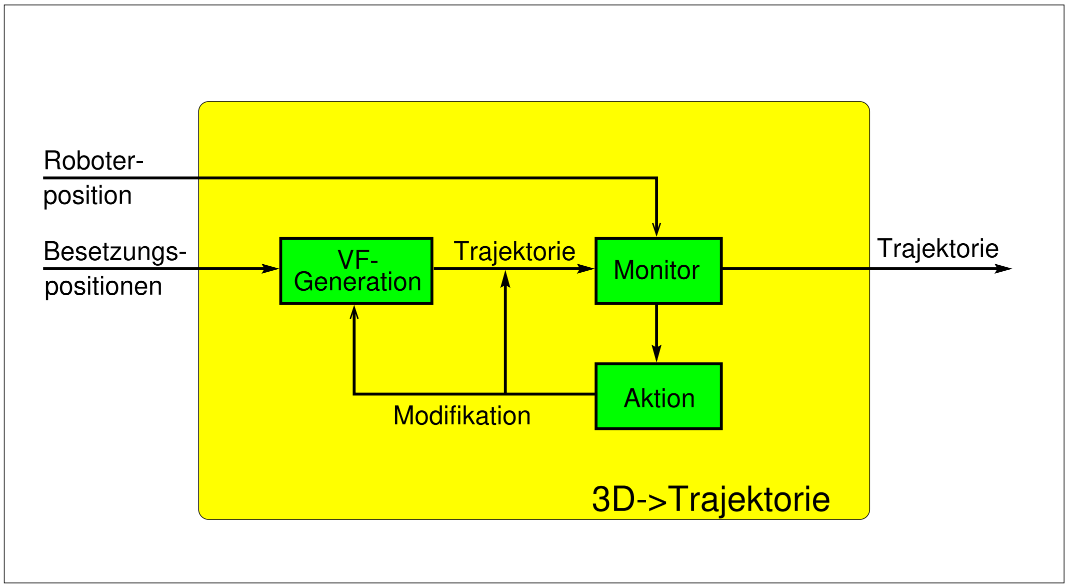

Abbildung 2.5: Konstruktion und Aktualisierung des Besetzungsraumes

1. *Vektorfeld-Generation*

   Die Liste der Besetzungspositionen wird auf der Basis von Gradienten radialer Funktionen in ein Vektorfeld, das Punkte der Anziehung (Attraktoren) als auch der Abstoßung (Repelloren) enthält, überführt. Dieses Vektorfeld liefert für eine Startposition eine zum Zielpunkt führende Trajektorie, die grundsätzlich zur Hindernisvermeidung geeignet ist. Lediglich lokale Minima sowie andere Punkte verschwindenden Potentials bedürfen weiterer Betrachtung. Dies geschieht durch die Einführung sogenannter *Monitore* und davon abhängiger *Aktionen*, siehe Kapitel 6.

2. *Monitor*

   Die Instanz, die eine Trajektorie bezüglich des in ihren Schritten gegebenen Fortschritts überwacht, ist der Monitor. Eine weitere Aufgabe des Monitors ist die Detektion des Erreichens des Zielpunktes. Seine Definition ist in Kapitel 6.6 dargestellt.

3. *Aktion*

   Detektiert der Monitor einen Punkt der Trajektorie, der ein Fortkommen unmöglich macht, wird gemäß einer Monitoraktionstabelle in Abhängigkeit vom detektierten Fall eine Aktion gewählt, die die Vektorfeldgeneration oder direkt die Trajektorie zur Umgehung des Hindernisses modifiziert. Aktionen sind Thema des Kapitels 6.7.

Kapitel 3

Strukturdetektion und Strukturzuordnung

Grundlage jeder Navigation ist ein Modell der Szene, die das navigierende Objekt, d.h. in unserem Fall den Manipulator, umgibt. Hierbei ist die Komplexität des Modells weit gefächert. Wir verwenden ein sehr einfaches Szenenmodell, das sich aus Punkten zusammensetzt, die Hindernisse beschreiben. Jeder Punkt des dreidimensionalen Szenenraumes, repräsentiert durch das Weltkoordinatensystem, ist die Rekonstruktion zweier Bildstrukturen, da Paare von Grauwertstrukturen in Bildsignalen meist mit dreidimensionalen Strukturen der abgebildeten Szene korrespondieren (eine Ausnahme bilden z.B. Schatten). Dabei findet das sogenannte *Bildfolgenstereo* Anwendung, das eine zeitlich versetzte Aufnahme zweier Bilder einer Szene aus unterschiedlichen Richtungen beinhaltet [30]. Die in diesem Bildpaar einer Bildfolge enthaltenen Szeneninformationen sind, eine entsprechende Kamerabewegung vorausgesetzt, identisch mit denen eines Stereobildpaares[1]. Neben den im Gegensatz zum Stereoverfahren kleineren Ausmaßen des Kameraaufbaus und der geringeren Anzahl der zu verarbeitenden Bilder (das letzte Bild eines Bildfolgenpaares dient als erstes Bild des nächsten Paares) ist ein Vorteil des Bildfolgenstereos die allgemeinere Bewegung der Kamera zwischen den Zeitpunkten der Bildgewinnung. Neben der beim Stereoverfahren zugrundeliegenden seitlichen Translation sowie einer möglichen, üblicherweise geringen Rotation ist in der Anwendung des Bildfolgenstereos eine beliebige Kombination von Translation und Rotation möglich, solange sich der interessierende Bildbereich in beiden Bildern der Bildfolge befindet. Daher ist die Basis aller Berechnungen des Manipulatornavigationssystems die Detektion sowie die anschließende Zuordnung von Grauwertstrukturpaaren aus je einem Bild eines Bildfolgenpaares, da die in diesem Strukturpaar zur Rekonstruktion der Korrespondenz im Weltkoordinatensystem benötigten Informationen

[1]Ein Stereobildpaar, das bei gleichzeitiger Bildaufnahme notwendigerweise mit zwei verschiedenen Kameras gewonnen wird, weist aufgrund der geringfügigen Ungenauigkeiten bei der Platzierung der CCD-Chips sowie deren leicht differierenden Helligkeiten in der Bildausgabe Unterschiede zum entsprechenden Bildfolgenpaar auf, das mit einer Kamera gewonnen werden kann. Diese Differenzen sind in der Regel gering und spielen daher eine untergeordnete Rolle.

wie Translation (zwischen den Strukturen) und dem Bildort (im letzteren der aufeinanderfolgenden Bilder) enthalten sind. Aufgrund der allgemeinen Bewegung der Kamera ist zu beachten, daß Bildinhalte nicht nur einer Translation unterzogen sein können, sondern auch einer Skalierung. Daher sind skalierungsinvariante Grauwertstrukturen wie Ecken oder Ecken enthaltene Strukturen besonders interessant. Ein weiterer Gesichtspunkt hinsichtlich der Wahl des zur Strukturdetektion verwendeten Operators ist seine Laufzeit. Nach der Einführung zweier Strukturdetektoren, dem Interest Operator sowie dem SUSAN Operator, ihrem Vergleich sowie der Beschreibung des parallelen Implementierung beschäftigen wir uns mit der Strukturzuordnung. Der beschriebene Ansatz der Strukturzuordnung wird modifiziert, um neben einer Beschleunigung auch die parallele Ausführung der Strukturzuordnung zu ermöglichen.

3.1 Strukturdetektion

Im Folgenden werden mit dem Interest Operator [29] sowie dem Operator SUSAN [31] zwei Strukturdetektoren verschiedenen Prinzips untersucht. Während der zuerst vorgestellte Interest Operator auf dem Prinzip der Grauwertvarianz basiert, detektiert der Operator SUSAN speziell Ecken.

3.1.1 Der Interest Operator (IOp)

Für die Ermittlung von Bildkorrespondenzen sind lokale Bildumgebungen mit guten Matchingeigenschaften gefordert. Dabei stellen Umgebungen mit hoher Varianz im Bildsignal von höherem Interesse für das Matching als Bereiche gleichförmigen Grauwertverlaufs. Dieser Motivation entspringt der Interest Operator, der die Grauwertdifferenzen in vier Richtungen berechnet und das Minimum zur Grundlage seiner Antwort benutzt. Damit handelt es sich um einen bezüglich der detektierten Strukturen allgemeinen Ansatz, da die Beschaffenheit der Grauwertstruktur unerheblich für die Operatorantwort ist, sondern die Varianz der in der Struktur enthaltenen Grauwerte die Antwort bestimmt.

3.1.1.1 Herleitung des Operators

Um jedem Bildpunkt (x_c, y_c) einen Interessenwert I zuzuordnen, der abhängig von der Grauwertvarianz eines ihn umgebenden Fensters S ist, werden zunächst für die Horizontalen, Vertikalen und die Diagonalen die summierten Richtungsdifferenzen berechnet:

$$I_1 = \sum_{(x,y)\in S} [f(x,y) - f(x,y+1)]^2 \tag{3.1}$$

$$I_2 = \sum_{(x,y)\in S} [f(x,y) - f(x+1,y)]^2 \tag{3.2}$$

$$I_3 = \sum_{(x,y)\in S} [f(x,y) - f(x+1,y+1)]^2 \tag{3.3}$$

$$I_4 = \sum_{(x,y)\in S} [f(x,y) - f(x+1,y-1)]^2 \tag{3.4}$$

wobei S die Menge aller Pixel des betrachteten Bildfensters und f das Bildsignal ist.

Die Summation gewährleistet durch den integrierenden Effekt, daß nicht nur Extremwerte, sondern alle Differenzen der Grauwertvarianzen für eine Richtung den resultierenden Richtungsinteressenwert beeinflussen. Geradensegmente enthaltene Bereiche sollen wegen der durch das Blendenproblem verursachten, nicht eindeutigen Rekonstruktion ihrer dreidimensionalen Korrespondenzstruktur von der Betrachtung ausgeschlossen werden. Diesbezüglich wird die Tatsache ausgenutzt, daß die Richtungssumme, die entlang des Geradensegmentes gebildet wird, gering ausfällt. Daher hat die Bildung des Minimums der vier Richtungssummen als Maß für die Operatorantwort einen niedrigen Interessenwert für einen solchen Bildbereich zur Folge, so daß dem Punkt (x_c, y_c) der folgende Wert zugewiesen wird:

$$I(x_c, y_c) \overset{Def}{=} \min(I_1, I_2, I_3, I_4) \tag{3.5}$$

Zwei weitere Kriterien liefern Verbesserungen des Endwertes: die ausschließliche Betrachtung lokaler Maxima verhindert, daß zwei benachbarte Punkte als interessant für die gleiche varianzreiche Umgebung gewählt werden. Schließlich sortiert der Vergleich mit einer Schranke Punkte mit zu niedrigem Interessenwert I aus.

3.1.1.2 Komplexität des Operators

Zur Berechnung der Ordnung des Interest Operators seien für eine Operatorgröße von $n = \mathit{Operatorhöhe} \times \mathit{Operatorbreite}$ die Anzahl der (arithmetischen) Operationen betrachtet:

1. Berechnung der gerichteten Varianzen (Gleichungen 3.1 bis 3.4): $8n$ Additionen und Subtraktionen, $4n$ Multiplikationen

2. Berechnung des Minimums der Varianzen: 3 Vergleiche

Zusammenfassend gilt:

Operation	Anzahl
Multiplikationen	4n
Additionen	8n
Vergleiche	3

Tabelle 1: Aufwand des Interest Operators

In der Summe werden für die Berechnung einer Operatorantwort $8n + 3$ Operationen verwendet, woraus eine Ordnung von $O(n)$ folgt. Die hohe Anzahl an Multiplikationen hat eine relativ hohe Laufzeit im Vergleich zu anderen Operatoren zur Folge. Aus diesem Grund haben wir den Algorithmus nach dem häufig verwendeten Verfahren der Zwischenspeicherung von Ergebnissen (Abbildung 3.1) sowie der Verwendung einer Look Up Table (LUT) für die Multiplikationsergebnisse der im Bildsignal vorkommenden Werte sowie ihren Differenzen $(-255^2 \ldots 255^2)$ optimiert. Damit reduziert sich der Aufwand wie folgt:

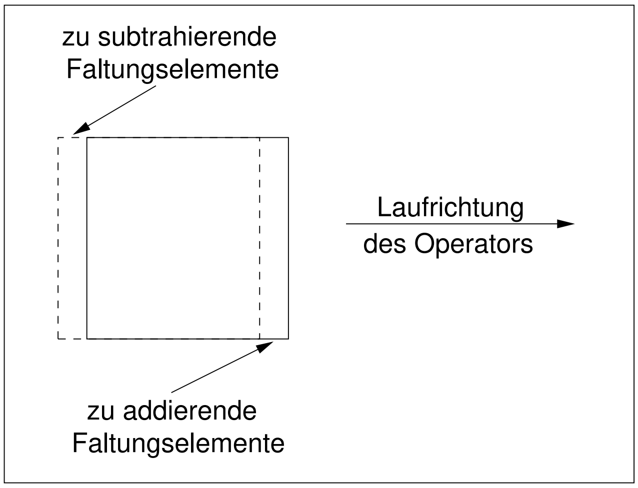

Abbildung 3.1: Die Berücksichtigung von schon berechneten Operatorantworten verkürzt die Laufzeit erheblich

1. Berechnung der Subtraktion der Bildelemente zwischen der letzten alten Bildpunktspalte und der neuen Spalte zur Eingabe in die LUT (siehe Abbildung 3.1): $4\sqrt{n}$ Subtraktionen[2]

2. Addition der neuen Spalte in die vier Richtungssummen: $4\sqrt{n}$ Additionen

[2] ausgehend von einem quadratischen Operator

3. Subtraktion der alten, herausfallenden Spalte: $4\sqrt{n}$ Subtraktionen
4. Berechnung des Minimums der Varianzen: 3 Vergleiche

Es ergibt sich ein Gesamtaufwand von:

Operation	Anzahl
Additionen / Subtraktionen	$12\sqrt{n}$
Vergleiche	3

Tabelle 2: Aufwand des optimierten Interest Operators

Nicht unterschätzt werden darf der Aufwand für die Bereitstellung der Operanden. Im Falle des Interest Operators müssen immerhin zu jeder Operatorposition für jeden Punkt des Operators vier Richtungen bedacht werden. Dies zieht zusätzliche arithmetische Operationen für Positionierungen nach sich. Eine allgemeine Abschätzung ist aufgrund der Abhängigkeit von der Implementierung nicht möglich.

Die Ausgabe des Interest Operator ist zwar sehr allgemein hinsichtlich der Berücksichtigung von Grauwertstrukturen verschiedener Art, da lediglich die Varianz der Grauwerte Maß für den Interessenwert ist; die Verwendbarkeit für die Ermittlung von Bildkorrespondenzen ist jedoch stark abhängig von der Beziehung der verwendeten Bilder untereinander. Bei einer Translation, wie sie (abgesehen von einer möglichen, üblicherweise geringen Rotation) bei Stereobildern auftritt, stimmen die Dimensionen zweier korrespondierender Grauwertstrukturen näherungsweise überein. Bei einer Translation in Richtung der optischen Achse einer Kamera verändert sich die Varianz jedoch mit der Skalierung der Bildstrukturen bei gleichbleibender Fenstergröße, so daß Punkte des einen Bildes mit hohem Interessenwert keineswegs einen entsprechenden Interessenwert am korrespondierenden Punkt des anderen Bildes implizieren. Für den Fall der beliebigen Kamerabewegung ist der Interest Operator daher nur bedingt geeignet, während für Stereobilder oder der eingeschränkten seitlichen Bewegung eine allgemeine Lösung vorliegt.

Um das Problem des nicht skalierungsinvarianten Interest Operators zu umgehen, bietet sich ein Operator für größenunabhängige Bildmerkmale an. Das einzige skalierungsinvariante Bildmerkmal ist die Grauwertecke, für deren Detektion verschiedene Operatoren zur Verfügung stehen. Dieser Ansatz verliert allerdings an Allgemeinheit bezüglich der Detektion von beliebigen Grauwertstrukturen. Daß sich dieser Verlust in der Praxis nicht als hinderlich erweist, wird später verdeutlicht werden. Den im Vergleich zu anderen Detektoren besonders schnellen Eckendetektor SUSAN stellt Smith in [31] vor.

3.1.2 Der Eckendetektor SUSAN

Bei SUSAN handelt es sich um einen rauschunempfindlichen und schnellen Detektor, den es in Versionen für ein- und zweidimensionale Bildstrukturen gibt. Für die hier gestellte Aufgabe beschränken wir uns auf die Detektion von speziellen zweidimensionalen Strukturen:

Ecken. Im Folgenden betrachten wir einen kreisförmigen Operator, dessen Mittelpunkt mit Nukleus bezeichnet wird. Die Kreisform des Operators bietet einerseits eine gleichmäßige Berücksichtigung der in die Operatorantwort einbezogenen Bildelemente, die bei einem rechteckigen Operator wegen der zum Mittelpunkt größeren Entfernung der an den Ecken des Operators liegenden Bildelemente nicht gegeben ist. Andererseits liegt durch die geringere Operatorfläche ein Laufzeitvorteil gegenüber der rechtechigen Variante vor. Die Ausgabe des Operators basiert auf der Anzahl der Bildpunkte unter der Maske, die den gleichen (oder einen ähnlichen) Grauwert wie der des Nukleus aufweisen. Dabei ist die Menge dieser Pixel gleichen Grauwertes genau dann minimal, wenn der Nukleus auf dem der Grauwertecke am nächsten liegenden Pixel lokalisiert ist (Abbildung 3.2). Abweichend von dieser Position steigt die Anzahl von Bildpunkten, die einen ähnlichen Grauwert wie der Nukleus aufweisen. Diese Eigenschaft verleiht dem Operator SUSAN seinen Namen: <u>S</u>mallest <u>U</u>nivalue <u>S</u>egment <u>A</u>ssimilating <u>N</u>ucleus.

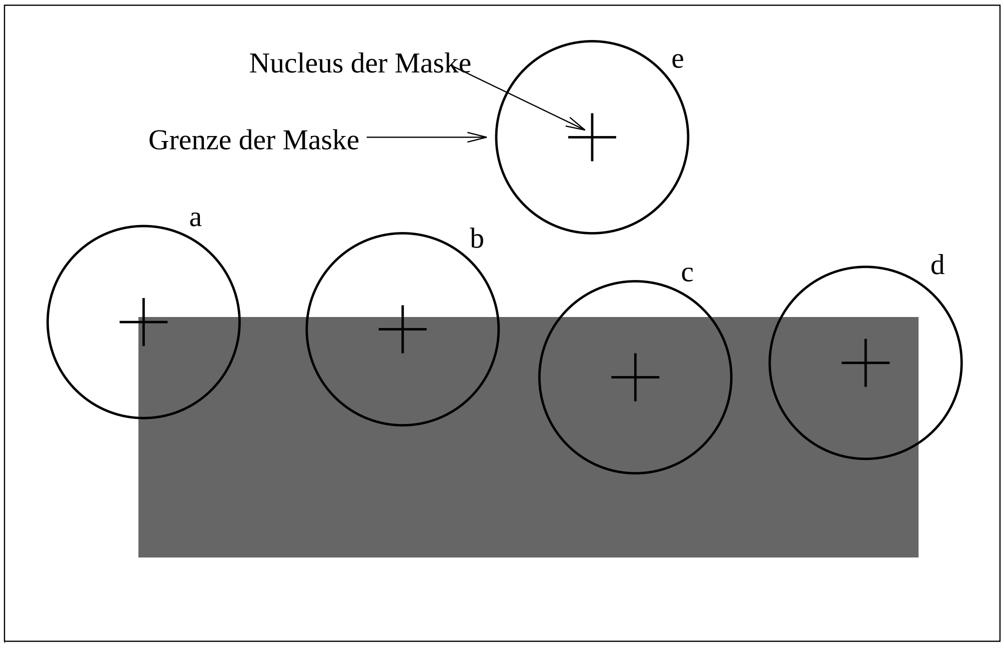

Abbildung 3.2: Verschiedene Plazierungen des Operators SUSAN (aus [31])

Das Zulassen von Bildpunkten nicht nur gleicher, sondern auch ähnlicher Grauwerte bei dem Vergleich mit dem Grauwert des Nukleus begründet zusammen mit der nichtlinearen Antwort die Rauschunempfindlichkeit des Operators.

3.1.2.1 Herleitung des Operators

Die grundlegende Berechnung des SUSAN Eckendetektors liegt in der Ermittlung der Bildpunkte unter der kreisförmigen Maske, deren Grauwert mit dem des Nukleus ähnlich ist.

Diese liegen im Regelfall, nicht aber notwendigerweise zusammen. Die Menge dieser Bildpunkte sei mit USAN (Univalue Segment Assimilating Nucleus) bezeichnet. Im Folgenden ist mit S der Bildbereich unter der Maske und mit r_0 der Nukleus beschrieben. Die Vergleichsfunktion für Bildpunkte $r \in S$ bezüglich der Grauwertähnlichkeit zum Nukleus ist mit

$$c(r_0, r) \stackrel{Def}{=} \begin{cases} 1 & falls \; |f(r) - f(r_0)| \leq t \\ 0 & falls \; |f(r) - f(r_0)| > t \end{cases} \tag{3.6}$$

gegeben. Dabei repräsentiert f das Bildsignal, und mit t ist die Schranke gegeben, die die maximal zulässige Grauwertdifferenz zweier als ähnlich einzustufender Bildpunkte beschreibt. Diese Schranke ist der einzige vorzugebende Parameter des SUSAN-Operators, der, für eine gegebene Operatorgröße n, einerseits die Empfindlichkeit der Operatorantwort und damit die Anzahl der detektierten Ecken beeinflußt, und andererseits den aufzufangenden Grad des Bildrauschens angibt. Dieser kann entweder manuell vorgegeben oder im Falle eines Bildstapels (z.B. bei einer Videoaufnahme) leicht automatisch durch Detektion des Bildrauschens bestimmt werden. Die Berechnung der Anzahl der Bildpunkte, für die die Vergleichsfunktion nicht verschwindet, wird durch

$$a(r_0) \stackrel{Def}{=} \sum_{r \in S} c(r, r_0) \tag{3.7}$$

vorgenommen. Liegt der Nukleus auf einer Grauwertecke, so weist die Funktion a im rauschfreien Fall ein lokales Minimum an der Stelle r_0 auf. In diesem Fall gilt: $a(r_0) < \frac{1}{2}n$, wobei n die Göße der Maske bezeichnet (Abbildung 3.3). Dieses Wissen läßt sich zur Eliminierung von rauschbedingten Grauwertfehlern, die $a(r_0)$ auf einen Wert größer als $\frac{1}{2}n$ anwachsen lassen, benutzen:

$$R(r_0) \stackrel{Def}{=} \begin{cases} \frac{1}{2}n - a(r_0) & \text{falls } a(r_0) < \frac{1}{2}n \\ 0 & \text{sonst} \end{cases} \tag{3.8}$$

Die Operatorantwort R an der Position des Nukleus (r_0) hat nun ein lokales Minimum, wenn r_0 der Ort einer Grauwertecke ist (Abbildung 3.4). Daher liefert eine nachgeschaltete sogenannte non-maximum suppression das endgültige Ergebnis des SUSAN Detektors.

Smith stellt in seinen Fehlerbetrachtungen fest, daß es für die Vergleichsfunktion (Gleichung 3.6) einen Ersatz mit geringerem zu erwartenden Fehler gibt. Die Neudefinition von c mit

$$c(r, r_0) \stackrel{Def}{=} e^{-\left(\frac{f(r)-f(r_0)}{t}\right)^6} \tag{3.9}$$

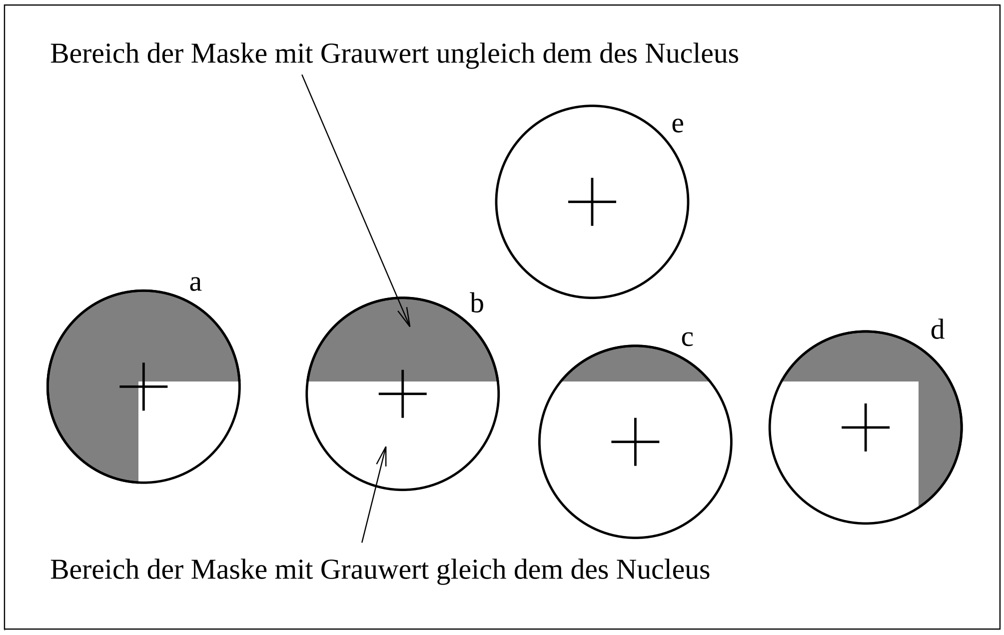

Abbildung 3.3: Bereiche gleichen Grauwertes wie der des Nukleus sind hell markiert (aus [31])

stellt ein fehlerminimierendes Mittel zwischen der Box-Funktion, wie sie durch die Schranke t gegeben ist, und der Gaussfunktion (e^{-x^2}) dar (siehe Abbildung 3.5, Kurve b)). Aus Effizienzgründen ist es sinnvoll, diese Funktion als Tabelle (Look Up Table) zu implementieren.

Es existieren Grauwertverteilungen, die, obwohl sie keine Ecke beinhalten, eine positive Antwort des Operators zur Folge haben (Abbildung 3.6). Diese Strukturen weisen jedoch die Eigenschaft auf, daß der geometrische Schwerpunkt des USAN-Bereiches mit dem Ort des Nukleus identisch ist, so daß ein Test auf den Abstand von Nukleus und Schwerpunkt diese Fehlerquelle eliminiert.

3.1.2.2 Komplexität des Operators

Zur Berechnung der Ordnung des SUSAN-Operators seien für eine Operatorgröße von n unter der kreisförmigen Maske liegenden Bildelementen die Anzahl der arithmetischen Operationen betrachtet:

1. Berechnung der Vergleichsfunktion c (Gleichung 3.9): $2n-2$ Subtraktionen, $n-1$ Divisionen, $3n-3$ Multiplikationen, $n-1$ Exponentialfunktionen

2. Berechnung der Summe a (Gleichung 3.7): $n-1$ Additionen

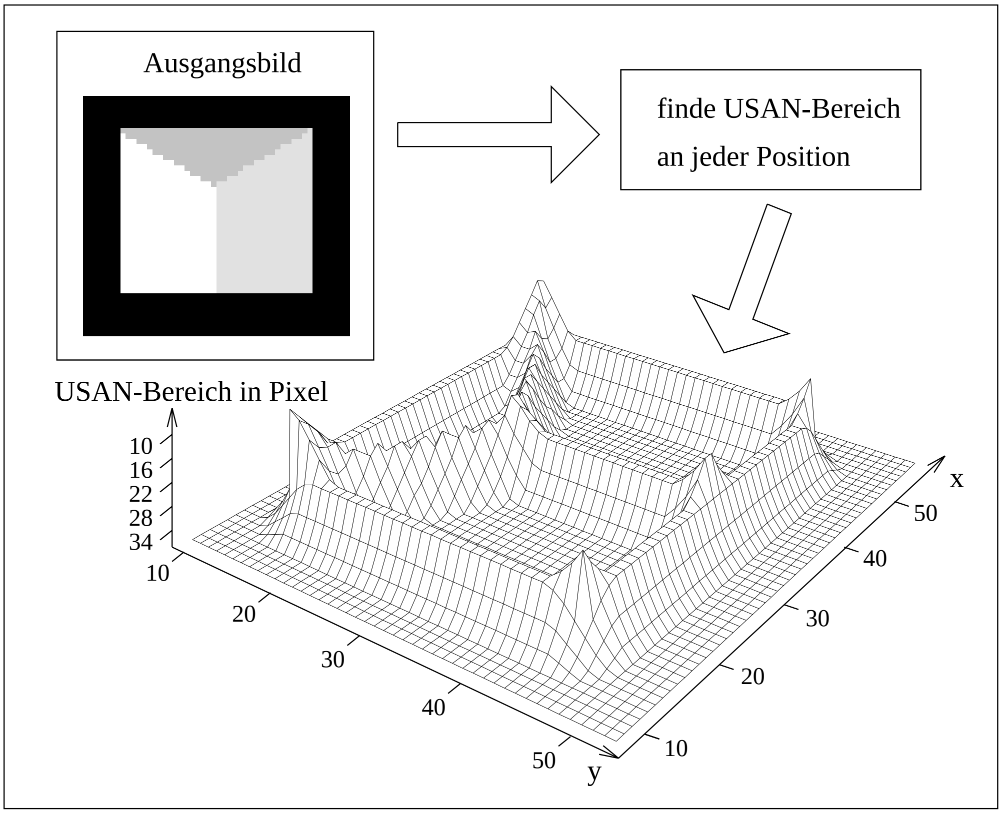

Abbildung 3.4: Die Ausgabe des SUSAN-Operators markiert Kanten und in stärkerem Maße Ecken (aus [31]). Zu beachten ist die umgekehrte Richtung der die Anzahl der Pixel markierenden, senkrechten Achse.

3. Vergleich mit geometrischer Schranke (Gleichung 3.8): 1 Vergleich, 1 Subtraktion

4. Berechnung des geometrischen Schwerpunktes und Vergleich mit Nukleus: $2n$ Additionen, 2 Divisionen, 2 Vergleiche

Zusammenfassend gilt:

Operation	Anzahl
Additionen/Subtraktionen	$5n-2$
Multiplikationen/Divisionen	$4n-2$
Exponentialfunktionen	$n-1$
Vergleiche	3

Tabelle 3: Aufwand (worst case) des SUSAN-Operators

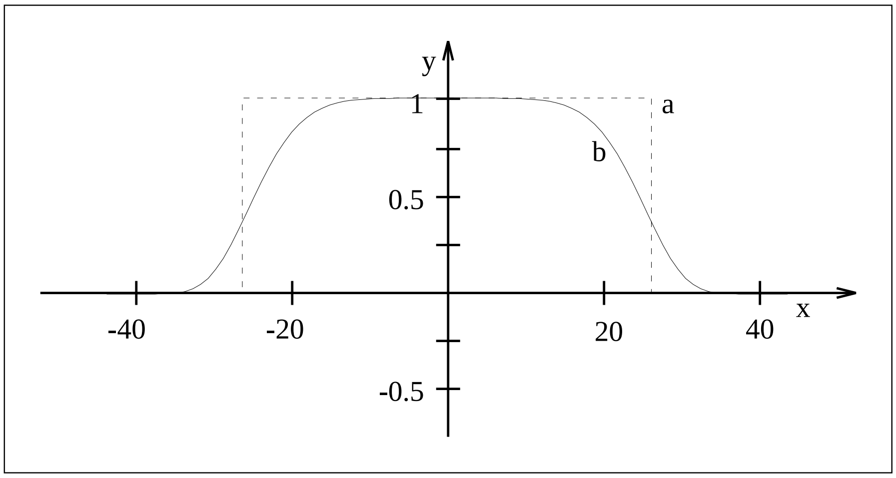

Abbildung 3.5: a) Die originale Vergleichsfunktion (x-Achse: Grauwertdifferenz der Bildpunkte, y-Achse: ohne Einheit). Hier hat die Schranke t den Wert 27, b) Die stabilere Vergleichsfunktion auf Basis der Exponentialfunktion (aus [31])

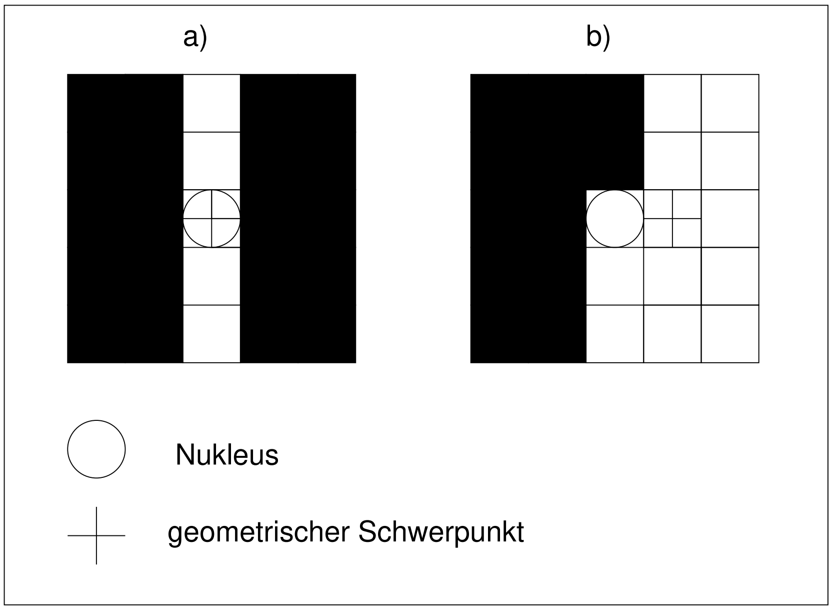

Abbildung 3.6: Der Abstand zwischen Nukleus und geometrischen Schwerpunkt des USAN-Bereiches ist Maß für die Korrektheit der Eckendetektion. Stimmen Nukleus und Ort des Schwerpunktes überein (a), so ist ein evtl. vorhandenes lokales Maximum nicht als Ecke zu werten. Differieren Nukleus und Schwerpunkt (b), korrespondiert ein lokales Maximum mit einer Ecke (nach [31]).

In der Summe werden für die Berechnung einer Operatorantwort höchstens $10n-2$ Operationen verwendet, woraus eine Ordnung von $O(n)$ folgt. Hierbei handelt es sich im Gegensatz zum Interest Operator, dessen Aufwand bei jeder Anwendung konstant bleibt, um eine worst case-Schätzung. Die Berechnung des geometrischen Schwerpunktes hängt von dem in Gleichung (3.8) vorgenommenen Vergleich mit der geometrischen Schranke und der Größe des USAN-Bereiches ab. Auch bei diesem Operator bietet sich eine Optimierung durch Implementation der Exponentialfunktion $e^{-\left(\frac{x}{t}\right)^6}$ als Look Up Table (LUT), so daß sich der Aufwand des Operators wie folgt reduziert:

1. Berechnung von $f(r) - f(r_0)$ zur Eingabe in die LUT: $n-1$ Subtraktionen
2. Berechnung der Summe a : $n-1$ Additionen
3. Vergleich mit geometrischer Schranke: 1 Vergleich, 1 Subtraktion
4. Berechnung des geometrischen Schwerpunktes und Vergleich mit Nukleus: $2n$ Additionen, 2 Divisionen, 2 Vergleiche

Zusammenfassend gilt:

Operation	Anzahl
Divisionen	2
Additionen/Subtraktionen	$4n-1$
Vergleiche	3

Tabelle 4: Aufwand (worst case) des optimierten SUSAN-Operators

Es entsteht als Oberschranke ein Aufwand von $4n+4$ Operationen. Wieviel Operationen weniger tatsächlich ausgeführt werden, ist von der zu Grunde liegenden Grauwertstruktur abhängig und wurde daher in Form von Laufzeitbetrachtungen untersucht. Der Aufwand für die Bereitstellung der Operanden ist vergleichsweise gering, da für jede Operatorposition für jeden Punkt des Operators zwei Operanden benötigt werden, von denen einer fest gewählt ist. Die Positionierungsarbeit bezieht sich daher nur auf einen Operanden.

3.1.3 Interest Operator versus SUSAN

Im Folgenden werden ermittelter Aufwand, reale Laufzeit sowie die Genauigkeit der Operatoren untersucht. Zwar ist der Interest Operator hinsichtlich der detektierten Grauwertstrukturen allgemeiner als der SUSAN-Operator. In diesem Zusammenhang ist jedoch von Interesse, welches Gewicht diese theoretischen Eigenschaften der Operatoren für die reale Anwendung haben. Aus diesem Grund werden sowohl synthetische als auch reale von einer Videokamera generierte Testbilder untersucht.

Aufwand und Laufzeit

Eine grobe Einschätzung der beiden Operatoren läßt der Aufwand der arithmetischen Operatoren bezüglich einer Operatorgröße von n zu. Hierbei ist zu beachten, daß die Größe n beim SUSAN-Operator, bedingt durch die kreisförmige Gestalt, einen um ca. 30% niedrigeren Wert als beim quadratischen Interest Operator aufweist. Diesem Umstand ist beim Vergleich des arithmetischen Aufwandes in Abbildung 3.7 durch entsprechende Skalierung der zum SUSAN-Operator gehörenden Kurve Rechnung getragen worden. Für eine Operatorgröße von 7×7 (IOp) sowie eines Operatordurchmessers von 7 (SUSAN) ergibt sich ein für den Interest Operator höherer arithmetischer Aufwand von etwa zwei Dritteln. Nicht einbezogen sind die implementationsabhängigen Positionierungsoperationen (z.B. Indizes in Arrays), die zum Teil ebenfalls arithmetischer Natur sind.

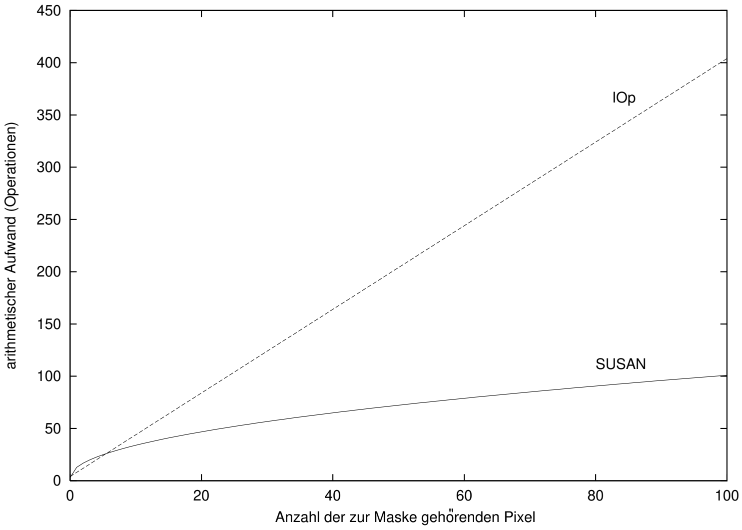

Abbildung 3.7: Vergleich des arithmetischen Aufwandes des Interest- und des SUSAN-Operators

In Tabelle 5 werden die realen Laufzeiten der Operatoren auf verschiedenen Rechnerarchitekturen hinsichtlich des Testbildes einer Größe von 512 x 512 Bildpunkten (Abbildung 3.8) gegenübergestellt.

Die Messungen beziehen sich beim Interest Operator auf eine Maskengröße von 7×7. An Optimierungsmaßnahmen sind eine Look Up Table (LUT) für die Multiplikationen sowie das Zurückgreifen auf schon berechnete Operatorantworten benutzt worden. Der verwendete Durchmesser des SUSAN-Operators beträgt 7, die Exponentialfunktion ist mit Hilfe eine LUT implementiert worden. Beide Operatoren haben sprachspezifische Optimierungen (zeigerbasierte Operationen in der Sprache C) erfahren.

Die reale Laufzeit (gerundet auf 1/10 Sekunde) auf verschiedenen Rechnerarchitekturen zeigt signifikante Unterschiede auf. Während die Laufzeiten beider Operatoren auf dem

Abbildung 3.8: Zur Laufzeitbestimmung verwendetes Testbild der Originalgröße 512×512 Pixel

AMD Pentium etwa gleich hoch liegen, differieren sie auf den SUN 167 MHz Rechnern um den Faktor 5. Der Vergleich zwischen den Rechnerarchitekturen ergibt für die Laufzeit des Operators SUSAN auf beiden SUN Prozessoren (167 MHz u. 300MHz) ca. 53% der Laufzeit des AMD Pentium. Der Interest Operator auf der SUN 167 MHz bringt es auf 138% mehr Laufzeit, als der Pentium in Anspruch nimmt (38% beim SUN 300 MHz Prozessor). Überraschend ist die ähnliche Laufzeit des Operators SUSAN (singlethreaded) auf den beiden SUN Prozessoren. Unabhängig von der Taktfrequenz von 167MHz oder 300MHz benötigt die Verarbeitung des Testbildes ca. eine Sekunde. Hier kommen Hardwarezugriffe zum Tragen, die nicht vom Prozessor abhängen. Die verwendete Rechnerarchitektur hat also ein entscheidendes Gewicht bezüglich der Performance der Operatoren. Es ergibt sich für den SUSAN-Operator ein je nach verwendeter Rechnerarchitektur verschieden ausgeprägtes Plus bezüglich der Laufzeit. Für die Implemenation auf den SUN Rechnern ist das Verhältnis der Laufzeiten von IOp zu SUSAN mit 3:1 gegeben.

Operatorantwort und Genauigkeit

Die Operatorantworten sind anhand synthetischer sowie realer Grauwertbilder untersucht worden. Die Operatorantworten bezüglich eines synthetischen Testbildes, welches auf wenigen Grauwerten basiert und kein Rauschen aufweist, sind in (Abbildung 3.9) festgehalten.

	Interest Operator	SUSAN-Operator
Ordnung	$O(n)$	$O(n)$
Anzahl der arithmetischen Operatoren	$12\sqrt{n}+3$	$4n+4$ (worst case)
AMD K5 Pentium 100 MHz	2.1 Sek.	1.9 Sek.
SUN 167 MHz Ultra 1	5.0 Sek.	1.0 Sek.
SUN 167 MHz Ultra 1, 4 Prozessoren, multithreaded	1.4 Sek.	0.4 Sek.
SUN 300 MHz Ultra 1	2.9 Sek.	1.0 Sek.
SUN 300 MHz Ultra 1, 2 Prozessoren, multithreaded	1.5 Sek.	0.5 Sek.

Tabelle 5: Aufwand und Laufzeiten der Operatoren IOp und SUSAN

Deutlich kommen die unterschiedlichen Ansätze der Operatoren zur Geltung: Der Interest Operator detektiert im Gegensatz zu SUSAN nicht nur die Ecken, sondern bei den Ellipsen auch Orte mit gewisser Krümmung. Diese weisen durch ihre Krümmung genug Abweichung von einer Geraden auf, so daß in jeder der vier Detektionsrichtungen des Interest Operators genug Grauwertvarianz für eine ausreichende Operatorantwort enthalten ist. Diese Krümmung ist jedoch für den SUSAN-Operator nicht ausreichend, um die anliegenden Gebiete ähnlicher Grauwerte klein genug zu halten, damit sie den Anforderungen eines mit USAN bezeichneten Gebietes genügen und so zu einer ausreichenden Antwort führen. Diese sind erwartungsgemäß an den Enden der Ellipsen zu finden.

Zur Positionierungsgenauigkeit läßt sich eine im Mittel 2 - 3 Bildpunkte betragende Abweichung bei der Antwort des Interest Operators feststellen. Die Ecken der rechteckigen Bildstrukturen werden vom SUSAN-Operator korrekt detektiert. Auch die Punkte größter Krümmung werden bei den Ellipsen korrekt detektiert. Lediglich in einem Fall liegen zwei detektierte Ecken am Ende einer Ellipse. Die Ursache hierfür ist eine Ungenauigkeit bei der Ellipsenzeichnung.

Um das Verhalten der Detektoren unter Rauschen zu prüfen, wurde dem Testbild der (Abbildung 3.9) weißes Rauschen (gaussian noise) mit einer Standardabweichung von 10 zugefügt. Auch unter Rauschen (Abbildung 3.10) behält der SUSAN-Operator seine Positioniergenauigkeit. Der Interest Operator detektiert zwar nicht ungenauer als im unverrauschten Bild, jedoch verändern sich Position und Anzahl der Detektionen.

Um die Frage zu klären, ob sich die Einschränkung, lediglich Ecken durch SUSAN im Gegensatz zu den vom Interest Operator detektierten allgemeineren Grauwertstrukturen zu berücksichtigen, bei der Anwendung auf Videobilder einer realen Szene signifikant auswirkt, werden beispielhaft Antworten beider Operatoren mit jeweils näherungsweiser Empfindlichkeit (d.h. annähernd gleicher Anzahl der Detektierten Grauwertstrukturen) gegenübergestellt (siehe Abbildung 3.11). Die verwendeten Operatorparameter sind für den Interest Operator eine Operatorgröße von 7×7 sowie ein Schrankenwert von 20. Für den SUSAN-Operator wurde ein Durchmesser von 7 sowie ein Schrankenwert von 12 verwendet.

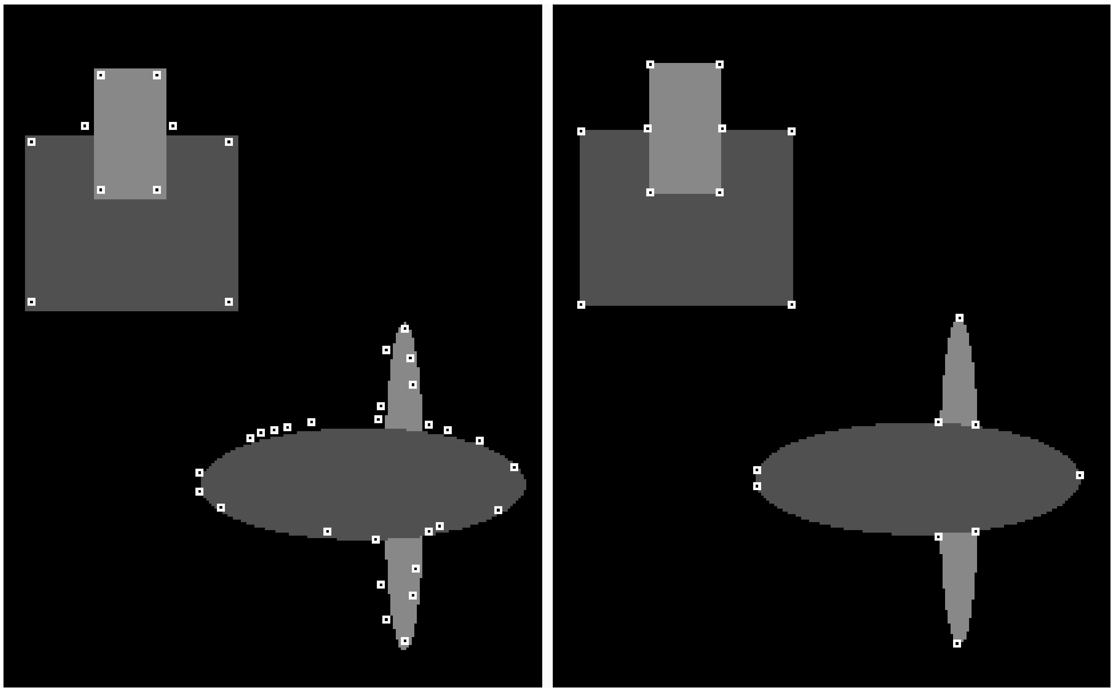

Abbildung 3.9: Links: Antwort des Interest Operators, rechts: Antwort des SUSAN-Operators

Die Unterschiede der beiden Operatorantworten sind marginal. Die Anzahl der Gebiete, die nur in einer Operatorantwort eine Detektion aufweisen können, ist verschwindend gering. Auch für höhere und niedrigere Empfindlichkeiten eines Operators lassen sich korrespondierende Schrankenwerte des anderen Operators finden, münden aber in wenig praktikable Mengen von detektierten Strukturen. Obige Werte entsprechen denen der realen Anwendung.

Zusammenfassend bietet der SUSAN-Operator die schnellere und genauere Lösung, die zudem den Vorteil der automatischen Schrankenwertbestimmung bietet. Praktisch wurde diese Bestimmung der Grauwertschranke durch Messung der bildpunktweisen Grauwertdifferenzen innerhalb einer Bildfolge solange durchgeführt, bis keine Erhöhung des Maximums der Grauwertdifferenzen mehr meßbar war. Das Maximum der Grauwertdifferenzen bietet sich als Schrankenwert für den SUSAN-Operator an. Der Interest Operator erfordert dagegen einen handgewählten Schrankenwert. Obwohl er die allgemeinere Lösung hinsichtlich der erkannten Grauwertstrukturen bietet, fällt dieser theoretische Aspekt in der praktischen Anwendung nicht signifikant ins Gewicht. Aus diesen Gründen wird der SUSAN-Operator für die Detektion von strukturbehafteten Umgebungen zur Ermittlung von Bildkorrespondenzen für das weitere Verfahren ausgewählt.

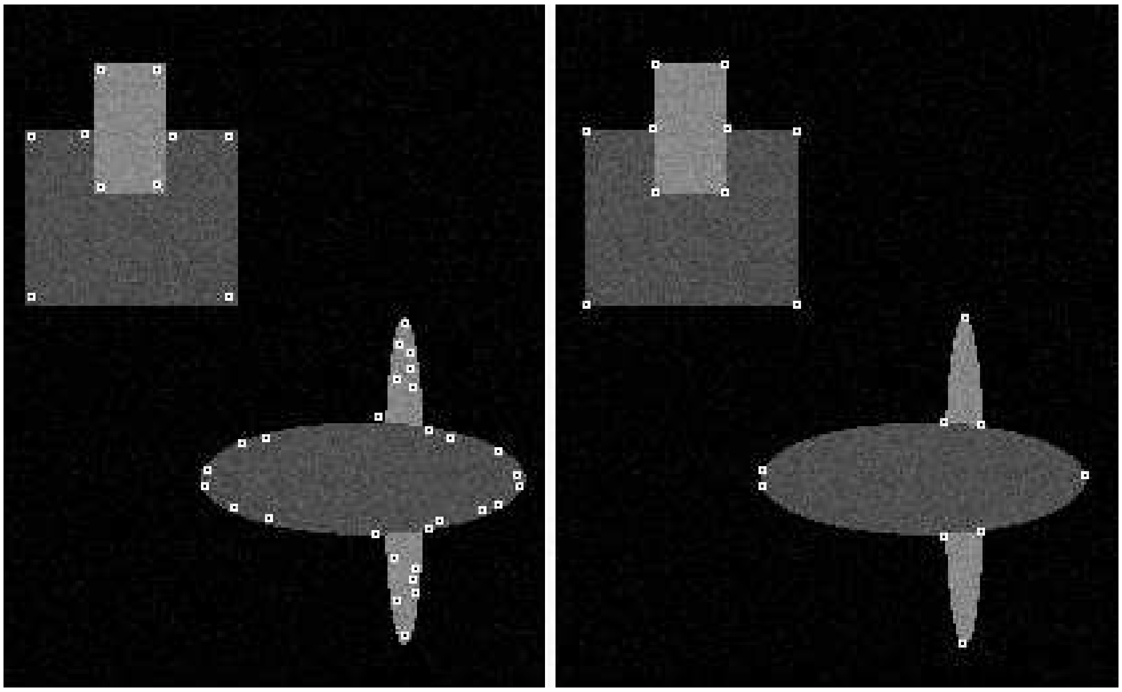

Abbildung 3.10: Weißes Rauschen (gaussian noise) mit Standardabweichung 10 beeinflußt nur die Positionierung des Interest Operators (Schrankenwert 400, links), die Antwort des SUSAN-Operators bleibt gleich (Grauwertschranke 50, rechts)

3.1.4 Parallele Implementierung der Operatoren

Bildverarbeitungsoperatoren eignen sich allgemein sehr gut für eine nebenläufige Implementierung zur Abarbeitung auf Mehrprozessorrechnern. Das Prinzip ist die gleichzeitige Anwendung mehrerer Inkarnationen eines Operators auf verschiedene Bildteile (Abbildung 3.12). Hierbei ist für eine Überlappung der Teilbereiche zu sorgen, damit keine Lücken in der Operatorantwort entstehen. Diese Überschneidung muß, in Bildpunkten gemessen, die Distanz aufweisen, die zwischen dem Zielpunkt der Operatorantwort und dem davon am weitesten entfernten Operatorrand liegt. Dieses Verfahren skaliert sehr gut mit der Anzahl der verwendeten Parallelkonstrukte[3]. Lediglich der zusätzliche Aufwand für die Überschneidungen der Bildbereiche sowie der für die parallele Implementation notwendige Overhead verringert den Laufzeitgewinn. In Kapitel 3.1.3 sind die Laufzeiten der parallelen und der nichtparallelen Implementationen gegenübergestellt; das Verhältnis der beiden Laufzeiten beträgt bei vier eingesetzten Prozessoren zwischen 1 : 2.5 (SUSAN) und 1 : 3.5 (Interest Operator).

[3]Parallele Algorithmen sind mittels *Multithreading* auf einem Mehrprozessorrechner implementiert worden. Ebenso kann eine Parallelisierung auf mehreren Prozessen oder sogar Rechnern (Cluster) basieren. Die hier gegebene Beschreibung der Parallelisierung abstrahiert von Implementationstechniken.

Abbildung 3.11: Der allgemeinere Interest Operator (links) weist in der Anwendung auf ein Realbild kaum Unterschiede zur Antwort des SUSAN-Operators (rechts) auf.

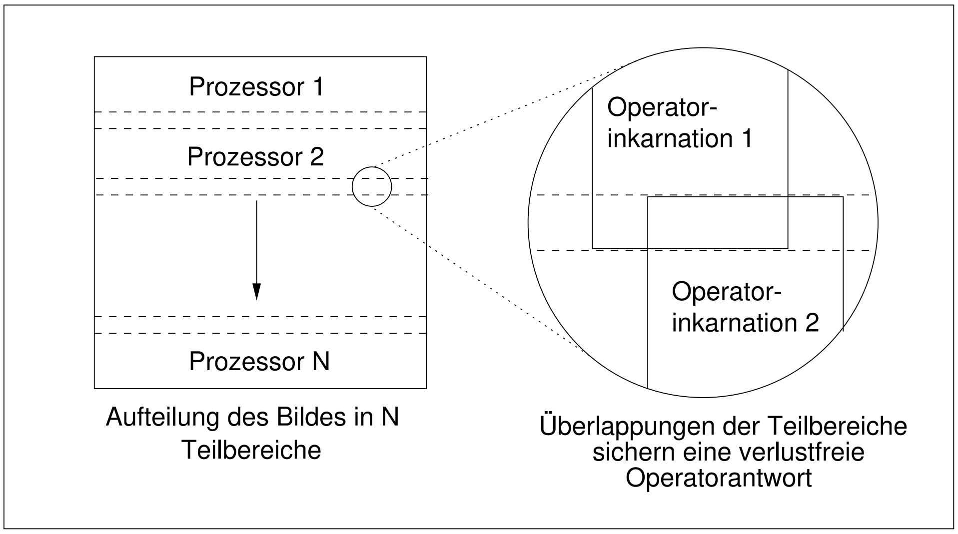

Abbildung 3.12: Beschleunigung der Operatorlaufzeit durch Parallelverarbeitung

3.2 Strukturzuordnung

Das Manipulatornavigationssystem generiert im Einsatz eine Folge von Bildern, zwischen deren Aufnahme jeweils eine Bewegung des Roboterarms und damit der Kamera liegt. Für zwei aufeinanderfolgende Bilder dieser Folge ist es mit dem aus den oben beschriebenen Operatoren ausgewählten Eckendetektor SUSAN möglich, das Bildpaar in zwei Listen von Ecken enthaltene Bildpositionen zu überführen. Die Aufgabe besteht nun darin, für zwei Bildpositionen der ersten und der zweiten Liste von detektierten Ecken das Korrespondenzproblem im Bildpaar zu lösen. Ist dies geschehen, gibt das Paar der zugeordneten Grauwertstrukturen Aufschluß über die zugrunde liegende Position im dreidimensionalen Raum (siehe Kapitel 4). Ein Standardverfahren zur Lösung des Korrespondenzproblems, welches auch hier Anwendung findet, ist mit der Maximierung des Kreuzkorrelationskoeffizienten gegeben [29]. Für eine Bildposition p_i einer aus dem Bild b_i generierten Liste von Ecken $L = \{p_1, \ldots, p_N\}$ existiert eine Bildumgebung u_i um die Position p_i im Bild b_i herum. Zu jedem Paar (p_i, q_j) von Bildpositionen der ersten und zweiten Liste von Ecken und den zugehörigen Bildumgebungen u_i (um p_i) und v_j (um q_j) ist der Kreuzkorrelationskoeffizient wie folgt definiert:

$$\gamma_{i,j} = \frac{\sum\limits_x \sum\limits_y (u_i(x,y) - \bar{u}_i)\,(v_j(x,y) - \bar{v}_j)}{\sqrt{\sum\limits_x \sum\limits_y (u_i(x,y) - \bar{u}_i)^2 \sum\limits_x \sum\limits_y (v_j(x,y) - \bar{v}_j)^2}}\,, \quad -1 \leq \gamma_{i,j} \leq 1 \tag{3.10}$$

wobei $\bar{u}_i$ und $\bar{v}_j$ die mittleren Grauwerte der Bildumgebungen u_i und v_j sind und u_i und v_j gleichgroß ($T \times T$ Pixel) sind. Dabei steht ein Kreuzkorrelationskoeffizient von -1 für völlig ungleiche Umgebungen, ein Wert von 1 wird bei Identität der Bildfragmente erreicht.

Für zwei Strukturlisten $L_1 = \{p_1 \; \ldots \; p_N\}$ und $L_2 = \{q_1 \; \ldots \; q_M\}$ ist nun für jedes mögliche Paar $(p_i, q_j) \in L_1 \times L_2$ der entsprechende Kreuzkorrelationskoeffizient $\gamma_{i,j}$ zu berechnen. Für eine Eckenposition $p_i \in L_1$ bestimmt das Maximum der Kreuzkorrelationskoeffizienten der Elemente von L_2 den zugehörigen Partner $q_k \in L_2$:

$$Partner\,(p_i) \quad \overset{Def}{=} \quad \begin{cases} q_k & \text{falls} \quad \gamma_{i,k} = \max\limits_j \{\gamma_{i,j} \mid i \in \{1,\ldots,N\}, j \in \{1,\ldots,M\}\} \\ & \quad \wedge \qquad \gamma_{i,k} > C \\ nil & \text{sonst} \end{cases} \tag{3.11}$$

wobei $C \in [-1, 1]$ eine Schranke für den Mindestwert des Ähnlichkeitsmaßes darstellt, unter dem ein Paar als nicht zueinander gehörig betrachtet wird. Auf der Grundlage der beiden Testbilder der Abbildung 3.13 wird zur Betrachtung der Auswirkungen des Parameters C auf die Ausgabe der Gleichung (3.11) ein Histogramm erstellt, das die Häufigkeit der Länge der Verschiebevektoren, die durch die berechneten Korrespondenzen bestimmt sind, in Abhängigkeit des Parameters C darstellt (Abbildung 3.14). Durch die Auswirkungen des Weitwinkelobjektivs ist der optische Fluß selbst bei seitlicher Translation (Bewegungsrichtung orthogonal zur optischen Achse) nicht an jeder Stelle des Bildes gleich. Auf diesen

Abbildung 3.13: Testbilder für die Strukturzuordnung bei seitlicher Translation von $5cm$ der Roboterhand. Links: Vor der Translation. Mitte: Nach der Translation. Rechts: Strukturzuordnung mit dem Parameterwerten $C = 0.94$ und $T = 11$.

Umstand wird in Kapitel 4.2.2 näher eingegangen. Daher ist das Histogramm für einen Wert C nicht nur an einer Stelle ungleich Null, sondern weist eine Bandbreite positiver Werte um das Maximum auf. Es ist zu beobachten, daß sich dieser Bereich des Histogramms schon bei Werten für C kleiner als 0.9 ausbreitet.

Da mit steigendem Parameterwert C die Mächtigkeit der Menge der ermittelten Korrespondenzen und damit die Menge der rekonstruierbaren Punkte der Szene sinkt, ist ein Kompromiß bezüglich des Parameters C sowie der Menge der berechenbaren Korrespondenzen zu treffen. Wir verwenden daher für C einen Parameterwert von 0.94 und wenden eine Heuristik zur Detektion und Löschung von falsch berechneten Korrespondenzen an. Diese nutzt die geringe Häufigkeit von falsch detektierten Zuordnungen aus, indem bei der Rekonstruktion der Punkte (Kapitel 4) ihre Korrektheit nach der Anzahl ihrer Nachbarn in einer lokalen Umgebung beurteilt wird. Die Auswirkung dieser Vorgehensweise ist bei der praktischen Erprobung in Kapitel 7 dargestellt.

3.2.1 Beschleunigung der Strukturzuordnung

Die Berechnung des Kreuzkorrelationskoeffizienten ist sehr rechenintensiv. Zusätzliche Kriterien als Voraussetzung der Paarbildung beschleunigen das Verfahren. Um den Rechenaufwand zu reduzieren, bieten sich zwei Verfahren an. Das erste basiert auf der Bestimmung der Translationslänge der Grauwertstrukturen im Bildpaar. Denn die Länge der Translation einer Grauwertstruktur in einem Bildpaar ist abhängig von der Entfernung der dreidimensionalen, korrespondierenden Struktur von der Kamera bzw. der Bildebene. Des weiteren ist die Translationslänge abhängig von der Lage der Grauwertstrukturen im Bildort, da die durch die Linse bedingten Bildverzerrungen besonders stark beim Einsatz

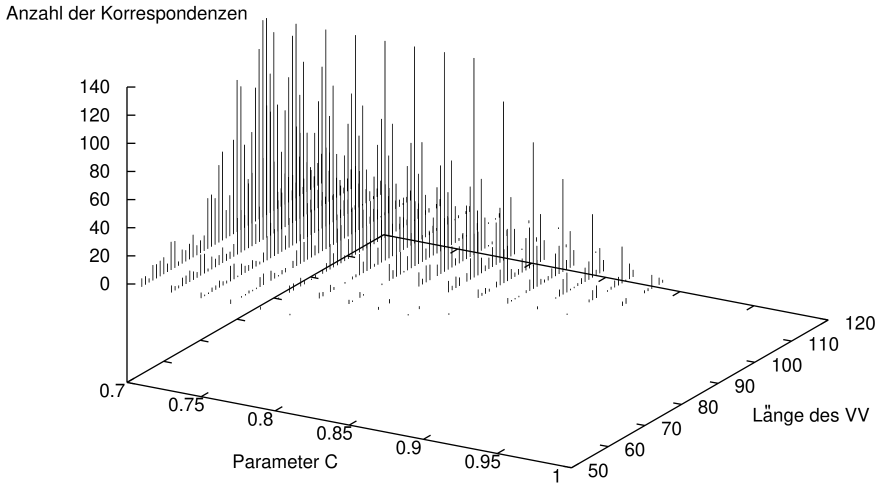

Abbildung 3.14: Histogramm der Verschiebevektorlängen (VV) der mit Gleichung (3.11) aus den Testbildern der Abbildung 3.13 ermittelten Korrespondenzen.

eines Weitwinkelobjektivs (Fischaugeneffekt) auftreten. Es ist ein Arbeitsbereich im Roboterhandkoordinatensystem festzulegen, innerhalb dem sich Hindernisse und der Zielpunkt befinden können. Durch die Definition dieses Arbeitsbereiches im Roboterhandkoordinatensystem befinden sich die Hindernisse und das Ziel relativ zur Roboterhand und der in der Nähe montierten Kamera. Aus diesem Bereich ergibt sich die minimale sowie die maximale Translationslänge einer Grauwertstruktur im Bildpaar, d.h. es existiert für jeden Bildort $b_{x,y}$ ein Intervall $\left[b_{x,y}^{min}, b_{x,y}^{max}\right]$, innerhalb dessen die Länge einer Translation liegen muß, wenn die entsprechende dreidimensionale Struktur im Arbeitsbereich liegt. Dieses Intervall kann automatisch erlernt werden. Dazu wird ein aus rechtwinklig im Abstand von 2 cm angeordneten Punkten bestehendes Muster erstellt. Zu einer relativ zu diesem Muster auszuführenden, durch den Arbeitsbereich festgelegten Bewegung der Kamera, welche auf das Muster gerichtet ist, werden Bildpaare des Musters angefertigt. Ein Bildpaar wird dabei mit einem Abstand des Musters von der Roboterhand (und der in der Nähe montierten Kamera) gewonnen, der dem Abstand der Roboterhand zum Beginn des oben festgelegten Arbeitsbereichs entspricht (siehe Abbildung 3.15). Das zweite Bildpaar wird bezüglich des Abstandes der Roboterhand zum Ende des Arbeitsbereiches angefertigt. Das Bildpaar (Abbildung 3.16) definiert durch Zuordnung der detektierten Punkte ein Vektorfeld, das den durch die ausgeführte Bewegung bedingten optischen Fluß in Abhängigkeit vom Bildort (Abbildung 3.17) repräsentiert. Damit existieren zwei Vektorfelder V_{Beginn} und V_{Ende}, die für einen Bildort $b_{x,y}$ mit den zwei Längenwerten des optischen Flusses,

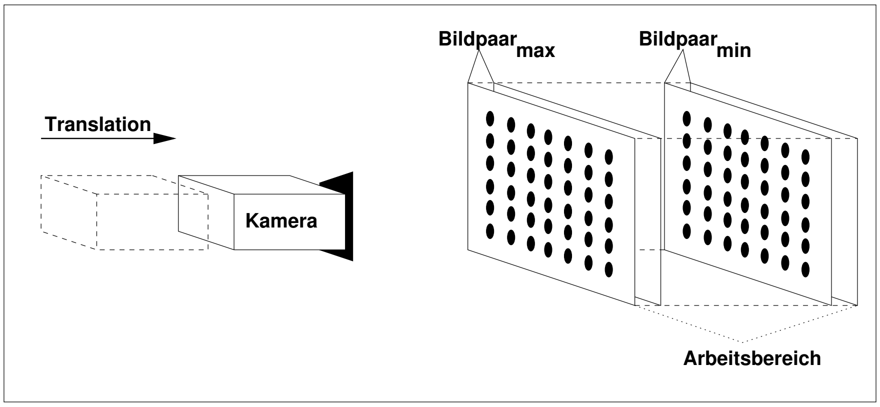

Abbildung 3.15: Bildpaargenerierung zur Messung des maximalen (Bildpaar$_{min}$) optischen Flusses am Beginn des Arbeitsbereiches und des minimalen (Bildpaar$_{max}$) optischen Flusses am Ende des Arbeitsbereiches.

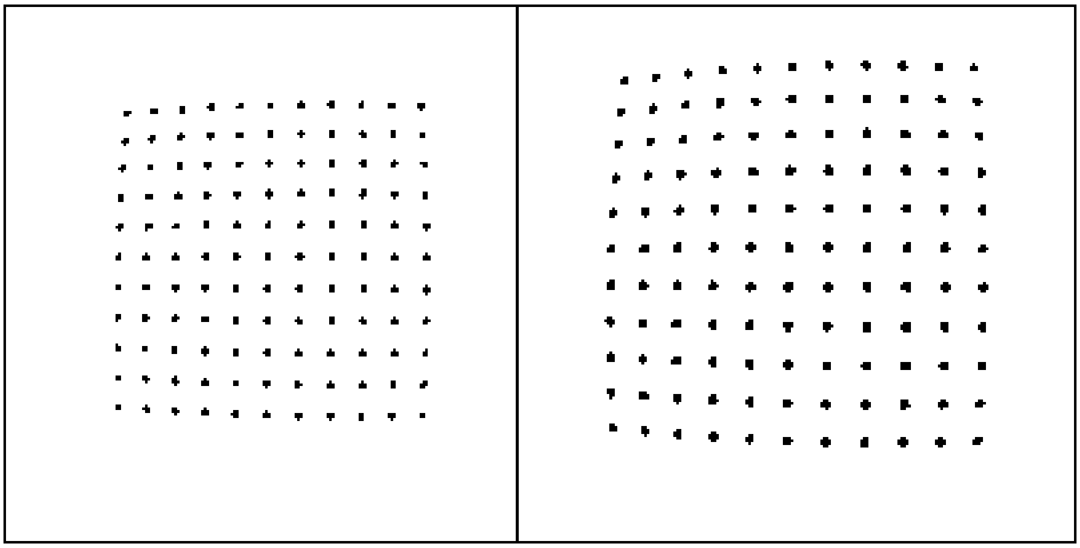

Abbildung 3.16: Binarisierte Aufnahme eines Kalibrierungsmusters vor und nach einer Translation in Richtung des Musters.

gemessen am Beginn und am Ende des Arbeitsbereiches, das geforderte Intervall

$$\left[b_{x,y}^{min}, b_{x,y}^{max}\right] \stackrel{Def}{=} \left[\|V_{Beginn}(x,y)\|,\ \|V_{Ende}(x,y)\|\right] \tag{3.12}$$

liefern. Damit können alle Strukturpaare von der Kreuzkorrelation ausgenommen werden, deren Translation nicht im ermittelten, positionsabhängigen Intervall liegt. Dieser Fall

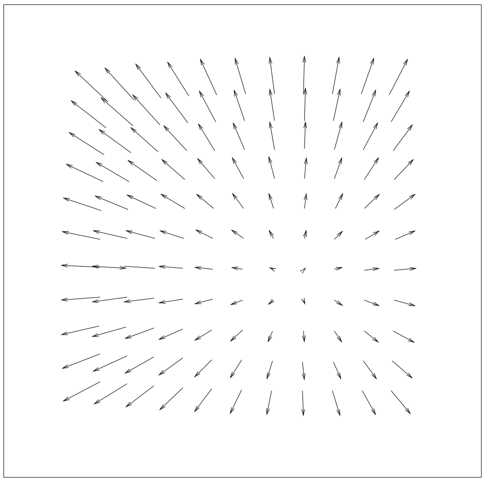

Abbildung 3.17: Der durch die Bilder der Abbildung 3.16 gegebene optische Fluß definiert Richtung und Länge einer Strukturverschiebung als Maß für die Korrektheit

tritt entweder bei Strukturpaaren auf, die nicht dieselbe verschobene Struktur repräsentieren oder bei Strukturpaaren, die zwar korrekt zugeordnet sind, jedoch außerhalb des Arbeitsbereiches positioniert sind (z.B. dem Hintergrund zugehörig).

Ein ähnliches Prinzip liegt der zweiten Möglichkeit zugrunde, eine Vorauswahl der möglichen Strukturpaare zu treffen. Mit den angefertigten Bildpaaren und den korrespondierenden Vektorfeldern liegt eine „Sollrichtung" des optischen Flusses hinsichtlich der Bewegung, die zwischen den Bildaufnahmen liegt, vor. Dabei ist es wegen der Unabhängigkeit der Richtung des optischen Flusses von der Entfernung der den Fluß erzeugenden Objekte grundsätzlich unerheblich, welches Vektorfeld V ausgewählt wird. Allerdings weist das Bildpaar, welches mit der größten Entfernung von der Kamera aufgenommen wurde, die meisten Punkte des Musters auf, so daß das entsprechende Vektorfeld die genaueren Aussagen hinsichtlich der Sollrichtung bietet. Das durch das Strukturpaar (p_i, q_j) gegebene Vektor $q_j - p_i$ legt durch die Lage der Bildposition q_j relativ zu p_i eindeutig einen Winkel fest, der mit $Winkel(q_j - p_i))$ bezeichnet wird. Dieser Winkel ist gerade derjenige, der in der Darstellung von q_j in Polarkoordinaten bezüglich p_i vorkommt. Die Abweichung dieses Winkels vom „Sollfluß" ist damit ein weiteres Maß für die korrekte Vorauswahl eines Paares (p_i, q_j). Die in Gleichung 3.11 gegebene Definition ändert sich wie folgt:

$$Partner'(p_i) \stackrel{Def}{=} \begin{cases} q_k & \text{falls} \quad \gamma_{i,k} = \max_j \{\gamma_{i,j} \mid i \in \{1,\ldots,N\}, j \in \{1,\ldots,M\}\} \\ & \wedge \quad \gamma_{i,k} > C \\ & \wedge \quad \|V_{Beginn}(x,y)\| \leq \|p_i - q_j\| \leq \|V_{Ende}(x,y)\| \\ & \wedge \quad |Winkel(p_i - q_j) - V(p_i)| < A \\ nil & \text{sonst} \end{cases} \tag{3.13}$$

wobei $A \in [0, 360)$ die Schranke für die zulässige Abweichung des für ein Strukturpaar gemessenen von dem durch den optischen Fluß gegebenen Winkel ist. Wir verwenden einen Wert von $A = 4$.

3.2.2 Parallele Ausführung der Strukturzuordnung

Die Parallelisierung der Strukturzuordnung findet nicht auf der Ebene der Kreuzkorrelation statt, sondern auf der Ebene der Listenverarbeitung. Jedem Parallelkonstrukt (Prozeß, Thread o.a.) wird dabei ein Element p_i aus der Liste L_1 zugewiesen (siehe Abbildung 3.18). Für dieses Element bestimmt das Parallelkonstrukt aus allen Elementen q_j der Liste L_2 das gegebenenfalls zum Element p_i zugehörige Element nach dem in dem Kapitel 3.2.1 vorgestellten Verfahren. Ist ein Paar zugeordnet bzw. eine Zuordnung ausgeschlossen, so wird dem frei werdenen Parallelkonstrukt das nächste, noch nicht in der Zuordnung befindliche Element $p_{i+offset}$ zugewiesen. Da bei diesem Verfahren keine Schreibzugriffe auf gleiche Speicherelemente zugreifen, ist keine Synchronisation der Parallelkonstrukte erforderlich. Damit skaliert das Verfahren bezüglich der Laufzeit sehr gut mit der Anzahl der eingesetzten Parallelkonstrukte.

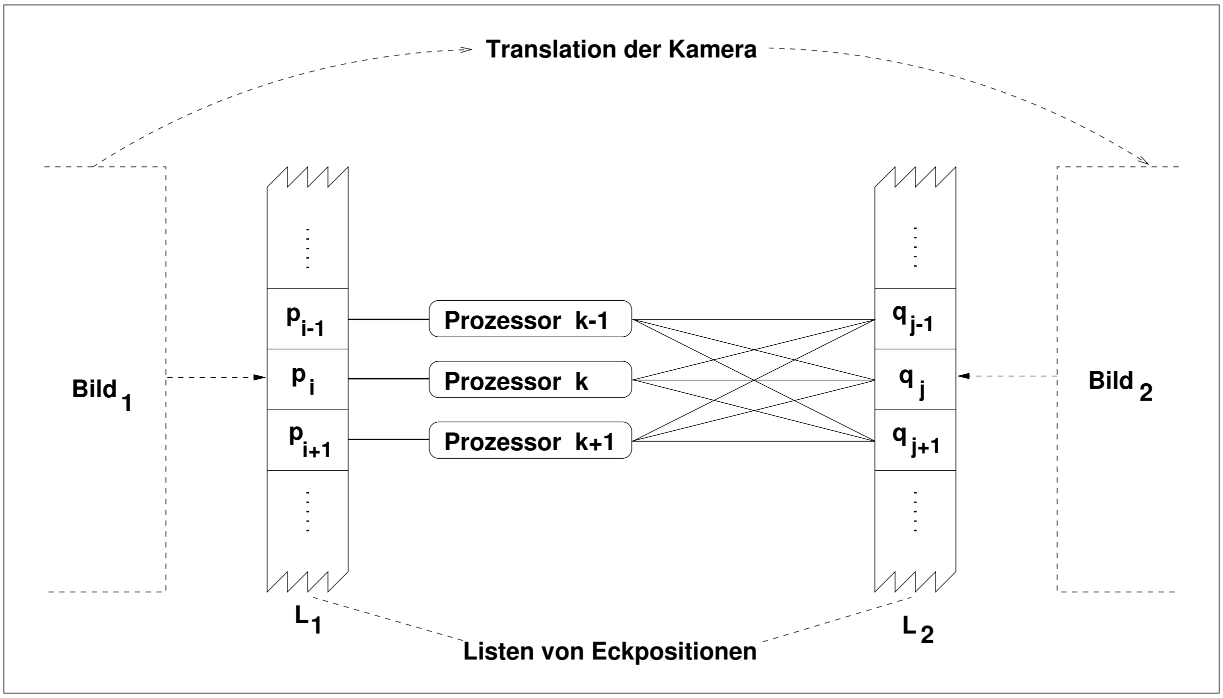

Abbildung 3.18: Parallelverarbeitung der Strukturzuordnung: Jedes Parallelkonstrukt (hier mit Prozessor bezeichnet) bestimmt zu einem Element p_i aus der Liste L_1 aus allen Elementen q_j der Liste L_2 das gegebenenfalls zugehörige Element

Kapitel 4

Neuronale Rekonstruktion von Besetzungspositionen

Die Zuordnung von Bildstrukturen liefert die Basis zur Rekonstruktion von 3D-Raumpositionen aus Bildpositionen. Diese Rekonstruktion ist die Umkehrabbildung der Kameraprojektion und kann z.B. geometrisch auf Basis der sogenannten Kamerakalibrierung vorgenommen werden[1]. Hierbei müssen die Kameraorientierung und -position im Weltkoordinatensystem sowie die Brennweite und das Pixel-Millimeter-Verhältnis bekannt sein, um die Koordinaten der Bildebene in das Weltkoordinatensystem zu überführen. Auch der Ursprung der Bildebene, das Seitenverhältnis (Aspect Ratio) und linsenbedingte Verzerrungen sind zu ermitteln [29]. Damit ist die korrekte Bestimmung einer Reihe von internen und externen Parametern ausschlaggebend für ein genaues Ergebnis.

Gelingt es dagegen, die internen und externen Kameraparameter implizit zu belassen, so vereinfacht sich das Problem der Rekonstruktion. Hierzu wird ein aus Rekonstruktionspunkten bestehendes Muster vor und nach einer Bewegung der Kamera aufgenommen. Durch Zuordnung der Bildpositionen der Rekonstruktionspunkte zu den korrespondierenden 3D-Raumpositionen werden interne und externe Kameraparameter mittels Neuronaler Netze in die zu erlernende Rekonstruktion integriert. Damit entfällt die Fehlerquelle der ungenauen Parameterbestimmung und reduziert die Kamerakalibrierung auf einen Lernprozeß von mehrdimensionalen Wertepaaren.

Wir verwenden für den Lernprozeß der Umkehrabbildung der Kameraprojektion sogenannte RBF-Netze. Dies sind Neuronale Netze auf der Grundlage radialer Basisfunktionen. Eine ausführliche Darstellung der RBF-Netze ist u.a. in [32] und [33] zu finden. Daher wird hier nur in einer Zusammenfassung auf RBF-Netze eingegangen, soweit es für den Kontext notwendig ist. Der Einführung in die RBF-Netze sowie deren parallelisiertem Training

[1] Da es sich um eine Projektion handelt, besteht die angesprochene Umkehrabbildung aus der Abbildung eines Bildpunktes auf einen Strahl von Punkten in der dreidimensionalen Szene. Erst aus zwei Bildern, zwischen denen eine Bewegung liegt, lassen sich Tiefeninformationen gewinnen.

folgen Betrachtungen zur Laufzeit der neuronalen Rekonstruktion, die mit Einführung des Netzsplittings zur Laufzeitverminderung führen.

4.1 RBF-Netze

Für eine gegebene Trainingsmenge von N Paaren von Vektoren

$$\{(\boldsymbol{x_i}, \boldsymbol{t_i}) \mid \boldsymbol{x_i} \in \mathbb{R}^I, \boldsymbol{t_i} \in \mathbb{R}^O, i \in \{1, \cdots, N\}\} \tag{4.1}$$

ist die Funktion $f : \mathbb{R}^I \rightarrow \mathbb{R}^O$ mit $f(\boldsymbol{x_i}) = \boldsymbol{t_i}$ für alle $i \in \{1, \cdots, N\}$ unbekannt. Die Dimension des Definitionsbereiches ist mit I gegeben, die des Wertebereiches sei mit O bezeichnet. Ein RBF-Netz hat die Aufgabe, die Funktion f zu interpolieren oder zu approximieren. Diese Aufgabe wird durch eine Linearkombination von radialen Basisfunktionen (RBF) ϕ_j gelöst, die folgende Gestalt[2] aufweisen:

$$\phi_j(\boldsymbol{x}) = \exp\left(-\frac{\|\boldsymbol{x} - \boldsymbol{\mu}_j\|^2}{2\sigma_j^2}\right) \tag{4.2}$$

Dabei gilt: $\sigma \in \mathbb{R}$, $\boldsymbol{x}, \boldsymbol{\mu}_j \in \mathbb{R}^I$ und $j \in \{1, \cdots, M\}$. σ ist ein Parameter, der die „Weichheit“ der interpolierenden oder approximierenden Funktion steuert. μ_j stellt einen im Eingaberaum positionierten Vektor dar, der die Rolle eines *Hidden Unit* übernimmt. Ist die Anzahl M der Hidden Units gleich der Anzahl der Trainingsdaten N, so wird die Funktion f interpoliert, anderenfalls liegt eine Approximation vor. Der Abstand eines Wertes $\boldsymbol{x}$ zu $\boldsymbol{\mu}_j$ ist ausschlaggebend für den Anteil der radialen Basisfunktion ϕ_j am Gesamtergebnis:

$$y_k(x) = \sum_{j=1}^{M} w_{kj}\phi_j(\boldsymbol{x})$$

für das k-te Element des Vektors $\boldsymbol{y}$ bzw.

$$\begin{pmatrix} y_1(\boldsymbol{x}) & \cdots & y_O(\boldsymbol{x}) \end{pmatrix} = \begin{pmatrix} \phi_1(\boldsymbol{x}) & \cdots & \phi_M(\boldsymbol{x}) \end{pmatrix} \begin{pmatrix} w_{11} & \cdots & w_{O1} \\ \vdots & \ddots & \vdots \\ w_{1M} & \cdots & w_{OM} \end{pmatrix} \tag{4.3}$$

[2]Es sind außer der angegebenen Funktion auch andere einsetzbar, siehe dazu [32].

mit $W^{\mathrm{T}} = (w_{ij})$. Die Elemente der hier aus Gründen der Herleitung transponierten Matrix W^{T} stellen die Gewichte der Linearkombination dar und bestimmen zusammen mit den durch die $\boldsymbol{\mu}_j$ repräsentierten Hidden Units und dem Parameter σ die Gestalt eines RBF-Netzes (Abbildung 4.1). Es gibt verschiedene Konzepte, diese Parameter zu bestimmen. In unserem Fall werden die Hidden Units $\boldsymbol{\mu}_j$ genau an den Stellen der Trainingsdaten $\boldsymbol{x}_i, i \in \{1, \cdots, M\}$ positioniert, was $M = N$ und damit eine Interpolation impliziert. Der Wert für σ ist gleich dem 1.25-fachen (maximalen) Abstand der Trainingsdaten[3]. Für die noch fehlende Berechnung der Gewichte des Netzes setzt man die Wertepaare der Trainingsmenge ein, so daß gilt:

$$\begin{pmatrix} \boldsymbol{t}_{11} & \cdots & \boldsymbol{t}_{1O} \\ \vdots & \ddots & \vdots \\ \boldsymbol{t}_{N1} & \cdots & \boldsymbol{t}_{NO} \end{pmatrix} = \begin{pmatrix} \phi_1(\boldsymbol{x}_1) & \cdots & \phi_M(\boldsymbol{x}_1) \\ \vdots & \ddots & \vdots \\ \phi_1(\boldsymbol{x}_N) & \cdots & \phi_M(\boldsymbol{x}_N) \end{pmatrix} \begin{pmatrix} w_{11} & \cdots & w_{O1} \\ \vdots & \ddots & \vdots \\ w_{1M} & \cdots & w_{OM} \end{pmatrix} \tag{4.4}$$

wobei der Zeilenvektor $(t_{k1} \cdots t_{kO})$ die Elemente des Trainingsvektors $\boldsymbol{t}_k$ darstellt. In Matrixschreibweise erhält man:

$$T = \Phi W^{\mathrm{T}} \tag{4.5}$$

und schließlich

$$\Phi^{\mathrm{T}} T = \Phi^{\mathrm{T}} \Phi W^{\mathrm{T}} \tag{4.6}$$

In [32] wird gezeigt, daß mit Gleichung (4.6) eine Minimierung der Summe der Quadrate der Fehler durchgeführt wird. Durch Inversion der Matrix $\Phi^{\mathrm{T}}\Phi$ löst man die Gleichung (4.6) durch die Pseudoinverse $\Phi^{\dagger}$ und erhält die gesuchten Gewichte des RBF-Netzes:

$$\Phi^{\dagger} T = W^{\mathrm{T}} \tag{4.7}$$

Damit ist der Trainingsvorgang des Netzes beendet. Um in der Praxis dem Problem einer möglicherweise schlechten Konditionierung der Matrix Φ zu begegnen, ist es sinnvoll, die Gleichung (4.6) durch Singulärwertzerlegung zu lösen [34].

[3]Dieser Wert resultiert aus einer Fehlerminimierung, die Dr. Josef Pauli am Lehrstuhl für Kognitive Systeme der Christian-Albrechts-Universität zu Kiel durchgeführt hat.

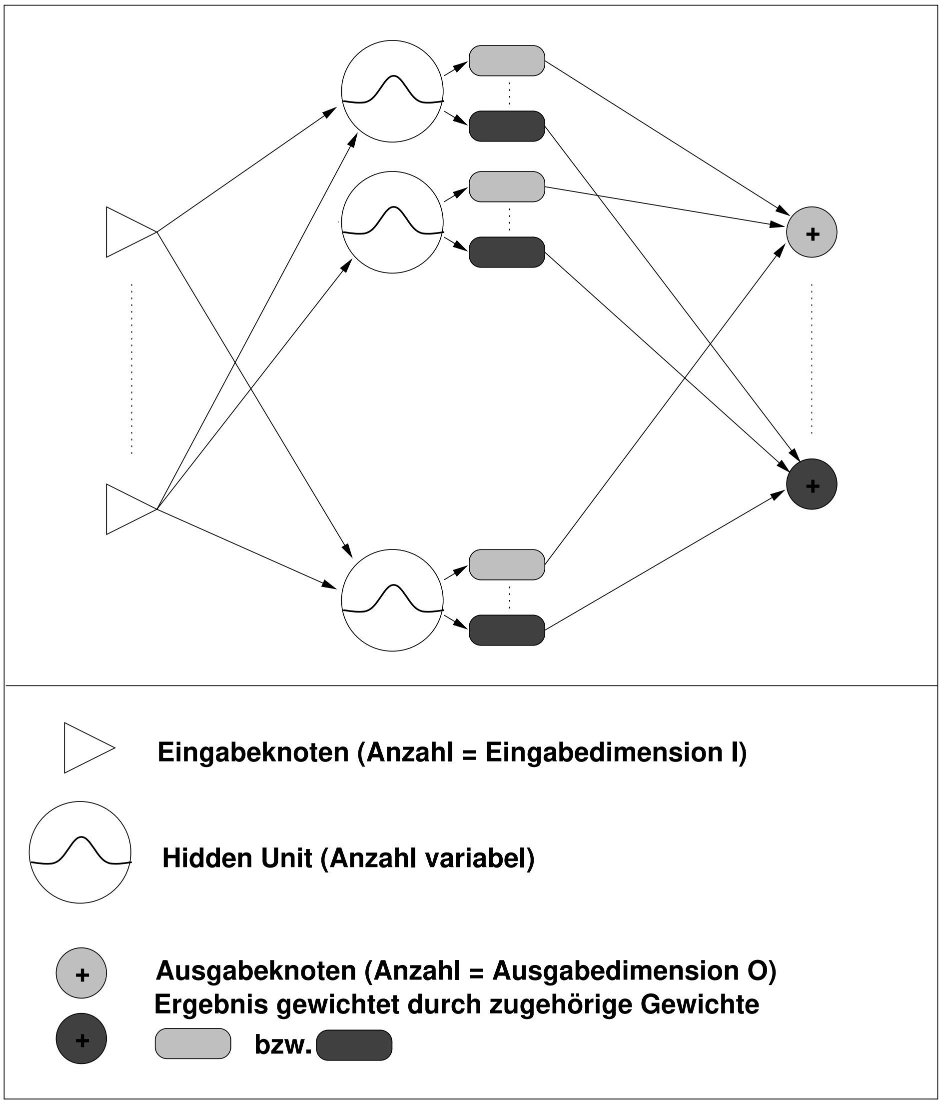

Abbildung 4.1: Struktur eines RBF-Netzes

4.1.1 Paralleles Training eines RBF-Netzes

Die Lösung der Gleichung (4.6) wird aus Gründen der Stabilität mit einer Singulärwertzerlegung vorgenommen. Dabei wird jeder Spaltenvektor von W^{T} einzeln und unabhängig von den anderen bestimmt, so daß diese Unabhängigkeit zur parallelen Implementation genutzt werden kann.

Zunächst sei $A := \Phi^{\mathrm{T}}\Phi$, $\Omega = W^{\mathrm{T}}$ und $P := \Phi^{\mathrm{T}}T$. Mit Ω_i sei der i-te Spaltenvektor von Ω und mit P_j der j-te Spaltenvektor von P bezeichnet. Die Gleichung (4.6) stellt sich nun wie folgt dar:

$$P = A\Omega \tag{4.8}$$

Dies ist äquivalent zum Gleichungssystem:

$$P_i = A\Omega_i\ , \quad i \in \{1, \cdots, O\} \tag{4.9}$$

Die gesuchten Gewichte Ω_i lassen sich nun nebenläufig nach Gleichung (4.9) berechnen.

4.2 Neuronale Rekonstruktion

Ziel der Rekonstruktion von 3D-Strukturen einer Szene mit Hilfe von Neuronalen Netzen ist die Interpolation der Umkehrabbildung der Kameraprojektion, ohne die explizite Bestimmung von in- und externen Kameraparametern vorauszusetzen. Dadurch fallen nicht nur die fehlerbehafteten, expliziten Parameterbestimmungen weg, es werden außerdem die kamerabedingten Besonderheiten wie Linsenverzerrungen oder nicht exakte Positionen der CCD-Chips (Verdrehungen und nicht zentrierte Position) automatisch in die Interpolation integriert. Nachdem die benötigten Neuronalen Netze zur Verfügung stehen, wird ein erstes Netz entworfen, das die Rekonstruktion implementiert.

4.2.1 Die Rekonstruktion mit einem RBF-Netz

Die neuronale Rekonstruktion der Szene implementiert die gesuchte Funktion f_{Rek}, die die Umkehrfunktion der Verkettung der Funktionen Projektion und Verzerrung durch die Kameralinse ist. Die Funktion f_{Rek} wird ebenfalls durch die durch die ungenaue Plazierung des bildaufnehmenden Elementes (CCD-Chip) bedingte Translation und Rotation der Bildebene bestimmt. Dabei stehen einem Punkt der dreidimensionalen Szene zwei Projektionen in den Bildebenen der jeweiligen Kamerapositionen, gemessen in den Bildpunktkoordinaten des Bildspeichers, gegenüber[4]:

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = f_{Rek}\left(\begin{pmatrix} x_1 \\ y_1 \end{pmatrix}, \begin{pmatrix} x_2 \\ y_2 \end{pmatrix}\right) \tag{4.10}$$

[4]Äquivalent zu f_{Rek} ist eine Funktion, die statt der beiden Bildpunktkoordinaten $(x_1, y_1)^{\mathrm{T}}$ und $(x_2, y_2)^{\mathrm{T}}$ eine Bildpunktkoordinate und die Translation zur zweiten berücksichtigt.

für die Bildpunktkoordinatenpaare $(x_1, y_1)^{\mathrm{T}}$ und $(x_2, y_2)^{\mathrm{T}}$ sowie den Szenenpunkt $(X, Y, Z)^{\mathrm{T}}$. Hierbei handelt es sich um Bildfolgenstereo [30], da zwischen der Aufnahme des zu $(x_1, y_1)^{\mathrm{T}}$ gehörenden Bildes sowie der Aufnahme des zu $(x_2, y_2)^{\mathrm{T}}$ gehörenden Bildes eine Kamerabewegung $\boldsymbol{m}$ ausgeführt wird. Ein Trainingsdatum besteht damit aus dem Quadrupel $\left(\boldsymbol{m}, (x_1, y_1)^{\mathrm{T}}, (x_2, y_2)^{\mathrm{T}}, (X, Y, Z)^{\mathrm{T}}\right)$, wobei $\left(\boldsymbol{m}, (x_1, y_1)^{\mathrm{T}}, (x_2, y_2)^{\mathrm{T}}\right)$ das Eingabedatum und $(X, Y, Z)^{\mathrm{T}}$ das Ausgabedatum des zu trainierenden Netzes ist. Hier wird die höhere Komplexität des Bildfolgenstereos im Vergleich zur Stereorekonstruktion (mit gleichzeitiger Bildaufnahme und fester Baseline) deutlich. Die in Orientierung und Länge variable Baseline, die durch die Kamerabewegung definiert ist, erzeugt zwei zusätzliche Dimensionen des Problems. Die Trainingsdatenmenge besteht nun aus einem Quadrupel für jeden möglichen Punkt der dreidimensionalen Szene sowie jeder möglichen Kamerabewegung. Damit wird eine Komplexität des Problems erreicht, die für eine praktische Anwendung ungeeignet ist (Abbildung 4.2). Für eine dreidimensionale Szene der Größe $1m \times 1m \times 1m$, diskretisiert in Schritten von $5cm$, ein bezüglich Polarkoordinaten auf Winkel von $5°$ diskretisiertes Bewegungsrichtungsgitter und drei verschiedenen Bewegungslängen ist zum Beispiel die Zahl der einzusetzenden Trainingstupel mit $21^3 \cdot (360/5)^2 \cdot 3 = 144.027.072$ gegeben[5]. Betrachtet man den Aufwand des Trainingsvorganges eines RBF-Netzes, so zeigt sich, daß die Anzahl der zu berechnenden Gewichte des Netzes linear abhängig von der Anzahl der verwendeten Netzknoten (Hidden Units) ist, die in unserem Fall stets gleich der Anzahl der Trainingsdaten ist. Für die Berechnungen der Matrix Φ steigt der Aufwand daher sogar quadratisch mit der Anzahl der Trainingspaare (Gleichung (4.4)). Dieses Skalierungsverhalten ist in Abbildung 4.3 dargestellt[6]. Bei vorgegebener Trainingszeit liegt damit eine Grenze hinsichtlich der Anzahl der zu trainierenden Knoten vor.

Es ist also notwendig, die Dimensionalität des Problems zu verringern. Ein erster Schritt ist das Training eines Netzes zu einer fest gewählten Translation, so daß für jede gewünschte Bewegung ein eigenes Netzwerk trainiert wird. Hierbei ist es nicht notwendig, einen Rotationsteil bei der Kamerabewegung zu berücksichtigen, da sich jede Bewegung B in eine Rotation R und eine Translation T zerlegen läßt:

$$B = R \circ T \tag{4.11}$$

Ist eine Bewegung gewünscht, die eine Rotation enthält, wird nach Gleichung (4.11) zuerst der Translationsteil der Bewegung ausgeführt, wobei vor und nach der Translation die Bildaufnahme vorgenommen wird. Anschließend wird der Rotationsteil der Bewegung ausgeführt. Auf diese Weise sind allgemeine Bewegungen der Kamera möglich, obwohl das Netztraining lediglich in Bezug auf Translationen erfolgen muß. In einer Aufnahme-Bewegungssequenz muß pro Bewegung eine zusätzliche Bildaufnahme ausgeführt werden

[5]Ein Clustering der Netzknoten hat durch die äquidistante Anordnung der Trainingsdaten keine geringere Anzahl von Knoten zur Folge.

[6]Die angegebenen Zeiten beziehen sich auf einen SUN ULTRASPARC Prozessor mit 167 Mhz, die Charakteristik der Kurve ist jedoch rechnerunabhängig.

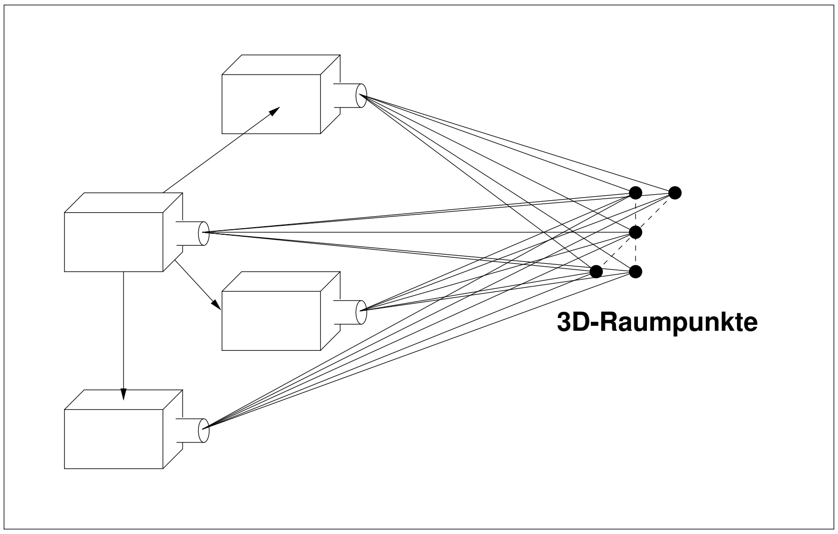

Abbildung 4.2: Jede mögliche Kamerabewegung und die Projektionen jedes möglichen Bildpunktes ergeben eine hohe Komplexität des Problems

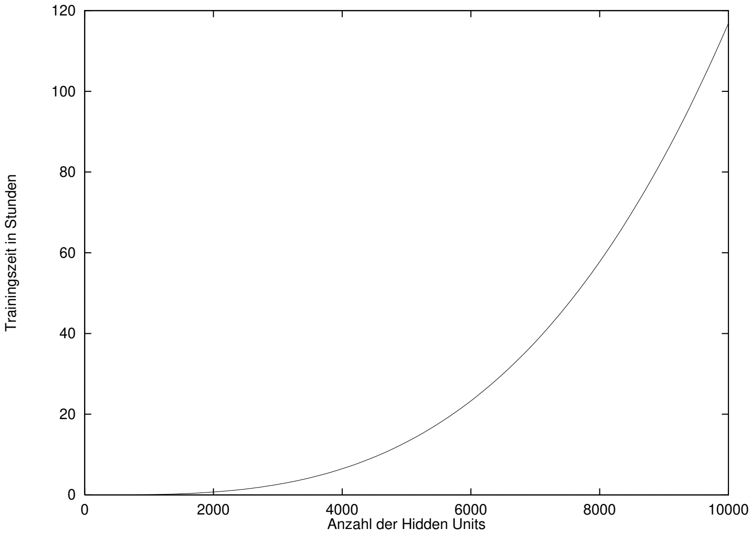

Abbildung 4.3: Trainingsaufwand von RBF-Netzen mit Eingangsdimension 4, Ausgangsdimension 3 und gleicher Anzahl von Trainingsbeispielen und Hidden Units

(Abbildung 4.4), was in der Praxis jedoch keinen nennenswerten zusätzlichen Zeitaufwand bedeutet.

Die Berechnung des optischen Flusses auf der alleinigen Basis einer Translation ist dabei biologisch motivierbar. Der jagende Gepard z.B. verwendet ausschließlich eine translatorische Bewegung seines Kopfes [35]. Die Anzahl der benötigten Trainingspaare nach obigem

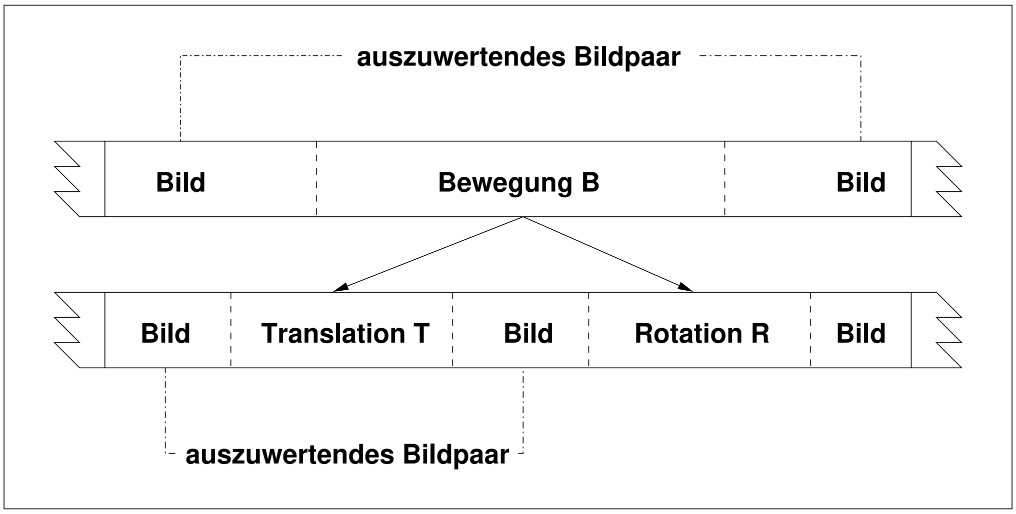

Abbildung 4.4: Ein zusätzliches Bild nach der reinen Translation ermöglicht die Auswertung der rotationsfrei erzeugten Bilder

Beispiel liegt nun für jede zu trainierende Translation bei $21^3 = 9261$. Mit diesen Schritten konnte zwar die Komplexität des Problems verkleinert werden, ohne die Leistungsfähigkeit des Ansatzes zu mindern, die Trainingszeit für ein Netz mit einer Anzahl von fast 10^4 Knoten liegt jedoch immer noch im Bereich von Tagen. Um eine weitere Beschleunigung des Trainings zu erreichen, ist es sinnvoll, die Größe der Netze zu betrachten.

4.2.2 Splitting des RBF-Netzes

Um die Trainingszeiten möglichst gering zu halten, muß die Anzahl der Trainingsdaten des bisherigen Neuronalen Netzes reduziert werden, ohne jedoch die Leistung des Lösungsansatzes zu mindern. Dazu führen wir eine Aufteilung des bisherigen Netzes in Teilnetze ein, was eine Zerlegung der bisher durch das Netz interpolierten Funktion f_{Rek} in Teilfunktionen erfordert[7]. Bei der Zerlegung eines Netzes erkauft man sich die geringere Trainingszeit mit der durch die Spaltung fehlenden Generalisierung zwischen den Netzteilen, die, z.B. durch eine Interpolation, ausgeglichen werden muß. Bei hinreichend feiner Zerlegung eines Netzes ist dieser Nachteil jedoch klein zu halten. In der in Gleichung (4.10) dargestellten Abbildung f_{Rek}, die das bisherige einzelne RBF-Netz interpoliert, werden beide Projektionen eines Punktes der Szene gleichzeitig auf die drei Weltkoordinaten des Szenenraumes abgebildet. Zwar sind beide Bildpunkte $(x_1, y_1)^T$ und $(x_2, y_2)^T$ für die Rekonstruktion

[7]Es zeigte sich, daß die mit *Hierarchical Mixture of Experts* (HME) bezeichneten Netzsysteme eine ähnliche Strategie des Netzsplittings (*divide-and-conquer*) verfolgen. Während die Gewichtung der Teilnetze bei HME-Netzen durch *Gates* erfolgt, deren Gewichte gelernt werden, verwenden wir verschiedene Interpolationen, die die Gewichtung der zu einem Eingabedatum benachbarten Teilnetze realisieren.

des zugehörigen Szenenpunktes $(X, Y, Z)^{\mathrm{T}}$ im Weltkoordinatensystem notwendig, aus den in den Bildpunktkoordinaten enthaltenen Informationen lassen sich jedoch sequentiell zuerst die Tiefe Z des Szenenpunktes und unter Verwendung dieser Tiefe die Koordinaten X und Y berechnen. Wir leiten dieses Verfahren von der Stereorekonstruktion mit einer Stereokameraanordnung, deren optische Achsen parallel sind, ab [29]. Die Tiefe Z eines Punktes in Weltkoordinaten $(X, Y, Z)^{\mathrm{T}}$ ist indirekt proportional zur Disparität $x_1 - x_2$ der Projektionen $(x_1, y_1)^{\mathrm{T}}$ und $(x_2, y_2)^{\mathrm{T}}$:

$$Z = \frac{b \cdot l}{x_1 - x_2} \tag{4.12}$$

wobei b die Brennweite der Objektive und l die Baseline des Stereokamerasystems ist. Ist die Tiefe des gesuchten Punktes bekannt, können die restlichen Koordinaten nach den Gleichungen der Zentralprojektion berechnet werden:

$$\begin{pmatrix} X \\ Y \end{pmatrix} = \frac{Z}{b} \begin{pmatrix} x_2 \\ y_2 \end{pmatrix} \tag{4.13}$$

Dieser Zweiteilung der Rekonstruktion folgend gestalten wir die Aufteilung des Netzes. Zwar liegen in unserem Verfahren die die Orientierungen der optischen Achse der Kamera ebenfalls parallel zueinander, da eine Rotation nach der Aufnahme eines Bildpaares vorgenommen wird. Effekte, die durch den Versatz der Kameraposition entlang der optischen Achse, die Bildverzerrungen der Linse und die ungenaue Platzierung des CCD-Chips hervorgerufenen werden, sind nicht in dem idealisierten Modell der Gleichungen (4.12) und (4.13) einbezogen, sondern werden erst durch die Neuronalen Netze integriert werden. Ein weiterer Unterschied zum Stereokamerasystem ist die Möglichkeit unseres monokularen Systems, die Translation auch in nicht-orthogonaler Richtung zu der optischen Achse der Kamera auszuführen. Daher ist nicht nur zwischen den Bildkoordinaten x_1 und x_2 ein Versatz vorhanden, sondern möglicherweise auch zwischen y_1 und y_2. Deshalb tritt an die in Gleichung (4.12) verwendete Disparität $x_1 - x_2$ die Länge des optischen Flusses $\|(x_1 - x_2, y_1 - y_2)^{\mathrm{T}}\|$, um das die Tiefe berechnende Netzwerk zu trainieren. Dieses Netzwerk implementiert damit die Funktion f^Z_{REK}:

$$Z = f^Z_{Rek}\left(\left\| \begin{pmatrix} x_1 - x_2 \\ y_1 - y_2 \end{pmatrix} \right\| \right) \tag{4.14}$$

Die Funktion (4.14) ist jedoch nicht für jeden Bildort gültig, da die Bildverzerrung des Weitwinkelobjektivs einen Einfluß auf den optischen Fluß zur Folge hat, die abhängig vom

Bildort ist. Um den optischen Fluß zu messen, wird die am Roboterarm montierte Kamera bezüglich eines in Abbildung 4.5 dargestellten Punktmusters verschoben. Die Punkte können über einen besonders markierten Punkt indiziert werden, so daß die Berechnung des optischen Flusses durch einen Positionsvergleich von Punkten gleichen Indizes möglich ist. In Abbildung 4.6 ist die Länge des optischen Flusses für eine Translation in seitlicher Richtung und in Richtung der optischen Achse dargestellt. Wäre die Aufnahme verzerrungsfrei, lägen die Werte der Länge des optischen Flusses für den Fall der seitlichen Translation auf einer Ebene. Auch die Orientierung der Kamera zum Punktmuster, welches auch zum Training der Netze verwendet werden wird, beeinflußt den optischen Fluß. Aus diesen Gründen ist nicht dieselbe Abbildung f^{Z}_{Rek} für alle Orte im Bild verwendbar, sondern es muß für jeden Bildort (bzw. einer äquidistant angeordneten Auswahl von Bildorten) eine eigene Abbildung $f^{xy->Z}_{Rek}$ gelernt werden:

$$Z = f^{xy->Z}_{Rek}\left(\left\|\begin{pmatrix} x_1 - x_2 \\ y_1 - y_2 \end{pmatrix}\right\|\right) \tag{4.15}$$

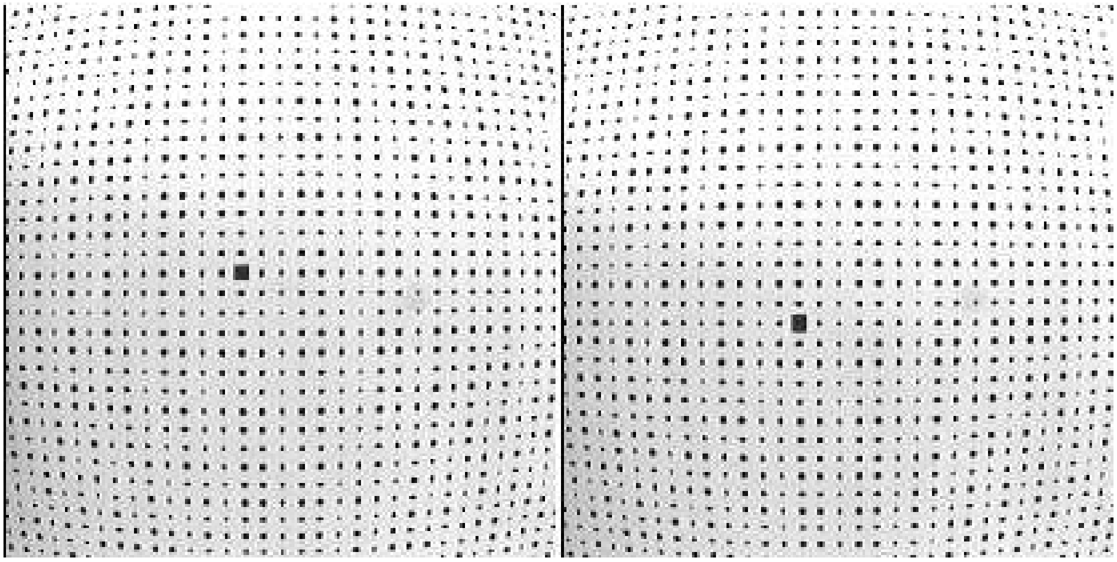

Abbildung 4.5: Bestimmung des optischen Flusses: Die Punkte des Rekonstruktionsmusters können durch den markierten Punkt indiziert werden, so daß durch Positionsvergleich von Punkten gleichen Indizes der optische Fluß für die Orte der Punkte berechnet werden kann.

Die berechnete Entfernung Z wird nun analog zur Gleichung (4.13) zur Grundlage der Bestimmung der fehlenden Weltkoordinaten X und Y. Hierzu werden die Koordinaten des Bildpunktes (x_2, y_2) benötigt, die der nach der Translation angefertigen Aufnahme

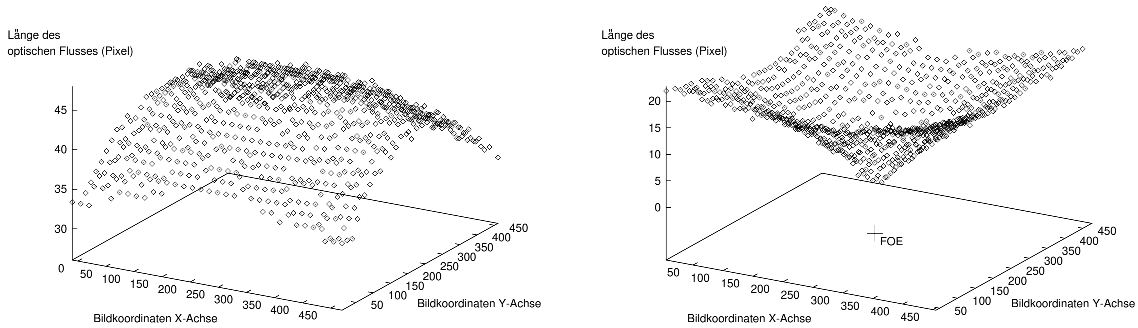

Abbildung 4.6: Die Länge der Vektoren des optischen Flusses des Punktmusters aus Abbildung 4.5. Links: Gemessen nach einer seitlichen Translation. Rechts: Gemessen nach einer Translation in Richtung der optischen Achse.

entstammen. Außerdem sind die Weltkoordinaten X und Y, analog zur Gleichung (4.13), von der ermittelten Tiefe abhängig, so daß die Funktion lautet:

$$\begin{pmatrix} X \\ Y \end{pmatrix} = f_{Rek}^{XY}\left(Z, \begin{pmatrix} x_2 \\ y_2 \end{pmatrix}\right) \tag{4.16}$$

Allerdings weist ein Netzwerk, das die Funktion f_{Rek}^{XY} implementiert, nicht den angestrebten Vorteil der geringeren Knotenzahl als das ursprüngliche Netzwerk auf, da die Trainingsdaten unverändert dem gesamten Arbeitsbereich entnommen werden müssen. Für eine berechnete Tiefe Z existiert jedoch eine in Abhängigkeit von Z gewählte Funktion $f_{Rek}^{Z \to XY}$, deren Implementation als Netzwerk effektiver ist, da die benötigten Trainingsdaten lediglich aus der durch die Tiefe Z gegebenen, parallel zur X-Y-Ebene liegenden Ebene entnommen werden müssen, und nicht aus dem gesamten Arbeitsbereich:

$$\begin{pmatrix} X \\ Y \end{pmatrix} = f_{Rek}^{Z \to XY}\left(\begin{pmatrix} x_2 \\ y_2 \end{pmatrix}\right) \tag{4.17}$$

für die Bildpunktkoordinatenpaare $(x_2, y_2)^{\mathrm{T}}$ sowie den Szenenpunkt $(X, Y, Z)^{\mathrm{T}}$. Zwar erfolgt die Beschreibung der resultierenden, zweistufigen Berechnung von Weltkoordinaten erst in Kapitel 4.2.5. Da diese jedoch untrennbar mit der Netzstruktur in Verbindung steht, wird diese vorgreifend in Abbildung 4.7 dargestellt.

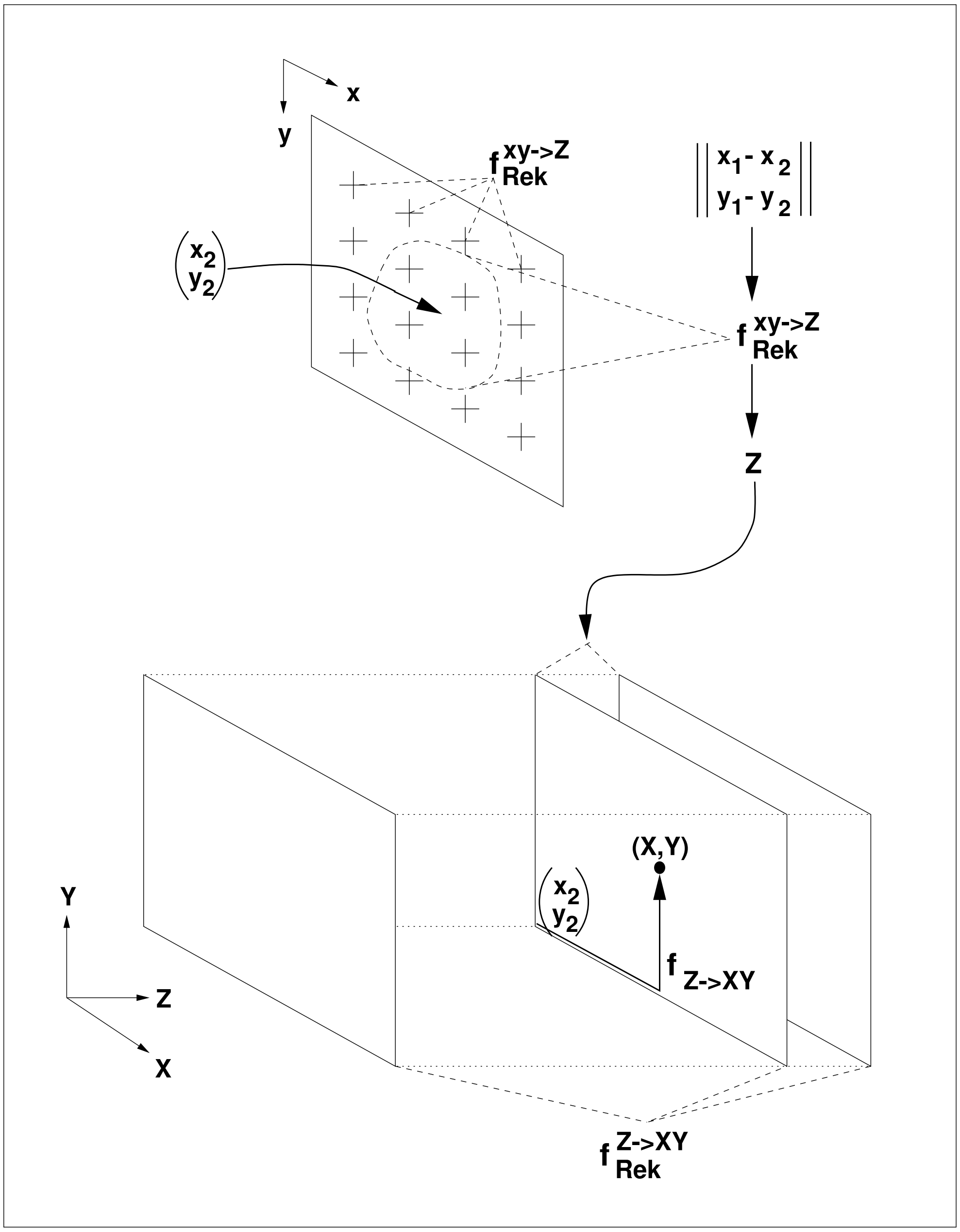

Abbildung 4.7: Die zweistufige Berechnung von Weltkoordinaten: Der Bildpunkt $(x_2, y_2)^{\mathrm{T}}$ ist Grundlage zur Auswahl eines oder mehrerer am nächsten liegenden Netzwerke $f_{Rek}^{xy \to Z}$. Mit diesen wird aus der Länge des optischen Flusses die Tiefe Z berechnet. Der berechnete Tiefenwert Z ist nun Grundlage zur Berechnung von X und Y aus den Antworten der Netze $f_{Rek}^{Z \to XY}$.

4.2.3 Training der einzelnen Netze

Nachdem das Splitting des Netzwerkes durch die Angabe der zu erlernenden Funktionen $f_{Rek}^{xy \to Z}$ und $f_{Rek}^{Z \to XY}$ motiviert wurde, sind nun die entsprechenden Trainingsdaten zu generieren. Dabei ist zur Bereitstellung der Trainingsdaten für die als Ergebnis der Netzzerlegung entstandenen Einzelnetze mehr Aufwand notwendig als zum Training des Einzelnetzes. Wie sich in Kapitel 4.2.4 zeigen wird, ist die Zeitersparnis größer als die durch den Mehraufwand zusätzlich benötigte Zeit.

Training der Netze zur Tiefenrekonstruktion

Wir betrachten zunächst das Training der mit Rz bezeichneten Netze, die die Funktion $f_{Rek}^{xy \to Z}$ aus Gleichung (4.15) implementieren. Die Aufteilung des Gesamtnetzes in Teilnetze ermöglicht es, die Knoten der Netze in Unterräumen des dreidimensionalen Szenenraumes anzuordnen. Für die Netze Rz bedeutet dies, daß die Knoten in einem eindimensionalen Unterraum liegen, da die Knotenposition durch die Länge des optichen Flusses bestimmt ist. Da diese Netze aus der Länge des optischen Flusses $\boldsymbol{o}$ an der ihnen zugeordneten Position (x, y) im Bild die zugehörige Tiefe Z berechnen, besteht ein Trainingsdatum aus dem Tupel $(\|\boldsymbol{o}\|, Z)$. Für die Berechnung des optischen Flusses ist ein Bildpaar notwendig, bei dem zwischen den Aufnahmen der Einzelbilder die Translation ausgeführt wurde, bezüglich der die Netze trainiert werden sollen. Dazu werden Bildpaare B_l durch Ausführung von Translationen der Kamera relativ zu dem in Abbildung 4.5 dargestellten Punktmuster erzeugt, wobei jedes Bildpaar aus den Bildern b_l^{pre} (vor der Translation aufgenommen) und b_l^{post} (nach der Translation aufgenommen) besteht. Jedes Bildpaar B_l wird mit der Entfernung $l \cdot d$ aufgenommen, so daß zwischen den Bildern b_l^{post} und b_{l+1}^{post} die Distanz d liegt (Abbildung 4.8).

Nun sei ein Gitter G_{sd} der Seitenlänge s mit Abstand d der Gitterpunkte über die das nach der Translation gewonnene Bild b_l^{post} gelegt:

$$G_{sd} = \{\, g_{ij} \mid i, j \in \{0, \cdots, s-1\},\ g_{ij} = (i \cdot d, j \cdot d)\,\} \tag{4.18}$$

Für jedes Bildpaar B_l existiert ein solches Gitter G_{sd}, so daß ein dreidimensionales Gitter G_{sdl} mit Gitterpunkten $\boldsymbol{g_{ijl}} = (x_{ijl}, y_{ijl}, Z_l)$ entsteht, das die Bildkoordinaten x_{ijl}, y_{ijl} in Relation mit dem Abstand $Z_l = l \cdot d$ der Roboterhand zu der Ebene des Punktmusters K_l setzt. Jedem i, j ist ein neuronales Netz Rz_{ij} zugeordnet, das eine vom Bildort $(\boldsymbol{g_{ijl}})$ abhängige Abbildung der Länge des optischen Flusses $(\|\boldsymbol{o_{ijl}}\|)$ auf die Entfernung Z_l erlernt. Dazu wird das Netz Rz_{ij} mit der Menge der Trainingspaare $(\|\boldsymbol{o_{ijl}}\|, Z_{ijl})$ für $l \in \{0, \ldots, L\}$ trainiert (L sei die Anzahl der eingesetzten Kalibrierungsebenen) (Abb. 4.8).

Es ist also der optische Fluß $\boldsymbol{o_{ijl}}$ am Gitterpunkt $\boldsymbol{g_{ijl}}$ zu bestimmen. Allerdings liefern die Bilder des Referenzmusters aufgrund von Verzerrung (Abbildung 4.5) und perspektivischer Verkleinerung Daten an (Bild-)Orten, die im allgemeinen nicht mit den Gitterpunkten $\boldsymbol{g_{ijl}}$ des Bildes b_l^{post} übereinstimmen. Die Daten an den äquidistanten Gitterpunkten

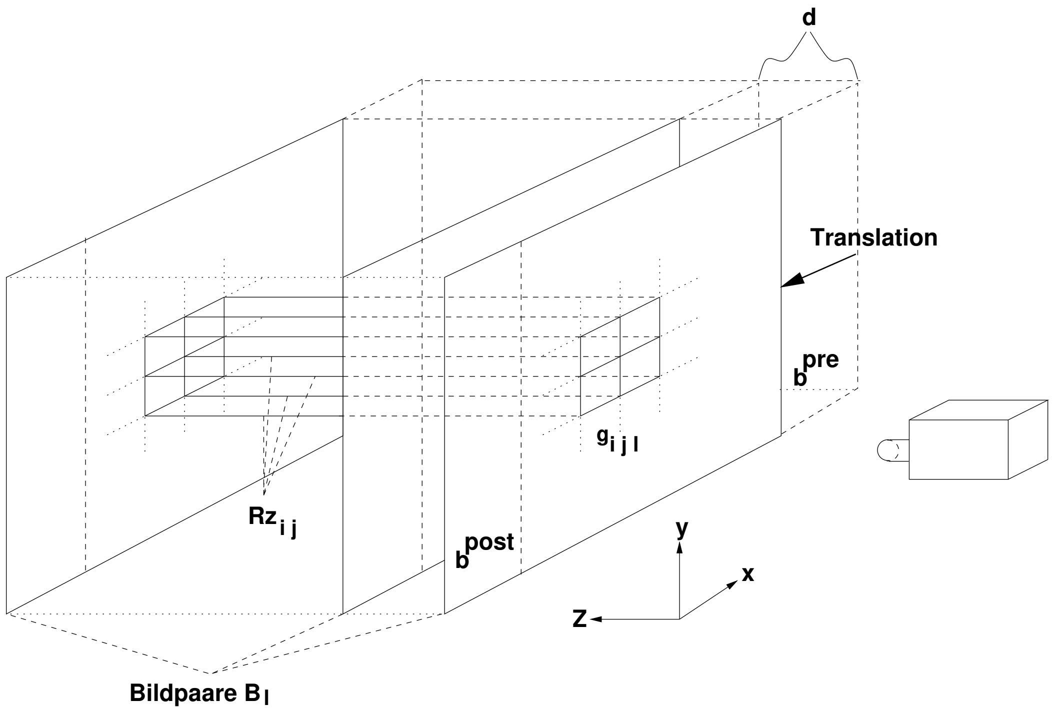

Abbildung 4.8: Anordnung der Netzwerke Rz_{ij} über der Bildfolge $(b_0^{post}, b_1^{post}, ...)$, die aus dem optischen Fluß, ermittelt aus einem Bildpaar $B_l = \left(b_l^{post}, b_l^{post}\right)$, die Tiefe berechnen

müssen daher aus den umliegenden Punkten des Muster, an denen der optische Fluß berechnet werden kann, gewonnen werden. Hierzu stehen einfachere Verfahren wie z.B. die Auswahl des zum Gitterpunkt nächsten Rekonstruktionspunktes oder das Ermitteln eines Zwischenwertes durch Triangulation zur Verfügung. Genauere Werte lassen sich jedoch durch Verfahren errechnen, die zu den durch die Bildorte der Rekonstruktionspunkte und den zugehörigen Längen der Flußvektoren gegebenen Stützstellen eine weiche Interpolationsfunktion liefern. Da es sich bei Funktionen der Länge des optischen Flusses typischerweise um glatte, differenzierbare Funktionen handelt (eine Ausnahme bildet der Focus of Expansion, wenn er im Bild liegt), sind Spline-Funktionen dazu geeignet, den optischen Fluß zu interpolieren (Abbildung 4.6). Zwar ist eine Interpolation des optischen Flusses auch mittels eines RBF-Netzwerkes möglich. Da die für diese Netze typischen, am Rand des Wertebereiches liegenden Ungenauigkeiten z.B. durch Extrapolation der Randpunkte zu beheben sind, wählen wir die Interpolation durch Spline Funktionen. Allerdings liegen, wie oben angeführt, die Stützstellen weder äquidistant noch regelmäßig (im Sinne einer Anzahl von $n \cdot m$, $n, m \in \mathbb{N}$ Stützstellen), so daß die Menge der anwendbaren Interpolationsverfahren, wie z.B. die Interpolation durch bikubische Splines [36], eingeschränkt ist. Für Stützpunkte beliebiger Anordnung existiert eine Lösung mit der sogenannten *Thin-Plate-Spline*-Interpolation [37], welche das Verhalten einer dünnen Metallplatte simuliert, die an verschiedenen Stellen senkrecht zur ihrer Oberfläche ausgelenkt wird. Die entstehende Form der Platte repräsentiert ein Minimum an Krümmung, während die Auslenkungspunkte durchlaufen werden. Diese Interpolation wird nun im Zusammenhang mit dem Problem der Interpolation der Länge des optischen Flusses vorgestellt.

Interpolation des optischen Flusses

Gegeben sei die Bildfolge $B = \left((b_0^{pre}, b_0^{post}), \ \ldots \ , (b_L^{pre}, b_L^{post}) \right)$ aus Bildpaaren, die je ein Bild vor (b_l^{pre}) und nach (b_l^{post}) der Translation, aufgenommen mit einem Abstand $l \cdot d$, $l \in \{0, \ldots, L\}$ beinhalten (gemäß Abb 4.8). Zu jedem Bild b existiert die Menge der Positionen im Bild der in ihm enthaltenen Punkte des Punktmusters

$$Punkt(b) = \{(x_0^b, y_0^b)^{\mathrm{T}}, \ (x_1^b, y_1^b)^{\mathrm{T}} \ \ldots \} \tag{4.19}$$

Des weiteren sei die Indexfunktion

$$Ind : Punkt(b) \rightarrow \mathbb{N} \times \mathbb{N} \tag{4.20}$$

gegeben. Durch die Zuordnung eines Indizes zu einem Punkt $\boldsymbol{v}^{pre} \in Punkt(b^{pre})$ ist es möglich, den korrespondierenden, im zweiten Bild translatierten Punkt $\boldsymbol{v}^{post} \in Punkt(b^{post})$ zu bestimmen, indem getestet wird, ob $Ind(\boldsymbol{v}^{pre}) = Ind(\boldsymbol{v}^{post})$ gilt.

Damit ist die Menge der Punktepaare gegeben, für die der optische Fluß berechnet werden kann:

$$\begin{aligned} Trans\,(\, b^{pre}, b^{post} \,) \stackrel{Def}{=} \{ \ \ (\boldsymbol{v}^{pre}, \boldsymbol{v}^{post}) \ \ \mid & \quad \boldsymbol{v}^{pre} \in Punkt\,(b^{pre})\,, \\ \wedge & \quad \boldsymbol{v}^{post} \in Punkt\,(b^{post})\,, \\ \wedge & \quad Ind(\boldsymbol{v}^{pre}) = Ind(\boldsymbol{v}^{post}) \quad \} \end{aligned} \tag{4.21}$$

$$LOpt\left(\boldsymbol{v}^{pre}, \boldsymbol{v}^{post}\right) = \left\| \begin{pmatrix} v_1^{pre} - v_1^{post} \\ v_2^{pre} - v_2^{post} \end{pmatrix} \right\| \tag{4.22}$$

für $\boldsymbol{v}^{pre} = (v_1^{pre}, v_2^{pre})^{\mathrm{T}}$, $\boldsymbol{v}^{post} = \left(v_1^{post}, v_2^{post}\right)^{\mathrm{T}}$.

Für die praktische Umsetzung der Interpolation wird die Implementation der Funktionen *Punkt* und *Ind* aus den Gleichungen (4.19) und (4.20) skizziert. Die Implementation der Funktion *Punkt* erfolgt in den folgenden Schritten:

Eingabe: Bild b des Punktmusters

1. Binarisierung des Grauwertbildes des Punktmusters (256 Grauwerte, Grauwertschranke 128)
2. Detektion des ersten Pixels, das gesetzt und somit vermutlich Teil eines Rekonstruktionspunktes ist. Existiert kein gesetztes Pixel $\rightarrow$ Ende.
3. Bestimmung des Freemancodes der Umrandung. Unterscheidung eines Rekonstruktionspunktes von einem durch Rauschen gesetzten Pixel durch die Länge des Freemancodes. Anschließender Bestimmung der Punktmitte als Element der Ausgabemenge. (Da es sich um gleichmäßige, konvexe Gebiete handelt, ist es legitim, vom Freemancode auf die Punktmitte zu schließen.)
4. Löschen des durch den Freemancode beschriebenen Bildbereichs.
5. Weiter mit 2.

Ausgabe: Menge von Punktmitten der Rekonstruktionspunkte ($Punkt(b)$).

Die Implementation der Funktion Ind, Gleichung (4.20), besteht aus den Stufen:

Eingabe: Binarisiertes Bild b des Punktmusters

1. Detektion eines besonders gekennzeichneten Rekonstruktionspunktes (z.B. doppelter Umfang des Punktes im Vergleich zu den restlichen) und Zuweisung des Indizes (0,0).
2. Bestimmung der (vier) Nachbarn zu jedem Punkt durch Positions- und Abstandsvergleich, rekursiv nach *floodfill*-Algorithmus

Ausgabe: Indizes $Ind(b)$ der Rekonstruktionspunkte

Die Interpolationsaufgabe zur Berechnung des optischen Flusses an den Gitterpunkten $\boldsymbol{g_{ijl}}$ für eine Bildpaar B_l lautet nun wie folgt [37]:

Zu den Punktepaaren $(\boldsymbol{v}_l^{pre}, \boldsymbol{v}_l^{post}) \in Trans(b_l^{pre}, b_l^{post})$ und den zugehörigen Werten der Länge des optischen Flusses $LOpt(\boldsymbol{v}_l^{pre}, \boldsymbol{v}_l^{post})$ ist eine Funktion S_l gesucht mit:

$$S_l(\boldsymbol{v}_l^{post}) = LOpt(\boldsymbol{v}_l^{pre}, \boldsymbol{v}_l^{post}) \tag{4.23}$$

und $j = 1, \ldots, NX, \quad NX \stackrel{Def}{=} |Trans(b_l^{pre}, b_l^{post})|$.

Die Lösung des Interpolationsproblems erfolgt mit der Thin-Plate-Spline-Interpolation:

Mit M sei die *derivative Ordnung* gegeben, die die Weichheit der Interpolation bestimmt. Mit $p_i^{(M)}$ seien die zweidimensionalen Monome des Grades kleiner oder gleich M gegeben (für $M = 2$ ist z.B. die Menge der Monome mit $p_1^{(2)} = 1, p_2^{(2)} = x, p_3^{(2)} = y, p_4^{(2)} = x^2, p_5^{(2)} = xy, p_6^{(2)} = y^2$ gegeben). Es sei $MM \overset{Def}{=} M(M+1)/2$.

Die Interpolationsfunktion S_l (für Werte der Bildebene B_l) hat die Gestalt:

$$S_l(w) \overset{Def}{=} \sum_j c_j S_l^j(w) \tag{4.24}$$

Die Funktionen S_l^j seien für $\boldsymbol{w} = (w_1, w_2)^{\mathrm{T}}, \boldsymbol{v_j^{post}} = (v_{j_1}^{post}, v_{j_2}^{post})^{\mathrm{T}} \in Punkt(\boldsymbol{b}^{post})$ wie folgt definiert:

$$S_l^j(\boldsymbol{w}) \overset{Def}{=} \begin{cases} E(w_1 - v_{j_1}^{post}, w_2 - v_{j_2}^{post}) & \text{falls} \quad j = 1, \ldots, NX \\ p_{j-NX}^{(M-1)}(w_1, w_2) & \text{falls} \quad j = NX+1, \ldots, NX+MM \end{cases} \tag{4.25}$$

mit $E(x,y) \overset{Def}{=} (x^2+y^2)^{M-1}\ ln\ (x^2+y^2)$ (Kernel Funktion). Es ergibt sich ein Gleichungssystem mit $NX + MM$ Reihen und Spalten für eine $NX \times NX$-Matrix G und eine $NX \times MM$-Matrix P der Form:

$$\begin{pmatrix} G & P \\ P^{\mathrm{T}} & 0 \end{pmatrix} \begin{pmatrix} c^{(1)} \\ c^{(2)} \end{pmatrix} = \begin{pmatrix} f \\ 0 \end{pmatrix} \tag{4.26}$$

wobei $G = \left(E(v_{i_1}^{post} - v_{j_1}^{post}, v_{i_2}^{post} - v_{j_2}^{post})\right),\ i,j = 1, \ldots, NX$
und $P = \left(p_j^{(M-1)}(v_{i_1}^{post}, v_{i_2}^{post})\right),\ i = 1, \ldots, NX,\ j = 1, \ldots, MM$
sowie $c^{(1)} = (c_1, c_2, \ldots, c_{NX})^{\mathrm{T}},\ c^{(2)} = (c_{NX+1}, c_{NX+2}, \ldots, c_{NX+MM})^{\mathrm{T}}$
und $f = \left(LOpt(v_1^{pre}, v_1^{post}), LOpt(v_2^{pre}, v_2^{post}), \ldots, LOpt(v_{NX}^{pre}, v_{NX}^{post})\right)$.

Gilt $det(P) \neq 0$, so ist mit der Lösung des Gleichungssystems (4.26) die Berechnung der Koeffizienten c_i und damit die Interpolationsaufgabe aus Gleichung (4.23) gelöst. Damit ist die Berechnung der Länge des optischen Flusses $\|\boldsymbol{o_{ijl}}\|$ bezüglich einer Bildebene B_l an der Stelle $\boldsymbol{g_{ijl}} = (x_{ijl}, y_{ijl}, Z_l)$ mit $S_l((x_{ijl}, y_{ijl}))$ möglich.

Es ist zu beachten, daß die Kondition der Matrix in Gleichung (4.26) mit steigenden Werten NX und M schlechter wird [37]. Daher betrachten wir die Interpolation nun in Abhängigkeit der Parameter NX und M. Für die Menge von 1354 Rekonstruktionspunkten wurde die Länge des optischen Flusses nach einer seitlichen Translation von $5cm$ berechnet. In

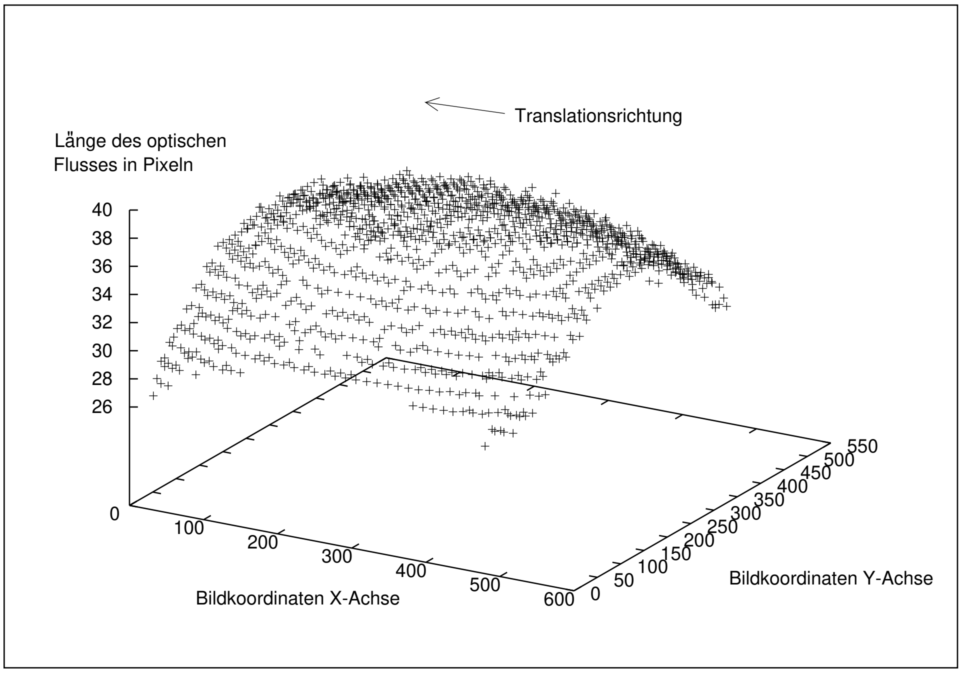

Abbildung 4.9: Länge des optischen Flusses nach seitlicher Translation, gemessen an 1354 Bildpunkten

Abbildung 4.10 wird die Auswirkung der Anzahl der in der Interpolation berücksichtigen Punkte (NX) betrachtet. Als Fehlermaß (mittlerer durchschnittlicher Fehler) dient der Abstand der Interpolation zu den Stützpunkten, die, je nach Anzahl der benötigten Stützstellen, zum Teil nicht in die Interpolation mit einbezogen werden. Die derivative Ordnung ist mit $M = 3$ zunächst fest gewählt. Die Auswirkungen der schlechten Kondition der Systemmatrix in Gleichung (4.26) bei einer großen Anzahl von Stützpunkten ist für den Fall $NX = 1354$ in Abbildung 4.10 zu sehen: das Ziel der möglichst glatten Interpolation ist verfehlt. Erst mit abnehmender Stützpunktzahl stellt sich mit günstigerer Kondition eine glattere Interpolationsfläche ein. Für einen Mindestabstand der Stützpunkte von 120 Pixeln und entsprechender Stützpunktzahl von 20 ist der minimale mittlere Fehler von 0.51 Pixeln erreicht. Nun variieren wir den Parameter M der derivativen Ordnung (Abbildung 4.11). Eine Polynombasis vom Grad $M = 1$ (Monome $1, x, y$) ist Ursache eines mittleren Fehlers von 1.59 Pixeln. Mit steigendem Grad M verkleinert sich der mittlere Fehler auf einen Wert von 0.51 Pixel für $M = 3$, so daß der minimale Fehler für den variierenden Grad M gefunden ist. Die vollständige Fehleraufstellung für die Parameter NX und M ist in der folgenden Tabelle zusammengefaßt. Die Parameterkombination $NX = 20$ und $M = 3$ minimiert den mittleren, durchschnittlichen Fehler.

NX	1354	207	67	32	**20**	14	10
Distanz	0	30	60	90	**120**	150	180
M=1	0.0	0.56	0.82	1.18	1.59	1.90	2.14
M=2	0.0	0.52	0.56	0.56	0.57	0.72	1.30
M=3	0.0	0.55	0.59	0.57	**0.51**	0.57	0.52
M=4	0.0	0.58	0.63	0.59	0.57	0.63	0.59
M=5	0.0	0.63	0.70	0.63	0.63	1.17	1.23

Tabelle 6: Mittlerer, durchschnittlicher Fehler der Thin-Plate Spline Interpolation in Abhängigkeit der derivativen Ordnung (M) und der Anzahl der Stützpunkte (NX) für das Interpolationsproblem der Abbildung 4.9.

Nachdem nun die Berechnung des optischen Flusses an beliebiger Bildposition möglich ist, können die für die Tiefenberechnung zuständigen Netzwerke Rz_{ij} trainiert werden. Dabei ist die Knotenzahl (Hidden Units) gleich der Anzahl der Bildpaare bzw. der Kalibrierungsebenen. In Abbildung 4.12 ist die Ausgabe eines mit 10 Trainingspaaren $(\|\boldsymbol{o}_{ijl}, Z_l)$, $l = 1, \ldots, 10$ trainierten Netzes zu finden. Ein bei RBF-Netzen auftretendes Randproblem macht sich hier deutlich bemerkbar: Die am Rand liegenden Hidden Units weisen weniger Nachbarn auf, die beim Trainingsvorgang Einfluß auf die entsprechenden Gewichte nehmen können. Hierdurch wird die Interpolation (bzw. Approximation) des Netzes in den Randbereichen ungüngstig beeinflußt. Dieser Effekt kommt bei Netzwerken mit wenigen Knoten besonders zu tragen, wie sie typischerweise in dieser Anwendung eingesetzt werden, da die Randbereiche einen größeren Anteil am Gesamtnetz aufweisen. Eine Lösung dieses Problems ist die Extrapolation von am Rand liegenden Wertepaaren und Einbeziehung dieser in den Trainingsvorgang. Hierdurch wird die Interpolation in den am Rand liegenden Gebieten qualitativ besser (Abbildung 4.13). Das Problem der ungenauen Interpolation wird gewissermaßen aus dem interessierenden Bereich verbannt.

Training der Netze zur X-Y-Rekonstruktion

Für die noch verbleibende Aufgabe der Berechnung von $X - Y-$Weltkoordinaten aus Bildpositionen (x_i, y_i) werden RBF-Netzwerke Rxy_l eingesetzt, die ihre Knoten (Hidden Units) innerhalb des zweidimensionalen Unterraumes, der durch die Bildebene des nach der Translation aufgenommenen Bildes b_l^{post} gegeben ist, positionieren. Innerhalb des Bildes werden die Bildorte der Rekonstruktionspunkte als Knotenpositionen gewählt, so daß eine Interpolation zum Training dieser Netze nicht erforderlich ist (Abbildung 4.14). Zwar liegt aufgrund der Verzerrungen keine rechtwinklige, jedoch eine gleichmäßige Punkteanordnung vor. Da das verwendete Punktmuster je nach Entfernung eine Anzahl von ca. 10^3 Punkten im aufgenommenen Bild aufweist (Abbildung 4.5), ausreichend gute Ergebnisse jedoch auch mit einer geringeren Anzahl von Knoten erreicht werden, ist ein Setzen der Knoten auf eine Untermenge der Punktpositionen ausreichend (Kapitel 4.2.6). Zur Klärung der Frage, in

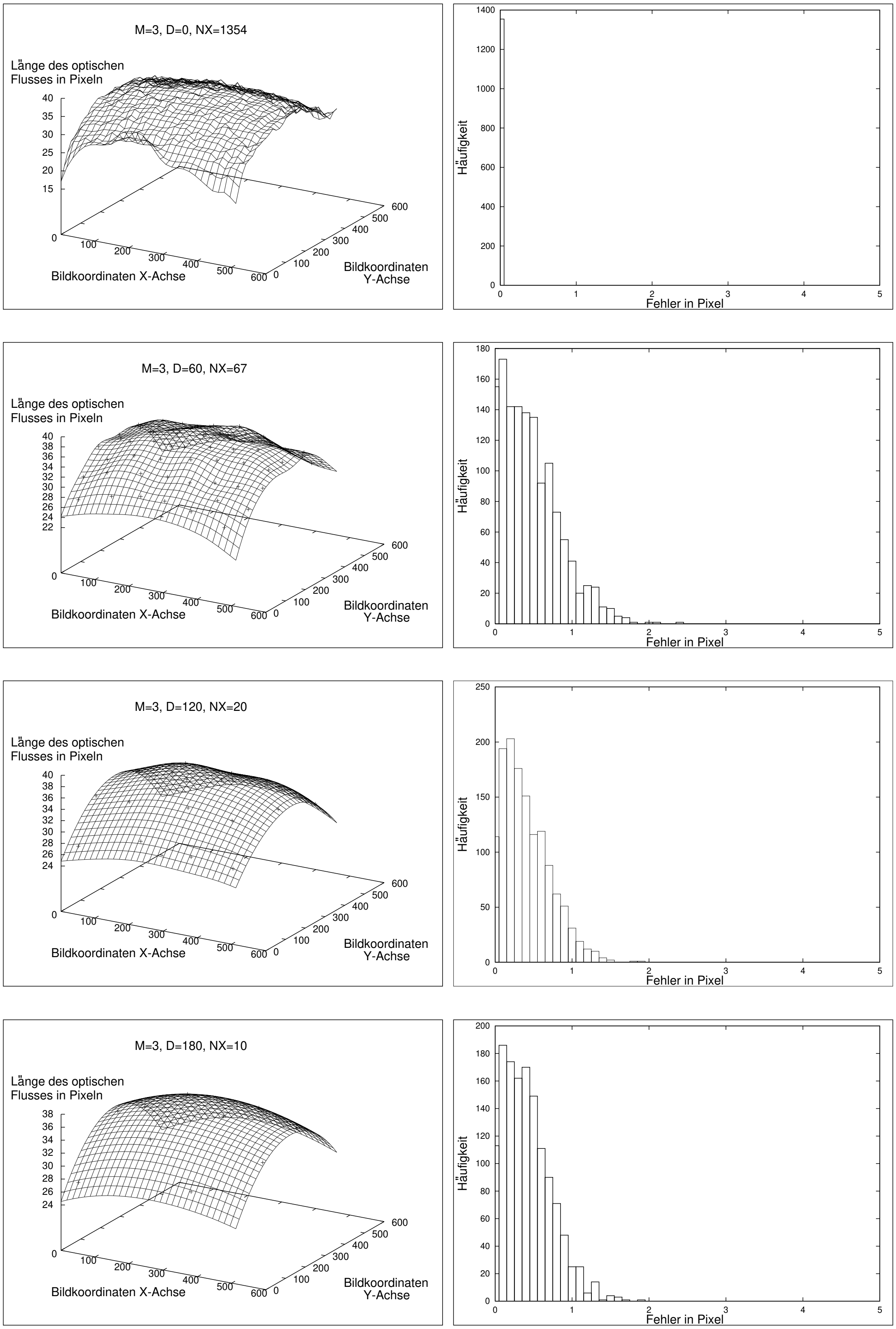

Abbildung 4.10: Links: Thin-Plate-Spline Interpolation mit variierender Stützpunktzahl (NX), entsprechender Stützpunktdistanz (D, in Pixel) und fester derivativer Ordnung von $M = 3$. Rechts: Fehlerhistogramm

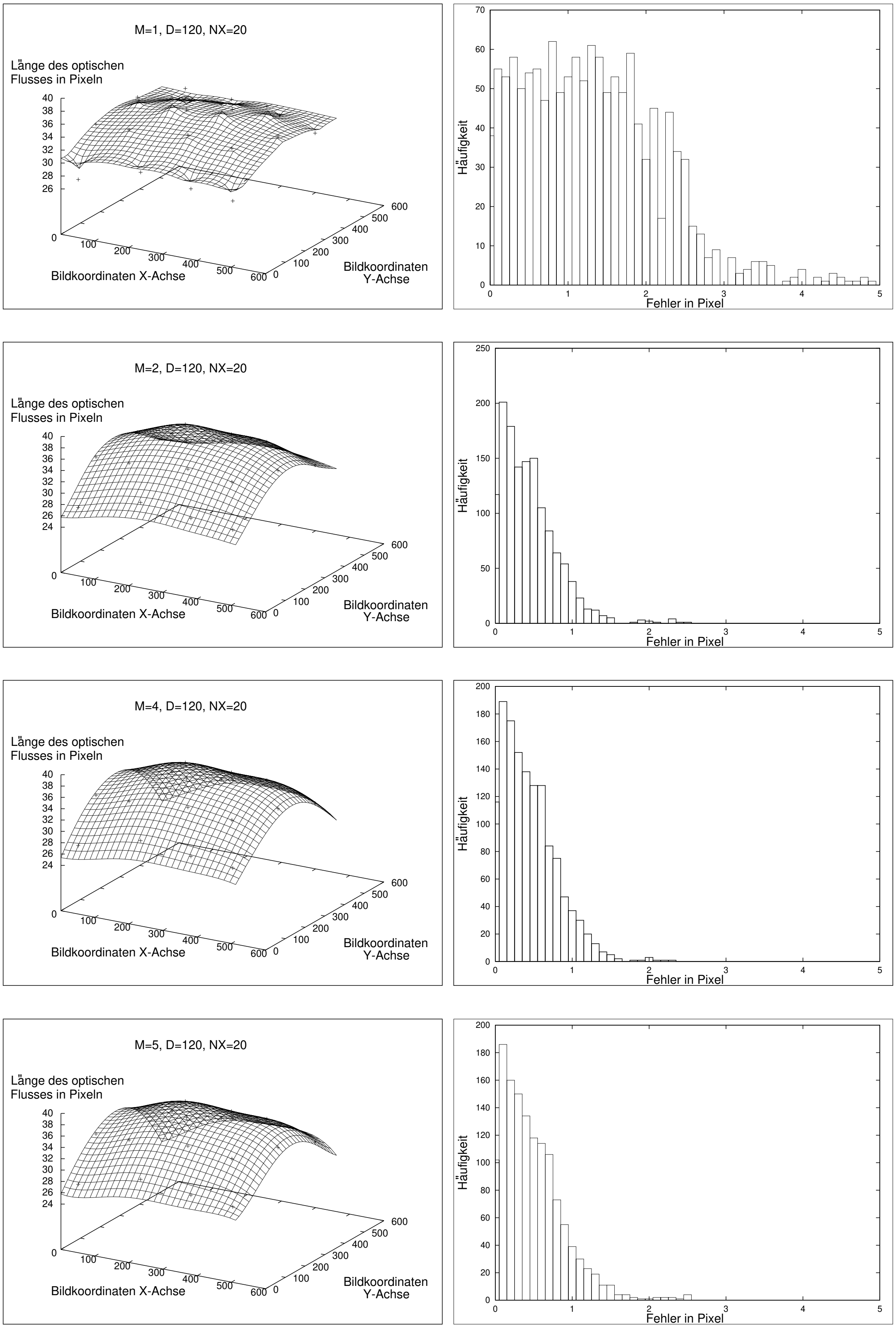

Abbildung 4.11: Links: Thin-Plate-Spline Interpolation mit fester Stützpunktzahl ($NX = 20$), entsprechender Stützpunktdistanz ($D = 120$, in Pixel) und variabler derivativer Ordnung M. Rechts: Fehlerhistogramm. (Fall $M = 3$ siehe Abbildung 4.10)

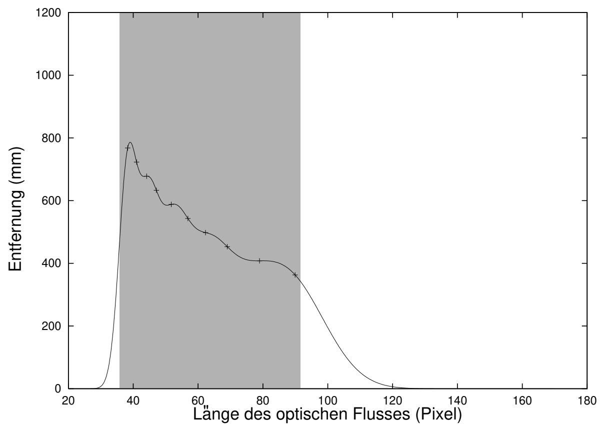

Abbildung 4.12: RBF-Netz Rz_{ij}, trainiert mit 10 Meßwerten $(\|\boldsymbol{o}_{ijl}\|, Z_l)$ im dunkelgrau markierten Segment. Aufgrund der wenigen Wertepaare wirken sich die am Rand befindlichen Ungenauigkeiten auf das gesamte, mit Meßwerten besetzte Segment aus.

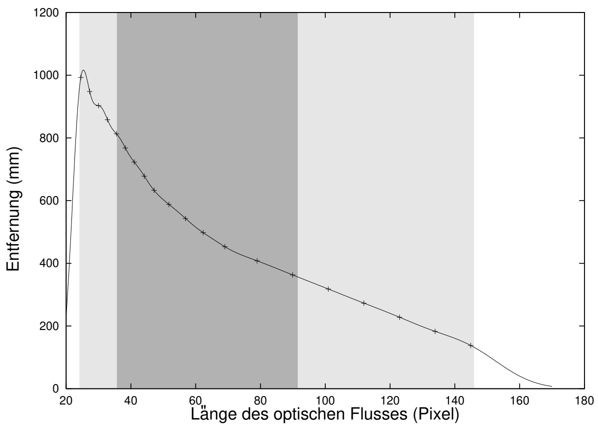

Abbildung 4.13: RBF-Netz Rz_{ij}, trainiert mit denselben 10 Meßwerten wie in Abbildung 4.12, jedoch zusätzlich mit je fünf extrapolierten Wertepaaren im hellgrauen Segment. Die Interpolation weist durch die geringere Krümmung eine bessere Interpolation auf.

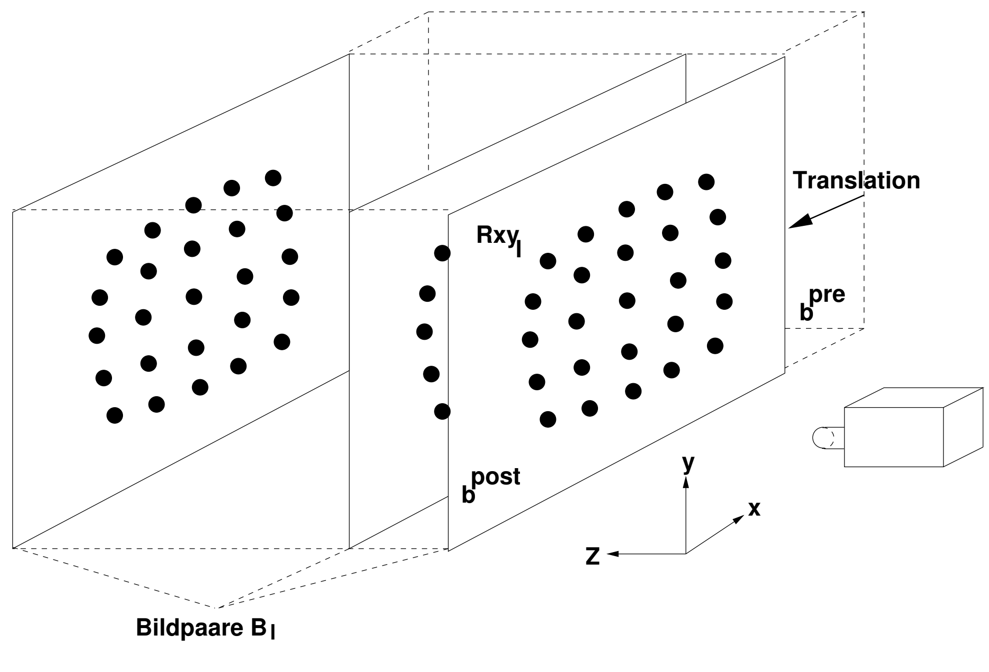

Abbildung 4.14: Die Knoten der Netzwerke Rxy_l liegen auf den Rekonstruktionspunkten innerhalb der Bildebenen der nach der Translation aufgenommenen Bilder b_l^{post}.

wievielen Ebenen tatsächlich Netzwerke zur Rekonstrunktion der $X-Y-$Weltkoordinaten benötigt werden, greifen wir auf das eingangs angeführte Stereokamerasystem und dessen Rekonstruktionsgleichungen zurück. Es gilt für die Rekonstruktion der Weltkoordinaten X und Y aus den Bildkoordinaten (x,y):

$$X = Z\frac{x}{f}, \quad Y = Z\frac{y}{f} \tag{4.27}$$

für eine (uns unbekannte) Brennweite f. Damit gilt für gegebenes x (und f): Die Weltkoordinate X ist direkt proportional zur Bildkoordinate x (Y und y respektive). Dies gilt allerdings für den verzerrungsfreien Fall. Die Bildverzerrung D kann als der Projektion nachgeschaltete Funktion auf die Bildkoordinaten aufgefaßt werden[8]:

$$D : \mathbb{R}^2 \rightarrow \mathbb{R}^2 \tag{4.28}$$

$$D\,(x, y) = (x + d_x, y + d_y) \tag{4.29}$$

für $d_x,\ d_y \in \mathbb{R}$. Damit ist die Rekonstruktion der Weltkoordinaten X und Y aus den Bildkoordinaten x und y durch die Nacheinanderausführung der folgenden Abbildungen gegeben:

1. $(x', y') = D^{-1}\,(x, y)$ (Entzerrung der Bildkoordinaten)
2. $(X, Y) = \left(Z\frac{x'}{f}, Z\frac{y'}{f}\right)$ (Rekonstruktion der Weltkoordinaten)

Der in Punkt 2. gegebene, lineare Zusammenhang zwischen den Weltkoordinaten und der Tiefe ist auch den Abbildungen 4.15 und 4.16 entnehmbar, die die Netzantworten der Rxy-Netzwerke, bezogen auf die jeweilige Tiefe, darstellen. Aufgrund dieses Zusammenhangs ist es ausreichend, zwei Netze Rxy_k und Rxy_l zu trainieren und die Zwischenwerte linear zu interpolieren bzw. zu extrapolieren. Dies können z.B. die am Anfang und am Ende des zu erlernenden Volumens plazierten Netze sein oder die zwei Netze mit der höchsten Knotenzahl. Wir wählen das erste und das letze Netz aus.

Durch das Training der Netze Rxy_l mit den die Rekonstruktionspunkte im Bild repräsentierenden Paaren (x_i, y_i) und den korrepondierenden Weltkoordinaten (X_i, Y_i) ist das System der Rekonstruktionsnetze vollständig trainiert.

[8]An diesem Punkt wird deutlich, daß es ebenfalls die Möglichkeit gibt, zuerst eine (neuronale oder geometrische) Bildrektifizierung vorzunehmen und dann die (neuronale oder geometrische) Rekonstruktion der Weltkoordinaten. Wir haben jedoch den naheliegenden Weg gewählt, diese beiden Abbildungen gleichzeitig durch ein Neuronales Netz Rxy_l zu erlernen.

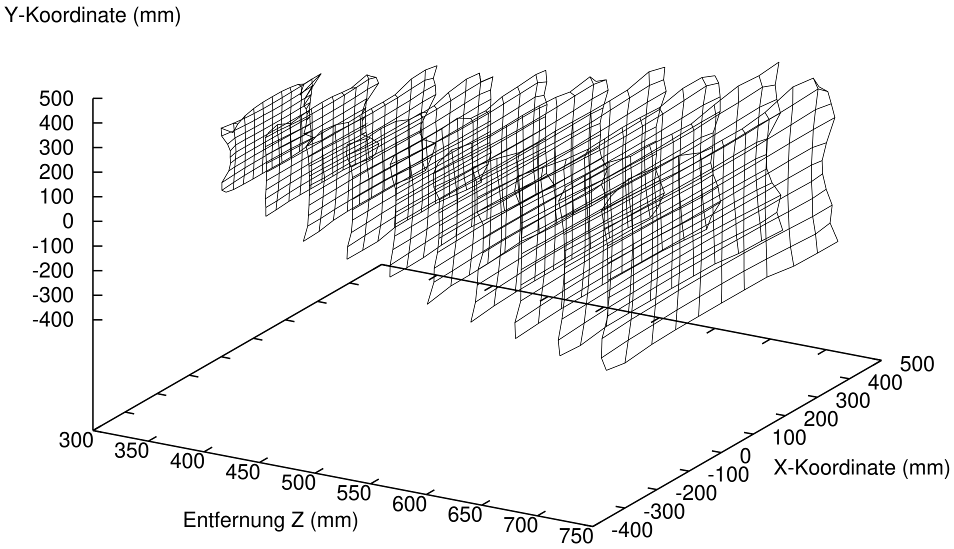

Abbildung 4.15: Die Antworten der Netzwerke Rxy_l für die gesamte Bildfläche.

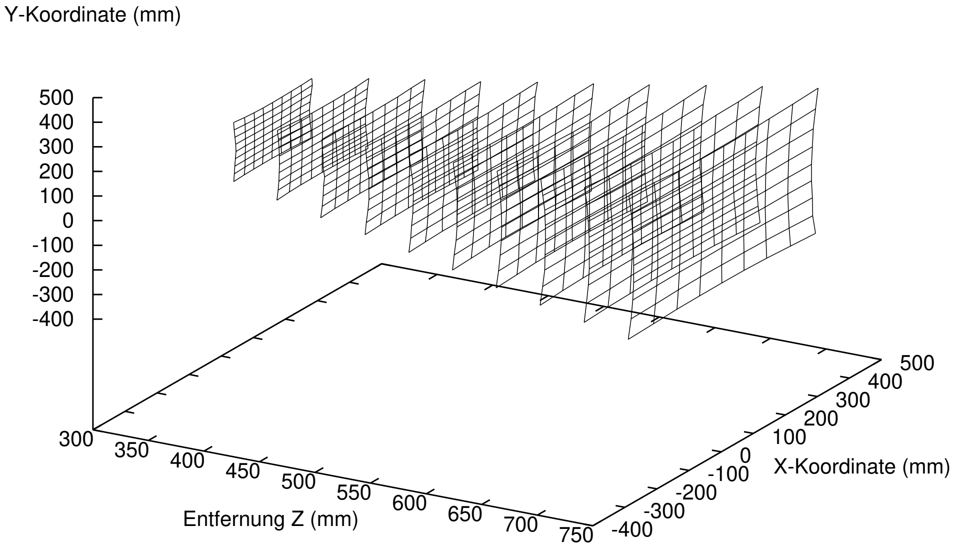

Abbildung 4.16: Die Antworten der Netzwerke Rxy_l für die Bildfläche ohne die Randbereiche, die im Gegensatz zum inneren Bereich mit einem größeren Fehler behaftet sind.

4.2.4 Vergleich des Gesamtnetzes mit dem gesplitteten Netz

In der folgenden Tabelle sind die Parameter des Gesamtnetzes und des Netzsystems vergleichend gegenübergestellt. Wir greifen wieder auf das Beispiel aus Kapitel 4.2.1 zurück, bei dem der Arbeitsbereich eine Größe von $1m^3$ aufweist und von einem rechtwinkligen Gitter mit einem Gitterknotenabstand von $5cm$ durchzogen wird. Die Gitterknoten sind Teil der Trainingsdaten und repräsentieren damit die Positionen der Netzknoten. Das Gesamtnetz weist damit eine Knotenzahl von $21^3 = 9261$ Knoten auf. Beim gesplitteten Netz haben die zur Rekonstruktion des Tiefenwertes zuständigen Netze Rz eine Anzahl von je 21 Knoten, wobei $21^2 = 441$ dieser Netze benötigt werden. Hinzu kommen zwei Netze zur X-Y-Rekonstruktion, die je 441 Knoten aufweisen. Die Trainingszeit und die Anzahl der Netze verhalten sich indirekt proportional zueinander[9].

	ohne Netzteilung	mit Netzteilung
Anzahl der Netze	1	443
Anzahl der Knoten	9261	21 je Netz Rz (441 Netze), 441 je Netz Rxy (2 Netze), 10143 insgesamt
Trainingszeit SUN 167MHz	$\approx$ 92 Stunden	45 Sekunden

Tabelle 7: Auswirkung der Netzteilung

4.2.5 Anwendung des geplitteten Netzes

Nachdem das Training der zwei Netzwerkkategorien abgeschlossen ist, besteht die Aufgabe nun darin, aus den Antworten der Teilnetze eine Antwort des Netzsystems zu bilden. Durch die Zerlegung des Gesamtnetzes in Teilnetze sind Bereiche entstanden, in denen die Generalisierung des Gesamtnetzes fehlt. Beispielsweise sind die Netze Rxy für bestimmte Ebenen, die parallel zur X-Y-Ebene liegen, trainiert worden. Wird eine Antwort für einen zwischen diesen Ebenen liegenden Punkt benötigt, muß ein geeignetes Verfahren die an diesem Punkt fehlende Netzgeneralisierung übernehmen. Neben einfachen Verfahren, wie z.B. der Auswahl der Antwort des Netzes, welches einem Datum am nächsten gelegen ist oder der Triangulation der Netzantworten liefern aufwendigere Verfahren, wie z.B. die Interpolation von Netzantworten bessere Ergebnisse.

Interpolation als Nachbildung der Netzgeneralisierung

Der Berechnungsweg einer Netzantwort $(X, Y, Z)^{\mathrm{T}}$ aus der Eingabe $((x_1, y_1)^{\mathrm{T}}, (x_2, y_2)^{\mathrm{T}})$ resultiert aus dem gewählten Netzsplitting aus Kapitel 4.2.2 und ist dort in Abbildung 4.7

[9] Der höhere Wert für die Trainingszeit des Gesamtnetzes wurde aus praktischen Gründen rechnerisch angenähert, indem, ausgehend von einer Meßreihe von Netztrainingszeiten, der Trainingsaufwand mit der Funktion $10^{-7} \cdot x^{3.156}$ (x ist die Anzahl der Knoten) für den ULTRASPARC Prozessor (167 MHz) geschätzt wurde.

dargestellt worden. In jeder der beiden Stufen der Berechnung einer Netzantwort ist eine Interpolation notwendig:

1. Berechnung eines Tiefenwertes Z als Interpolationsergebnis der um den Punkt $(x_2, y_2)^{\mathrm{T}}$ liegenden Netze Rz (Funktion $f_{Rek}^{xy \to Z}$).
2. Berechnung der Werte $(X, Y)^{\mathrm{T}}$ als Interpolationsergebnis der Netzwerke Rxy in Abhängigkeit der berechneten Tiefe Z (Funktion $f_{Rek}^{Z \to XY}$).

Schon im Kapitel 4.2.3 wurde begründet, daß die lineare Interpolation zweier am Anfang und am Ende des Arbeitsbereiches plazierten Rxy-Netze die Netzgeneralisierung hinsichtlich der X-Y-Rekonstruktion ersetzen kann. Es bleibt also zu klären, wie die Interpolation der Stufe 1 beschaffen sein muß.

Für die Länge des an der Bildpunktposition $(x_2, y_2)^{\mathrm{T}}$ berechneten optischen Flusses $\boldsymbol{o} = (x_2 - x_1, y_2 - y_1)$ als Anfrageobjekt antworten die diesen Punkt lokal umgebenden Netzwerke $Rz_{k,l},\ k, l \in \{\lfloor \frac{x_2}{d} \rfloor - w,\ \lfloor \frac{x_2}{d} \rfloor + w,\}$ für einen die angesprochenen benachbarten Netzwerke bestimmenden Wert $w \in \mathbb{N}$ und ein Gitter G_{sd} nach Gleichung (4.18), siehe dazu Abbildung 4.17.

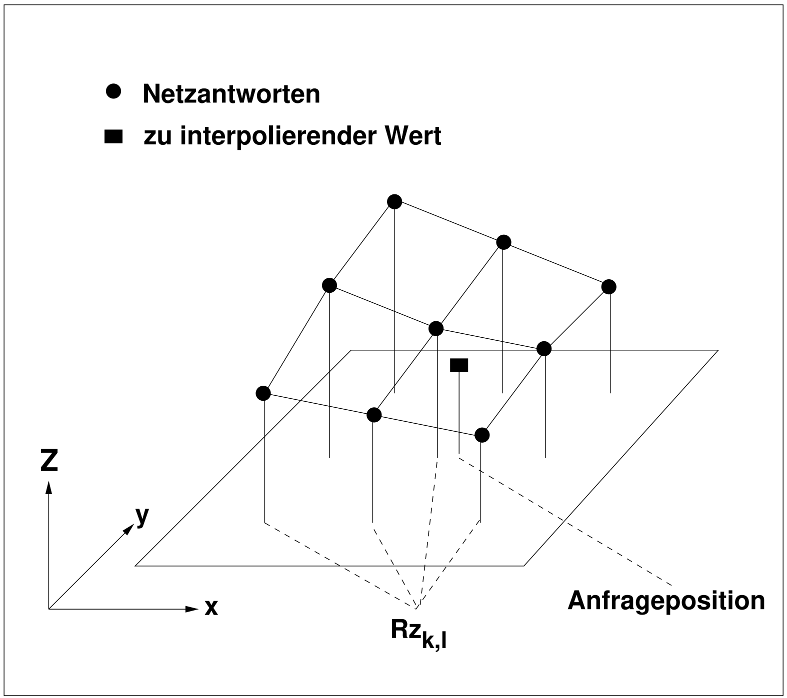

Abbildung 4.17: Zu einer Anfrageposition antworten die benachbarten Netzwerke Rz_{kl}. Der zu interpolierende Wert wird mittels bikubischer- oder Thin-Plate-Splineinterpolation berechnet.

Da die Stützpunkte nach Maßgabe des Gitters G_{sd} äquidistant über das Bildkoordinatensystem verteilt sind, eignen sich z.B. auf Splines basierende Standard-Interpolationsverfahren wie z.B. die bikubische Splineinterpolation [36] zur Ermittlung des zur Position $(x_2, y_2)^{\mathrm{T}}$ gehörenden Interpolationswertes. Aufgrund der geringen Punktemenge ist auch das oben vorgestellte Thin-Plate-Spline-Interpolationsverfahren geeignet.

Nachdem der optische Fluß an beliebiger Bildposition zu einer Tiefenantwort Z geführt hat, können nun die fehlenden Koordinaten X und Y in Stufe 2 durch lineare Interpolation der Netzantworten der Rxy-Netze in Abhängigkeit des Tiefenwertes Z ermittelt werden. Diese Netzantworten sind für die jeweils gesamte Bildfläche in Abbildung 4.15 bzw. für die Bildfläche ohne den Rand, der die äußeren Knoten beherbergt, in Abbildung 4.16 dargestellt. Wie schon bei den eindimensionalen Netzen in Kapitel 4.2.3 gezeigt, liegen auch hier an den Rändern der durch die Knoten eingenommenen Volumina fehlerhafte Antworten aufgrund der geringeren Anzahl der den Knoten umgebenden Traininsbeispiele vor. Durch Ausschluß der Antworten, die durch randständige Eingaben zustande kommen und damit in der Regel fehlerhaft sind, kann dem Problem begegnet werden. Wie in Kapitel 4.2.3 gezeigt, ist auch eine Extrapolation geeignet, das Problem der am Rand beheimateten Fehler zu lösen. Die leichte Wölbung der Gitter nach innen in den Abbildungen 4.15 und 4.16 resultiert aus der durch die Netzwerke erlernten Kompensation der Bildverzerrung (Fischaugeneffekt) und der Eingabe äquidistanter Positionen. Mit diesen Methoden der Netzinterpolation ist die Generalisierung eines das interessierende Gesamtvolumen umspannenden Netzes nachgebildet.

4.2.6 Genauigkeit des gesplitteten Netzes

Zur Bestimmung der Genauigkeit der Rekonstruktion des Netzsystems verwenden wir die folgenden Parameter: Die Tiefe (Z) der beiden zur X-Y-Ebene parallel liegenden Netzwerke Rxy_1 und Rxy_2 beträgt $20mm$ bzw. $670mm$. Jedes der Netzwerke weist eine Knotenzahl von $20 \times 20 = 400$ Knoten auf. Die Anzahl der Rz-Netzwerke beträgt 400, die der Knoten dieser Netzwerke jeweils 14, d.h. die Trainingspaare werden entlang der Z-Achse in einem Abstand von $50mm$ gewonnen. Jeweils zwischen den Trainingsebenen werden die Testebenen mit gleichen Abstand positioniert, aus denen die Testdaten gewonnen werden (Abbildung 4.18). Für die Testebenen wird das gleiche Muster genommen, das zum Training verwendet wird (Abbildung 4.5).

Wir betrachten die Genauigkeit der Rekonstruktion des Netzsystems sowohl hinsichtlich der einzelnen Dimensionen als auch insgesamt. Dazu berechnen wir für jede Testebene an Zwischenpositionen des Punktmusters den Positionierungsfehler und ermitteln das Fehlerhistogramm für die Dimensionen X, Y und Z, als auch für den Gesamtrekonstruktionsfehler. Der resultierende, mittlere Fehler ist in Abbildung 4.19 gegeben und für sechs ausgewählte Testebenen in der Tabelle 8 aufgeschlüsselt.

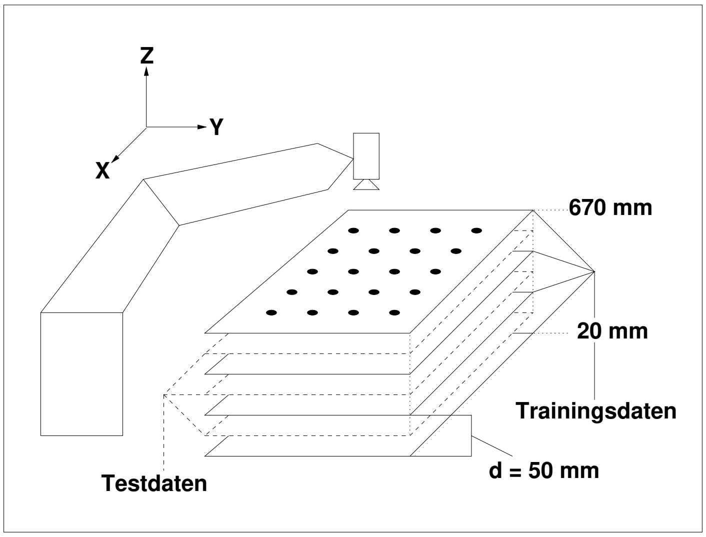

Abbildung 4.18: Anordnung zur Bestimmung der Rekonstruktionsgenauigkeit des Netzsystems.

Höhe	Fehler X-Koord.	Fehler Y-Koord.	Fehler Z-Koord.	Gesamtfehler
$95mm$	3.01	1.98	4.28	5.66
$195mm$	2.80	1.93	4.41	5.60
$295mm$	2.77	1.93	4.59	5.73
$395mm$	2.72	2.01	4.89	5.96
$495mm$	2.72	2.21	5.37	6.40
$595mm$	2.84	2.00	5.61	6.47

Tabelle 8: Mittlerer, quadratischer Fehler von sechs ausgewählten Testebenen.

Der Gesamtrekonstruktionsfehler erreicht sein Maximum von $6.47mm$ bei einer Höhe der Testebene von $595mm$. Für die einzelnen Koordinaten gelten die folgenden Fehlermaxima: $2.84mm$ (X-Koordinate), $2.21mm$ (Y-Koordinate) und $5.61mm$ (Z-Koordinate). Damit weist die Rekonstruktion der Tiefe einen ungefähr doppelt so großen mittleren Fehler auf wie die Rekonstruktion der Werte der X- bzw. Y-Koordinaten.

Da für die Testebene der Höhe $595mm$ der maximale Gesamtfehler vorliegt, beschränkten wir uns statt der Angabe aller auf die Darstellung der zu diesem Wert korrespondierenden Histogramme in Abbildung 4.20.

Der gemessene Fehler ist klein genug, um eine ausreichende Genauigkeit bezüglich der Rekonstruktion der betrachteten Szene zu erreichen. Durch Anwendung einer in Kapitel 3.2 erläuterten sowie in Kapitel 7.3 in ihrer Wirkung dargestellten Heuristik, die auf der geringeren Häufigkeit nicht korrekt rekonstruierter Punkte beruht, kann das Rekonstruktionsergebnis verbessert werden. Wird eine höhrere Genauigkeit der Rekonstruktion

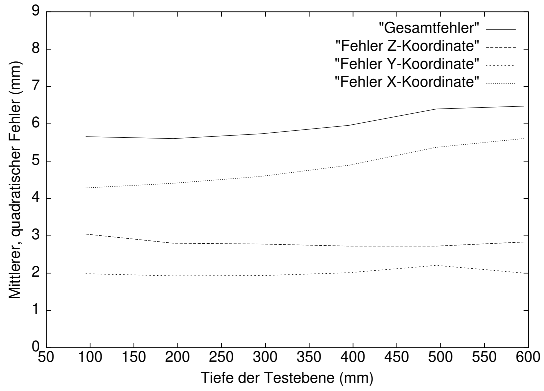

Abbildung 4.19: Mittlerer, quadratischer Fehler der Rekonstruktion.

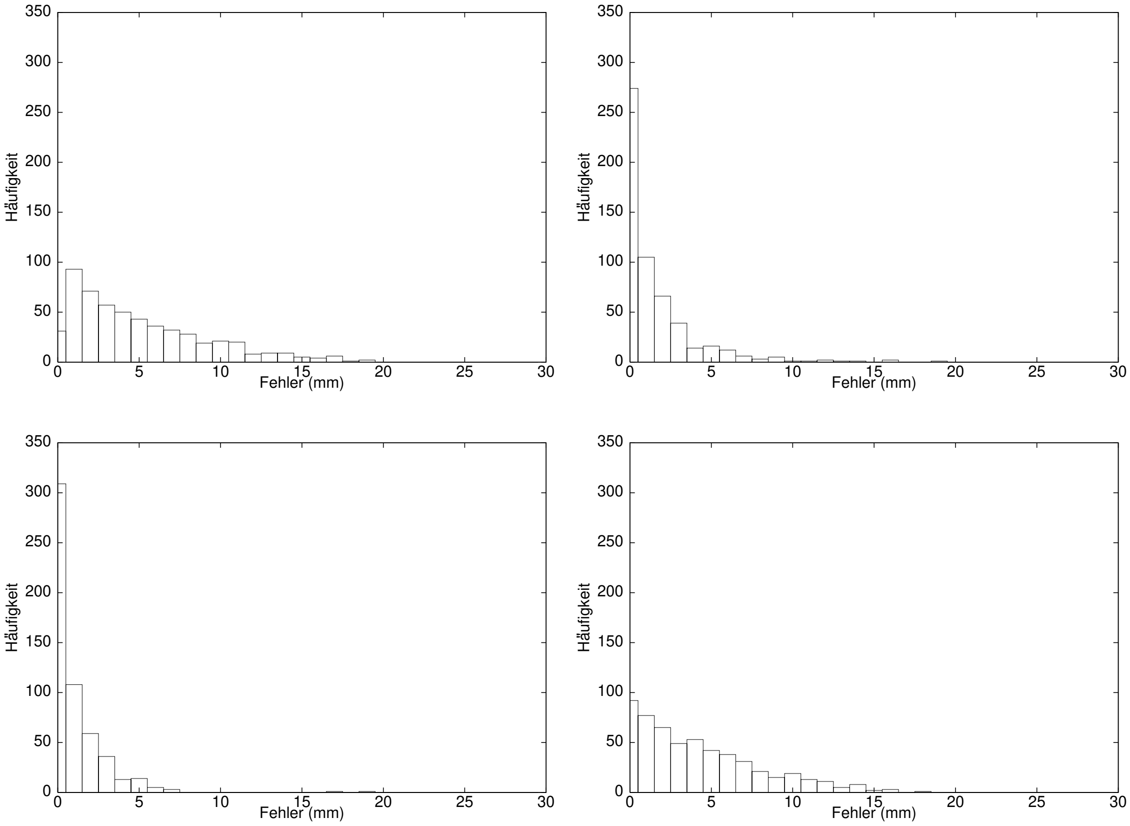

Abbildung 4.20: Mittlerer quadratischer Fehler. Oben links: Gesamtfehler. Oben rechts: Fehler X-Koordinate. Unten links: Fehler Y-Koordinate. Unten rechts: Fehler Z-Koordinate.

benötigt, kann der Fehler durch Erhöhung der Anzahl der Netze Rxy sowie Rz verringert werden.

Kapitel 5

Begriffsdefinition Hindernis, Trajektorie und Ziel

Aus der Menge $H \subset \mathbb{R}^3$ der aus zweidimensionalen Strukturpositionen neuronal berechneten dreidimensionalen Hindernispunkte der Szene ist nun eine Trajektorie zu generieren, die die Roboterhand zu einem definierten Zielpunkt $\boldsymbol{z} \in \mathbb{R}^3$ unter Umgehung der Hindernispunkte führt. Grundsätzlich läßt sich Hindernisvermeidung sowohl im Konfigurationsraum des Roboters als auch im kartesischen Raum der Szene betreiben. Die erste Variante erfordert die Abbildung aller Punkte $\boldsymbol{h} \in H$ auf den den Konfigurationsraum des Roboters, der mit den einzelnen Gelenkwinkeln für den hier verwendeten Roboter einen sechsdimensionalen Raum darstellt. Eine Trajektorie besteht in diesem Fall aus einer Folge von sechsdimensionalen, die Gelenkstellungen beschreibenden Vektoren. Üblicherweise ist der die Hindernispunkte enthaltene Teilraum jedoch nicht mit dem den Aktionsbereich des Roboters umfassenden Raum identisch, sondern örtlich eng begrenzt (Abbildung 5.1).

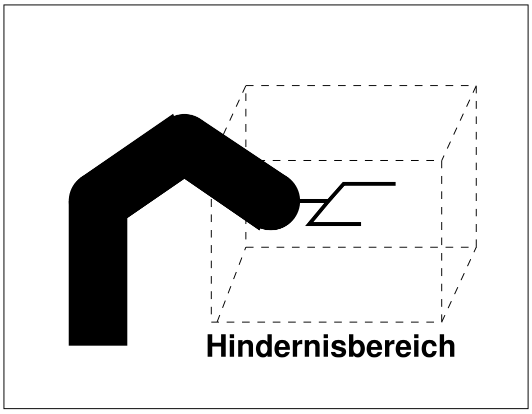

Abbildung 5.1: Mögliche Kollisionen beschränken sich auf den in den Hindernisbereich eintauchenden Roboterarm.

Es ist daher erwägenswert, zur Reduzierung des Aufwandes die Kollisionsvermeidung auf das letzte Armsegment des Roboters sowie auf die Roboterhand zu beschränken, obwohl das hier vorgestellte Verfahren auf alle Segmente anwendbar ist. Die Beschränkung der Kollisionsdetektion auf das letzte Segment und einen begrenzten Arbeitsbereich bringt im kartesischen Raum eine Aufwandsverminderung. Da das Volumen des begrenzten Arbeitsbereiches im Konfigurationsraum aus mehreren, kompliziert geformten Teilvolumina besteht, ist die Restriktion auf einen Bereich in diesem Raum kaum von Vorteil. Ein weiterer Grund, der für die Anwendung der Hindernisvermeidung im kartesischen Raum spricht, ist die geringere Dimensionalität des kartesichen Raums im Gegensatz zum Konfigurationsraum, die einen geringeren Aufwand der Hindernisvermeidung zur Folge hat. Da jedoch im kartesischen Raum die Roboterposition lediglich als Position der Hand vorliegt, ist sie nicht dazu geeignet, Aussagen über die Kollisionsfreiheit des gesamten Roboters zu begründen. Denn für eine gegebene Roboterhandposition im Weltkoordinatensystem, die kollisionsfrei bezüglich der Roboterhand ist, kann ein Hindernispunkt im Volumen eines Roboterarmsegmentes liegen. Daher ist der Hindernisbegriff neu zu definieren, so daß der Begriff *hindernisfrei* für eine Position im kartesischen Koordinatensystem das vom Roboter eingenommene Volumen berücksichtigt. Ziel ist es also, die im Konfigurationsraum vorhandene, explizite Angabe der vollständigen, aktuellen Roboterkonfiguration in den kartesischen Raum zu übertragen und so die Vorteile des kartesischen Raumes zu nutzen, ohne auf die Vorteile des Konfigurationsraumes zu verzichten. Nach der Erweiterung des Hindernisbegriffs wird die dazu notwendige Berechnung der Gelenkwinkel (Vorwärtskinematik) als auch die ebenfalls benötigte neuronal gelernte Inverse Kinematik behandelt. Mit der anschließenden Betrachtungen der Bestimmung einer Zielposition sowie der Definition des Begriffs Trajektorie werden die Grundlagen zur Navigation gelegt.

5.1 Erweiterung der Hindernisdefinition

Das Ziel der Neudefinition des Begriffes *Hindernispunkt* ist es, nicht nur Aussagen hinsichtlich der Hinderniseigenschaft des Punktes selber, sondern auch hinsichtlich des durch den Roboter eingenommenen Raumes zu treffen. Für diese Erweiterung legen wir zunächst die folgenden Begriffe fest:

Mit $\boldsymbol{k}_i \in \mathrm{I\!R}^3,\ i \in \{0 \ldots 4\}$ seien die Anfangs- und Endpunkte der Roboterarmsegmente (inklusive der Hand) bezeichnet. Dabei sind alle segmentbegrenzenden Punkte $\boldsymbol{k}_i$ abhängig von der Position der Roboterhand $\boldsymbol{p}_{Hand} \in \mathrm{I\!R}^3$ (Abbildung 5.2a):

$$Pos(\boldsymbol{p}_{Hand}) = (\boldsymbol{k}_0, \boldsymbol{k}_1, \boldsymbol{k}_2, \boldsymbol{k}_3, \boldsymbol{k}_4) \tag{5.1}$$

Mehrdeutigkeiten sind lediglich durch die nach oben bzw. unten gerichtete Armhaltung möglich (Abbildung 5.3), die bei dem verwendeten Modell RX90 fest konfigurierbar ist, so daß mit Gleichung (5.1) tatsächlich eine Funktion gegeben ist.

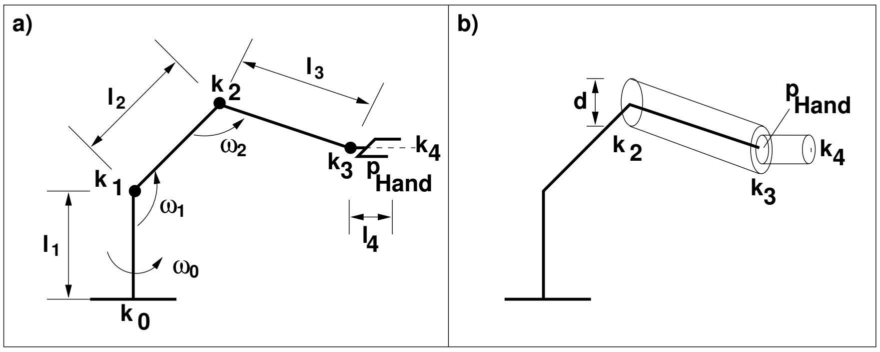

Abbildung 5.2: a) Die Gelenkpunkte $\boldsymbol{k}_i$ lassen sich aus den bekannten Segmentlängen l_i sowie den Gelenkwinkeln ω_i berechnen. b) Die auf Hindernispunkte zu überprüfenden Volumina des letzten Roboterarmsegmentes und der Hand.(Eingezeichnet sind nur die für die Position der Roboterhand notwendigen Gelenke.)

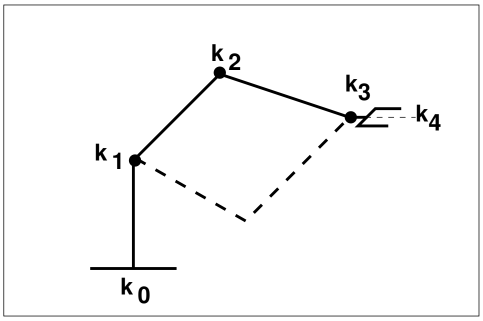

Abbildung 5.3: Mehrdeutigkeiten der Gelekpositionen $\boldsymbol{k}_0, \boldsymbol{k}_1, \boldsymbol{k}_2, \boldsymbol{k}_3, \boldsymbol{k}_4$ für eine Handposition $\boldsymbol{p}_{Hand}$.

Zur Implementierung der Funktion Pos sind die Winkel der Gelenke notwendig, die entweder aus einer dreidimensionalen Roboterhandposition durch inverse Kinematik oder einem neuronalen Netz berechnet werden. Die Implementierung der Funktion Pos wird in Kapitel 5.1.1 gezeigt. Der Durchmesser des Armsegmentes i sei mit d_i bezeichnet (Abbildung 5.2b). Die Projektion eines Hindernispunktes $\boldsymbol{h}$ auf die Gerade, die das Roboterarmsegment $\boldsymbol{k}_i, \boldsymbol{k}_{i+1}$ definiert, ist gegeben durch:

$$Proj(\boldsymbol{h}, \boldsymbol{k}_i, \boldsymbol{k}_{i+1}) = \frac{(\boldsymbol{h} - \boldsymbol{k}_i)(\boldsymbol{k}_{i+1} - \boldsymbol{k}_i)}{(\boldsymbol{k}_{i+1} - \boldsymbol{k}_i)^2}(\boldsymbol{k}_{i+1} - \boldsymbol{k}_i) \tag{5.2}$$

Für den uns interessierenden Fall, daß $Proj(\boldsymbol{h}, \boldsymbol{k}_i, \boldsymbol{k}_{i+1})$ auf der Strecke $\overline{\boldsymbol{k}_i \boldsymbol{k}_{i+1}}$ liegt, wird der Abstand eines Punktes $\boldsymbol{h}$ zur Strecke $\overline{\boldsymbol{k}_i \boldsymbol{k}_{i+1}}$ mit

$$L\left(\boldsymbol{h}, \boldsymbol{k_i}, \boldsymbol{k_{i+1}}\right) \stackrel{Def}{=} \|\boldsymbol{h} - Proj(\boldsymbol{h}, \boldsymbol{k_i}, \boldsymbol{k_{i+1}})\| \tag{5.3}$$

berechnet. Nun läßt sich für eine Handposition $\boldsymbol{p}_{Hand}$ testen, ob sie das Roboterarmsegment mit den Gelenkpunkten $\boldsymbol{k_i}, \boldsymbol{k_{i+1}}$ so positioniert, daß Hindernispunkte $\boldsymbol{h} \in H$ in das durch den Abstandsparameter d_i bestimmte Volumen des Armsegmentes fallen:

$$Test^H_{i,i+1}\left(\boldsymbol{p}_{Hand}\right) \stackrel{Def}{=} \begin{cases} 1 \quad \text{falls} & \exists\, \boldsymbol{h} \in H : L\left(\boldsymbol{h}, \boldsymbol{k_i}, \boldsymbol{k_{i+1}}\right) < \frac{d_i}{2} \\ & \wedge \quad Pos(\boldsymbol{p}_{Hand}) = (\boldsymbol{k_0}, \boldsymbol{k_1}, \boldsymbol{k_2}, \boldsymbol{k_3}, \boldsymbol{k_4}) \\ 0 \quad \text{sonst} & \end{cases} \tag{5.4}$$

für $i \in \{0, \ldots, 3\}$. Abhängig von der Anzahl der einzubeziehenden Roboterarmsegmente läßt sich nun die Zuordnung eines Punktes zur erweiterten Menge der Hindernispunkte definieren. Wird z.B. nur das letzte Segment berücksichtigt, lautet die erweiterte Definition der Menge der Hindernispunkte:

$$H' = \{\ \boldsymbol{p} \in \mathbb{R}^3 \ \mid \ Test^H_{2,3}(\boldsymbol{p}) = 1\ \} \tag{5.5}$$

wobei die in H' enthaltenen Punkte Positionen der Roboterhand ($\boldsymbol{k}_3$) beschreiben. In Kapitel 6.3 wird gezeigt, daß die Berechnung von H' für eine lokale Umgebung der aktuellen Position ausreichend ist. Zur Überprüfung des durch die Roboterhand eingenommenen Volumens läßt sich der Test (5.4) ebenfalls auf die Hand anwenden, denn der Anfangspunkt des zur Hand zählenden Segmentes ($\boldsymbol{k}_3$) ist bekannt. Der mit $\boldsymbol{k}_4$ bezeichnete Endpunkt des Handvolumens läßt sich mit der bekannten Handorientierung berechnen. Eine zweite Möglichkeit, das Handvolumen zu approximieren, ist z.B. durch einen Quader gegeben, der die Maße der Hand sowie die auf der Hand montierte Kamera berücksichtigt. Die erweiterte Definition der Menge der Hindernispunkte basiert dann zusätzlich auf den Positionen, für die der Quader Hindernispunkte (der alten Definition) enthält. Damit ist die Neudefinition des Hindernisbegriffes gegeben, zur praktischen Anwendung dieser Definition fehlt jedoch noch die Implementierung der Funktion Pos zur Berechnung der Gelenkpositionen.

5.1.1 Berechnung der Gelenkpositionen

Die Positionen der Gelenke im Roboterbasis- bzw. Weltkoordinatensystem sind mit den Gelenkwinkeln und dem Wissen um die Segmentlängen schrittweise zu bestimmen. Dazu seien die Koordinatensysteme $K_0, \ldots, K_3$ betrachtet, deren Ursprünge P_{K_iORG} in den korrespondierenden Gelenkpunkten $\boldsymbol{k_i}$ liegen und deren Ausrichtung durch die Gelenkwinkel ω_0, ω_1 rund ω_2 gegeben ist. Im folgenden sei mit ${}^K\boldsymbol{p}$ die Darstellung eines Punktes $\boldsymbol{p}$ in

Koordinaten des Koordinatensystems K, bezeichnet. Dann ist die Transformation eines Punktes ${}^{K_{i+1}}\boldsymbol{p}$ des Koordinatensystems K_{i+1} zu seiner Darstellung im Koordinatensystem K_i mit

$$ {}^{K_i}\boldsymbol{p} \;=\; {}^{K_i}_{K_{i+1}}R \;\; {}^{K_{i+1}}\boldsymbol{p} \;+\; {}^{K_i}\boldsymbol{p}_{K_{i+1}ORG} \tag{5.6} $$

gegeben [38]. Auf der Grundlage homogener Koordinaten lassen sich die beiden Transformationen Rotation und Translation der Gleichung (5.6) zusammenfassen:

$$ {}^{K_i}_{K_{i+1}}T = \left(\begin{array}{ccc|c} & {}^{K_i}_{K_{i+1}}R & & {}^{K_i}\boldsymbol{p}_{K_{i+1}ORG} \\ \hline 0 & 0 & 0 & 1 \end{array}\right) \tag{5.7} $$

Der translatorische Anteil der Transformation ${}^{K_i}_{K_{i+1}}T$ ist bekannt: ${}^{K_i}\boldsymbol{p}_{K_{i+1}ORG} = (0, 0, l_{i+1})^{\mathrm{T}}$. Der Rotationsteil der Transformation ${}^{K_i}_{K_{i+1}}T$ läßt sich aus ω_i berechnen. Damit ist die Bestimmung einer Gelenkpositionen $\boldsymbol{k_i}$ im Weltkoordinatensystem durch schrittweise Überführung des der Gelenkposition $\boldsymbol{k_i}$ entsprechenden Ursprungs des Koordinatensystems ${}^{K_i}\boldsymbol{p}_{K_iORG}$ durch die Koordinatensysteme der Gelenkpunkte möglich. Im Falle des uns interessierenden Armsegmentes berechnen sich die Gelenkpunkte $\boldsymbol{k_2}$ und $\boldsymbol{k_3}$ durch wiederholte Transformationen mit

$$ \boldsymbol{k}_3 \;=\; {}^{K_0}\boldsymbol{p}_{K_3ORG} \;=\; {}^{K_{WK}}_{K_0}T \;\; {}^{K_0}_{K_1}T \;\; {}^{K_1}_{K_2}T \;\; {}^{K_2}_{K_3}T \;\; {}^{K_3}\boldsymbol{p}_{K_3ORG} \tag{5.8} $$

und analogem Vorgehen für $\boldsymbol{k}_2$. Hierbei ist ${}^{K_3}\boldsymbol{p}_{K_3ORG} = \vec{\mathbf{0}}$. Mit K_{WK} ist das Weltkoordinatensystem bezeichnet. Die Transformation vom Koordinatensystem K_0 zum Weltkoordinatensystem beinhaltet lediglich die Rotation ω_0, während die Transformation von K_3 zu K_2 lediglich aus der durch l_3 gegebenen Translation besteht. Mit den Größen $l_1, \ldots, l_4$ und $\omega_0, \ldots, \omega_2$ lassen sich die gesuchten Gelenkpositionen zur Implementation der Funktion Pos berechnen. Allerdings sind nur die statischen Größen l_i bekannt. Wie die Gelenkwinkel ω_i für eine aktuelle Position durch Inverse Kinematik berechnet werden können, ist Thema des folgenden Abschnitts.

5.1.2 Berechnung der Gelenkwinkel

Zwar unterstützt das verwendete Robotersystem RX90 die Inverse Kinematik auch von virtuellen, nicht angefahrenen Positionen des kartesischen Raumes, die Ausführungszeit der Abfrage von ca. 0.2 Sekunden genügt unseren Ansprüchen jedoch nicht, da diese Abfrage häufig benötigt wird. Aus diesem Grund erlernen wir die Inverse Kinematik neuronal, da die Abfragezeit des neuronalen Netzes mit ca. 0.002 Sekunden um den Faktor 100 schneller als die Systemabfrage ist. Mit der Bereitstellung der Inversen Kinematik ist die letzte Voraussetzung zur Vorhersage der Hindernisfreiheit des Robotervolumens mit Gleichung (5.4) für eine virtuelle Position gegeben.

Die Inverse Kinematik berechnet zu einer Roboterhandposition $\boldsymbol{p} = (X, Y, Z)^{\mathrm{T}}$ die Gelenkwinkel $\boldsymbol{\omega} = (\omega_0, \ldots, \omega_m)^{\mathrm{T}}$ des Roboters. Die zugehörige Funktion sei mit IK bezeichnet:

$$IK \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \begin{pmatrix} \omega_0 \\ \vdots \\ \omega_m \end{pmatrix} \tag{5.9}$$

wobei z.B. $m = 6$ für einen 6 DOF-Roboter gilt. Für die Implementierung der Inversen Kinematik mit Neuronalen Netzen verwenden wir die im Kapitel 4.1 vorgestellten RBF-Netze. Hierzu werden der Menge der über den in Frage kommenden Arbeitsbereich des Roboters äquidistant verteilten dreidimensionalen Handpositionen $\boldsymbol{p}_i$ die zugehörigen Gelenkwinkel $\boldsymbol{\omega}_i$ gegenübergestellt. Wegen der Äquidistanz der Trainingsvektoren ist kein Clustering notwendig. Das mit diesen Daten trainierte Netzwerk R_{IK} berechnet die Gelenkwinkel $\boldsymbol{\omega}'$:

$$R_{IK} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} \omega'_0 \\ \vdots \\ \omega'_m \end{pmatrix} \tag{5.10}$$

Nimmt man für einen Roboterarbeitsbereich der Größe $800mm \times 800mm \times 500mm$ $(X \times Y \times Z)$ eine Knotenverteilung des Netzes mit einem Abstand von $25mm$ an, beträgt die Anzahl der Hidden Units $33 \cdot 33 \cdot 21 = 22869$. Wie in Kapitel 4 dargestellt, ist das Training eines Netzwerks dieser Größe sehr aufwendig. Um die Trainingszeit zu reduzieren, greifen wir wieder auf die Technik des Netzsplittings zurück. Dazu wird das Gesamtnetz R_{IK} in Teilnetze R_{IKi} gespalten, deren Knoten jeweils in einer Ebene angesiedelt sind. Wir wählen dazu die (waagerechten) XY-Ebenen, deren vertikaler Abstand mit d gegeben sei. Da für ein solches Netzwerk mit der ihm zugeordneten Höhe der Z-Wert schon gegeben ist, besteht ein Eingabevektor nur noch aus den XY-Koordinaten, während die Ausgabedimension unberührt bleibt (Abbildung 5.4).

Die Berechnung eines Ausgabewertes des Netzsystems R_{IK} gestaltet sich wie folgt: Zunächst werden durch eine Wahlfunktion W zu einer Position $\boldsymbol{p} = (x, y, z)^{\mathrm{T}}$ die Nummer der höchsten Ebene (bezüglich der Z-Achse) aus einer Menge von N Ebenen bestimmt, die unterhalb von $\boldsymbol{p}$ liegt.

$$W(z) = \left\lfloor \frac{z}{d} \right\rfloor \tag{5.11}$$

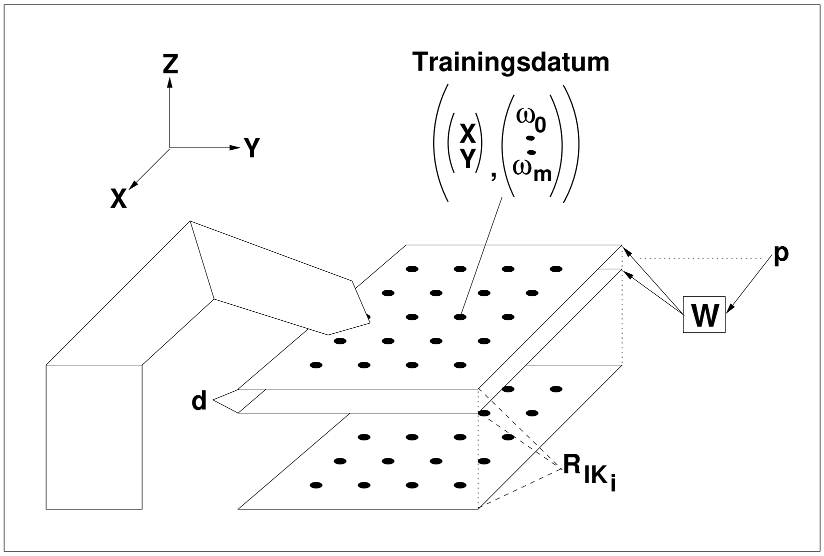

Abbildung 5.4: Lage der waagerechten Netze R_{IK_i} und Wahlfunktion zur Zuordnung der beiden Netze R_{IK_n} und $R_{IK_{n+1}}$ mit $W(\boldsymbol{p}) = n$, die über und unter der Z-Komponente eines Punktes $\boldsymbol{p}$ liegen.

Es ist $z \in [\ 0,\ (N-1)\cdot d)$ und $W(z) \in \{0, \ldots, N-2\}$. Mit $W(z)+1$ ist dann die Nummer der tiefsten Ebene bezeichnet, die oberhalb von $\boldsymbol{p}$ liegt.

Für die beiden Ebenen $n = W(z)$ und $n+1$ werden die Antworten $R_{IKn}(\boldsymbol{p})$ und $R_{IKn+1}(\boldsymbol{p})$ der zu diesen Ebenen korrespondierenden Netze berechnet. Diese werden in Abhängigkeit vom Abstand des Punktes $\boldsymbol{p}$ zu den beiden Ebenen n und $n+1$ linear interpoliert, um die durch die Aufteilung verlorene Generalisierung des Netzes zwischen den Ebenen zu simulieren, so daß sich die Antwort R_{IK} des Systems wie folgt ergibt:

$$R_{IK}\begin{pmatrix} x \\ y \\ z \end{pmatrix} \stackrel{Def}{=} R_{IKn}\begin{pmatrix} x \\ y \end{pmatrix} + \frac{|z-d\cdot n|}{d}\cdot\left(R_{IKn+1}\begin{pmatrix} x \\ y \end{pmatrix} - R_{IKn}\begin{pmatrix} x \\ y \end{pmatrix}\right) \qquad (5.12)$$

mit $n = W(z)$.

Die einzelnen Netzwerke R_{IKi} weisen, ausgehend von obigem Beispiel, eine Knotenzahl von 1089 auf, was ein Training in einem Bruchteil der Zeit des ursprünglichen Netzes ermöglicht (die Trainingszeit beträgt ca. 6 Minuten gegenüber einer geschätzten Trainingszeit von zwei Monaten für das Gesamtnetz, bezogen auf einem SUN 167MHz Prozessor). Da jedes Netzwerk R_{IKi} in kurzer Zeit trainiert werden kann, ist eine Halbierung des Abstandes der Netze von $d = 25mm$ auf $d = 12mm$ zur Steigerung der Genauigkeit der Netzantwort möglich. Für den Arbeitsbereich von $800mm \times 800mm \times 500mm$ (X $\times$ Y $\times$ Z) werden

41 Ebenen im Abstand von $d = 12.5mm$ mit Netzwerken belegt. Der Knotenabstand der Netze innerhalb einer Ebene beträgt $25mm$. Um den Fehler des Gesamtnetzes zu bestimmen, wird eine Referenzdatenmenge von Paaren $(\boldsymbol{p}, \boldsymbol{\omega})$ generiert, deren Punkte $\boldsymbol{p}$ mit einem Abstand von $1.5mm$ in Z-Richtung ($3mm$ in X- bzw. Y-Richtung) liegen. Die feine Diskretisierung ermöglicht die Bestimmung der maximalen Abweichung zwischen den Positionen der Knoten. Zu jeder der Testpositionen $\boldsymbol{p}$ im Arbeitsbereich liefert das Netzwerk einen Vektor der Winkelstellungen $\boldsymbol{\omega}$, der mit dem vom Robotersystem gelieferten Wert verglichen werden kann. Die Anwendung der Vorwärts-Kinematik (Gleichungen (5.6) und (5.8)) ermöglicht es, ein Winkeltupel wieder auf eine Position des Arbeitsbereiches abzubilden und mit der Eingabe des Netzes zu vergleichen. Der bezüglich des Netzparameters σ entstehende Fehler ist in Tabelle 9 aufgelistet.

σ	0.7	0.9	1.1	1.3	1.5
max. Fehler $\boldsymbol{\omega}_0$	0.98°	1.21°	0.74°	0.35°	11.05°
max. Fehler $\boldsymbol{\omega}_1$	0.99°	1.00°	0.89°	0.73°	5.80°
max. Fehler $\boldsymbol{\omega}_2$	1.44°	1.77°	1.09°	0.59°	22.98°
max Fehler x	7.68 mm	9.48 mm	5.79 mm	2.61 mm	62.43 mm
max Fehler y	12.14 mm	14.77 mm	9.62 mm	4.96 mm	160.41 mm
max Fehler z	8.00 mm	8.49 mm	7.93 mm	6.94 mm	64.64 mm
max. Positionierungsfehler	14.49 mm	17.69 mm	11.22 mm	7.65 mm	183.87 mm
mittl. Positionierungsfehler	3.14 mm	3.62 mm	3.27 mm	2.83 mm	28.66 mm

Tabelle 9: Fehler des Netzwerksystems

Während die durchschnittlichen Positionierungsfehler meist um 3 mm liegen, läßt sich anhand des maximalen Positionierungsfehlers ein optimaler Wert für σ ermitteln. Bei $\sigma = 1.3$ liegt der maximale Positionierungsfehler bei 7.65 mm, der durchschnittliche Positionierungsfehler liegt bei 2.83 mm. Der höchste, in nennenswertem Umfang vorkommende Positionierungsfehler liegt bei 6 mm, wie der Fehlerverteilung für $\sigma = 1.3$ in Abbildung (5.5) zu entnehmen ist. Der Fehler ist bei Bedarf durch Verfeinerung des Knotengitters in den XY-Ebenen sowie einer Erhöhung der Anzahl dieser Ebenen unter Erhöhung des Rechenaufwandes zu verkleinern.

Mit dem in diesem Kapitel geschaffenen erweiterten Begriff des Hindernispunktes ist es nun möglich, die Hinderniseigenschaften einer Szene in eine zur Roboternavigation geeignete Darstellung zu überführen. Um die Grundlagen zur Navigation zu schaffen, wird im folgenden Kapitel die Ermittlung eines hindernisfreien Zielpunkts beschrieben und anschließend der Begriff der Trajektorie definiert.

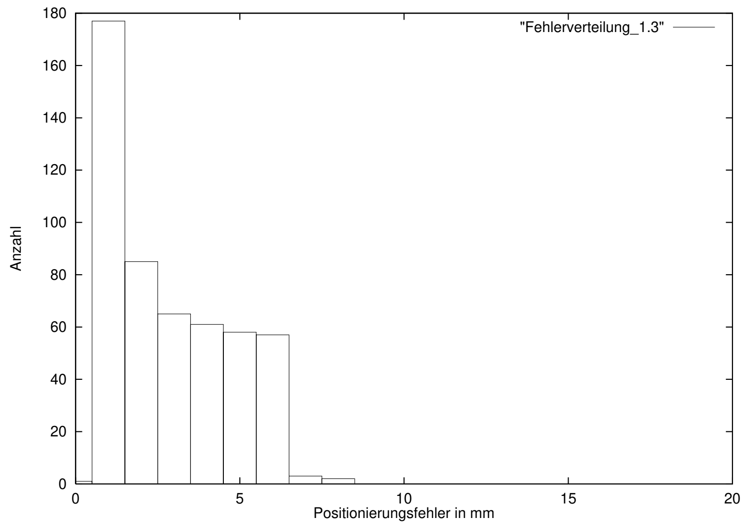

Abbildung 5.5: Verteilung des Positionierungsfehlers ($\sigma = 1.3$) der Referenzdatenmenge von 509 Wertepaaren

5.2 Ziel und Zugriffsposition

Zu einem vorgegebenen Zielpunkt $\boldsymbol{z} \in \mathbb{R}^3$ sind mehrere Zugriffspositionen der Roboterhand möglich (Abbildung 5.6). Damit ist der vom Roboter anzusteuernde Zugriffspunkt $\boldsymbol{r_z}$, der durch die Position der Roboterhand gegeben ist (die Hand ist notwendigerweise zum Zielpunkt orientiert), keineswegs eindeutig. Deshalb ist vor Generierung der Trajektorie der Zugriffspunkt $\boldsymbol{r_z}$ in Abhängigkeit von den Hindernispunkten H und dem Zielpunkt $\boldsymbol{z}$ zu bestimmen.

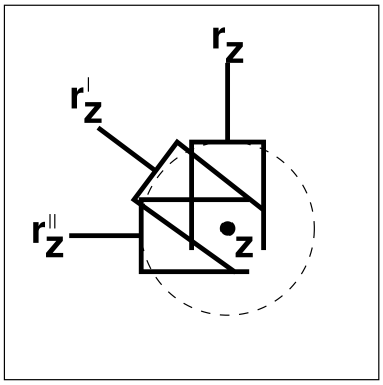

Abbildung 5.6: Zu einem Zielpunkt existieren mehrere Möglichkeiten des Zugriffs.

Zu diesem Zweck muß das Volumen, das die Roboterhand beim Zugriff auf den Zielpunkt einnimmt, als hindernisfrei überprüft worden sein. Der von der Roboterhand eingenommene Raum läßt sich durch einen Zylinder oder einen Quader gut annähern (Kapitel 5.1). Die Menge der Zugriffspositionen um den Zielpunkt lassen sich in Polarkoordinaten (ϕ, θ, r) beschreiben, wobei der Abstand r des Roboterhandgelenkes vom Zielpunkt durch die Länge der Roboterhand festgelegt ist. Die zum Zugriff auf den Zielpunkt geeigneten Roboterhandkoordinaten lassen sich durch die zwei Parameter ϕ und θ der Polarkoordinaten als Fläche interpretieren (Abbildung 5.7). Für eine Diskretisierung liefert diese eine Besetzungsmatrix, anhand der für jede Zugriffsposition gespeichert werden kann, ob ein kollisionsfreier Zugriff möglich ist.

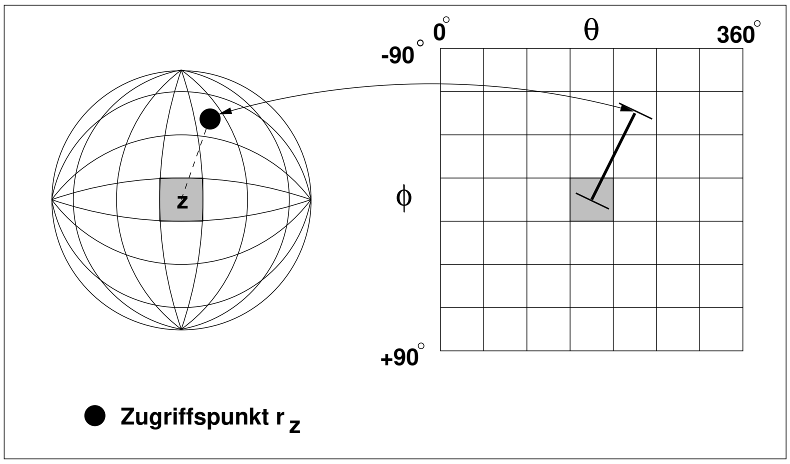

Abbildung 5.7: Polarkoordinaten ermöglichen die Schaffung einer Besetzungsmatrix bezüglich der Zugriffspositionen und Anwendung einer Metrik zur Bestimmung der nächsten möglichen Zugriffsposition

Innerhalb dieser Besetzungsmatrix ist die euklidische Metrik anwendbar. Damit ist es möglich, ausgehend von der initialen Zugriffsposition $(\phi, \theta) = (0, 0)$ diejenige auszuwählen, deren Abstand von der initialen Zugriffsposition minimal ist (und damit auch der einzustellende Winkel der Roboterhand).

Zur Markierung einer Zugriffsposition der Besetzungsmatrix sind die folgenden Tests durchzuführen:

1. Die Kollisionsfreiheit des zur Hand führenden Roboterarmsegmentes hinsichtlich der Zugriffsposition $\boldsymbol{r_z}$ läßt sich nach Gleichung (5.4) mit $Test^H_{2,3}(\boldsymbol{r_z})$ prüfen. Bei Bedarf werden weitere Armsegmente einbezogen.

2. Die Kollisionsfreiheit des Handvolumens zu der gegebenen Zugriffsposition $\boldsymbol{r_z}$ wird mit $Test^H_{3,4}(\boldsymbol{r_z})$ überprüft. Alternativ wird das Handvolumen nicht mit einem Zylinder, sondern mit einem Quader approximiert[1].

[1]In diesem Fall werden alle Hindernispunkte gemäß der Handrotationsparameter $(-\phi, -\theta)$ rotiert, so daß

Wird einer der beiden Tests nicht bestanden, wird das entsprechende Element der Besetzungsmatrix als besetzt gekennzeichnet. Damit bildet die Berechnung eines Zielpunktes $\boldsymbol{r}_z$ in Roboterhandkoordinaten aus einem Zielpunkt $\boldsymbol{z}$, der mit den Roboterfingern erreicht werden soll, die Basis zur Generation einer Trajektorie zu dem Zielpunkt $\boldsymbol{r}_z$.

5.3 Definition der Trajektorie

Nachdem nun die Gestalt des Raumes, in dem die Trajektoriegeneration ausgeführt werden soll, festgelegt ist, kann der Begriff der Trajektorie genau spezifiziert werden. In dem hier vorgestellten Manipulatornavigationssystem sind die Parameter Geschwindigkeit und Beschleunigung bezüglich eines Trajektorieteilstücks fest gewählt. Die Eingaben des Systems sind:

- aktuelle Position $\boldsymbol{p}$ und Orientierung $\boldsymbol{o}$ der Roboterhand
- Zugriffsposition $\boldsymbol{r}_z$
- Hindernispunkte $\boldsymbol{h} \in H'$

Die folgende, zu berechnende Ausgabe beschreibt das nächste Trajektorieteilstück:

- Positionsänderung $\dot{\boldsymbol{p}}$
- Orientierungänderung $\dot{\boldsymbol{o}}$ der Roboterhand

Dabei ist es zweckmäßig, entweder die Orientierung der Hand von der Bewegungsrichtung abhängig zu machen, indem die Hand bezüglich ihrer Orientierung der Bewegung folgt oder die Orientierung der Hand in Abhängigkeit von der berechneten Zugriffsposition $\boldsymbol{r}_z$ festzulegen. Die erste Variante minimiert das während der Bewegung des Roboters von ihm eingenommene Volumen, die zweite Variante verringert wegen der fehlenden Freiheitsgrade der Rotation der Roboterhand die Laufzeit der Berechnung der Trajektorie, benötigt jedoch ein größeres Volumen während der Bewegung. Während das erste Verfahren eine allgemeine Lösung bietet, zeigte sich in der Praxis, daß das Verfahren der fixen Handorientierung in der Regel ausreicht[2]. Die Trajektoriegenerationsfunktion TG bildet damit die Position $\boldsymbol{p} \in \mathbb{R}^3$ und die Menge von Hindernispunkten H' sowie den Zielpunkt $\boldsymbol{z}$ auf die auszuführende Bewegung der Roboterhand ab:

sich die Überprüfung, ob Hindernispunkte im Handvolumen liegen, auf den einfachen, koordinatenweisen Vergleich der Hindernispunkte mit den Quadermaßen beschränkt.

[2]Es lassen sich jedoch Szenen konstruieren, für die eine Trajektorie nur dann erfolgreich ist, wenn auch die Orientierung der Hand Gegenstand der Regelung ist

$$\dot{\boldsymbol{p}} = TG\,(\boldsymbol{p}, H', \boldsymbol{z}) \tag{5.13}$$

und setzt gegebenenfalls die Orientierung $\boldsymbol{o}$ der Hand gemäß der Bewegungsrichtung,

$$\boldsymbol{o} = t\,(\dot{\boldsymbol{p}}) \tag{5.14}$$

wobei t eine Transformation des Richtungsvektors $\dot{\boldsymbol{p}}$ in die entsprechenden Positionierungswinkel der Hand ist. Damit definieren wir eine Trajektorie Q bezüglich eines Startpunktes $\boldsymbol{q}_0 \in \mathbb{R}^3$ als (zeitliche) Folge dreidimensionaler, die Position der Roboterhand im Weltkoordinatensystem beschreibenden Vektoren:

$$Q_{\boldsymbol{q}_0} \stackrel{Def}{=} \langle \boldsymbol{q}_0, \boldsymbol{q}_1, \ldots \rangle\,, \quad \boldsymbol{q}_{i+1} = \boldsymbol{q}_i + TG\,(\boldsymbol{q}_i, H', \boldsymbol{z}) \tag{5.15}$$

Das k-te Element der Trajektorie $Q_{\boldsymbol{q}}$ sei mit $Q_{\boldsymbol{q}}^k$ bezeichnet.

Eine Trajektorie $Q_{\boldsymbol{q}}$ erreicht den Zielpunkt $\boldsymbol{z}$, wenn gilt:

$$\exists\, k \in \mathbb{N} \;:\; Q_{\boldsymbol{q}}^k = \boldsymbol{z} \tag{5.16}$$

Das bedeutet, daß eine Trajektorie ständig auf die Eigenschaft (5.16) überprüft werden muß. Die Implementierung der Funktion TG wird durch einen Algorithmus auf der Basis eines Potentialfeldes vorgenommen, der aus diesem ein Vektorfeld generiert.

Kapitel 6

Vektorfeldgeneration durch Potentialfelder

Der im Kapitel 5.3 eingeführte Begriff der Trajektorie beschreibt eine Folge von Wertepaaren (Zustandsänderungen des Roboters), die mit Fortschreiten des Parameters Zeit durch ein dynamisches System erzeugt werden. Dieses zu erstellende System liefert in Abhängigkeit vom aktuellen Systemzustand die auszuführende Zustandsänderung (Gleichung 5.13). Die in das System einzubindenen Ziele sind Hindernisvermeidung und Zielansteuerung. Eine geeignete Repräsentation eines solchen dynamischen Systems ist das Vektorfeld, das einerseits jedem Systemzustand (Position der Roboterhand) eine Zustandsänderung (Bewegung der Roboterhand) zuordnet, andererseits die Ziele Hindernisvermeidung und Zielansteuerung durch geeignete Vektoranordnungen, die Gegenstand der folgenden Untersuchungen sein werden, ermöglicht. Grundlage der Vektorfeldgeneration ist eine Potentialfunktion, deren Gradient mit *Repellor* bezeichnet wird, da er bezüglich der Position zu seinem Zentrum abweisende Eigenschaften besitzt. Das zur Hindernisvermeidung und Zielansteuerung gesuchte Vektorfeld wird durch eine Kombination verschiedener Repelloren sowie des Zielpunktes definiert. Nach der Betrachtung eines skalierbaren Repellors werden die Probleme der auf Potentialfeldern basierenden Algorithmen sowie deren Lösungen präsentiert. Das Verfahren zur Lösung dieser Probleme ist das sogenannte *Monitoring*, das Probleme detektiert und Lösungsstrategien initiiert.

6.1 Die Potentialfunktion und ihr Gradient

Schon im Zusammenhang mit RBF-Netzen in Kapitel 4.1 wurden radiale Basisfunktionen vorgestellt, deren Antwort von dem Abstand eines Zentrums $\boldsymbol{\mu}$ zu einem Ortsvektor $\boldsymbol{x}$ abhängt. Die hier gewählte Potentialfunktion entspricht bis auf Vorzeichen und Faktoren dieser radialen Basisfunktion. Sie lautet:

$$Pot\,(\boldsymbol{x}, \boldsymbol{\mu}) = -\frac{\exp\left(-\|\boldsymbol{x} - \boldsymbol{\mu}\|^2\right)}{2} \tag{6.1}$$

Das durch diese Funktion generierte Potentialfeld ist die (auf den Kopf gestellte) Gauß'sche Glockenkurve, deren Gradient das gewünschte Vektorfeld liefert. Da der Gradient der Potentialfunktion Vektoren um sein Zentrum $\boldsymbol{\mu}$ definiert, die von diesem Zentrum wegweisen, gehört diese Funktion zur Klasse der *Repelloren* [19],[20]. Der Repellor ist durch

$$R\,(\boldsymbol{x}, \boldsymbol{\mu}) \stackrel{Def}{=} \frac{\partial Pot\,(\boldsymbol{x}, \boldsymbol{\mu})}{\partial \boldsymbol{x}} = (x - \mu) \exp\left(-\|\boldsymbol{x} - \boldsymbol{\mu}\|^2\right) \tag{6.2}$$

definiert. Für den eindimensionalen Fall ist die Potentialfunktion sowie ihr Gradient (Repellor) in Abbildung 6.1 dargestellt. Für einen zum Zentrum ($\boldsymbol{\mu} = 0$) positiven Wert x gibt der Repellor einen positiven Positionsänderungswert $\dot{x}$ zurück, d.h. der Positionwert wird vergrößert, die Entfernung zum Zentrum steigt. Positionswerte $x < 0$ ergeben einen negativen Positionsänderungswert, auch hier vergrößert sich der Abstand zum Zentrum.

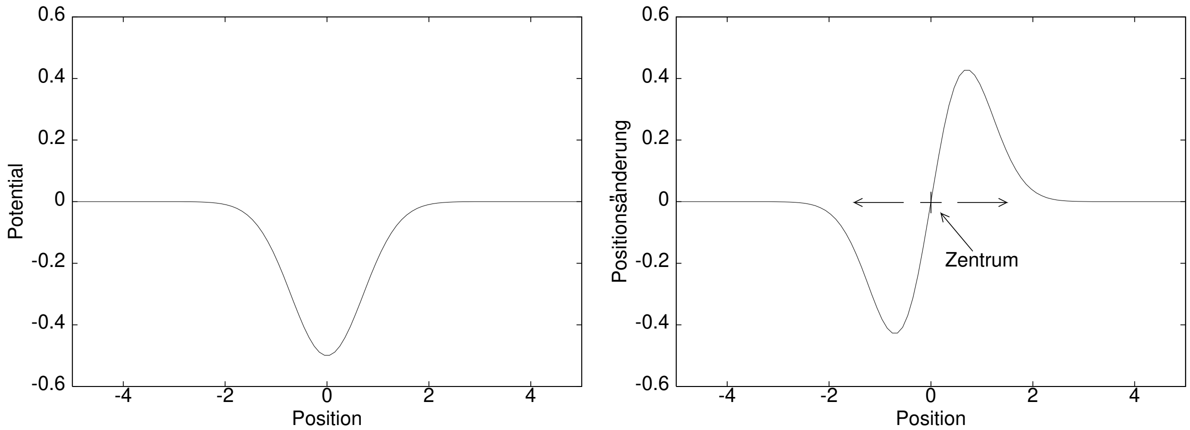

Abbildung 6.1: Links: Die Potentialfunktion Pot um das Zentrum 0. Rechts: Der Gradient R der Potentialfunktion definiert einen Repellor.

6.2 Hindernisvermeidung und Zielansteuerung

Die in Gleichung (6.2) gegebenen Vektorfelder sind geeignet, die Hindernisvermeidung lokal an einer Position $\boldsymbol{x}$ zu repräsentieren (Abbildung 6.2, links). Zum Zweck der globalen Hindernisvermeidung wird jeder Hindernispunkt $\boldsymbol{h} \in H'$ als Zentrum eines Repellors $R\,(\boldsymbol{x}, \boldsymbol{h})$

betrachtet. Die Summe aller durch die Repelloren definierten Vektorfelder generiert das Repellorfeld $RF_{H'}$:

$$RF_{H'}\left(\boldsymbol{x}\right) = \sum_{h \in H'} R\left(\boldsymbol{x}, \boldsymbol{h}\right) \tag{6.3}$$

Das Repellorfeld $RF_{H'}$ wird für den Zielpunkt $\boldsymbol{z}$, der mit Attraktor bezeichnet sei, mit einem Attraktorfeld $AF_{\boldsymbol{z}}$

$$AF_{\boldsymbol{z}}\left(\boldsymbol{x}\right) \;=\; \alpha \, \frac{(z - x)}{\|z - x\|} \tag{6.4}$$

überlagert. Die Vektoren des Feldes $AF_{\boldsymbol{z}}$ sind auf eine Länge von $\alpha \in \mathbb{R}$ skaliert und weisen zum Zielpunkt z (Abbildung 6.2, rechts: Lokales Umfeld des mit einem Attraktorfeld überlagerten Repellorfeldes, der Attraktor liegt links unten, außerhalb des Bildes.). Es entsteht das Gesamtvektorfeld $V_{H',\boldsymbol{z}}$:

$$V_{H',\boldsymbol{z}}\left(\boldsymbol{x}\right) = RF_{H'}\left(\boldsymbol{x}\right) + AF_{\boldsymbol{z}}\left(\boldsymbol{x}\right) \tag{6.5}$$

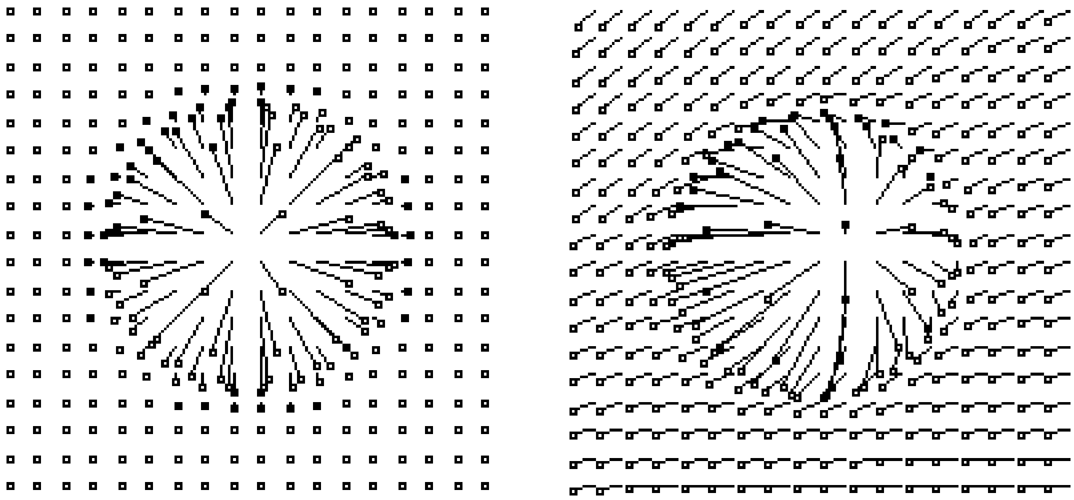

Abbildung 6.2: Zweidimensionale Repelloren. Links: Das durch einen Repellor R erzeugte Vektorfeld RF. Rechts: Gleiches, mit einem Attraktorfeld AF überlagertes Vektorfeld. Das Zentrum $\boldsymbol{\mu}$ des Repellors liegt jeweils im Bildmittelpunkt.

Damit implementiert das Vektorfeld V gerade die Trajektoriengenerationsfunktion TG:

$$TG\left(\boldsymbol{x}, H', \boldsymbol{z}\right) \stackrel{Def}{=} V_{H',\boldsymbol{z}}\left(\boldsymbol{x}\right) \tag{6.6}$$

6.3 Skalierung der Repelloren

Zur Generation einer Trajektorie ist ein Mindestabstand jedes Trajektorieelementes zu den Repellorzentren wünschenswert. Daher ist es sinnvoll, die „Breite“ der Funktion R, d.h. die Länge des Intervalls, für das sich die Repellorantworten von Null nennenswert unterscheiden, beeinflussen zu können und damit den Abstand der Trajektorie zu den Hindernissen zu definieren. Eine Skalierung des Repellors in Breite (Parameter b) und Höhe (Parameter h), die dieses leistet, ist durch die Neudefinition des Repellors mit

$$R_{skal}\left(\boldsymbol{x}, \boldsymbol{\mu}\right) \stackrel{Def}{=} h \cdot b \cdot (x - \mu) \exp\left(-\|b\left(\boldsymbol{x} - \boldsymbol{\mu}\right)\|^2\right) \tag{6.7}$$

gegeben mit $b, h \in \mathbb{R}$. Mit den Parametern b, h und α steht ein Werkzeug zur Beschreibung des trajektoriengenerierenden Systems zur Verfügung. Allerdings ist der Attraktorwert α nur unter der Berücksichtigung einer oberen, durch die Parameter b und h bestimmten Schranke zu wählen. Betrachtet man die Länge der Vektorantworten der Repelloren, so definiert jeder Attraktorwert α zu gewählten Werten b und h ein Abstandsintervall I_{dist} bezüglich des Repellors. Für jeden Punkt, der einen Abstand vom Repellorzentrum aus dem Intervall I_{dist} aufweist, ist die Antwort des Repellors größer als der Attraktorwert (Abbildung 6.3). Wäre nun die Länge dieses Intervalls kleiner als der Wert α, so wäre ein „Überspringen“ der Repellorvektoren möglich. Für einen definierten Repellor, also gegebenen Parametern b und h, läßt sich der größtmögliche Wert von α wie folgt automatisch bestimmen:

$$\alpha \stackrel{Def}{=} \arg\min_{\alpha}\{\, ||I_{dist}| - \alpha| \quad | \quad \forall x \in I_{dist} : \|R\left(x\right)\| > \alpha\} \tag{6.8}$$

An dieser Stelle wird deutlich, daß es zur Berechnung des Vektors V zu einer gegebenen Position $\boldsymbol{x}$ ausreicht, lediglich die Hindernisse $h \in H'$ in die Berechnung der Gleichung (6.5) einzubeziehen, deren Abstand zur Position $\boldsymbol{x}$ kleiner als 2α ist, da der Funktionswert des Repellors an der Stelle 2α so klein ist, daß er als Null anzusehen ist (Abbildung 6.3).

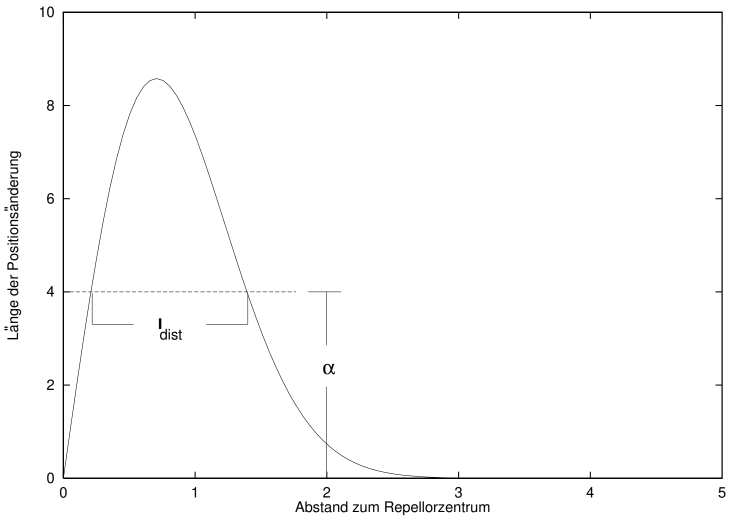

Abbildung 6.3: Länge der Vektoren der Repellorantwort. Der Attraktorwert α definiert ein Intervall $[a, b]$, dessen Länge der Wert α selber nicht überschreiten darf, um den Repellor nicht zu „überspringen“.

6.4 Statische und dynamische Szenen

Bisher wurde der Fall betrachtet, daß die Umgebung des Systems als auch das Ziel statisch sind. Unterliegt die Position der Hindernisse $h \in H'$ oder des Ziels $\boldsymbol{z}$ einer Änderung in Abhängigkeit von der Zeit, so ist das Vektorfeld $V_{H',\boldsymbol{z}}(\boldsymbol{x})$ für eine Diskretisierung der Zeit jeweils neu zu berechnen, so daß die Parameter H' und $\boldsymbol{z}$ zu Variablen des Verfahrens der Vektorfeldgeneration werden:

$$V(H', \boldsymbol{z}, \boldsymbol{x}) = RF(H', \boldsymbol{x}) + AF(\boldsymbol{z}, \boldsymbol{x}) \tag{6.9}$$

Mit dieser Modifikation ist die Navigation in Szenen dynamischer Natur möglich. Ein weiteres dynamisches Element wird in Kapitel 6.7 eingeführt, wo es um die Einführung von virtuellen Repelloren, d.h. Repelloren ohne entsprechende Hindernisse in der realen Szene zur Markierung von zu umgehenden Positionen geht.

6.5 Probleme der Potentialfeldalgorithmen

Das Vektorfeld $V = R + A$ bietet bereits die Fähigkeit, hindernisumgehende Trajektorien zu bilden (Abbildung 6.4). Allerdings gilt das Grundproblem der Potentialfeldalgorithmen auch hier: Lokale Minima „fangen“ eine Trajektorie an einem Punkt ein, an dem das Vektorfeld, bedingt durch ein Aufheben der Attraktor- und Repellorkräfte, den Wert $\vec{0}$ annimmt (Abbildung 6.5). Ein Punkt $\boldsymbol{p}_{min}$ ist ein *lokales Minimum*[1], wenn gilt:

$$\begin{aligned} & V(\boldsymbol{p}_{min}) = \vec{0} \\ \wedge \quad & \exists\, \delta \in \mathbb{R} . \forall \boldsymbol{q} \in \mathbb{R}^3 . \, |\boldsymbol{p}_{min} - \boldsymbol{q}| < \delta \, . \exists k \in \mathbb{N} \; : \; Q_{\boldsymbol{q}}^k = \boldsymbol{p}_{min} \end{aligned} \tag{6.10}$$

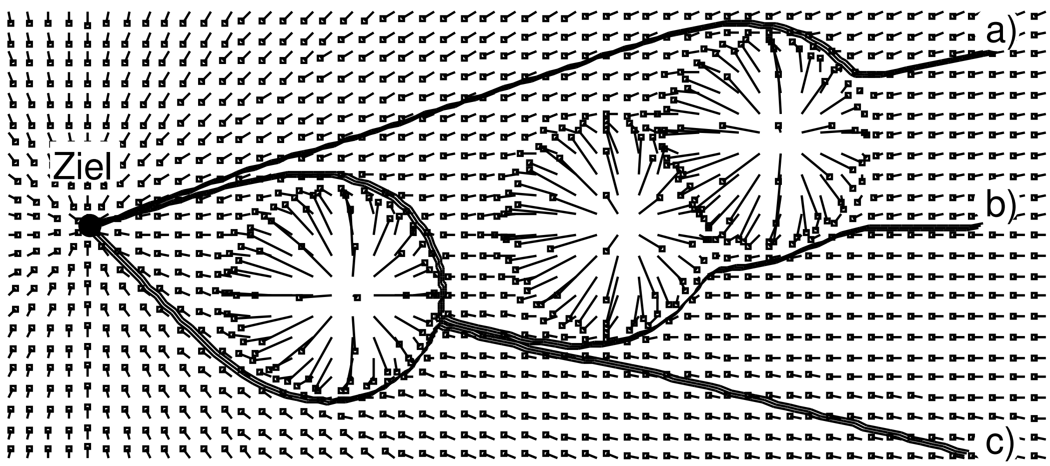

Abbildung 6.4: Hindernisvermeidung durch Überlagerung von Repellor- und Attraktorfeldern. Die Startpunkte a), b) und c) führen zu verschiedenen Trajektorien, die jeweils unter der Prämisse der Hindernisvermeidung den kürzesten Weg zum Zielpunkt darstellen (siehe b) und c)).

Allerdings gilt (6.10) nur im kontinuierlichen Fall des Vektorfeldes. In der Praxis liegt V jedoch in einer Diskretisierung vor, so daß sich die Repellor- und Attraktorkräfte an der Stelle $\boldsymbol{p}_{min}$ nicht genau aufheben, so daß $V(\boldsymbol{p}_{min})$ lediglich einen kleinen Wert annimmt. Dies kann in periodische Bewegungen münden. Diesem Problem wird durch das sogenannte Monitoring begegnet, bei dem eine Instanz (der Monitor) über den Fortlauf der Trajektoriengeneration wacht. Bevor der Monitor geschildert wird, sei noch ein Fall betrachtet, bei dem eine Trajektorie ebenfalls in einem potentialfreien Punkt anhält, jedoch ohne das Vorliegen eines lokalen Minimums: In diesem Fall zeigt für eine Trajektorie $Q_{\boldsymbol{q}_0} = \langle \boldsymbol{q}_0, \ldots, \boldsymbol{q}_k, \ldots \rangle$ der vom Repellor erzeugte Vektor der Richtung $(\boldsymbol{q}_k - \boldsymbol{\mu})$ in die Gegenrichtung des Attraktorvektors $(\boldsymbol{z} - \boldsymbol{q}_k)$, so daß diese beiden Vektoren linear abhängig sind. Dies hat zur Folge, daß der vom Repellor erzeugte Vektor den vom Attraktor produzierten Vektor aufhebt. Es gilt:

[1]Das lokale Minimum ist ein Potentialminimum, das sich auf die Länge der Gradientenvektoren auswirkt.

$$\exists k \in \mathbb{N} \ : \ R\left(Q^k_{\boldsymbol{q}_0}\right) = \vec{0} \ \wedge \ \left(Q^k_{\boldsymbol{q}_0} - \boldsymbol{\mu}\right) \times \left(\boldsymbol{z} - Q^k_{\boldsymbol{q}_0}\right) = \vec{0} \tag{6.11}$$

Im folgenden sei $Q^k_{\boldsymbol{q}_0}$ mit q_k bezeichnet. Die Wahrscheinlichkeit, daß das Prädikat (6.11) wahr ist, ist im kontinuierlichen Fall zwar gleich Null, im diskretisierten Vektorfeld tritt dieser Fall jedoch durchaus auf, wie Abbildung 6.6 zeigt.

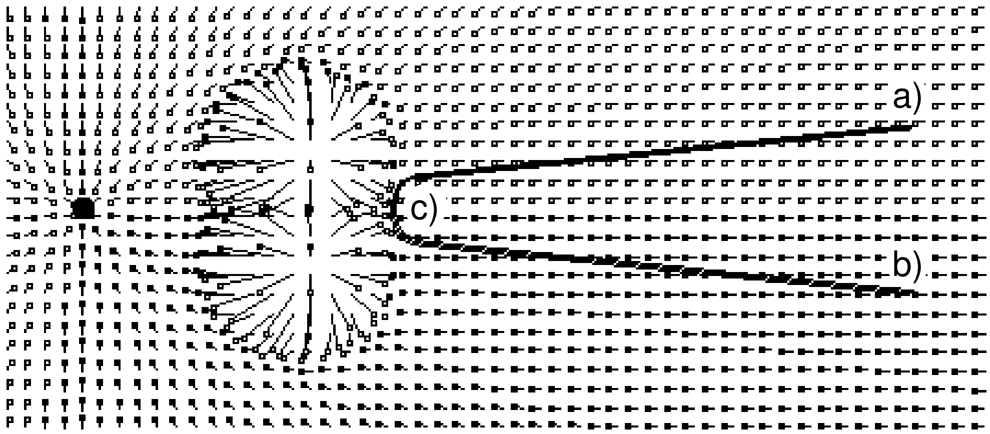

Abbildung 6.5: Beide Trajektorien (Startpunkt a) bzw. b)) laufen in ein lokales Minimum (Punkt c)), an dem sich Anziehungs- und Abstoßungskräfte durch die Punkt c) umlagernden Repelloren aufheben. Die Trajektorie erfährt in diesem Punkt keinen Fortschritt mehr.

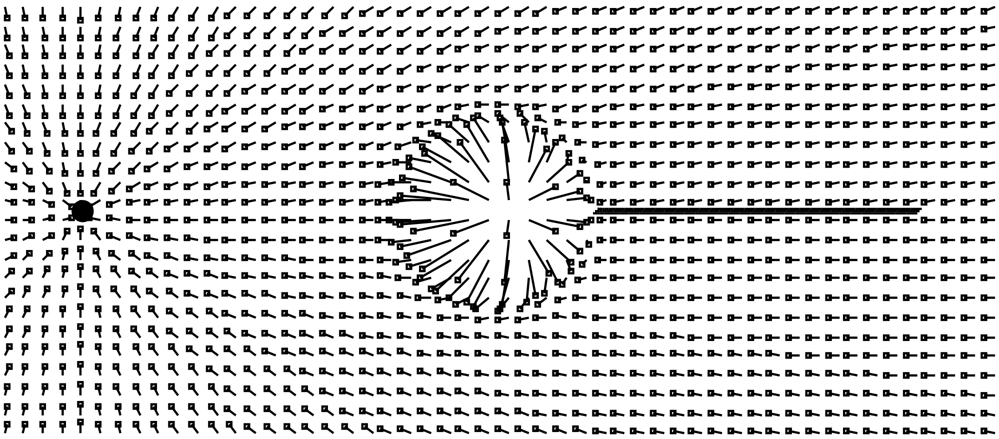

Abbildung 6.6: Liegen Hindernis und Ziel genau auf einer Geraden mit einer Position der Trajektorie, so existiert ein Punkt, an dem die Trajektorie keinen Fortschritt mehr erfährt.

6.6 Monitoring

Ziel des Monitoring ist es, das Eintreten der bestimmter Ereignisse zu detektieren, um geeingete Maßnahmen zu treffen. Dazu überwacht ein der Trajektoriegeneration übergeordneter Mechanismus die aktuelle Trajektorie. Der Abstand des letzten Punktes einer Trajektorie zum Zielpunkt ist z.B. Gegenstand des Monitorings, um das Erreichen des Zielpunktes zu detektieren. Dabei ist mit dem k-ten Punkt der Trajektorie $Q_{\boldsymbol{q}}$ ein Ziel erreicht, wenn $\|Q_{\boldsymbol{q}}^k - \boldsymbol{z}\| = 0$ gilt. Aus den schon erwähnten Gründen der Disketisierung und Rechenungenauigkeit muß dies jedoch nicht erfüllt sein. Daher ist eine Schranke zu setzen, die dieses Kriterium ersetzt:

$$Monitor_{Ziel}\left(Q_{\boldsymbol{q}}^k\right) \overset{Def}{\Leftrightarrow} \|Q_{\boldsymbol{q}}^k - \boldsymbol{z}\| < \epsilon\,, \quad \epsilon \in \mathbb{R}^+ \tag{6.12}$$

Selbstverständlich muß ϵ (deutlich) kleiner als der Attraktorwert α sein. Eine weitere Aufgabe des Monitorings ist das Detektieren des Eintretens in den Bereich eines lokalen Minimums gemäß Prädikat (6.10) oder des Stillstands nach Prädikat (6.11), um einen Algorithmus, der geeignet ist, die Position des Minimums wieder zu verlassen, starten zu lassen. Im Falle eines lokalen Minimums gilt für einen bestimmten und alle nachfolgenden (im Kontinuierlichen identischen, im Diskreten nahegelegenen) Punkte der Trajektorie, daß das Vektorfeld an dieser Stelle den Wert $\vec{0}$ aufweist. Dies führt zum leicht anwendbaren Kriterium, daß die Summe S der Längen der Schritte der Trajektorie in diesem Fall nicht steigt. Für eine Trajektorie $Q_{\boldsymbol{q}_0}$ sei die Summe S nach j Schritten wie folgt definiert:

$$S^j\left(Q_{q_0}\right) \overset{Def}{=} \sum_{k=1}^{j} \|\boldsymbol{q}_k - \boldsymbol{q}_{k-1}\| \tag{6.13}$$

Es gilt für den Fall des Eintretens von (6.10) oder (6.11):

$$\lim_{j\to\infty} S^j\left(Q_{\boldsymbol{q}_0}\right) = c\,,\ c \in \mathbb{R}^+ \tag{6.14}$$

Diesem Kriterium kann man sich durch

$$Monitor_{Halt}\left(Q_{\boldsymbol{q}}^k\right) \overset{Def}{\Leftrightarrow} \forall i \in \{k-l, \ldots, k-1\} : |\boldsymbol{q}_{i+1} - \boldsymbol{q}_i| < \epsilon' \tag{6.15}$$

mit $\epsilon' \in \mathbb{R}^+$ annähern. Dabei sind l und ϵ' festzulegende Werte. Ein praktikabler und sicherer Wert für l ist z.B. mit 10 gegeben, ϵ' ist mit einem Bruchteil von α (z.B. $\epsilon' = \frac{\alpha}{10}$) sinnvoll gewählt.

6.7 Monitorgesteuerte Aktionen

Im vorhergehenden Kapitel wurden die beiden Monitore $Monitor_{Ziel}$ und $Monitor_{Halt}$ vorgestellt, zu denen nun geeignete Aktionen definiert werden müssen. Im ersten Fall ist ein Beenden des Algorithmus' angezeigt, da das Ziel erreicht wurde. Im letzten Fall, wenn die Trajektorie keinen Fortschritt zeigt, sind die Fälle zu unterscheiden:

1. Die Trajektorie ist in einem lokalen Minimum gefangen (Fall nach Prädikat (6.10)).
2. Der durch die letzten beiden Positionen der Trajektorie implizierte Vektor ist auf einen Repellormittelpunkt mit dahinterliegendem Zielpunkt gerichtet (Fall nach Prädikat (6.11)).

Dies kann genau mit dem Kriterium der linearen Abhängigkeit des zu der aktuellen Position gegebenen Repellor- und Attraktorvektors[2] getestet werden: Sind beide Vektoren linear unabhängig, liegt ein lokales Minimum vor, andernfalls liegen Repellorzentrum und Ziel mit dem letzten Trajektorieelement auf einer Geraden.

Es ergibt sich die Monitoraktionstabelle für die aktuelle Position $\boldsymbol{q}_k$:

Monitor	Situation	Bedeutung	Aktion
$Monitor_{Ziel}$	eindeutig	Trajektorie hat Ziel erreicht	STOP
$Monitor_{Halt}$	$(\boldsymbol{q}_k - \boldsymbol{\mu}) \times (\boldsymbol{z} - \boldsymbol{q}_k) \neq \vec{0}$	Trajektorie ist in lokalem Minimum gefangen	Markiere aktuelle Position $\boldsymbol{q}_k$ als neuer (virtueller) Repellor und starte vom Anfangspunkt der Trajektorie neu
$Monitor_{Halt}$	$(\boldsymbol{q}_k - \boldsymbol{\mu}) \times (\boldsymbol{z} - \boldsymbol{q}_k) = \vec{0}$	Trajektorieelement liegt mit Repellorzentrum und Ziel auf einer Geraden	Modifiziere aktuelle Position $\boldsymbol{q}_k$

Tabelle 10: Monitoraktionstabelle

Die im letzten Fall notwendige Modifikation der aktuellen Position kann dabei z.B. nach den folgenden Strategien vorgenommen werden:

- Modifikation in eine oder zwei Richtungen (sinnvoll im Zweidimensionalen)

[2]oder Skalierungen davon, wie die Vektoren Position–Repellor und Position–Attraktor

- Modifikation der Position einer Spirale um die aktuelle Position folgend, die in Polarkoordinaten mit für jeden Schritt wachsendem Abstand zur ursprünglich aktuellen Position beschrieben werden kann (dreidimensionaler Raum)
- Modifikation durch Simulation der Brown'schen Bewegung (dreidimensionaler Raum)

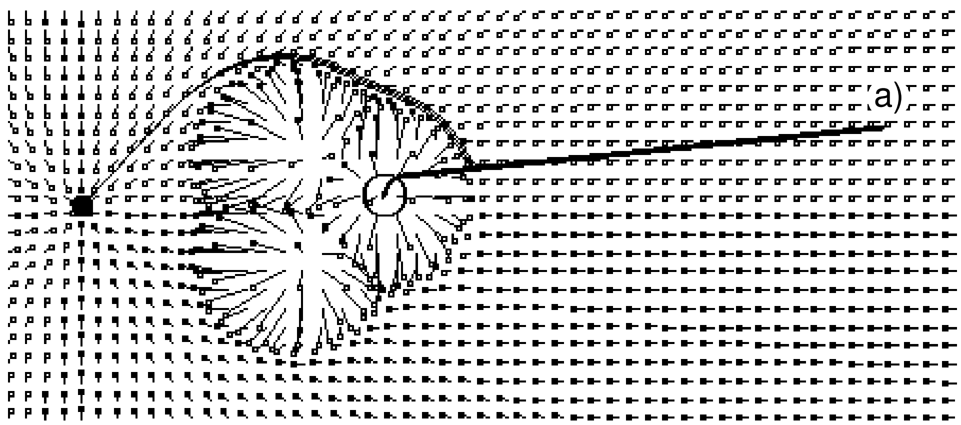

Abbildung 6.7: Durch Einfügen neuer Repelloren an Haltestellen der Trajektorie (Kreis) und erneutem Starten von Punkt a) wird das lokale Minimum umgangen (vgl. Abbildung (6.5)).

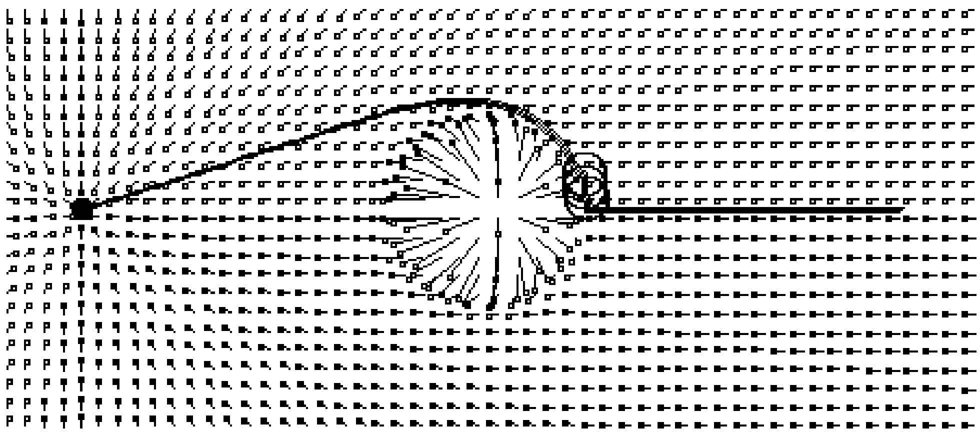

Abbildung 6.8: Detektion des Haltezustandes der Trajektorie und wiederholte Modifikation der Position (eingekreist, vgl. Abbildung (6.6))

Die vom Monitor gesteuerten Aktionen erlauben nun auch in den oben geschilderten Fällen die Umgehung von Hindernissen, wie die Abbildungen (6.7) und (6.8) zeigen.

Kapitel 7

Erprobung des Systems

Mit den in den Kapiteln 3 bis 6 vorgestellten Techniken der Detektion von Grauwertstrukturen, der Strukturzuordnung, der Anwendung von RBF-Netzen zur neuronalen Rekonstruktion einer Szene und der vektorfeldbasierten Generation von Trajektorien sind die Komponenten des Manipulatornavigationssystems vollständig beschrieben. Nachdem bereits in den einzelnen Kapiteln Betrachtungen bzgl. des (teilweise von den jeweiligen Parametern abhängigen) Fehlers der Komponenten angestellt wurden, wird nun der Ablauf des Gesamtsystems im Zusammenhang beschrieben und getestet.

7.1 Testszene und Aufgabe

Bei der Testszene handelt es sich um eine statische Anordnung von Hindernissen und einem Zielobjekt. Vor dem Roboter befindet sich ein Tisch, auf dem fünf unterschiedliche Flaschen angeordnet sind. Die vier äußeren Flaschen, die um die in der Mitte plazierte Flasche angeordnet sind, repräsentieren die Hindernisse, die mittlere Flasche ist das zu ergreifende Ziel. Diese Szene ist in Abbildung 7.1 aus zwei Perspektiven dargestellt. Die Annäherung des Roboters erfolgt gemäß der verwendeten Testszene mit waagerecht gehaltenen Fingern, die Roboterhand wird ebenfalls innerhalb einer zum Tisch parallelen Ebene positioniert[1]. Nachfolgend werden die einzelnen Schritte des Manipulatornavigationssystems dokumentiert.

[1]Die verwendeten Verfahren erlauben bei Bedarf ein beliebige Dimensionalität der generierten Trajektorie z.B. in fünf Dimensionen: Drei Dimensionen bestimmen die Position der Roboterhand, zwei die Orientierung der Hand in Polarkoordinaten.

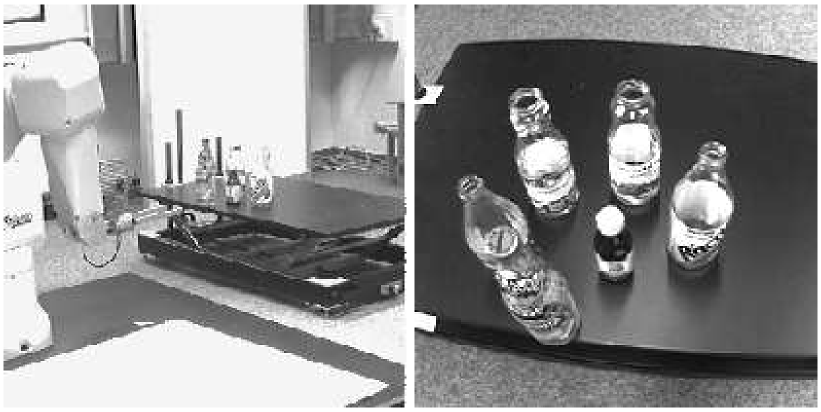

Abbildung 7.1: Die Testszene mit den äußeren Flaschen als Hindernisse und der inneren Flasche als zu ergreifendes Ziel. Links: Gesamtszene mit Startposition des Roboters. Rechts: Perspektive von oben, ein erfolgreicher Zugriff mit waagerechter Annäherung des Roboters kann nur von unten rechts erfolgen.

7.2 Erlernen der Rekonstruktion der Szene

Die Rekonstruktion der Szene wird in Abhängigkeit der Kameraorientierung am Roboter, die bei jeder erneuten Montage der Kamera an die Hand variieren kann, sowie einer auszuführenden Translation der Roboterhand erlernt (Kapitel 4.2). Der Aufbau des zum Lernvorgang positionierten Robotersystems ist der Abbildung 7.2 zu entnehmen.

Abbildung 7.2: Aufbau der Trainingsszene: Bzgl. der Orientierung der Kamera und der gewünschten Translation wird mit dem auf dem Boden liegenden Muster die Rekonstrukion erlernt.

7.3 Rekonstruktion der Szene

Die Szene wird von rechts nach links mit der im Trainingsschritt ausgeführten Translation mit einer Schrittlänge von $5cm$ und einer Gesamtlänge der Strecke von $40cm$ abgefahren. Durch die Aufnahme eines Bildes an jeder Handposition wird eine Bildfolge von 9 Bildern generiert, von der je zwei aufeinanderfolgende Bilder durch die Komponente der neuronalen Rekonstruktion in dreidimensionale Szeneninformationen überführt werden (Kapitel 4.2.5). Auf diese Weise wird die rekonstruierte Szene durch die schrittweise Einbettung der im letzten Schritt berechneten Hindernispunkte in die Menge der schon ermittelten an die reale Szene angenähert. Die Abbildung 7.3 zeigt die verschiedenen Stadien der Rekonstruktion. Die linke Spalte zeigt das aktuelle Bild, die rechte Spalte die aktuelle Rekonstruktion der Szene, wobei der jeweils am weitesten unten gelegene Punkt die Zugriffsposition der Hand repräsentiert. Um die Rekonstruktion visualisieren zu können, wurde die von der Firma Silicon Graphics, Inc. (SGI) entwickelte Grafik-Bibliothek *OpenGL* eingesetzt. Diese bietet ein Software-Interface zu der verwendeten Grafikhardware an.

Nach der schrittweisen Rekonstruktion der Szene wird eine Heuristik zur Reduzierung der falsch zugeordneten Korrespondenzen und damit der falsch rekonstruierten Szenenpunkte angewandt (Kapitel 3.2). Dazu wird die Anzahl der zu einem betrachteten, rekonstruierten Szenenpunkt benachbarten Punkte als Maß für die Korrektheit der Rekonstruktion dieses Punktes herangezogen. Ein falsch rekonstruierter Punkt weist in der Tendenz weniger Nachbarn als ein korrekt rekonstruierter Punkt auf. Daher werden alle Punkte gelöscht, deren nächster Nachbar weiter als $2cm$ entfernt ist. Die Wirksamkeit dieser Heuristik wird im Vergleich der Abbildungen 7.4 und 7.5 sichtbar.

7.4 Berechnung der Zugriffsposition

Nach der Rekonstruktion der Testszene ist die Ermittlung der Zugriffsposition Grundlage zur Berechnung der Trajektorie. Die Zugriffposition beinhaltet in Abhängigkeit von der Zielposition die Orientierung der Roboterhand. Diese ist für die Berechnung der Trajektorie fest gewählt. Eine andere Möglichkeit besteht in der Einbeziehung der Handorientierung in die Berechnung der Trajektorie. Allerdings hat sich gezeigt, daß die gestellte Aufgabe mit der Position der Hand als freier und der Handorientierung als fest gewählter (berechneter) Parameter gelöst werden kann. Die Zugriffsposition wird (ohne Orientierung) in den Abbildungen 7.3 bis 7.5 dargestellt, Abbildung 7.7, rechts oben, zeigt die Roboterhand die Einnahme der zum Zugriff notwendigen Orientierung (Berechnung nach Kapitel 5.2).

7.5 Berechnung der Trajektorie

Zu dem aus der Zielposition berechneten Zugriffsposition erfolgt die Generation der Trajektorie (Kapitel 6). Die berechnete Trajektorie ist in Abbildung 7.6 dargestellt. Die hier eingezeichneten Hindernispunkte sind nach der erweiterten Hindernisdefinition des Kapitels 5.1 zu verstehen. Nach anfänglicher Ablenkung der Roboterhand nach rechts (relativ zum Roboter) ist ein Zufahren der Hand zur Zugriffsposition möglich, wobei zum Ende der Trajektorie zwei weitere Korrekturen der Trajektorie bis zum Erreichen der Zugriffsposition erfolgen. Eine Auswahl der Schritte der Trajektorie, die aus 50 Elementen besteht, ist in der Abbildung 7.7 dargestellt.

7.6 Verwendete Parameter

In der folgenden Tabelle werden die Parameter sowie ihre für diesen Testlauf gewählten Werte gegenübergestellt, wobei der Parameter t der Grauwertschranke des SUSAN Operators aus dem pixelweise betrachteten, maximalen Bildrauschen der Bildfolge ermittelt wurde.

Komponente	Parameter	Wert
Strukturdetektion	Grauwertschranke t d. Operators SUSAN	15
Strukturzuordnung	Bestimmung der notwendigen Strukturähnlichkeit C	0.94
s.o.	Bestimmung der max. Abweichung der Sollrichtung des optischen Flusses A	4
Training der Rekonstruktion	Seitliche Translation	$50mm$
s.o.	Abstand der Trainingsebenen	$50mm$
Rekonstruktion	Anzahl der Netze Rxy / Knoten pro Netz	2 / 400
s.o.	Anzahl der Netze Rz	400 /14
Berechnung der Zugriffsposition	-	-
Berechnung der Trajektorie	Skalierung der Repelloren h / b	30 / 0.08
s.o.	Attraktorwert α	automatisch aus b und h bestimmbar
$Monitor_{Halt}$	Fortschritt ϵ' in den jeweils l letzten Schritten	$\frac{\alpha}{10}$, 10

Tabelle 11: Die Parameterwerte des Testlaufs. Testebenen.

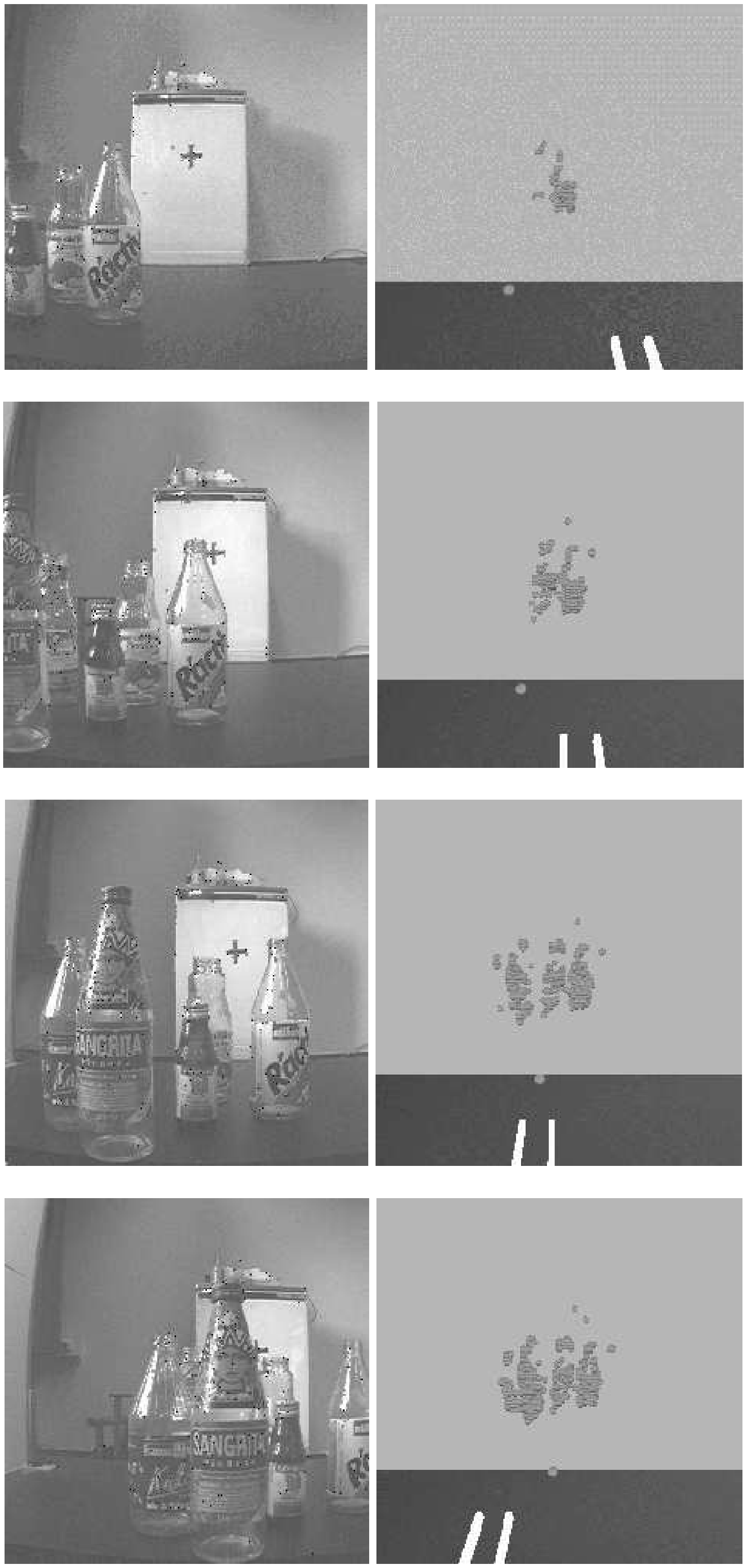

Abbildung 7.3: Schrittweise Rekonstruktion der Szene. Schritte 1 (oben), 3, 5 und 7 (unten) der insgesamt 9 Rekonstruktionsschritte. Der unterste Punkt der jeweils rechten Bilder stellt nicht eine Rekonstruktion, sondern die Zugriffsposition der Roboterhand (vorne im Bild) dar.

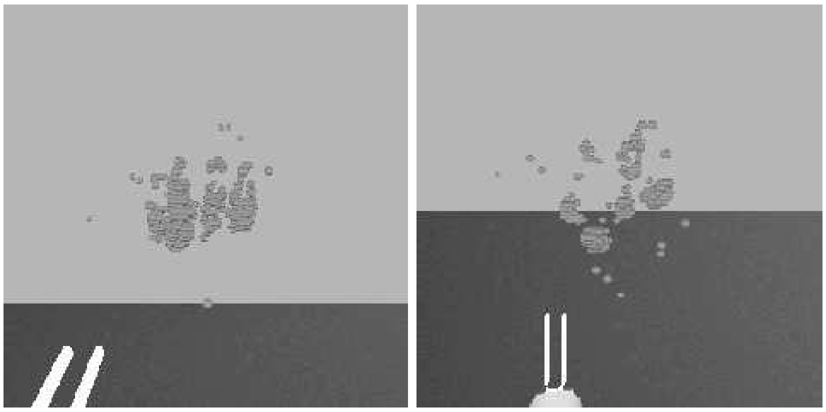

Abbildung 7.4: Abgeschlossene Rekonstruktion der Szene nach Schritt 9 (mit vorner liegendem Zugriffspunkt). Links: Perspektive von vorne. Rechts: Perspektive von oben (vgl. Abbildung 7.1, rechts)

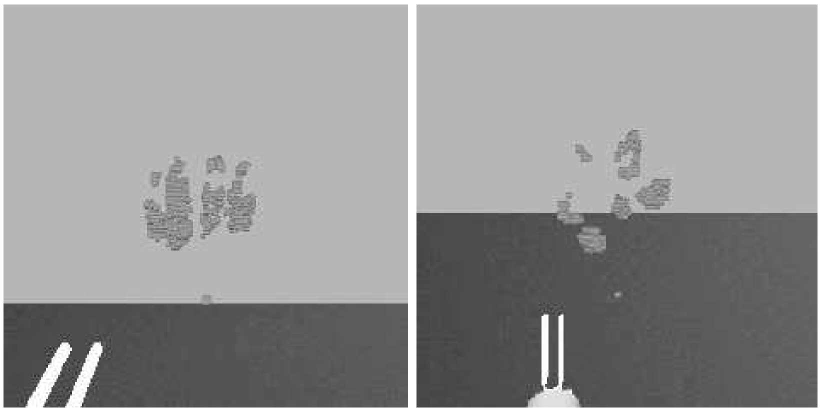

Abbildung 7.5: Die Löschung von rekonstruierten Punkten, die keine Nachbarn in einer Entfernung von $2cm$ aufweisen, befreit die Szene von Fehlrekonstruktionen (vgl. Abbildung 7.4)

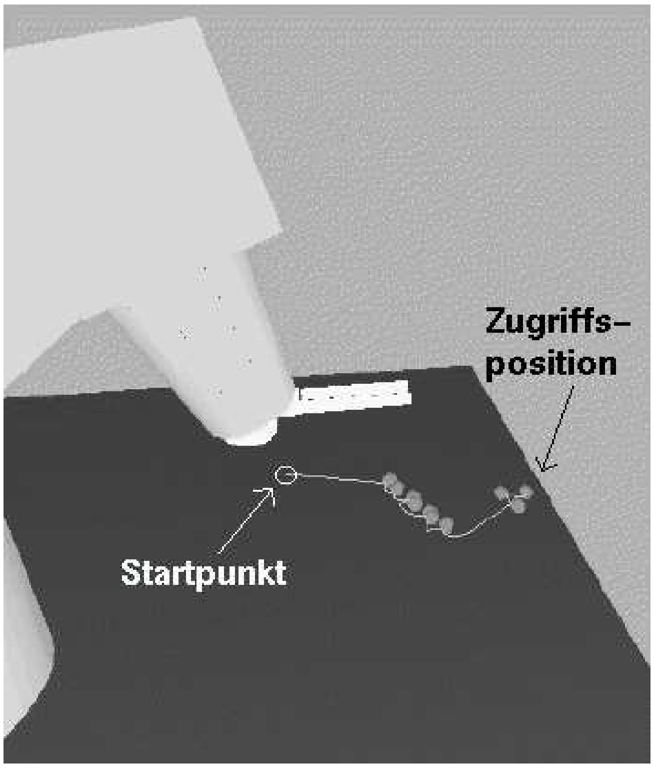

Abbildung 7.6: Die berechnete Trajektorie vom Startpunkt zur Zugriffsposition (letzter Punkt der Trajektorie) und die Hindernispunkte nach der erweiterten Definition

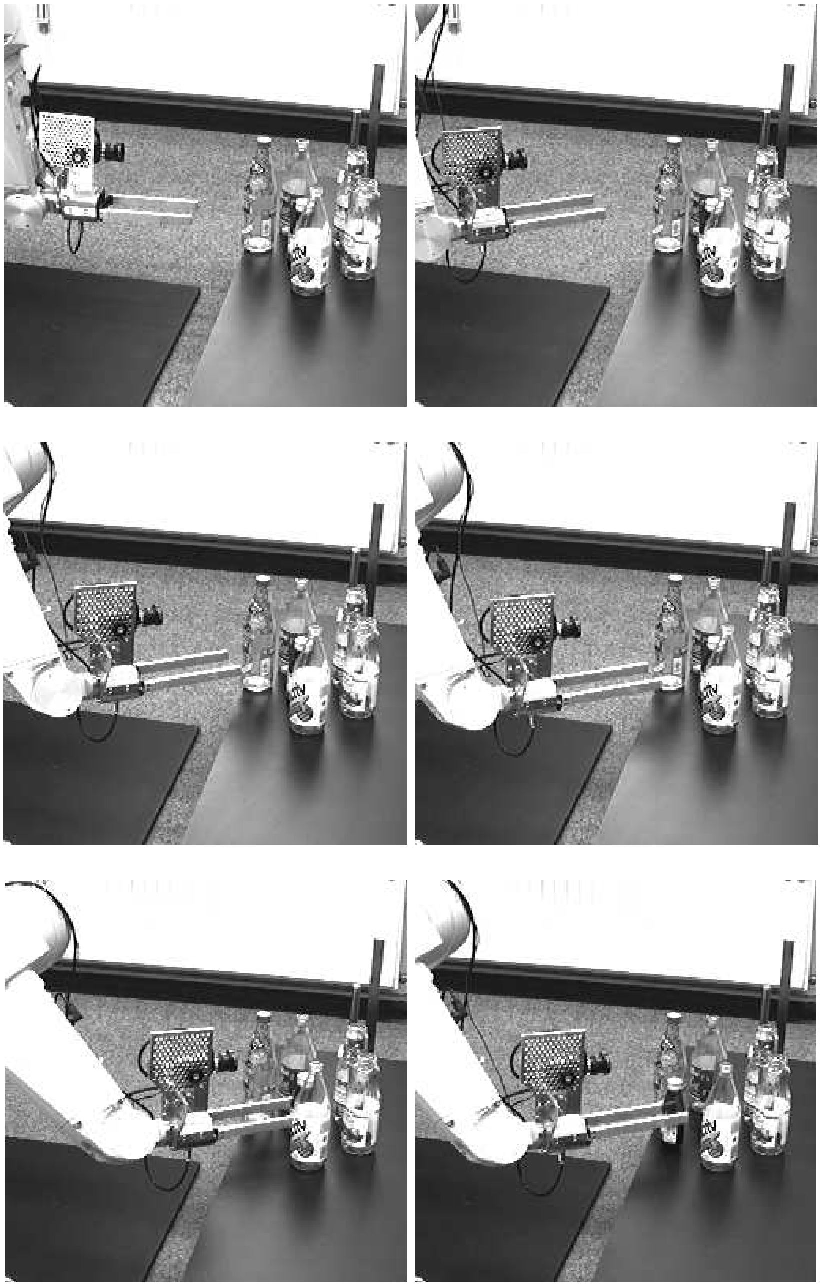

Abbildung 7.7: Ausführung der berechneten Trajektorie. Oben links: Startposition. Oben rechts: Einnahme der Handorientierung der Zugriffsposition. Mitte links und Mitte rechts: Schritte der Trajektorie. Unten links: Erreichen der Zugriffsposition. Unten rechts: Nach dem Zugriff der Roboterhand wird das Zielobjekt durch umgekehrtes Abfahren der berechneten Trajektorie aus dem Hindernisbereich entfernt.

Kapitel 8

Diskussion und Ausblick

Die vorherigen Kapitel wurden dem Entwurf, der Beschreibung und der Erprobung des Manipulatornavigationssystems gewidmet. In diesem Kapitel werden die Ergebnisse dieser Arbeit diskutiert und abschließend ein Ausblick auf mögliche Erweiterungen und Verbesserungen der Arbeit gegeben.

8.1 Diskussion

Das Manipulatornavigationssystem ist ein System, dessen Struktur durch Wahrnehmungs-Aktions-Zyklen bestimmt wird. In Abhängigkeit von der Dynamik der Szene handelt es sich um einen Zyklus oder eine Abfolge von Wahrnehmung und Aktion. Die dargestellten Verfahren sind für beide Arten der Szene geeignet, auch wenn wir in der Erprobungsphase eine statische Anordnung von Hindernissen und Zielobjekt gewählt haben. Die Komponenten der Wahrnehmung/Verarbeitung der Wahrnehmung sind Strukturdetektion, Strukturzuordnung sowie neuronale Rekonstruktion der Szene. Die Komponente, die die auf die Wahrnehmung folgende Aktion bestimmt, ist mit der Generation der Trajektorie gegeben.

Bei der Auswahl des Strukturdetektors ist mit dem SUSAN-Operator ein im Vergleich zu anderen Operatoren schneller Detektor zur Auswahl gekommen, dessen Ausgabequantität mit einem Parameter geregelt werden kann, ohne die Qualität der detektierten Ecken zu beeinflussen. Dieser Parameter ist bei Bedarf automatisch (durch die Bestimmung des Rauschens in einer Bildfolge) bestimmbar. Die Parallelisierung des Operators beschleunigt die Anwendung und skaliert mit der Anzahl der eingesetzten Prozessoren des verwendeten Computers. Bei der Strukturzuordnung ist mit der normierten Kreuzkorrelation ein Standardverfahren zur Anwendung gekommen. Dieses Verfahren ist zwar rechenintensiv, es kann jedoch durch Vorauswahlverfahren der in Frage kommenden Grauwertstrukturen beschleunigt werden. Auch hier hat die Parallelisierung des Verfahrens Eingang gefunden.

Die Rekonstruktion der Szene durch neuronale Netze macht die Bestimmung von internen und externen Kameraparametern unnötig. Nicht nur der Ausschluß dieser Fehlerquelle, sondern auch die automatische Integration veränderbarer Parameter wie die Orientierung und die Position der Kamera relativ zur Roboterhand sind Vorteile dieses Verfahrens. Die unpraktikablen Trainingszeiten des einzelnen Rekonstruktionsnetzes konnten durch die Anwendung des divide-and-conquer-Verfahrens und der daraus resultierenden Netzteilung auf einen Bruchteil des vorherigen Wertes reduziert werden. Dies konnte jedoch nur um den Preis der zusätzlichen Interpolation der Netzantworten an den Stellen erfolgen, die nicht im Kompetenzbereich eines Netzes lagen, da die Generalisierung des Einzelnetzes durch die Teilung entfallen ist. Trotzdem liegt die Antwortzeit des Netzsystems im Bereich des Gesamtnetzes, da die Anzahl der zu berücksichtigenden Knoten näherungsweise gleich der Summe der Knoten der Einzelnetze ist.

Mit der Generierung der Trajektorie auf der Basis eines Vektorfeldes konnten die gegensätzlichen Ziele der Hindernisvermeidung und der Annäherung des Roboters an das Ziel in einem gemeinsamen Rahmen integriert werden. Die Grundlage dazu konnte mit der Definition eines allgemeinen Repellortyps in Anlehnung an radiale Basisfunktionen geschaffen werden, die in Breite und Höhe des von Null verschiedenen Definitionsbereiches manipuliert werden können. Der Detektion und Bewältigung der bei der Verwendung von Potentialfeldalgorithmen auftretenden Probleme lokaler Minima dienen die eingeführten Monitore sowie die monitorgesteuerten Aktionen. Das angegebene Verfahren bietet die Möglichkeit der dynamischen Aktualisierung des Vektorfeldes, um Änderungen der Umwelt in die generierte Trajektorie des Manipulatornavigationssystems einfließen zu lassen.

Das Manipulatornavigationssystem benötigt einige a priori Informationen. Sie ermöglichen Verfahren des automatischen Informationserwerbs. Die beiden Kategorien der automatisch bestimmten und der a priori Informationen sind in Tabelle 12 gegenübergestellt.

Information	Verwendung	a priori	durch MNS bestimmt
Lage des Rekonstruktionsmusters in Weltkoordinaten	Erlernen der Rekonstruktionsabbildung	●	
Rekonstruktionsabbildung	Rekonstruktion der Szene		●
Maße des Roboters	Test auf Kollision des Roboters mit Hindernissen	●	
Position von Hindernispunkten	Setzen von Repelloren		●
Zielpunkt	Setzen des Attraktors	●	
Trajektorie	Greifen des Zielobjekts unter Hindernisvermeidung		●

Tabelle 12: A priori und zur Laufzeit des Manipulatornavigationssystems bestimmte Informationen

Zusammenfassend ermöglichen die geschilderten Verfahren ein System, das in einer Umwelt zielorientiert agieren kann, die mit strukturbehafteten (typischerweise vom Menschen geschaffenen) Hindernissen besetzt ist. Dabei zeichnet sich das Manipulatornavigationssystem durch die Fähigkeit zur autonomen Orientierung in seiner statisch oder dynamischen Umgebung aus, die in seine Verhaltensweise Eingang findet. Weiterhin ist das System flexibel bezüglich systeminterner, variabler Parameter (Kameraposition) und weist eine praktikable Laufzeit in Training und Ausführung der Aufgabe vor. Anwendungen finden sich sowohl im Bereich der Industrie als auch im Bereich der Behindertenunterstützung.

8.2 Ausblick

Obwohl das Manipulatornavigationssystem für dynamische Szenen geeignet ist, haben wir uns bisher auf Tests statischer Szenen beschränkt, so daß weitere Forschungsarbeit hinsichtlich dynamischer Umgebungen sinnvoll ist.

Die Rekonstruktion der Umwelt geschieht in Form der Berechnung von Punkten im dreidimensionalen, für die eine Korrespondenz in zwei aufeinanderfolgenden Bildern bestimmt werden kann. Wir haben bewußt auf eine Modellbildung von Objekten im Sinne einer Interpretation von Strukturgruppierungen verzichtet, da eine Szeneninterpretation für die Aufgabe der Hindernisvermeidung nicht notwendig ist. Zwar bietet sie Vorteile wie z.B. die Schätzung des durch das Hindernis eingenommenen Raumes bei nicht vollständig sichtbarem Objekt, der zusätzliche Aufwand ist jedoch bei geeigneter, d.h. strukturbehafteter Szene und entsprechender Raumerkundung nicht notwendig. Allerdings sind Szenen konstruierbar, die dem System keine oder wenig Strukturen zur Berechnung des optischen Flusses bieten. Soll das System in solchen Umgebungen eingesetzt werden, ist eine Erweiterung der Menge der Grundelemente der Rekonstruktion denkbar. Wurde bisher die Rekonstruktion auf der Basis des Punktes realisiert, könnten im Bild vorhandene Elemente wie Geradensegmente und Flächen ihre Entsprechung in den Grundelementen der Rekonstruktion finden. Die Rekonstruktion eines Grundelements könnte durch ein eigenes Neuronales Netz(-system) vertreten werden.

Schließlich ist die Integration einer Objekterkennung in das Manipulatornavigationssystem eine Möglichkeit, das System autonom sein Ziel finden zu lassen. Bisher haben wir das Ziel vorgegeben, so daß das System bezüglich der Wahrnehmung noch nicht autonom handelt. Eine parallel zu der bisher vorhandenen Wahrnehmung installierte Objekterkennung vervollständigt das Manipulatornavigationssystem.

Literaturverzeichnis

[1] H. P. Moravec. Robot spatial perception by stereoscopic vision and 3d evidence grids. Technical Report CMU-RI-TR-96-34, Carnegie Mellon University, Robotics Institute, 1996.

[2] X. Zhang and H. Deng. Distributed image edge detection methods and performance. In *Proceedings of the Sixth IEEE Symposium on Parallel and Distributed Processing*. IEEE Computer Society Press, 1994.

[3] R. Szeliski and S.B. Kang. Recovering 3d shape and motion from image streams using non-linear least squares. Technical Report CRL 93/3, Cambridge Research Laboratory, 1993.

[4] J.-Y. Bouguet and P. Perona. Visual navigation using a single camera. In *Proceedings of the 5th International Conference on Computer Vision*, pages 645–652, 1995.

[5] N. Cui, J.J. Weng, and P. Cohen. Recursive-batch estimation of motion and structure from monocular image sequences. *Computer Vision, Graphics, and Image Processing, Image Understanding*, 59(2):154–170, 1994.

[6] J. Mendelsohn, E. Simoncelli, and R. Bajcsy. Discrete-time rigidity-constrained optical flow. In *Computer Analysis of Images and Patterns*, pages 255–262, Kiel, Germany, 1997.

[7] K. Kanatani. Self-evaluation for active vision by the geometric information criterion. In *Computer Analysis of Images and Patterns*, pages 247–254, Kiel, Germany, 1997.

[8] A. Dev, B. Kröse, L. Dorst, and F. Groen. Observer curve and object detection from the optic flow. *Intelligent Robots and Computer Vision XIII*, pages 38–50, 1994.

[9] F. G. Meyer. Time-to-collision from first-order models of the motion field. *IEEE Transactions on Robotics and Automation*, 10(6):792–798, 1994.

[10] Z. Zhang, R. Weiss, and A. R. Hanson. Qualitative obstacle detection. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 554–559, 1994.

[11] H. A. Mallot, H. H. Bülthoff, J. J. Little, and S. Bohrer. Inverse perspective mapping simplifies optical flow computation and obstacle detection. *Biological Cybernetics*, 64:177–185, 1994.

[12] S. Seitz and C. R. Dyer. Complete scene structure from four point correspondences. In *Proceedings of the 5th International Conference on Computer Vision*, pages 330–337, 1995.

[13] G. Sandini, F. Gandolfo, E. Grosso, and M. Tistarelli. Vision during action. In *Active Perception*, pages 151–190, 1993.

[14] M. Jordan and R. Jacobs. Hierarchical mixtures of experts and the em algorithm. *Neural Computation*, 6:181–214, 1994.

[15] T. Cord and D. Pallmer. Distance measuring for mobile robots using axial motion stereo. In *Proceedings of the 3rd International Workshop on Robotics*, pages 193–196, 1994.

[16] R. A. Brooks. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, RA-2(1):14–23, 1986.

[17] J. H. Connell. A behavior based arm controller. *IEEE Journal of Robotics and Automation*, 5(6):784–791, 1989.

[18] R. Hartley and F. Pipitone. Experiments with the subsuption architecture. In *Proceedings of the 1991 IEEE International Conference on Robotics and Automation*, pages 1652–1658, 1991.

[19] G. Schöner, M. Dose, and C. Engels. Dynamic fields endow behavior-based robots with representations. *Robotics and Autonomous Systems*, 14:55–77, 1995.

[20] G. Schöner, M. Dose, and C. Engels. Dynamics of behavior: theory and applications for autonomous robot architectures. *Robotics and Autonomous Systems*, 16:213–245, 1995.

[21] J. BorenStein and Y. Koren. Real-time obstacle avoidance for fast mobile robots. *Transactions on Systems, Man and Cybernetics*, 19(5):1179–1187, 1989.

[22] F. A. Mussa-Ivaldi. From basis functions to basis fields: Vector field approximation from sparse data. *Biological Cybernetics*, 67:479–489, 1992.

[23] F. A. Mussa-Ivaldi and S. F. Giszter. Vector field approximation: A computational paradigm for motor control and learning. *Biological Cybernetics*, 67:491–500, 1992.

[24] Y. Koren and J. BorenStein. Potential fild methods and their inherent limitations for mobile robot navigation. In *Proceedings of the 1991 IEEE International Conference on Robotics and Automation*, pages 1398–1404, 1991.

[25] S. Quinlan and O. Khatib. Elastic bands: Connecting path planning and robot control. In *Proceedings of the IEEE International Conference on Robotics and Automation*, volume 2, pages 802–807, Atlanta, Georgia, 1993.

[26] W. E. Kelly. Neuro-fuzzy control of a robotic arm. In *International Conference on Artificial Neural Networks in Engineering – ANNIE*, pages 837–842. St. Louis, Missouri, USA, 1996.

[27] J. P. Johnson and et al. A multi-neural network intelligent path-planner for a robot arm. In *International Conference on Artificial Neural Networks in Engineering – ANNIE*, pages 843–848. St. Louis, Missouri, USA, 1996.

[28] C. Torras. From geometric motion planning to neural motor control in robotics. *AI Communications*, 6(1):3–17, 1993.

[29] R. Jain et al. *Machine Vision*. McGraw-Hill, 1995.

[30] B. Jähne. *Digitale Bildverarbeitung*. Springer-Verlag, 4 edition, 1997.

[31] S.M. Smith and J.M. Brady. Susan - a new approach to low level image processing. *International Journal of Computer Vision*, 1996.

[32] C. M. Bishop. *Neural Networks for Pattern Recognition*. Clarendon Press, Oxford, 1996.

[33] M. Orr. Introduction to radial basis function networks. http://www.cns.ed.ac.uk/people/mark/intro/intro.html.

[34] W. H. Press and et al. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, 1993.

[35] A. Naeve. Structure from translational observer motion. In *Algebraic Frames for the Perception-Action Cycle*, pages 235–248, Kiel, Germany, 1997.

[36] F. Locher. *Numerische Mathematik für Informatiker*. Springer Verlag, Berlin, 1991.

[37] G. Engeln-Müllges and F. Uhlig. *Numerical Algorithms with C*. Springer Verlag, Berlin, 1996.

[38] J. Craig. *Intoduction to Robotics*. Addison-Wesley Publishing Company, Massachusetts, 1989.

Danksagung

Allen Personen, die direkt oder indirekt an der Entstehung dieser Diplomarbeit beteiligt waren, möchte ich herzlich danken.

Insbesondere danke ich meinem Betreuer Dr. Josef Pauli nicht nur für die fruchtbaren Diskussionen, sondern auch für die innerhalb des Rahmens der Beratung gegebene größtmögliche Freiheit und Selbständigkeit meiner Arbeit.

Nicht zuletzt bin ich der umfassenden Unterstützung meiner Freundin Susanne dankbar, wie auch der Leser für die grammatikalische Bereinigung der Arbeit durch Susanne und meine Schwester Sigrun dankbar sein wird.

Erklärung:

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

Kiel, den