Modifikation einer geeigneten Stereobild-verarbeitungsmethode für die Anwendung in einer Echtzeitumgebung

Diplomarbeit

Otto-von-Guericke-Universität Magdeburg

Fakultät für Informatik

Institut für Verteilte Systeme

Richard Bade

Juni 2003

Kurzfassung

Mobile Roboter müssen in der Lage sein, Informationen über ihre Umgebung zu erfassen, um mit ihr in Interaktion treten zu können. Dafür gibt es eine Vielzahl von unterschiedlichen Sensoren. Die vorliegende Arbeit leistet einen Beitrag, die Genauigkeit der Umgebungsabbildung bei mobilen Robotern zu verbessern. Das Kernstück dieser Arbeit ist die Auswahl und Modifikation einer Stereobildverarbeitungsmethode in Hinblick auf die Verwendung in einem Echtzeitsystem.

Im Rahmen dieser Arbeit wird aufbauend auf dem Block Matching Algorithmus ein echtzeitfähiges Verfahren basierend auf dem Konzept der Gaußpyramiden implementiert. Mit einem eigenen Versuchsaufbau wurden Daten gewonnen, mit deren Hilfe der Algorithmus getestet wurde. Die Ergebnisse zeigen, dass sich sowohl die Ausführungszeit als auch die Qualität im Vergleich zum Ausgangsverfahren verbesserten. Neben dem implementierten statischen Ansatz werden zwei weitere Ansätze, der dynamische und der Fusionsansatz, konzeptionell betrachtet. Diese erweitern den statischen Ansatz durch Ausnutzung redundanter Sensorinformationen.

I Inhaltsverzeichnis

II Abbildungsverzeichnis

III Tabellenverzeichnis

IV Verzeichnis der Abkürzungen

BM	Block-Matching
ECET	Expected Case Execution Time
EDF	Earliest Deadline First
EP	Exception Part
LRT	Latest Release Time
MP	Main Part
MSE	Mean Square Error
RTLinux	Real Time Linux
SBV	Stereobildverarbeitung
SV	Stereo Vision
TAFT	Time Aware, Fault Tolerant
TP	Task Pair
WCET(EP)	WCET des EP
WCET	Worst Case Execution Time

V Verzeichnis der wichtigsten Formelzeichen

C_l, C_r	Brennpunkt der linken bzw. rechten Kamera
D	Disparität zwischen Blöcken bzw. Pixeln
d_{max}	Disparitätslimit
E_l, E_r	Intensität eines Pixels aus dem linken bzw. rechten Bild
f	Brennweite
h	Kameraabstand vom Ursprung des Weltkoordinatensystems
P	Punkt im Weltkoordinatensystem
P_l, P_r	Projektion von P auf das linke bzw. rechte Bild
$t_{deadline}$	Zeitpunkt der Deadline einer Task
x	x-Koordinate im Weltkoordinatensystem
x_l, x_r	x-Koordinaten im lokalen Koordinatensystem des linken bzw. rechten Bildes
z_{max}	Maximale Entfernung für Disparitätsberechnung

1 Einleitung

Mobile Roboter müssen die Umgebung, in der sie sich befinden, sowie ihre eigene Position kennen, um sich autonom in der Umwelt bewegen zu können. Das Wahrnehmen ihrer Umgebung erfolgt durch verschiedene Sensoren, die aktiv (Laserscanner, Ultraschall) oder passiv (Kraftsensoren) Objekte, wie Hindernisse oder ähnliches, erkennen. Zusätzlich zu den Informationen, die sich direkt durch die separate Auswertung der Sensorinformationen gewinnen lassen, lässt sich durch Sensorfusion ein besseres Abbild der Umwelt gewinnen.

Der Aufgabensteller, die Arbeitsgruppe Echtzeitsysteme und Kommunikation der Fakultät für Informatik an der Otto-von-Guericke-Universität Magdeburg, beschäftigt sich in ihrer wissenschaftlichen Arbeit unter anderem mit der Fusionierung von Daten verteilter Sensoren. Die vorliegende Diplomarbeit untersucht den Sensortyp Kamera, im speziellen die Fusionierung der Daten von mehreren Kameras, was die Rekonstruktion von Entfernungsinformationen ermöglicht.

1.1 Motivation

Sensoren dienen der Gewinnung von Informationen über die Umwelt und interner Zustände des Roboters. Sie bilden physikalischen Größen der Umwelt auf elektrische Größen so ab, dass sie von einem Rechnersystem verarbeitet werden können. Es werden verschiedene Sensoren benötigt, um zum Beispiel die Entfernung (Laserscanner) oder die Geschwindigkeit (Tachometer) zu messen, da kein Sensor existiert, der alles zu messen in der Lage ist. Wenn es also von Interesse ist, ein System zu entwickeln, das zum Beispiel im Straßenverkehr den Abstand eines Autos zum Vorfahrenden konstant halten soll, müssen entsprechende Sensoren ausfindig gemacht und die von ihnen bereitgestellten Informationen geeignet verarbeitet werden.

Da es in dieser Arbeit um die Abbildung der Welt im Kontext mobiler Roboter geht, werden hier Sensoren benutzt, die Entfernungen, Beschleunigungen und Rotationen messen können, um Roboterbewegungen festzustellen, Objekte zu lokalisieren und Änderungen in der Umwelt zu erfassen. Wesentlich hierbei ist in der Robotik die Objekterkennung. Das

bildet eine bedeutende Grundlage, um die Umgebung abzubilden und wiederzugeben. Dabei spielen natürlich die Kosten eine nicht unerhebliche Rolle. Es ist von Interesse, Verfahren zu entwickeln, die es erlauben die Informationen mehrerer heterogener oder homogener Sensoren so zu fusionieren, dass auf den Einsatz einzelner hoch spezialisierter und teurer Sensoren verzichten werden kann und die Lösung günstig, zuverlässig, flexibel und skalierbar wird.

Es ist nun zu überlegen, welche Art von Sensoren verwendet werden müssen, um die Kosten gering und gleichzeitig den Einsatz flexibel und skalierbar halten zu können. Ein Weg, das zu ermitteln, ist es, die vom Sensor bereit gestellte Informationsmenge und die effektiv genutzten Informationen den Anschaffungskosten gegenüber zu stellen. Je weniger Kosten pro Information vorhanden sind, desto günstiger ist der Sensor. Tabelle 1-1 versucht hier einen Überblick zu geben, wie ausgewählte Sensoren aus dem Bereich mobiler Roboter einzuordnen sind. Natürlich sind nicht allein die Kosten ausschlaggebend. Wozu die Informationen, die ein Sensor zu messen in der Lage ist, verwendet werden können, ist ebenfalls ein wichtiger Faktor. In Tabelle 1-2 sind einige wichtige Sensoren und die von ihnen gemessenen, sowie ableitbaren Informationen dargestellt. Die Kamera erzeugt dabei als einzige eine zweidimensionale Objektabbildung. Sie bildet Punkte im dreidimensionalen Raum auf Pixel in zwei Dimensionen mit den Parametern *R*, *G* und *B* ab. *x* und *y* definieren dabei die Position des Pixels im Bild und *R*, *G* und *B* die Intensität des Pixels durch rot, grün und blau. Laserscanner sind dagegen aber nur in der Lage in einer Ebene zu messen. Die gelieferten Informationen des Laserscanners sind eine Punktmenge, bei dem jeder Punkt durch einen Winkel φ und dem Parameter *s* definiert wird. *s* ist die Entfernung vom Laserscanner. Damit wird eine eindimensionale Objektabbildung des Raumes in Form von Konturen erzeugt. Die Kamera hat des Weiteren den Vorteil, Farbinformationen bereitzustellen, wozu Laserscanner und auch Ultraschall nicht in der Lage sind. Diese können aber im Gegensatz zur Kamera, auf Grund des Sensorprinzips, Entfernungen bestimmen. Das ist beim Einsatz einer Kamera ohne Vorwissen über die Objekte nicht möglich. Aus diesem Grund müssen die Informationen mehrerer, mindestens zwei Kameras, fusioniert werden, um so ebenfalls Entfernungsinformationen gewinnen zu können. Dieses Prinzip ist unter dem Begriff Stereo Vision (SV) bekannt. Die grundlegende Idee dahinter ist, in Anlehnung an das menschliche Sehsystem, die Betrachtung einer Szene mit zwei Kameras. Auf Grund

geometrischer Beziehungen zwischen den Kameras lassen sich in einer Szene die Tiefeninformationen berechnen. Diese Tiefeninformationen lassen sich gewinnen, nachdem die Pixel des einen Bildes den korrespondierenden Pixeln des anderen Bildes zugeordnet worden sind. Korrespondierende Pixel bedeutet, dass sie den selben Punkt im dreidimensionalen Raum repräsentieren.

Also lassen sich mit dem Kamerasensor ebenfalls Entfernungen gewinnen. Das bei der SV mehr als eine Kamera verwendet werden muss, ist, dadurch das sie ein kostengünstiger Sensor ist, kein Nachteil. Vielmehr ergibt sich der Vorteil, dass damit auf den Einsatz von hoch spezialisierten und teuren entfernungsmessenden Sensoren verzichtet werden kann.

Sensor	Anschaffungskosten	Bereitgestellte Informationsmenge	Kosten pro Information
Gyroscope	Mittel – Hoch	Klein	Mittel
Kamera	Gering –Mittel	Sehr groß	Gering
Laserscanner	Hoch	Groß	Mittel
Odometrie (Motor-/Radencoder)	Gering	Klein	Gering
Ultraschall	Gering	Mittel	Gering

Tabelle 1-1: Sensoren aus ökonomischer Sicht

Sensor	Art der Information	Ableitbare Information
Gyroscope	Orientierung	Orientierung, Drehung
Kamera	Farben	Formen, Objekte
Laserscanner	Entfernungen	Entfernung, Konturen
Odometrie (Motor-/Radencoder)	Rotationen	Richtung, Bewegung, Fahrweg, Geschwindigkeit
Ultraschall	Entfernungen	Entfernungen

Tabelle 1-2: Information von Sensoren

Im folgenden wird der Begriff SV Sensor verwendet. Damit ist ein Aufbau mit zwei Kameras gemeint, die in der Lage sind, eine gleiche Szene von zwei leicht verschiedenen Raumpositionen zur selben Zeit betrachten und aufnehmen zu können. Die gleichzeitige Aufnahme der Bilder stellt besonders in dynamischen Umgebungen sicher, das sich die Objekte in den Bildpaaren am selben Ort im Raum befinden. Erst dadurch wird die Suche nach korrespondierenden Pixeln ermöglicht.

Was heißt Gewinnung von Tiefen- oder Entfernungsinformationen nun aber im speziellen für mobile Roboter? Informationen zu erhalten ist der erste Schritt, sie zu verarbeiten und entsprechend darauf zu reagieren der nächste. Das rechtzeitige Reagieren auf Ereignisse setzt das rechtzeitige Vorliegen von Informationen voraus. In dem Bereich mobiler Roboter heißt rechtzeitig mitunter wenige Sekunden oder Millisekunden, je nach Geschwindigkeit des Roboters und Applikation. Wird beispielsweise von einem sich bewegenden Roboter in Bewegungsrichtung ein Hindernis erkannt, muss er entweder anhalten oder ausweichen. Welche Entscheidung zu treffen ist, ist an dieser Stelle nicht wichtig und hängt auch noch von anderen Faktoren ab. Wichtig ist aber, dass er auf das Ereignis Hindernis *rechtzeitig* reagiert, bevor er dagegen stößt.

1.2 Aufgabenstellung

Die Gewinnung von Entfernungsinformationen mittels mehrerer Kameras benötigt sehr viel Rechenkapazitäten, da die Bildverarbeitungsalgorithmen im Allgemeinen eine große Datenmenge zu verarbeiten haben. Um applikationsbezogenen Echtzeitanforderungen gerecht zu werden und außerdem eine hohe Qualität der Bildanalyse zu erzielen, ist es von Interesse, rechtzeitig Ergebnisse für nachgeschaltete Prozesse zur Robotersteuerung bereitzustellen. Die Rechtzeitigkeit des Vorliegens von Ergebnissen in bestehenden Arbeiten ist nur unter Verwendung spezieller Hardware oder Einschränkungen in der Einsatzumgebung realisiert worden. Um sie für klassische Echtzeit Scheduling Ansätze einsetzen zu können, muss eine WCET (Worst Case Execution Time) bekannt sein. Diese ist meist schwer zu ermitteln und wird in der Regel zu groß bemessen. Auch die bestehenden Algorithmen arbeiten großteils mit Laufzeiten der vollständigen Suche. Der Begriff vollständige Suche bezieht sich dabei auf den Suchraum, den die Algorithmen für die Zuordnung der Pixel des

einen Bildes auf das andere Bild durchlaufen müssen. Der maximale Aufwand wird erreicht, indem jedes Pixel mit jedem verglichen wird. Bei kleinen Bildern mag das vertretbar sein, bei Auflösungen im PAL Bereich von 720x576 Pixeln jedoch nicht denkbar.

Ein anderer Aspekt betrifft die Beschaffung der Informationen des SV Sensors. Im allgemeinen werden nach der Bestimmung von Entfernungsinformationen durch den SV Sensor wieder Bilder für eine erneute Berechnung aufgenommen, ohne zu überprüfen, ob sich die Umgebung geändert hat, also neue Informationen vorhanden sind. Eine solche Überprüfung ist zwar nur unter zu Hilfenahme weiterer Sensortypen möglich, aber wenn sie vorhanden sind, durchaus sinnvoll, um eine übermäßige Ressourcenbelastung zu vermeiden. So ist beispielsweise bei einer statischen Umgebung durch die einfache Überprüfung der Radencoder feststellbar, ob die SV Sensordaten erneut ausgewertet werden müssen.

Des Weiteren ist bezüglich der Fusion von Daten des SV Sensors mit anderen Sensordaten zu beachten, das nicht nur die gemeinsamen Ergebnisse eine Verbesserung der Umgebungsabbildung bewirken. Die Berücksichtigung beispielsweise der Ergebnisse anderer Sensoren innerhalb der Auswertung der SV Sensordaten kann ebenfalls nützlich sein.

Zusammenfassend können zwei Probleme formuliert werden, die bei bestehenden SV Techniken bezüglich mobiler Roboter vorhanden sind und in dieser Arbeit gelöst werden sollen. Das sind

1. die Echtzeitfähigkeit ist nur durch spezielle Hardware oder durch Restriktionen bezüglich der Einsatzumgebung realisierbar und
2. die Beschaffung zum einen zeitlich redundanter Informationen durch den SV Sensor als auch die ungeeignete Bereitstellung redundanter Informationen von unterschiedlichen Sensoren stellen eine übermäßige Ressourcenbelastung dar.

Aufgabe dieser Diplomarbeit ist daher die Auswahl und Modifizierung einer geeigneten Stereobildverarbeitungsmethode zur Bestimmung von Entfernungsinformationen. Diese muss gewährleisten, dass Ergebnisse rechtzeitig und möglichst immer vorliegen.

Zur Lösung dieser Aufgabe, soll ein Szenario mit zwei Kameras und bekannten Parametern, wie Abstand und Winkel, als Grundlage für die Auswahl und Modifizierung eines geeigneten Algorithmus dienen. Ein solches System mit zwei Kameras wird als Stereo Vision bezeichnet. Abstand und Winkel der Kameras sind so zu wählen, das ein Stereobildszenario entsteht, in dem bestehende Algorithmen funktionieren. Es ist zwar möglich und zukünftig auch vorgesehen, dass diese Parameter gelockert werden, aber nicht im Rahmen dieser Arbeit, die sich zunächst dem Problem der Echtzeit zuwendet.

Es ist ferner von Interesse, in welcher Weise Vorinformationen aus bereits analysierten Kamerabildern oder Bewegungsinformationen zur Verbesserung des Algorithmus beitragen.

1.3 Ergebnisse der Diplomarbeit

In dieser Diplomarbeit ist ein Konzept auf die Frage entwickelt worden, wie ein Stereobildverarbeitungsalgorithmus umgesetzt werden muss, damit er rechtzeitig und möglichst immer Ergebnisse vorweist. Zur Erlangung der Rechtzeitigkeit arbeiten klassische Scheduler mit der WCET (Worst Case Execution Time). Diese sind in der Stereobildverarbeitung (SBV) in der Regel sehr hoch und fern jeglicher Realität. Aufgrund von variablen Merkmalsmengen, die in der SBV extrahiert werden, ist die Verwendung der ECET (Expected Case Execution Time) realistischer. Allerdings ist die Schätzung der ECET immer mit dem Risiko verbunden, nicht eingehalten zu werden. Aus diesem Grund wird der TAFT Scheduler verwendet, der das nicht rechtzeitige Beenden einer solchen Task feststellen kann und in der Lage ist, mit einer entsprechenden Ausnahmebehandlung zu reagieren. Eine Möglichkeit der Ausnahmebehandlung ist das Einsammeln aller bisherigen Zwischenergebnisse der Task, und diese als Ergebnis zu verwerten. Dieses Verhalten kann mit Anytime Algorithmen umgesetzt werden.

Für die Auswahl eines geeigneten SBV Algorithmus für die Entfernungbestimmung war die Anpassung an dieses Verhalten maßgeblich. Das konnte beim Block Matching Algorithmus erfolgreich umgesetzt werden. Dazu wurde inspiriert durch die Idee der Gaußpy-

ramide das Korrespondenzproblem in verschiedenen Auflösungsstufen gelöst. Durch geschickte Einschränkung des Suchraumes konnte die Lösung des Korrespondenzproblems auf jeder Stufe effizienter gestaltet werden.

Konzeptionell wurde des Weiteren beschrieben, wie der Algorithmus unter Verwendung von Redundanzen Ressourcen effizienter nutzen kann und es wurde außerdem gezeigt, wie eine weitere Verbesserung der Ergebnisse durch den Einsatz von Sensorfusion herbeigeführt werden kann.

1.4 Aufbau der Diplomarbeit

Die Diplomarbeit ist wie folgt gegliedert. In Kapitel 2 werden die Grundlagen zur Stereobildverarbeitung einerseits sowie der TAFT Scheduler erläutert. Anschließend werden in Kapitel 3 verwandte Arbeiten zu den Schwerpunkten dieser Diplomarbeit vorgestellt und es wird diskutiert, inwieweit die Eigenschaften hinsichtlich der Nutzbarkeit im Rahmen dieser Arbeit verwendet werden können. Kapitel 4 stellt dann das Konzept für die Entwicklung eines Anytime Block Matching Algortihmus für die SV in drei Stufen dar. Danach gibt Kapitel 5 einen Überblick über die unter Linux in *C++* vorgenommene Implementierung. In Kapitel 6 werden Ergebnisse präsentiert und erläutert. Die Arbeit schließt mit Kapitel 7, in dem eine Zusammenfassung und ein Ausblick auf mögliche Folgearbeiten gegeben wird.

2 Grundlagen

In diesem Kapitel werden die zum Verständnis notwendigen Grundlagen erläutert. Das Verstehen des Block-Matching (BM) Algorithmus sowie des TAFT (Time Aware, Fault Tolerant) Schedulers sind für den in dieser Diplomarbeit beschriebenen Stereo Vision Algorithmus erforderliche Voraussetzungen.

Im folgenden wird allgemein erklärt, was unter Stereobildverarbeitung (SBV) zu verstehen und wie ein Standard SBV System aufgebaut ist. Nachfolgend stellt der Hauptteil den Block Matching Algorithmus dar, welcher als Basis für das weitere Vorgehen dient. Im Anschluss daran wird die Funktionsweise des TAFT Schedulers erläutert.

2.1 Stereobildverarbeitung

Der Begriff Stereobildverarbeitung definiert sich aus den Wörtern Stereo und Bildverarbeitung. Die Bildverarbeitung ist die Auswertung und Aufbereitung von Bildsignalen und Bildsignalfolgen. Bildsignale stellen hier zweidimensionale Funktionen des Ortes und der Zeit dar, denen eine spezifische physikalische Größe zugeordnet ist [7]. Das zweite Wort Stereo – Zwei – beschreibt in Anlehnung an das menschliche Sehen die Grundidee der SBV. Sie lautet wie folgt:

> *Wird eine Szene im dreidimensionalen Raum aus zwei unterschiedlichen Blickrichtungen betrachtet, ist die Bestimmung von Tiefeninformationen der betrachteten Objekte auf Grund geometrischer Beziehungen möglich* [8].

Werden beide Begriffe verknüpft, wird ein Prozess beschrieben, bei dem ausgehend von zwei Bildern, welche die gleiche Szene von verschiedenen Standorten zeigen, Tiefeninformationen der sichtbaren Objektoberflächen bestimmt werden können.

Die Berechnung von Tiefeninformationen ist aber nur der letzte Schritt eines SBV Systems. Es besteht im allgemeinen aus 4 Arbeitsschritten. Die Arbeitsschritte im einzelnen sind:

1. Bildaufnahme
2. Vorverarbeitung des Bildpaares
3. Korrespondenzanalyse
4. Tiefenwertbestimmung.

In den meisten Fällen findet die Bildaufnahme unter Zuhilfenahme von zwei Kameras statt, damit eine Szene zur gleichen Zeit aufgenommen werden kann. Das ist von entscheidender Bedeutung, denn dynamische Objekte in der Szene müssen sich bei beiden Bildern des Stereobildpaares an der gleichen Weltposition befinden, um eine korrekte Tiefenberechnung zu gewährleisten.

Eine bewährte Kameraanordnung für die SBV ist die kanonische Konfiguration (Abbildung 2-1). Bei diesem Aufbau sind beide Kameras C_l und C_r so in einer Ebene angeordnet, dass die Grundlinien ihrer Bilder horizontal und in derselben Höhe liegen. Des Weiteren gilt für die optischen Achsen der Kameras $\vec{c}_l$ und $\vec{c}_r$, dass sie senkrecht zu dieser Ebene stehen. Daraus ergibt sich außerdem

$$\vec{c}_l \parallel \vec{c}_r . \tag{2.1}$$

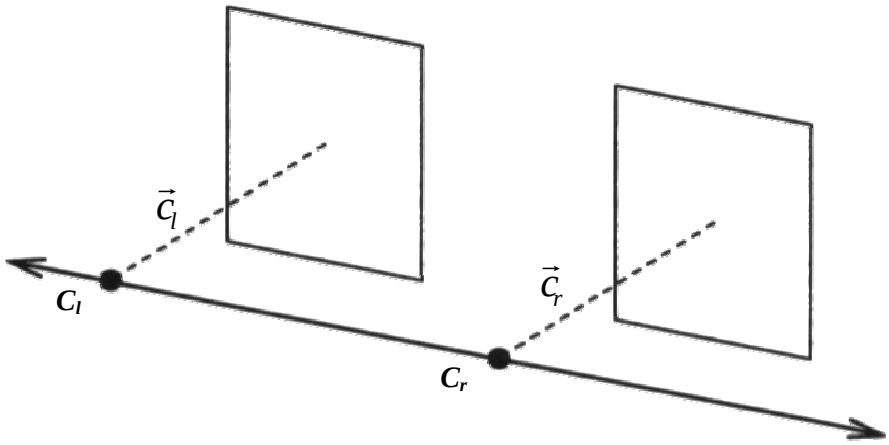

Abbildung 2-1: Kanonische Kamerakonfiguration für Stereo Vision

Die kanonische Kamerakonfiguration hat einige positive Effekte, die sie für den Einsatz mit mobilen Robotern prädestinieren. Das sind, bedingt durch den einfachen Aufbau, die Suche nach Gemeinsamkeiten in den Stereobildpaaren und die anschließende Tiefenrekonstruktion.

Nach der Bildaufnahme folgt im Allgemeinen eine Vorverarbeitung der Bilder. Sie ist bei der SBV von besonderer Wichtigkeit. Zum einen müssen die durch die Bildaufnahme durch fehlerhafte Sensorelemente entstandenen Fehler korrigiert werden, zum anderen das in dem Bildpaar vorhandene hochfrequentes Rauschen herausgefiltert werden. Das kann zum Beispiel durch elektrische Störfelder im Rechner bei der Bilddigitalisierung auftreten. Durch die Anwendung von linearen verschiebungsinvarianten Filtern werden diese Probleme aber weitestgehend behoben, sodass eine Korrespondenzanalyse möglich wird. Eine genaue Beschreibung dieser Filter im Zusammenhang mit der Vorverarbeitung findet sich in Kapitel 4.3.1. Dort sind sie sehr detailliert in Zusammenhang mit der Echtzeitproblematik erläutert. An dieser Stellte sei so viel erwähnt, dass durch diese Filter die Bilder ‚unschärfer' werden und damit insgesamt störungsfreier und rauschärmer. Optional muss der Einsatz von Methoden vorgesehen werden, die Linsenverzerrungen und daraus entstehende Abbildungsfehler korrigieren. In dieser Arbeit fällt dieses Problem jedoch nicht an, da die Kameraverzerrungen so minimal sind, dass sie nicht ins Gewicht fallen.

Auf die Vorverarbeitung folgt die Korrespondenzanalyse. Sie stellt die Suche nach Gemeinsamkeiten zwischen den Bildern dar. Das ist der wichtigste Operationsschritt in der SBV, denn die korrespondierenden Pixel in den Bildern beschreiben die selben Punkte im Raum. Dies ist natürlich nur möglich, wenn die beobachtete Szene genügend differenzierte Texturen aufweist. Werden korrespondierende Pixel gefunden, kann mittels Anwendung des Strahlensatzes unter Benutzung von Kameraparametern (Brennweite, Größe des Aufnahmechips) und der Kamerakonfiguration (Abstand zwischen beiden Kameras, gegenseitige Orientierung) die Entfernung des Punktes, der in beiden Bildern als gleich markiert wurde, bestimmt werden.

Bevor Gemeinsamkeiten aber gesucht werden können, muss definiert werden, wann zwei Punkte des Bildpaares gleich aussehen, also gleich sind. In der SBV wird hier für die Kor-

respondenzanalyse die Annahme getroffen, das korrespondierende Pixel einen ähnlichen Intensitätswert besitzen.

Sehr häufig werden für die Korrespondenzanalyse korrelationsbasierte Methoden verwendet. Ein Verfahren aus dieser Klasse, welches auch hier Verwendung gefunden hat, ist das *Block-Matching (BM)*. Dieser Algorithmus hat seinen Namen auf Grund seiner Lösungsstrategie bekommen, welche die Definition von gleichen Pixeln zwischen den Bildern auf die Gleichheit von Blöcken ausdehnt. Die Annahme, das ein Pixel auf Basis seines Intensitätswertes eindeutig zugeordnet werden kann, wird als nicht ausreichend bewertet. Die Ursache ist, das normalerweise eine große Anzahl identischer Intensitätswerte in einem Bild existieren. Aus diesem Grund werden zur Lösung mehrere benachbarte Pixel in einem Fenster, auch Block genannt, zusammengefasst, um eine höhere Eindeutigkeit zu erhalten. Die Größe des Fensters wird, standardmäßig auf 8x8 Pixel festgelegt, wie in der Literatur beschrieben, aus Experimenten ermittelt worden ist [8]. Die Zuordnung zwischen den Pixeln erfolgt anschließend anhand eines Ähnlichkeitsmaßes zwischen den Intensitätswerten der Fenster.

Das BM Verfahren arbeitet im Detail nach folgenden Muster. Als erster Verarbeitungsschritt wird ein Bild des Bildpaares (zum Beispiel das linke) in eine konstante Anzahl von gleich großen Blöcken aufgeteilt. Die Suche nach einem korrespondierendem Block im rechten Bild wird nur für die festgelegten Blöcke des linken Bildes durchgeführt. Gesucht werden muss im rechten Bild, dank der kanonischen Kamerakonfiguration, nur in horizontaler Richtung, das heißt nur in einer Zeile (Abbildung 2-2). Als Maß für die Ähnlichkeit zweier Blöcke wird der mittlere quadratische Fehler *MSE* (mean square error) zwischen den Intensitätswerten der Pixel innerhalb der entsprechenden Blöcke verwendet.

Die Intensitätsfunktionen des linken bzw. rechten Bildes werden mit E_L bzw. E_R bezeichnet. Das Ähnlichkeitsmaß ist für einen Offset Δ, der die Differenz $(x_L\text{-}x_R)$ zwischen den Spaltenpositionen im rechten und im linken Bild angibt, und eine Blockgröße von n x m Pixeln durch

$$MSE(x, y, \Delta) = \frac{1}{n \cdot m} \sum_{i=0}^{n-1} \sum_{j=0}^{m-1} \left| E_L(x+i, y+j) - E_R(x+i+\Delta, y+j) \right|^2 \qquad (2.2)$$

definiert.

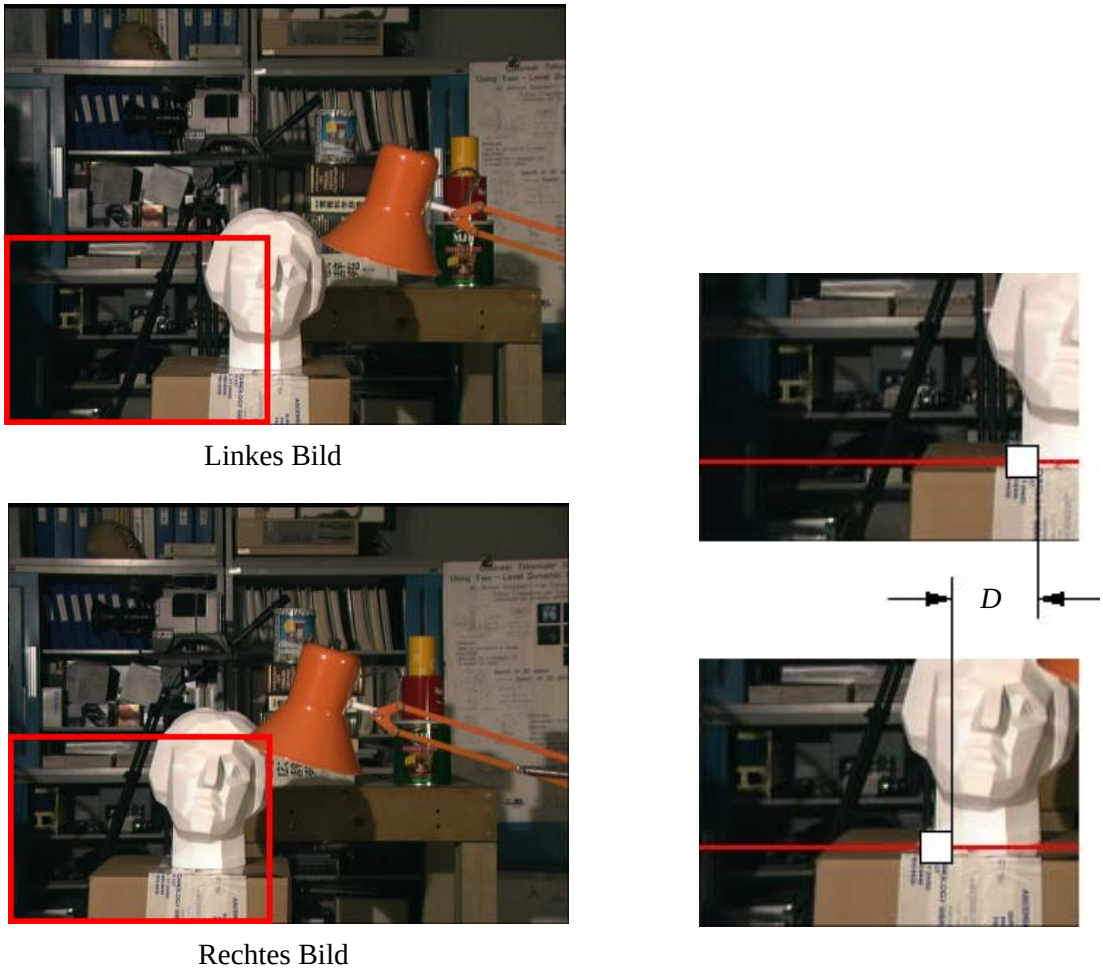

Abbildung 2-2: Block-Matching-Verfahren; der weiß markierte Block im linken Bild wird im rechten Bild wieder gefunden; die Disparität *D* ergibt sich aus der Differenz der Positionen der Blöcke aus den Bildern

(x, y) bezeichnet hier jeweils die linke obere Ecke eines Blockes im linken Bild. Die Disparität *D* zwischen den Blöcken ist definiert durch den Abstand zwischen den Positionen der Blöcke, die die minimale Abweichung aufweisen.

Der Suchbereich in horizontaler Richtung im rechten Bild wird zusätzlich durch ein Disparitätslimit d_{max} beschränkt. d_{max} ergibt sich direkt aus der definierten Einsatzumgebung des

Verfahrens. Je dichter Objekte der beobachteten Szene am Kamerasystem liegen, desto größer ist die Disparität der zu dem Objekt gehörenden Pixel. Durch die Festlegung einer Mindestdistanz lässt sich d_{max} also leicht ermitteln.

Innerhalb des Suchbereiches (zwischen 0 und d_{max}) wird der n x m-große Block punktweise verschoben. Der Verschiebungswert Δ für den die MSE-Funktion ihr Minimun annimmt, bestimmt den so genannten Blockdisparitätswert D. Ein Disparitätswert ist ein Vektor, welcher die Änderung der Lage eines Blockes (später eines Bildpunktes) zwischen den Bildern des Bildpaares beschreibt. Er ist nur dann eindeutig bestimmt, wenn die MSE-Funktion im Suchbereich ein eindeutiges Minimum besitzt. In den Fällen, in denen kein eindeutiges Minimum existiert, wird ein zusätzliches Entscheidungskriterium verwendet.

Unter der Annahme, dass sich die Disparitätswerte benachbarter Blöcke nur geringfügig unterscheiden, werden alle Disparitätswerte, für die die MSE-Funktion ein Minimum annimmt, mit dem Wert des benachbarten Blocks verglichen. Ausgewählt wird die Disparität mit dem geringsten Unterschied zu der Disparität des Nachbarblockes.

Das Ergebnis der Anwendung des BM Verfahrens ist eine Disparitätsmatrix, in der jeweils Blöcke fester Größe einen identischen Wert besitzen. Dieses Ergebnis lässt sich unter Verwendung eines Pixelselektionsverfahrens weiter verfeinern, sodass für jedes Pixel ein Disparitätswert bestimmt werden kann. Das Verfahren von T. Reuters [25] zum Beispiel setzt sich aus drei Verarbeitungsschritten zusammen:

1. Anwendung des Medianoperators auf die Disparitätswerte der Blöcke
2. Pixelselektion und
3. Anwendung des Medianoperators auf die Disparitätswerte, die für jedes einzelne Pixel bestimmt wurden.

Der Medianoperator bestimmt in einer Menge von Werten denjenigen Wert, der bei deiner sortierten Reihenfolge der Werte die mittlere Position einnehmen würde. Als erster Schritt wird zunächst der Medianoperator auf die Blockdisparitäten innerhalb einer 3x3-Blockumgebung angewendet, um einzelne Ausreißer in den Disparitätswerten zu eliminieren. Anschließend wird bei der Pixelselektion die Disparität für jeden Pixel (x', y') eines

Blockes unter Verwendung der Disparitätswerte dieses und der benachbarten Blöcke bestimmt.

Für die Bestimmung der Disparität eines einzelnen Pixels an der Position (x', y') werden die Differenzen *DIFF(k)* zwischen dem Intensitätswert des linken Bildes an der Position (x', y') und den Intensitätswerten des rechten Bildes an den Positionen $(x' + D(k), y')$ für alle Disparitäten *D(k)* mit *(1<k<9)* aus der 3x3-Blockumgebung gebildet (vgl. dazu Abbildung 2-3):

$$DIFF(k) = \left| E_R(x', y') - E_L(x' + D(k), y') \right| \qquad \text{mit } k=1,...,9. \tag{2.3}$$

Der Disparitätswert $DISP(x', y')$ ist definiert durch den Wert *D(k)*, für den die Betragsdifferenz *DIFF(k)* ihr Minimum annimmt. Bei der Anwendung der Pixelselektion auf jeden Bildpunkt ergibt sich eine Disparitätsmatrix in Originalbildgröße. Abschließend wird der Medianoperator auf die mittels Pixelselektion berechneten Disparitätswerte angewendet.

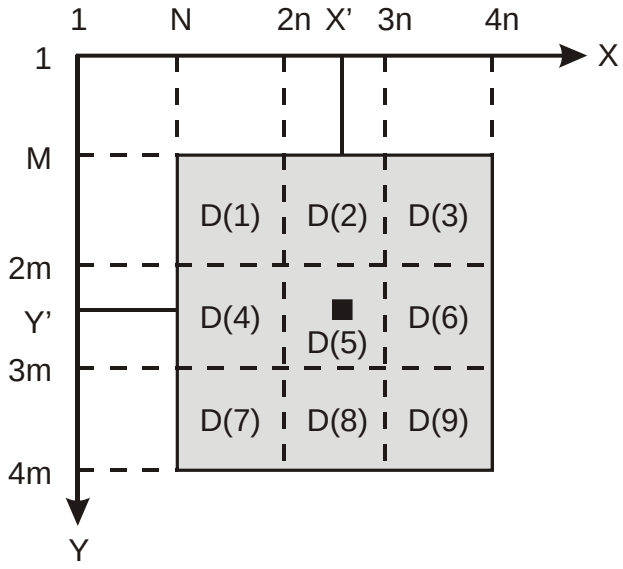

Abbildung 2-3: Pixelselektion des Disparitätswertes an der Position (x', y') unter Verwendung der Blockdisparitäten *D(k)* in der Achterblocknachbarschaft (aus [8])

Der letzte Schritt in einem SBV System ist die Bestimmung der Entfernung jedes einzelnen Punktes, für den zwei Pixel in dem Bildpaar als gleich identifiziert werden konnten. Dieses Problem ist unter der Annahme einer kanonischen Kamerakonfiguration, wie sie hier verwendet wurde, ohne Komplikationen lösbar. Abbildung 2-4 stellt die Vorgehensweise schematisch dar. Die zwei für die Aufnahmen verwendeten Kameras C_l und C_r haben den Abstand *2h* voneinander. In den beiden entsprechenden Bildern wurden die Punkte P_l und P_r als gleich identifiziert. Sie sind die Projektionen des Weltpunktes *P* mit den Koordinaten (*x, y, z*). Das Koordinatensystem ist folgendermaßen definiert. Die *z*-Achse repräsentiert die Entfernung von den Kameras. An den Kameras gilt somit *z=0*. Die horizontale Entfernung wird durch die *x*-Achse definiert. *x=0* ist die Position in der Mitte zwischen den beiden Kameras. Jedes Bild hat zusätzlich ein lokales Koordinatensystem, welches die Bildmitte als Ursprung verwendet. Die x-Koordinaten der Projektionen P_l und P_r sind dementsprechend x_l für das linke, x_r für das rechte Bild. Über eine einfache Translation kann aus dem globalen *x* dann x_l *und* x_r berechnet werden. *y* stellt die vertikalen Abstände dar, wird aber für die Herleitung der Formel für die Tiefenberechnung nicht benötigt.

Dass zwischen x_l und x_r eine Disparität besteht ist auf Grund verschiedener Kamerapositi-

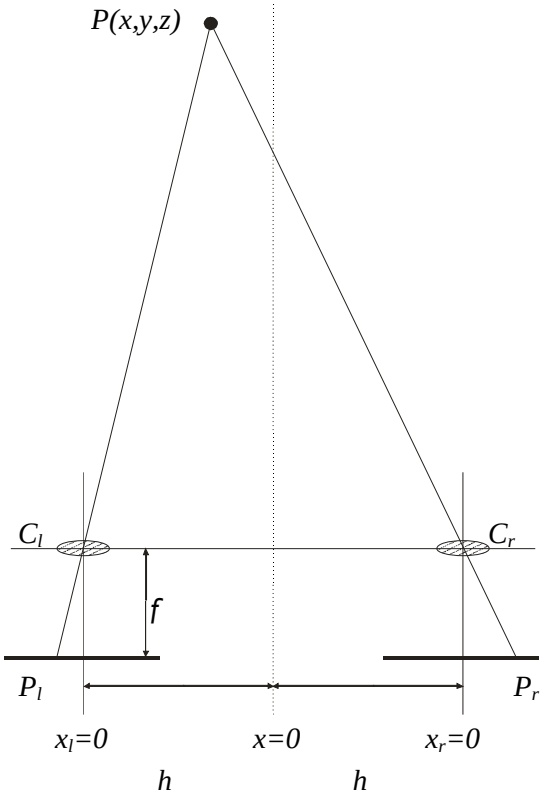

Abbildung 2-4: Geometrie der SV mit kanonischer Kamerakonfiguration

onen klar. Daher gilt $|x_l - x_r| > 0$. Um die *z*-Koordinate von *P* zu berechnen, wird der Strahlensatz angewendet.

Die Strecken P_l, C_l und C_l, *P* sind, auf Abbildung 2-4 bezogen, die Hypotenusen von ähnlichen rechtwinkligen Dreiecken. Ähnlich heißt, dass ihre Seiten unterschiedlich lang sind, sie aber in den Winkeln übereinstimmen. Des Weiteren ist anzumerken, dass *h* und *f* immer positiv sind. Ebenso die *z*-Koordinate, sie ist im Gegensatz zu *x*, x_l und x_r, welche positiv und negativ sein können, immer positiv. Für die Berechnungen ist außerdem vereinfacht angenommen worden, dass beide Kameras C_l und C_r die gleiche Brennweite *f* haben. Eine Anpassung an jeweils eigene Werte ist auf Basis dieser Formeln natürlich trotzdem möglich. Für die linke Seite von Abbildung 2-4 ist nun folgende Ableitung möglich:

$$\frac{x_l}{f} = -\frac{h+x}{z} \tag{2.4}$$

Folglich gilt für die rechte Seite:

$$\frac{x_r}{f} = \frac{h-x}{z}. \tag{2.5}$$

Werden 2.4 und 2.5 nach –*x* aufgelöst, können sie zu

$$\frac{x_l}{f} \cdot z + h = \frac{x_r}{f} \cdot z - h \tag{2.6}$$

zusammengefasst werden. Nach *z* umgestellt, ergibt sich

$$z = \frac{2hf}{x_r - x_l}. \tag{2.7}$$

Letztendlich sind die Koordinaten des beobachteten Punktes *P* nur von der Disparität $x_r - x_l$ abhängig. Natürlich kann $x_r - x_l = 0$ auch auftreten. Ist das der Fall, so folgt daraus $z \to \infty$. Das bedeutet das, dass der Punkt *P* von den beiden Kamerapositionen C_l und C_r aus betrachtet, auf Grund der begrenzten Kameraauflösung und der begrenzten Genauigkeit der Kameraparameter, jeweils auf die gleiche Stelle der Bilder projiziert wird. Daraus

folgt, es existiert ein $z_{\max} < \infty$ bei $x_l = x_r$. Somit ist durch d_{max} und z_{max} der Abbildungsbereich der Kamera bezüglich der ermittelbaren Entfernung klar definierbar eingegrenzt.

Mit der Berechnung von Entfernungsdaten aus der Disparitätsmatrix, hat das SBV System seine Aufgaben erfüllt. Abbildung 2-5 zeigt zusammenfassend den BM Algortihmus mit anschließender Pixelselektion, wie er in [8] zu finden ist.

```
begin
    unterteile linkes Bild in Blöcke der Größe n x m;
    for (jeden Block im linken Bild) do begin          {BM}
        initialisiere min und D;
        setze (x, y) gemäß linker oberer Ecke des aktuellen Blockes;
        for Δ := 1 to d_max do begin
            berechne MSE(x, y, Δ);
            if MSE(x,y, Δ) < min then
                min := MSE(x,y, Δ);        D := Δ
            else
                if MSE(x, y, Δ) = min then
                    D := Disparität mit geringstem Unterschied zum Disparitäts-
                    wert des Nachbarblockes
                end {if}
            end {if};
            speichere Blockdisparitätswert
        end {for}
    end {for};
    filtere Blockdisparitätswerte mit Medianoperator;
    for (jedes Pixel im linken Bild) do begin          {Pixelselektion}
        for k := 1 to 9 do begin berechne DIFF(k);
            bestimme Wert D für den DIFF(k) minimal ist;
            DISPARITÄT(Pixelposition) := D
        end {for}
    end {for};
    filtere Matrix DISPARITÄT mit Medianoperator
end
```

Abbildung 2-5: Block Matching Algorithmus mit Pixelselektion (aus [8])

Die weitere Verarbeitung der Daten, sei es für eine Objekterkennung oder Pfadplanung auf Basis von Distanzen möglicher Hindernisse, kann anschließend in einem nachgeschalteten Algorithmus übernommen werden. Damit diese Algorithmen ihre Aufgaben korrekt erfüllen können, muss ihnen das Vorhandensein von Eingangsdaten gesichert werden können. Dafür sind, wie im folgenden Kapitel erläutert wird, TAFT Schedulers verantwortlich.

2.2 TAFT Scheduler

Scheduler sind ein wichtiger Bestandteil von Betriebssystemen. Sie bestimmen die Strategie, wie einzelnen Tasks Rechenzeit zugeteilt wird. Bei Echtzeitbetriebssystemen ist der Scheduler zusätzlich für die Garantie der rechtzeitigen Ausführung einer Task verantwortlich, wenn die Taskmenge, zu der auch dieser Task gehört, als machbar eingestuft worden ist. Voraussetzung dafür ist, das von den Tasks die maximalen Laufzeiten bekannt sind. Nur dann kann der Scheduler, wenn der Schedule machbar ist, eine Garantie geben, den Task bis zu seiner Deadline abarbeiteten zu können. Allerdings kann bei dem hier betrachteten Problem eine derartige Vorhersage der maximalen Laufzeit nur schlecht getroffen werden. Des Weiteren würde sie weit über der realen Ausführungszeit liegen, was eine nicht akzeptable Systemauslastung zur Folge hätte. Abhilfe schafft an dieser Stelle der TAFT (Time Aware, Fault Tolerant) Scheduler [21].

Der TAFT Scheduler ist ein echtzeitfähiger Scheduler, der die rechtzeitige Ausführung dynamischer Tasks garantieren soll. Die Idee des TAFT Schedulers basiert auf der des TP Schedulings [20], welches jede Task als Task Paar darstellt. Bei diesem Konzept wird nur die Ausführung der Subtask garantiert, welche die minimal notwendige Funktionalität implementiert. Für diese Subtask ist die maximale Ausführungszeit bekannt. Für die zweite Subtask, die normalerweise ausgeführt wird, wird eine statistische Laufzeit [18] angenommen. Daraus abgeleitet besteht beim TAFT Scheduler die Task aus einem Main Part (MP) und einem Exception Part (EP). Der MP implementiert die Hauptfunktionalität. Für den Fall, dass der MP abgebrochen wird, beinhaltet der EP eine Ausnahmebehandlung. Sie gewährleistet die Herstellung eines sicheren Systemzustandes. Aus diesem Grund wird beim EP die WCET (Worst Case Execution Time) als Laufzeitparameter verwendet, damit ihr immer genügend Ressourcenzeit zur Verfügung steht. Im Gegensatz dazu wird für den MP die durchschnittlich erwartete Ausführungszeit, die ECET (Expected Case Execution Time), angenommen. Aus den Ausführungszeiten von MP und EP ergibt sich zusammen mit der Periodendauer der Task eine Deadline für dieses Paar. Innerhalb dieser Deadline garantiert TAFT entweder die vollständige Ausführung des MP oder mindestens des vollständigen EP. Das Abbrechen des MP einer Task muss daher spätestens zum Zeitpunkt $t_{deadline}$-*WCET(EP)* erfolgen.

Durch die ECET, welche eine realistische Schätzung für die Laufzeit der Task ist, entsteht eine bessere Systemauslastung als beispielsweise mit dem EDF (Earliest Deadline First) Scheduler, dem Standardscheduler von RTLinux. Zum TAFT Scheduler gehört des Weiteren eine Monitoring Komponente, mit der die Ausführungszeiten der Task angepasst werden können [19]. Insgesamt kann TAFT die Performance im Vergleich zum EDF Scheduler erheblich steigern. Nicht nur, dass EDF mit den WCETs der Tasks, welche sich unter Umständen schwer ermitteln lassen [16], arbeitet, zusätzlich gilt meist noch:

$$WCET_{MP} > ECET_{MP} + WCET_{EP}. \qquad (2.8)$$

Die WCET des EP muss zwar noch bestimmt werden, doch diese sollte im Vergleich zum MP sehr kurz und einfach bestimmbar sein. Infolgedessen gilt bei normalen Anwendungen:

$$ECET_{MP} >> WCET_{EP}. \qquad (2.9)$$

Das TAFT Konzept, in Abbildung 2-6 kurz schematisch dargestellt, nutzt damit die vorhandenen Prozessorressourcen besser aus.

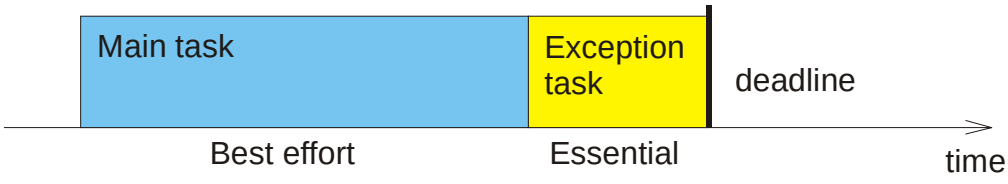

Abbildung 2-6: Taskform für den TAFT Scheduler

Tasks, die mit TAFT geschedult werden, arbeiten stets periodisch. Dabei kommen für den MP und den EP unterschiedliche Schedulingstrategien zum Einsatz. Das Ziel ist es, den MP so lange wie möglich ausführen zu können und den EP, wenn er denn gebraucht wird, zum spätest möglichen Zeitpunkt zu starten. Somit entsteht eine bessere CPU Ausnutzung. Deshalb kommt für die MPs ein EDF und für die EPs ein LRT (Latest Release Time) Scheduling zur Anwendung. LRT ist auch unter Reverse Order EDF bekannt, da es, wenn Startzeit und Deadline vertauscht werden, wie EDF arbeitet. Die Prioritäten der EPs werden so vergeben, dass EPs, die eine spätere Deadline besitzen, eine höhere Priorität haben.

EPs mit früherer Deadline müssen sich also den EPs mit späterer Deadline unterordnen und gegebenenfalls früher beginnen, damit sie beendet sind, wenn ein höherpriorer EP startet. Des Weiteren besitzen EPs eine höhere Priorität als MPs. Dadurch wird eine Unterbrechung der MPs durch die EPs jederzeit ermöglicht.

Bevor eine Task jedoch vom TAFT Scheduler eingeplant wird, muss sie einen Akzeptanztest erfolgreich absolvieren. Können die ECET des MP und die WCET des EP garantiert werden, ist der Test bestanden. Sonst muss die Task abgelehnt werden. Durch diesen Akzeptanztest wird sichergestellt, dass die Deadlines aller akzeptierten Tasks eingehalten werden können.

Eine Implementierung von TAFT wurde in RTLinux vorgenommen [26]. Die Wahl auf RTLinux fiel auf Grund die freien Verfügbarkeit des Quelltextes, was die Möglichkeit bot, notwendige Modifikationen am Scheduler direkt vorzunehmen. Ein Beispiel für die Struktur einer Thread Einsprungfunktion zeigt Abbildung 2-7.

```
void * thread_fkt(void *arg) {
    pthread_make_periodic_tp (pthread_self(), starttime);
    while (1) {
        pthread_wait_np_ex();
        BEGIN_MAIN_PART
            …
        END_MAIN_PART

        BEGIN_EXCEPTION_PART
            …
        END_EXCEPTION_PART
    }
    return 0;
}
```

Abbildung 2-7: Struktur einer Thread Einsprungfunktion für den TAFT Scheduler

Ein Vorteil ergibt sich durch die Tatsache, das sich MP und EP einer Task im selben Adressraum befinden und intern wie *ein* Thread behandelt werden. Dadurch ist dem EP der Zugriff auf lokale Daten des MP problemlos möglich.

Dieses Konzept des TAFT Scheduling bietet die Verwendung von Anytime Algorithmen an, welche die Grundidee von Imprecise Computation realisieren. Von einem Anytime Algorithmus wird gesprochen, wenn der Algorithmus nach einer kurzen Anlaufzeit jederzeit unterbrochen werden kann und ein Ergebnis liefert. Das heißt im Sinne von Imprecise Computation, je mehr Rechenzeit dem Algorithmus zur Verfügung steht, desto genauer wird das Ergebnis. Dieses Konzept bildet die Grundlage für die Modifikation des BM Algorithmus, wie sie im Hauptteil dieser Arbeit beschrieben ist.

3 Verwandte Arbeiten

Dieses Kapitel zeigt Lösungen, wie sie in aktuellen Verfahren bezüglich der Aufgabenstellung dieser Arbeit eingesetzt werden. Es wird erörtert, inwieweit diese Methoden hier einsetzbar sind und wo Probleme auftreten.

3.1 Stereobildverarbeitung

Aktuelle Verfahren der Stereobildverarbeitung im Bereich mobiler Roboter, die für echtzeitbezogene Applikationen genutzt werden können, setzen klar bestimmte Applikationsumgebungen voraus und benötigen meist spezielle Hardware. Beispielsweise arbeiten die in [5] und [6] beschriebenen Systeme mit Signalprozessoren für die Bildverarbeitung und sind für die Analyse von Straßenszenen angepasst. Diese Systeme dienen der Korrektur von Trägheitssensor- und Odometriedaten eines Autos. Damit das Vorliegen der Ergebnisse in diesem System vorhergesagt werden kann, wird mit den Laufzeiten der vollständigen Suche gearbeitet. Da die vollständige Suche für das gesamte Bild aber bei weitem zu lange im Sinne der Steuerung von Autos dauer würde, werden für die Entfernungsberechnung kleine Teilbereiche des Bildes benutzt. Das sind meist markante Stellen, zum Beispiel Begrenzungspfeiler, Schilder oder Straßenmarkierungen, wie sie im Straßenverkehr häufig vorkommen. In [1], [2] und [3] werden echtzeitfähige Stereobildverarbeitungsalgorithmen auf mobilen Robotern zur frühen Hindernisserkennung verwendet, um ihnen gegebenenfalls rechtzeitig ausweichen zu können. Auch hier wird ein Hardwareboard für die Bildberechnungen benutzt. Zusätzlich sind in [2] die Bilder auf 256x45 Pixel zu Gunsten einer akzeptablen Geschwindigkeit beschränkt worden.

Das größte Problem der SBV besteht im Finden der richtigen Korrespondenzen zwischen den Punkten in den Bildern, um dann mit Hilfe der Kamerageometrie die Entfernung der Punkte ermitteln zu können. Für das Lösen des Korrespondenzproblems existieren zwei grundlegende Prinzipien.

Das erste Prinzip, auf das auch alle bisher genannten Verfahren basieren, arbeitet nach dem lokalen Ansatz. Lokaler Ansatz heißt, die Korrespondenz der betrachteten Pixel wird auf

Grund einer kleinen Anzahl sie umgebender Pixel bestimmt. Es werden dadurch iterativ die Pixel des einen Bildes den Pixeln des zweiten Bildes zugeordnet. Damit ergibt sich aber auch das Problem, dass durch diese Lokalität Zweideutigkeiten auftreten können. Das ist zum Beispiel bei gleichfarbigen Flächen oder Verdeckungen der Fall. Diese Probleme sind im Gegensatz dazu bei einem globalen Ansatz weniger sensitiv. Dieser versucht, alle Pixel entweder einer Zeile oder des gesamten Bildes gleichzeitig zuzuordnen. Der Preis dafür schlägt sich allerdings bei der Berechnungskomplexität nieder. Globale Ansätze sind viel rechenintensiever als lokale, welche aber dennoch durch geschickte Parameterbestimmung sehr effizient gestaltet werden können. Ein Vergleich beider Ansätze und ausführliche Erläuterungen zu diesem Thema können in [8] und [4] nachgelesen werden.

3.2 Echtzeitscheduling

Was heißt zunächst Echtzeit? Die Definition nach DIN 44330 [12] lautet:

> *Ein Betrieb eines Rechensystems, bei dem Programme zur Verarbeitung anfallender Daten ständig betriebsbereit sind, derart, dass die Verarbeitungsergebnisse innerhalb einer vorgegebenen Zeitspanne verfügbar sind. Die Daten können je nach Anwendungsfall nach einer zeitlich zufälligen Verteilung oder zu vorherbestimmten Zeitpunkten anfallen.*

Der wichtigste Punkt in dieser Definition ist, und das ist auch der Grundgedanke der Echtzeit, dass Verarbeitungsergebnisse nach einer von der Umgebung definierten Zeit, also rechtzeitig, zur Verfügung stehen. Rechtzeitigkeit wird erreicht, indem ein Scheduler, basierend auf bestimmten Taskparametern, wie im allgemeinen der WCET, versucht einen Schedule zu erstellen, in dem alle Tasks vor ihrer Deadline ausgeführt werden. Das erfolgt nach bestimmten Kriterien. Zum Beispiel kann das die früheste Deadline (EDF Scheduler) oder auch die Periode (Rate Monotonic Scheduler) einer Task sein ([10], [11]). Die für die Planung notwendigen Taskparameter können auf unterschiedlichste Weise und zu unterschiedlichen Zeiten ebenso wie die Erstellung des Schedules ermittelt werden. Tabelle 3-1 gibt hier einen Überblick. Die Frage ist, wann ist was sinnvoll. Bei statischen Schedulingverfahren wird die Analyse und die Bestimmung der Parameter offline durchgeführt. Das

ist beispielsweise bei dem Laufroboter *Katharina* der Fall [13]. Aber wie auch schon in [13] erläutert wird, ergibt sich beim offline Scheduling das Problem, dass sich die Taskplanung zur Laufzeit nicht an aktuelle Bedingungen anpassen kann. So kann die WCET deutlich länger sein, als die im statistischen Mittel ermittelte Laufzeit einer Task. Diese Erscheinung tritt besonders bei heutigen Prozessoren, die Caches und Pipelining benutzen, auf. Damit die Tasks ebenfalls auf diesen Prozessoren in Echtzeit ablaufen können, werden für die WCET sehr konservative Vorschläge unterbreitet. So kann nur dann eine Ausführungsgarantie gegeben werden, wenn auch der schlechteste aller Fälle für die Ausführung in Betracht gezogen wird. Das heißt die Laufzeitmessungen müssen ohne Caches oder sonstige prozessorspezifische Beschleunigungen erfolgen. Durch dieses pessimistische Vorgehen wird natürlich sehr viel Prozessorleistung verschenkt. Zu Problemen bei der Bestimmung der maximalen Tasklaufzeit wird in [14], [15] und [16] ausführlich eingegangen.

Schedulingverfahren	Planung	Laufzeitanalyse
Statisch	Offline	Offline
Dynamisch, nicht adaptiv	Online	Offline
Dynamisch, adaptiv	Online	Online

Tabelle 3-1: Einteilung von Schedulingverfahren nach Plan ung und Laufzeitanalyse (aus [13])

Auf der anderen Seite stehen aber die Anforderungen der mobilen Robotik. So müssen WCETs für Sensortasks im Millisekunden, höchstens im kleinen Sekundenbereich liegen, ansonsten wäre der Einsatz beispielsweise im Straßenverkehr [6] überhaupt nicht denkbar, denn die mögliche Bewegungsgeschwindigkeit mobiler Roboter ist unter anderem wesentlich von der Geschwindigkeit ihrer Umweltwahrnehmung und die damit verbundene rechtzeitige Reaktion auf Ereignisse abhängig. Das letzte Kapitel zeigte, dass dieses Hindernis zumeist durch Brute Force Lösungen in Form von sehr leistungsstarken Rechnern überwunden wird.

Eine kostengünstigere Variante wäre es, könnten als Taskparameter Ausführungszeiten verwendet werden, die mehr der Realität entsprechen. Dadurch wäre es möglich, die ver-

fügbaren Ressourcen effizienter zu nutzen. Fehlertolerante statische Schedulingverfahren bieten hier eine denkbare Lösung. Bei diesen Verfahren wird eine funktionale Redundanz bei der Planung der Task ausgenutzt [17]. Es wird für die im Normalfall auszuführende Task von einer erwarteten Ausführungszeit, der Expectet Case Execution Time (ECET), ausgegangen [18]. Kann die Task nicht rechtzeitig beendet werden, stehen verschiedene Methoden zur Verfügung, diese Ausnahmesituation zu behandeln. Eine Möglichkeit das zu erreichen ist das Verfahren der ungenauen oder schrittweisen Berechnung (Imprecise Computation [9]). Hier wird in kurzer Zeit ein erstes Ergebnis berechnet, welches dann schrittweise verbessert wird. So kann die Task nach Vorliegen des ersten Ergebnisses auch vor ihrer vollständigen Ausführung abgebrochen werden und es liegt in jedem Fall ein verwendbares Ergebnis vor.

Bezogen auf das Thema dieser Diplomarbeit würde bei SBV zu vermuten sein, das statisches Schedulingverfahren genügen. Aber in Hinblick auf das zweite bestehende Problem (redundante Sensorinformation), ist davon auszugehen, dass die Taskparameter nicht im Voraus bekannt oder konstant sind. Der Grund dafür ist, das erst zur Laufzeit bekannt ist, wie viel Information von welchen Sensoren verwendet werden kann. Insbesondere bei der SBV kann die extrahierbare Merkmalsmenge nicht geschätzt werden, da die Einsatzumgebungen nicht im Voraus bekannt sind. Es ist also notwendig, auch online eine Taskplanung durchführen zu können. Für diese Probleme kommen dynamische Schedulingverfahren zum Einsatz. Im Gegensatz zu statischen Schedulingverfahren, welche die Machbarkeit der Taskmenge überprüfen, greifen dynamische Scheduler auf einen Akzeptanztest zurück. Es wird überprüft, ob die ankommende neue Task zusätzlich zu den bereits zur Ausführung akzeptierten Tasks einplanbar ist. Fällt der Akzeptanztest positiv aus, kann die Ausführung der Task garantiert werden. Aber auch eine Ablehnung ist ein mögliches Ergebnis. Tabelle 3-2 gibt einen Überblick zur Einordnung der Schedulingverfahren mit Bezug auf den Akzeptanztest.

Auch dynamische Schedulingverfahren arbeiten mit der WCET, diese ist aber wie oben bereits erwähnt eine äußerst ungenaue Nachbildung der Ausführungszeit einer Task. Die effiziente Ausnutzung von Prozessorressourcen kann nur mit realistischen Schätzungen durch ein Monitoring [19] ermittelt werden. Diese so genannten adaptiven Scheduler tref-

fen natürlich nicht die exakte Ausführungszeit einer Task. Aus diesem Grund ist es notwendig, im Falle einer zu kurz ermittelten Ausführungszeit, eine Fehlerbehandlung zu integrieren. Das kann zum Beispiel durch das bereits besprochene TP Scheduling erfolgen, welches mit dem TAFT (Time Aware, Fault Tolerant) Scheduler [21] umgesetzt wurde (siehe dazu Kapitel 2.2).

	Statisches Scheduling	Dynamisches Scheduling	
		Nicht Adaptives	Adaptives
Ohne Garantie bzw. Akzeptanztest	Keine Echtzeit, nur best effort		
Mit Garantie bzw. Akzeptanztest / ohne Fehlertoleranz	z.B. rate monotonic	z.B. Spring Kernel	Nicht sinnvoll
Mit Garantie bzw. Akzeptanztest / mit Fehlertoleranz	z.B. kalenderbasiert + Task Pair Scheduling	z.B. EDF + Task Pair Scheduling	Fehlertoleranz erforderlich, z.B. TAFT

Tabelle 3-2: Einordnung verschiedener Schedulingverfahren (aus [13])

Der TAFT Scheduler macht es sinnvoll, die Main Parts als Anytime Algorithmen umzusetzen, welche die Grundidee von Imprecise Computation realisieren. Ein Anytime Algorithmus ist ein Algorithmus, der nach einer kurzen Anlaufzeit jederzeit unterbrochen werden kann und trotzdem in der Lage ist, ein Ergebnis zu liefern. Das heißt ähnlich wie bei Imprecise Computation, je mehr Rechenzeit dem Algorithmus zur Verfügung steht, desto genauer wird das Ergebnis. Es heißt aber auch, es kann nach sehr kurzer Vorlaufzeit immer von vorliegenden Ergebnissen ausgegangen werden. Der SBV Algorithmus muss also so arbeiten, dass ein erstes Ergebnis sehr schnell vorliegen und dieses dann gegebenenfalls weiter verbessert werden kann.

4 Echtzeitfähige Stereobildverarbeitung

In diesem Kapitel wird das Konzept zur Lösung der in der Einleitung ermittelten Probleme entworfen. Dafür werden im folgenden die vier Schritte eines SBV Systems, die Bildaufnahme, die Vorverarbeitung, die Korrespondenzanalyse und die Entfernungsberechnung betrachtet. Dabei wird in den einzelnen Phasen auf die Auswahl und Modifizierung der entsprechenden Algorithmen sowie die Vermeidung redundanter Informationen eingegangen. In einem ersten Schritt soll aber zunächst der genaue Versuchsaufbau, auf den sich bezogen wird, dargestellt werden.

4.1 Versuchsaufbau

Für die Entwicklung des Algorithmus wurde eine statische Versuchsumgebung gewählt, das heißt sowohl das Kamerasystem als auch die Objekte in der beobachteten Szene bewegen sich nicht. Wie schon in Kapitel 2.1 erwähnt, wird ein kanonische Kameraaufbau verwendet. Abbildung 4-1 zeigt die Experimentierumgebung. Es werden Kameras vom Typ VCAM-006 der Firma PHYTEC verwendet. Sie sind an einer Experimentiereinrichtung auf der Tischplatte montiert, mit dessen Hilfe sich Kameraabstand und Winkel einstellen lassen. In dieser Arbeit wird der Abstand zwischen den Kameras auf 160 mm und der Winkel auf 0° festgelegt, das heißt ihre optischen Achsen sind parallel (kanonische Konfiguration). Der Framegrabber eGRABBER-2, ebenfalls ein Produkt der Firma PHYTEC, liefert Bilder mit voller PAL Auflösung von 720 x 576 Pixeln [34]. Das Bild wird in der Kamera auf einen Sensorchip der Größe 4,888 x 3,6375 mm² projeziert. Damit entspricht ein Pixel 6,78 x 6,32 µm². Die Objektive haben eine Brennweite von 8 mm. Die Aufnahmen werden nur gering verzerrt, sodass dies vernachlässigt werden kann.

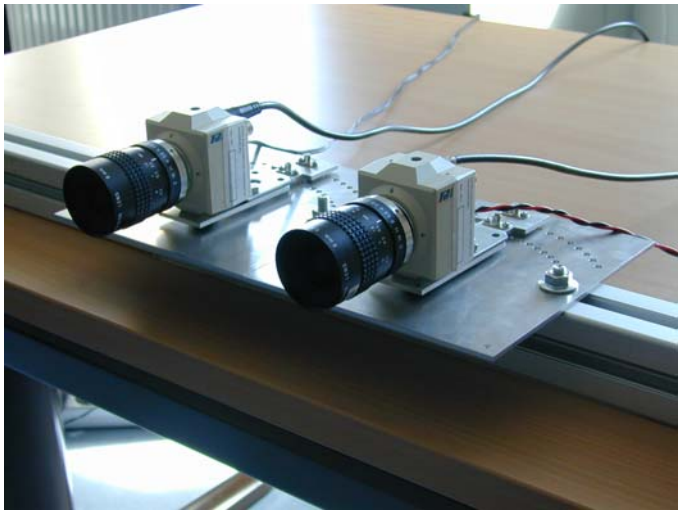

Abbildung 4-1: Versuchsaufbau

4.2 Bildaufnahme

Bilder für die Verarbeitung werden in der Regel durch eine Kamera geliefert. Bei der SBV stellen zwei Kameras die Bildquellen dar. An diese wird neben dem speziellen Aufbau noch zusätzlich die Anforderung gestellt, die Bilder zur gleichen Zeit aufzunehmen. Diese Bedingung ist von entscheidender Wichtigkeit für die SBV, denn um die Entfernung eines Punktes in einer Szene bestimmen zu können, müssen die Abbildungen beider Kameras den Punkt an der selben Raumposition erfassen. Nur so können dynamische Umgebungen, das heißt Umgebungen, in denen sich bewegende Objekte vorhanden sind, korrekt analysiert werden. Da der Schwerpunkt in dieser Arbeit auf der Rechtzeitigkeit der SBV liegt und von einer statischen Umgebung ausgegangen wird, konnte diese Restriktion jedoch gelockert werden. Das war auch notwendig, weil der Framegrabber die Bilder der Kameras nacheinander digitalisieren muss. Für die Aufnahme eines Bildes braucht er 40 ms. Zur Aufnahme des zweiten Bildes muss der Kanal gewechselt, was einige Millisekunden dauert. Insgesamt werden also 80 ms und die Zeit für die Kanalumschaltung benötigt, um zwei Bilder bereitzustellen. Dabei sind die Bilder zeitlich um etwa 40 ms versetzt.

Aber auch das Laden von Bildern von der Festplatte ist für die Entwicklung des Algorithmus von Interesse. Erst dadurch wird ein Test mit unterschiedlichen Parametern ermöglicht, da ein Vergleich der Ergebnisse bei gleichen Eingangsdaten vorgenommen werden kann.

Bezüglich der Modifikation in einen Anytime Algorithmus ist für die Bildaufnahme keine Änderung möglich, da die Laufzeiten der Bildaufnahme konstant sind. Die WCET für das Bereitstellen der Bilder vom Framegrabber sind auf Grund sich nicht ändernder Bildparameter gut bestimmbar. Das Laden der Bilder von der Festplatte wird hier nicht in die Echtzeitbetrachtung mit einbezogen, da es nur für die Entwicklung des Algorithmus notwendig war und für den praktischen Einsatz keine Rolle mehr spielt.

4.3 Bildvorverarbeitung

In Abbildung 4-2 ist der Prozess der Bildaufnahme und der Digitalisierung schematisch dargestellt. Der Weg des Bildes von der Kamera in den Rechner ist durch einige Störstellen gekennzeichnet, die die Bildqualität, die letztendlich für die Weiterverarbeitung zur Verfügung steht, mindern. In diesem Kapitel werden daher Methoden beschrieben, die in der Lage sind, die Bildqualität zu verbessern.

Die erste Störstelle ist der CCD Chip der Kamera. Hier können durch das Auftreten einzelner fehlerhafter Sensorelemente auf dem Chip Pixelfehler im Bild entstehen, beispielsweise falschfarbige Pixel. Auch produzieren CCD Sensoren bei geringer Beleuchtungsstärke ein Farbrauschen. Des Weiteren leidet die Bildqualität durch die analoge Übertragung des Bildsignals von der Kamera zum Rechner auf Grund elektrischer Störungen verursacht durch Leitungswiderstand und elektromagnetischer Strahlung. Die nächste Störung tritt bei der Digitalisierung der Daten am Framegrabber auf. Dazu gehören beispielsweise Quantisierungsfehler oder Effekte durch elektromagnetische Störfelder. Diese Strörungen stellen sich in der Regel durch hochfrequentes Rauschen dar, das heißt die Intensitätswerte der Pixel sind verändert.

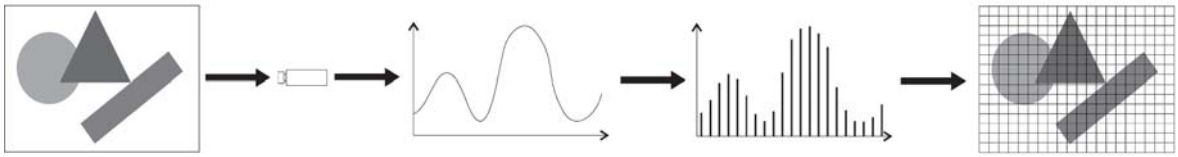

Abbildung 4-2: Bildaufnahme und Digitalisierung

Das Bildrauschen ist dabei auch das schwierigste Problem für das nachfolgende SV Verfahren. Da dort ja farblich gleiche Flächen zwischen den Bildpaaren gefunden werden sollen. Wenn fehlerhafte Pixel in einem der beiden Bilder auftreten, können diese Pixel nur schwer oder gar nicht den korrespondierenden Pixeln im zweiten Bild zugeordnet werden. Im schlechtesten Fall findet eine falsche Zuordnung statt. Aus diesem Grund wird in der Regel eine *Glättung* der Bilder mit einem Filter vorgenommen. In dieser Arbeit kommen zwei verschiedene Filtertypen zum Einsatz, lineare verschiebungsinvariante Filter zur Verminderung des Bildrauschens und das Medianfilter zur Beseitigung von Pixelfehlern.

4.3.1 Lineare verschiebungsinvariante Filter

Diese Art von Filtern wird auf ein Bild durch eine Faltungsoperation mit einer entsprechenden Filtermaske angewendet. Dieser Vorgang wird auch als Konvolution, die Filtermasken dementsprechend als Konvolutionskerne bezeichnet. Die Funktionsweise einer Konvolution würde an dieser Stelle den Rahmen sprengen und ist daher nur in Grundzügen dargestellt. Weiterführende Informationen hierzu finden sich in [7].

Eine Faltung ist eine Operation, die einen Bildbereich auf einen Pixel abbildet. Dabei ist sowohl das Bild als auch der Konvolutionskern als zweidimensionale Matrix zu betrachten. Abbildung 4-3 stellt die Berechnung einer Konvolution für den allgemeinen Fall dar. Sie läßt sich formal durch

$$[G * O](m,n) = \sum_{k=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} G(k,l) \cdot O(k-m, l-n) \qquad (4.1)$$

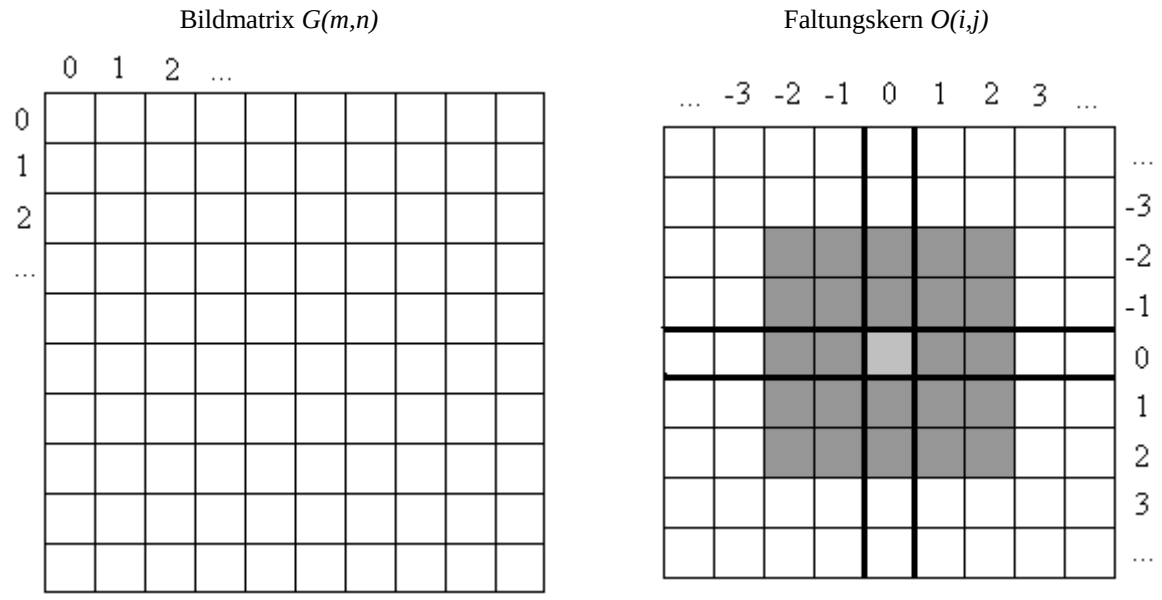

Abbildung 4-3: Bildmatrix und allgemeiner Konvolutionskern

Der Faltungskern wird mit seinem Zentrum über das zu verändernde Bildpixel gelegt, welches dann entsprechend der Filterwerte neu berechnet wird. Jedes Element der Matrix des Faltungskerns ist dabei ein Gewicht, wie stark der jeweilige Bildpixel an dieser Position, das Zielpixel im Zentrum beeinflusst. Der Faltungskern wird für die Faltung des ganzen Bildes sukzessive Pixel für Pixel über das gesamte Bild geschoben. Damit ein eindeutiges Zentralpixel der Filtermatrix als Zielpixel vorhanden ist, werden in der Regel Faltungskerne mit ungerader Kantenlänge bevorzugt.

Faltungskerne lassen sich auch als Operatoren auffassen. Ein solcher Operator bildet den *M x M*-dimensionalen Bildraum auf sich selbst ab und überführt ein Bild *G* in ein Bild *G´* [7]:

$$G' = OG\,. \qquad (4.2)$$

Diese Operatoren besitzen folgende Eigenschaften:

Linearität

$$O(aG + bG') = aOG + bOG'\,. \qquad (4.3)$$

Dabei sind *a* und *b* beliebige Konstanten.

Additivität

Operatoren können additiv zusammengefasst werden:

$$O_1 G + O_2 G = (O_1 + O_2) G \,. \tag{4.4}$$

Kommutativität

Die Reihenfolge der Filteroperationen darf vertauscht werden:

$$O_1 O_2 G = O_2 O_1 G \,. \tag{4.5}$$

Assoziativität

Mehrere Operatoren nacheinander auf ein Bild angewendet, können vertauscht und auch zu einem Operator O_3 zusammengefasst werden:

$$O_1 (O_2 G) = (O_1 O_2) G = O_3 G \,. \tag{4.6}$$

Der Beweis dieser Eigenschaften ist in [7] nachzulesen. Wichtig ist die Tatsache, das sich mehrere Operatoren zu einem komplexeren zusammenfassen lassen. Denn dann ist es umgekehrt auch möglich, komplexe Faltungskerne in einfache Operatoren zu zerlegen. Faltungskerne, mit denen eine solche Zerlegung entlang der Achsen in zwei eindimensionale Filter möglich ist, heißen Seperable Filter. Diese Eigentschaft kommt der Geschwindigkeit, mit der die Faltung auf einem Bild ausgeführt wird, auf Grund weniger auszuführender Operationen zu Gute.

Als Beispiel wird folgender Faltungskern mit dazugehöriger Zerlegung betrachtet:

$$\frac{1}{256}\begin{pmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{pmatrix} = \frac{1}{16}\begin{pmatrix} 1 \\ 4 \\ 6 \\ 4 \\ 1 \end{pmatrix} * \frac{1}{16}\begin{pmatrix} 1 & 4 & 6 & 4 & 1 \end{pmatrix} \tag{4.7}$$

Die Faktoren vor den Matrizen bewirken nur eine Skalierung des Ergebnisses, damit dieses innerhalb der gültigen Grenzen des Intensitätswertes eines Pixels bleibt. Für die Betrachtung der Operationen ist er nicht wichtig, da er auch in die Matrix multipliziert werden

kann und dann effektiv nicht mehr vorhanden ist. Für den Faltungskern, die linke Seite der Gleichung 4.7, sind 25 Multiplikationen und 24 Additionen pro Pixel notwendig. Wird der Kern so in zwei Vektoren zerlegt, wie auf der rechten Seite von 4.7 ersichtlich, sind nur noch 10 Multiplikationen und 8 Additionen für die Faltung notwendig.

Das Filter, das konkret für die Bildglättung in dieser Arbeit eingesetzt wird, ist das Binomialfilter. Ein Beispiel einer Binomialfiltermaske ist bereits in Gleichung 4.7 zu sehen. Binomialfilter werden benutzt, um den Mittelwert der Intensitätswerte eines Pixels und dessen Nachbarn zu ermitteln. Das Ergebnis ist die Verringerung des Rauschens im Bild. Die Konstruktion eines Binomialfilters geht auf die Binomialverteilung zurück, welche im eindimensionalen Fall der Berechnung des Pascalschen Dreiecks (Abbildung 4-4) entspricht.

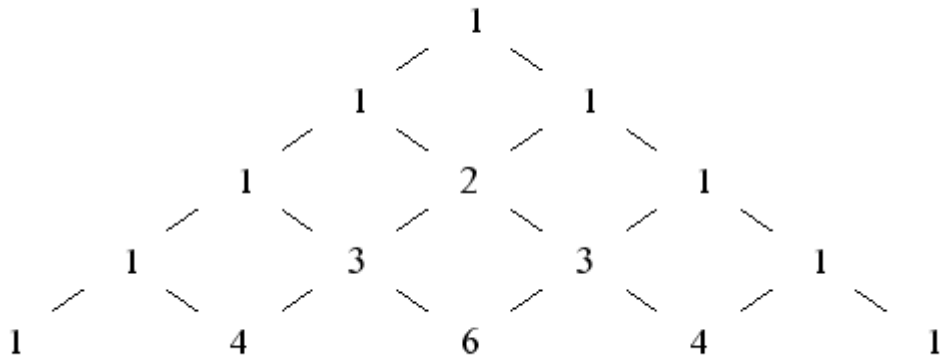

Abbildung 4-4: Berechnungsschema des Pascalschen Dreiecks

4.3.2 Medianfilter

Lineare Mittelwertfilter wie das Binomialfilter verringern das Bildrauschen recht wirkungsvoll. Treten jedoch Bildfehler auf, die auf Übertragungsfehler oder defekte Pixel zurückzuführen sind, stimmen die Intensitätswerte dieser Pixel überhaupt nicht mehr. Die Anwendung des Binomialfilters würde zwar den Fehler dieser Pixel etwas verringern, aber die Nachbarpixel würden unnötig stark verfälscht werden. Diese Art der Bildfehler wird als Salt-And-Peper Rauschen bezeichnet.

Um die Pixel des Salt-And-Peper Rauschens zu korrigieren, müssen sie gezielt gesucht werden. Oft werden sie auch als Ausreißer bezeichnet, weil sie in keinem Zusammenhang zu ihrer Nachbarschaft stehen. Eine Möglichkeit, diese Ausreißer zu finden, ist das Medianfilter. Der Median ist eine statistische Größe, welche den mittleren Wert einer geordneten Wertereihe beschreibt. Im Gegensatz zum Binomialfilter stören ihn Ausreißer weniger. Das Medianfilter funktioniert für Bilder nach folgendem Schema. Innerhalb einer definierten Nachbarschaft eines Pixels (*i*,*j*) werden alle Intensitätswerte der Größe nach geordnet. Dem Pixel (*i*,*j*) wird anschließend der Median dieser Werte zugewiesen (vgl. dazu Abbildung 4-5).

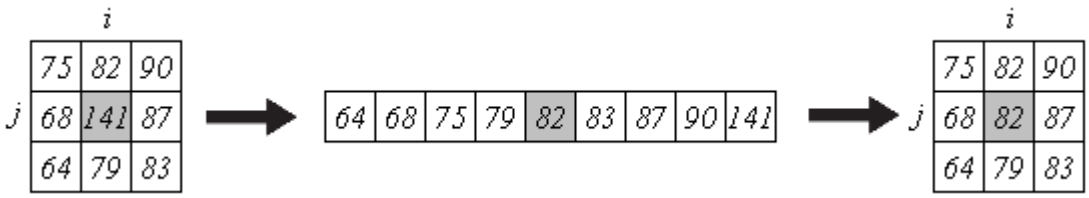

Abbildung 4-5: Medianfilter

Die Anzahl an Operationen, die sowohl für eine Faltung, als auch für den Medianfilter auszuführen sind, ist abhängig von der Größe der Filtermatrix. Ist die Größe konstant, haben die Algorithmen eine Laufzeit von O(1). Eine WCET kann demnach bestimmt werden.

4.4 Anytime Block Matching Algorithmus

In diesem Kapitel wird nach der Auswahl des Verfahrens zur Lösung des Korrespondenzproblems das Konzept des Anytime Block Matching Algorithmus (statischer Ansatz) vorgestellt. Danach wird aufgezeigt, wo die Redundanzen des SV Sensors liegen und wie diese vermieden werden können (dynamischer Ansatz). Abschließend wird diskutiert, wie Verbesserungen durch den Einsatz weiterer Sensoren erreicht werden können (Fusionsansatz).

4.4.1 Auswahl eines geeigneten Verfahrens

In der SBV kann das Korrespondenzproblem, wie in den Grundlagen erläutert, mit zwei grundlegend verschiedenen Ansätzen, dem globalen und dem lokalen, gelöst werden. Eine Methode, die für den globalen Ansatz repräsentativ ist, ist das Horn Schunk Verfahren [27].

Das Prinzip des Horn Schunk Verfahrens ist die rekursive Anwendung von zwei konkurrierenden Funktionen auf das Bildpaar. Die erste Funktion, die sogenannte Horn Schunk Bedingung, versucht korrespondenzen auf Basis des optischen Flusses zu finden. Das heißt, es werden von einem Pixel (i,j) o.B.d.A. im rechten Bild korrespondierende Pixel im linken Bild nur in der unmittelbaren Nachbarschaft der Position (i,j) gesucht. Durch den rekursiven Aufruf, wächst so die Verschiebung des Pixels (i,j) aus dem rechten Bild um höchstens ein Pixel pro Durchlauf in Richtung des korrespondierenden Pixels im linken Bild. Die zweite Funktion verfolgt das Ziel, die Verschiebungen der Pixel möglichst gleichmäßig zu halten. Sie geht davon aus, das benachbarte Pixel zu einem Objekt gehören können und sich deshalb um die selbe Position verschieben müssen.

Auf den ersten Blick erscheint dieses Verfahren durch die schon vorhandene Rekursivität und die dabei sukzessive Verbesserung des Ergebnisses prädestiniert für die Modifikation in einen Anytime Algorithmus. Allerdings hängen dem Verfahren einige grundsätzliche Nachteile an. Die sukzessive Annäherung der korrespondierenden Pixel funktioniert nur, wenn die Entfernung zwischen ihnen sehr kurz ist – maximal einigen Pixel weit. Des Weiteren kann die Korrespondenz auch nur dann gefunden werden, wenn sich zwischen den Pixeln eine Ähnlichkeitsstrecke befindet. Das heißt, liegen zwischen den korrespondierenden Pixeln zu diesen völlig verschiedene, wird der Suchprozess diese ‚Barriere' nicht überwinden können und ein vollkommen falsches Resultat ermitteln. Der dritte und schwerwiegendste Nachteil ist jedoch die Geschwindigkeit. Es dauert mehrere Minuten bis ein Ergebnis vorliegt.

In der Regel laufen Algorithmen basierend auf dem globalen Ansatz länger als Algorithmen, die auf dem lokalen Ansatz beruhen. Daher ist die Anwendung globaler Ansätze in mobilen Robotern ungeeignet.

Das hier Verwendung findende Verfahren eines lokalen Ansatzes ist der Block Matching Algorithmus. In Bezug auf die Rechtzeitigkeit des Vorliegens der Ergebnisse, ist er der einzige, der eine ausreichende Entfernungsinformation mit einer algorithmischen Struktur liefern kann, die dank einfacher zu Grunde liegenden Berechnungen effizient zu implementieren ist. Außerdem ist es möglich die Parameter des BM Algorithmus so zu ändern, dass er mit mehr als zwei Kameras funktioniert und auch eine andere Kameraanordnung vorgegeben werden kann. Das ist für die nachfolgenden Arbeiten ein wichtiger Faktor. Zusammenfassend waren für die Entscheidung folgende Kriterien ausschlaggebend:

- Die Initialisierung des Algorithmus mit seinen eigenen Ergebnissen ist umsetzbar.
- Der Algorithmus ist effizient in Software implementierbar.
- Die Erweiterbarkeit auf andere Anwendungsparameter.

4.4.2 Statischer Ansatz

Ausgehend von dem im Grundlagenkapitel 2.1 beschriebenen BM Algorithmus müssen Modifikationen vorgenommen werden, um einen für den TAFT Scheduler (Kapitel 2.2) verwendbaren Anytime Algorithmus zu entwickeln. Zunächst wird zur Vereinfachung der Modifikation von nur einem Stereobildpaar ausgegangen. Daher auch der Name statischer Ansatz, da keine weiteren Informationen vom SV Sensor geliefert werden.

Da der BM Algorithmus sein Ergebnis ermittelt, indem er *einmal* über das gesamte Bild iteriert, muss seine Arbeitsweise angepasst werden, um die für einen Anytime Algortihmus notwendige funktionale Redundanz zu bekommen. Das bedeutet, es muss ermöglicht werden, ein erstes Ergebnis, welches noch nicht sehr genau sein muss, möglichst schnell zu ermitteln und dieses dann, wenn Zeit bleibt, sukzessive zu verbessern.

Eine denkbare Möglichkeit hier schnell ein erstes Resultat zu ermitteln, ist die Betrachtung von Bildausschnitten. So kann beispielsweise ein Bild in beliebig große Teilbilder zerlegt werden, für welche dann jeweils ein Maß bestimmt werden muss, wie wichtig es für das Gesamtergebnis ist. Anschließend können diese Teilbilder nacheinander berechnet werden,

geordnet nach dem wichtigsten. Problematisch sind bei diesem Ansatz aber mehrere Punkte. Zum einen ist es sehr schwierig, ein Maß für die Bewertung der Wichtigkeit eines Teilbildes zu finden. Zum anderen ist schwer abzuschätzen, wann hier ausreichend Informationen für ein erstes Ergebnis zur Verfügung stehen. Des Weiteren wirft sich die Frage auf, was mit den nicht berechneten Teilbildern geschieht, würde der Algorithmus dennoch im Sinne der Anytime arbeiten. Es wäre möglich an diesen Stellen auf Grund der schon berechneten Umgebung zu interpolieren, würde aber höchstwahrscheinlich in den meisten Fällen Fehlschätzungen liefern, die inakzeptabel sind. Eine in diesem Fall vermutlich bessere Lösung wäre es, für diese Bereiche des Bildes nichts auszusagen. Damit würde das Problem verschoben werden und müsste an anderer Stelle, beispielsweise unter Zuhilfenahme der Sensorfusion gelöst werden.

Der bessere und hier gewählte Ansatz ist die gleichzeitige Reduzierung der gesamten Bildinformation auf ein definiertes Minimum. Möglich wird das durch den Einsatz von Tiefpassfiltern. Dieser Begriff stammt ursprünglich aus der Elektrotechnik und beschreibt ein elektronisches Bauteil, das hochfrequente Signale aus einem Spektrum herausfiltert und tieffrequente passieren lässt [22]. Hochfrequente Anteile repräsentieren Details in einem Signal. Werden diese ‚weggeschnitten', reduziert sich die Information in diesem Signal. Da ein Bild ebenfalls mit Hilfe einer Fouriertransformation [23] im Frequenzraum dargestellt werden kann, ist auch hier das Konzept der Tiefpassfilterung von Frequenzen möglich. Das bereits vorgestellte Binomialfilter löst genau diese Aufgabe auch im Ortsraum, sodass keine Überführung durch die Fouriertransformation notwendig ist.

Die Reduzierung der Bildinformation allein ist aber nicht ausreichend, um das angestrebte Ziel, schnell ein ‚erstes' Ergebnis berechnen zu können, zu erreichen. Der Grund ist, dass die Bildgröße für die Ausführungszeit des BM Algorithmus maßgeblich ist. Durch die Reduzierung von Informationen im Bild, werden Redundanzen zwischen den Pixeln erzeugt, welche es erlauben, genau diese Pixel ohne weiteren Informationsverlust zusammenzufassen. Dies ermöglicht Reduzierung der Bildgröße und damit der Laufzeit.

Die Idee dahinter stammt von dem Konzept des Aufbaus von Gaußpyramiden [24]. Hier wird ein Tiefpassfilter mit anschließender Bildverkleinerung erst auf das Orginalbild, da-

nach jeweils auf die Ergebnisbilder angewendet. Es entstehen Bilder unterschiedlicher Größe und entsprechender Informationsmenge. Abbildung 4-6 zeigt schematisch eine dreistufige Gaußpyramide. Der Name ‚Pyramide' hat sich auf Grund der Form, die entsteht, wenn alle Ergebnisbilder übereinander gelegt werden, etabliert.

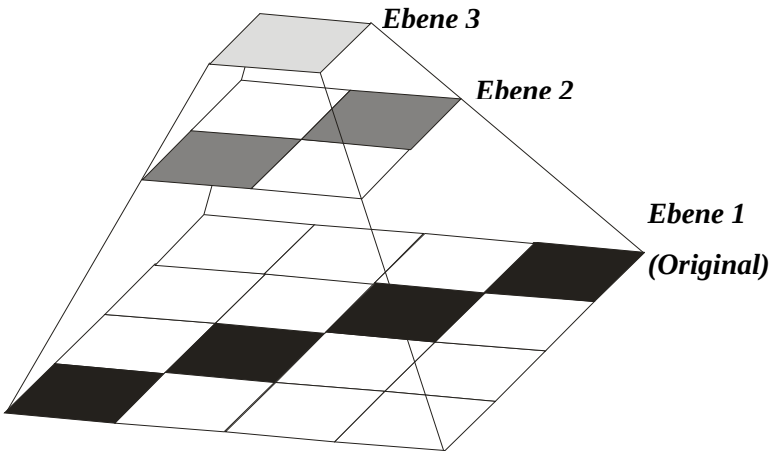

Abbildung 4-6: Schema der Gaußpyramide

Die Anwendung eines 5x5 Binomialfilters mit anschließender Halbierung der Bildhöhe und -breite erstellt aus dem vorverarbeiteten Bildpaar jeweils eine Gaußpyramide für das rechte und linke Bild. Dadurch wird die Bildgröße in jeder Stufe effektiv um den Faktor vier verkleinert. Insgesamt wird die Gaußpyramide in dieser Arbeit aus vier Ebenen zusammengestellt. Die Hinzunahme weiterer Ebenen macht wenig Sinn, da die Informationen dort schon so weit reduziert sind, dass während der Korrespondenzanalyse eine falsche Zuordnung der Blöcke sehr wahrscheinlich ist.

Mit dem Bild der Pyramidenspitze kann der Anytime Block Matching Algorithmus ein ‚erstes' Resultat auf das um den Faktor 64 verkleinerte Originalbild ermitteln. Natürlich ist die Qualität dadurch ebenfalls stark gemindert worden. Um sie verbessern zu können, muss der Anytime Block Matching Algorithmus mit den Bildern aus der nächst feineren Stufe arbeiten. Damit die letzten Informationen nicht umsonst berechnet worden sind, werden sie in die nächste Stufe integriert. Das geschieht folgendermaßen. Das Resultat der Korrespondenzanalyse einer Stufe ist die Disparitätsmatrix, die für jeden Block des rechten Bil-

des einen Disparitätsvektor beschreibt, der die Verschiebung bis zu der Stelle darstellt, an der der passendste Block im linken Bild gefunden wurde. Diese Informationen werden als Initialisierung der Disparitätsmatrix des Anytime Block Matching Algorithmus für die Bilder aus der nächst feineren Stufe der Gaußpyramide verwendet. Da sich Bildhöhe und –breite verdoppeln, ist auf eine entsprechende Vergrößerung der Informationen der Disparitätsmatrix aus der letzten Stufe zu achten.

Eine Anforderung an den Algorithmus ist die Fähigkeit, jederzeit ein Ergebnis liefern zu können. Wie bereits erwähnt, liegt auf gröbster Ebene frühzeitig ein Ergebnis vor. Ab diesem Zeitpunkt enthält die Disparitätsmatrix immer verwertbare Informationen, die als Ergebnis verwendet werden können. Wird durch den TAFT Scheduler eine Ausnahmebehandlung initiiert, muss die aktuelle Disparitätsmatrix nur noch auf Pixelebene umgerechnet werden, d.h. die Blockdisparitäten werden auf die einzelnen Pixel mit Hilfe der Pixelselektion (siehe Kapitel 2.1) abgebildet.

Um die Suche nach korrespondierenden Pixeln möglichst effizient zu gestalten, ist es sinnvoll, den Suchraum so eng wie möglich zu fassen. Bereits im klassischen BM wird die Suche nach Korrespondenzen auf nur eine Zeile, der so genannten Epipolarlinie, eingeschränkt. Diese Annahme kann auf Grund der kanonischen Kameraanordnung auch hier gemacht werden. Des Weiteren kann im klassischen BM die Annahme gemacht werden, dass die Reihenfolge der Punkte auf den korrespondierenden Epipolarlinien im rechten und linken Stereobild gleich ist. Dies trifft hier nicht mehr zu, da sich Objekte unterschiedlich weit entfernt von der Bildebene befinden können (vgl. dazu Abbildung 4-7). Der horizontale Suchraum kann aber durch die Vorgabe eines maximalen Disparitätswertes d_{max} eingeschränkt werden. Ein maximaler Disparitätswert entspricht einem Mindestabstand zwischen den Objekten der Szene und dem Kamerasystem. Wenn für die Korrespondenzanalyse eine Mindestüberlappung des Bildpaares von 50% verlangt, kann über Gleichung 2.7 und den Parametern des Versuchsaufbaus, ein Mindestobjektabstand von 524 mm ermittelt werden. Also sollte ein Abstand von einem halben Meter nicht unterschritten werden. Andernfalls ist eine ausreichende Überlappung der Szene in den beiden Bilder des Bildpaares nicht mehr gewährleistet. Die maximale Disparität d_{max} beträgt dann 360 Pixel. Bei der Halbierung von Bildhöhe und –breite durch die Gaußpyramide halbiert sich d_{max} ebenfalls.

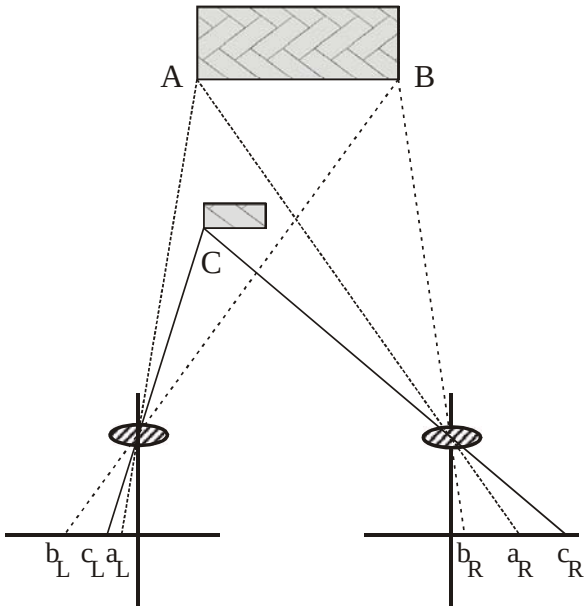

Abbildung 4-7: Reihenfolge der abgebildeten Punkte im linken und rechten Bild

Die Initialisierung der Disparitätsmatrix durch die Ergebnisse der vorherigen Stufe der Gaußpyramide, ermöglicht die Suche nach genaueren Disparitäten an diesen Werten zu orientieren. Anstatt die Blöcke pixelweise über den Suchraum von links nach rechts für einen Vergleich zu durchlaufen, kann zunächst die Umgebung der erwarteten Lösung durchsucht werden. Nur wenn dort kein erfolgreicher Vergleich zu Stande kommt, wird in weiterer Entfernung gesucht. Ein erfolgreicher Vergleich wird über die quadratische Differenz der Pixelintensitäten zwischen den Blöcken definiert. Dazu musste ein geeigneter Grenzwert gefunden werden. In Experimenten mit dem klassischen BM Algorithmus wurde der Wert 250 als vernünftige Grenze ermittelt. Dabei wurde wie folgt vorgegangen. Nach dem Durchlauf des Algorithmus sind manuell etwa 50 korrekte Zuordnungen herausgesucht worden. Aus den dazugehörigen quadratischen Differenzen wurde ein Mittelwert gebildet, an dem sich der Grenzwert orientiert.

Die Größe des Suchraumes bestimmt maßgeblich über die Geschwindigkeit des Algorithmus. Die Qualität der Ergebnisse wird entscheidend von der Blockgröße beeinflusst. In der Literatur wird eine Blöckgröße von 8 x 8 Pixeln favorisiert [8]. In dieser Arbeit wurden

ausgehend davon verschiedene Blockgrößen getestet. Abbildung 4-8 und Abbildung 4-9 zeigen die entsprechenden Ergebnisse in Form einer Disparitätsmatrix. Der Betrag der Länge eines Disparitätsvektors ist als Grauwert abgebildet. Helle Grauwerte stellen längere Vektoren, dunklere kürzere dar. Das bedeutet, helle Flächen liegen dichter am Kamerasystem. Als Ausgangspunkt wurde das Bildpaar aus Abbildung 2-2 verwendet. Das ist das typisches Referenzbild in der SBV. Es stellt mehrere Objekte in unterschiedlicher Entfernung dar.

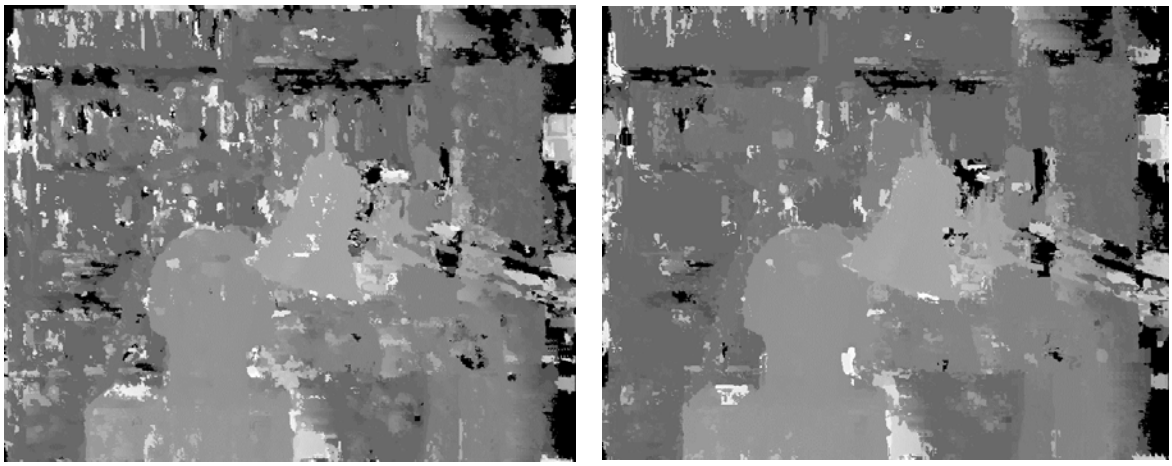

Abbildung 4-8: Disparitätsmatrix durch eine Blockgröße von 4x4 Pixeln (links) und 8x8 Pixeln (rechts) erzeugt

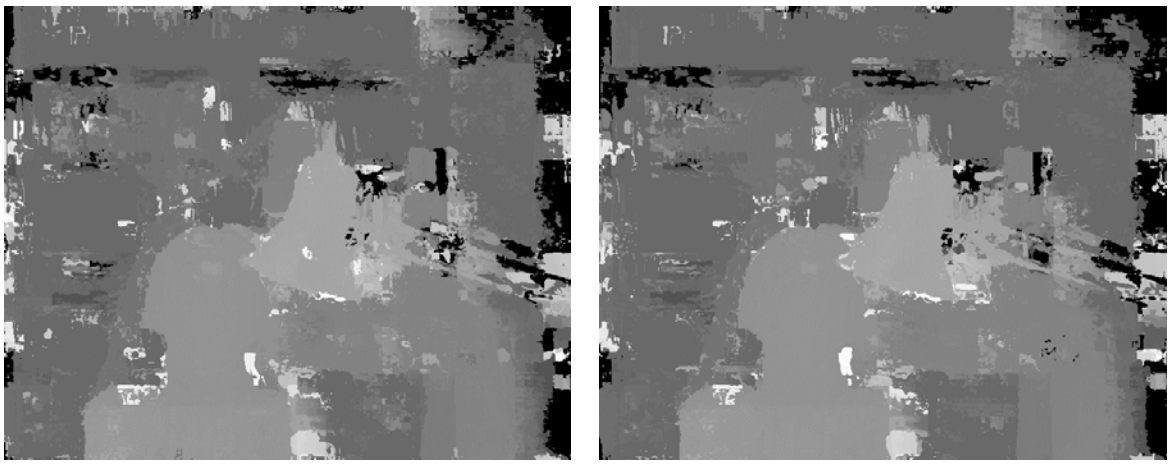

Abbildung 4-9: Disparitätsmatrix durch eine Blockgröße von 12x8 Pixeln (links) und 16x8 Pixeln (rechts) erzeugt

Deutlich erkennbar sind die Qualitätsunterschiede zwischen der Disparitätsmatrix mit 4x4 Blockgröße und den anderen Matrizen. Im 4x4 Block Bild gibt es viele Fehlzuordnungen, in Form kleiner weißer und schwarzer Flächen. Die Unterschiede zwischen den anderen Matrizen scheinen nicht mehr so deutlich, dennoch können im 8x8 Bild etwas mehr Fehler erkannt werden. Zwischen 12x8 und 16x8 sind markante Unterschiede in der Qualität nicht mehr ersichtlich. In dieser Arbeit wurde eine Blockgröße von 12x8 Pixeln gewählt. Dies ist zum einen damit zu begründen, das bei dieser Blockgröße qualitativ bessere Ergebnisse ermittelt werden können als bei kleineren Blockgrößen. Zum anderen wurde bei den qualitativ gleichwertigen Blockgrößen 12x8 und 16x8 die kleinere gewählt, um feinere Strukturen erkennen zu können.

Zusammenfassend ermöglicht das hier vorgestellte Konzept des statischen Ansatzes die Anpassung des BM zu einem Anytime Algorithmus. Dazu werden die Bilder in verschiedenen Detailstufen analysiet. Ist die Zeit nicht ausreichend, um alle Dateilstufen zu berechnen, wird das letzte erreichte Zwischenergebnis zurückgegeben. Auch wenn dies unter Umständen nicht so genau ist, ermöglicht es doch die Entfernungen zu Hindernissen grob zu schätzen

4.4.3 Dynamischer Ansatz

Der dynamische Ansatz wurde im Hinblick auf die Anwendung des Kamerasystems auf einem mobilen Roboter entwickelt. Die Szene wird nach wie vor als statisch angenommen, aber der beobachtete Ausschnitt kann sich auf Grund von Eigenbewegungen des Roboters ändern. Diese ist aber zunächst nicht bekannt.

Im dynamischen Ansatz werden Sequenzen von Stereobildpaaren verwendet. Für jedes Stereobildpaar wird separat über den Anytime Block Matching Algorithmus eine Disparitätsmatrix berechnet. Es besteht keine Möglichkeit, zu erkennen, ob die Disparitätsmatrix noch einmal berechnet werden muss oder nicht. Wenn sich der beobachtete Ausschnitt der Szene nicht ändert, müssen Ressourcen eingesetzt werden, um das selbe Ergebnis wie im letzten Schritt noch einmal zu berechnen. Ziel dieses Ansatzes ist es, diesen Ressourcenaufwand zu minimieren.

Zwischen aufeinanderfolgenden Stereobildpaaren, die gleich oder ähnlich sind, ist eine hohe Redundanz vorhanden. Hier bietet sich ein Ansatzpunkt, Ressourcen einzusparen. Analog zur Übernahme von Ergebnissen einer Ebene der Gaußpyramide in die nächste, können auch Ergebnisse von einem Stereobildpaar an das nachfolgende weitergegeben werden. Für zwei aufeinanderfolgende Stereobildpaare können zwei verschiedene Fälle eintreten, entweder der beobachtete Ausschnitt der Szene ändert sich oder er bleibt gleich. Im ersten Fall ist davon auszugehen, dass durch die geringe zeitliche Differenz zwischen den Aufnahmezeitpunkten aufeinanderfolgender Stereobildpaare, die Unterschiede zwischen ihnen relativ klein sind. Eine Initialisierung der Disparitätsmatrix in der gröbsten Auflösungsebene der Gaußpyramide verringert die Berechnungsdauer, denn ohne sie müsste auf dieser Ebene der komplette Suchraum durchlaufen werden. Die eingesparten Berechnungen sind analog zu denen von einer Ebene der Gaußpyramide in die nächste. Im zweiten Fall können die Berechnungen ganz eingespart werden. Das Problem ist, diesen Fall zu erkennen. Dazu wird wieder auf gröbster Ebene die Disparitätsmatrix mit dem Ergebnis des vorhergehenden Stereobildpaares initialisiert. Das Ergebnis wird außerdem für einen späteren Vergleich gespeichert. Nun wird die Disparitätsmatrix für das aktuelle Bildpaar auf der gröbsten Ebene berechnet. Ist das Ergebnis dieser Berechnung und das Ergebnis des vorhergehenden Bildpaares identisch, kann davon ausgegangen werden, dass sich der beobachtete Ausschnitt der Szene nicht geändert hat. Klar ist, dass zwei aufeinanderfolgende Stereobilder auf Grund von Fehlern bei der Bildaufnahme niemals vollkommen identisch sind (vgl. dazu Kapitel 4.3). Daher muss der Begriff ‚identisch' mit einer gewissen Toleranz belegt werden. Ist eine Matrix als ‚identisch' identifiziert, kann das Ergebnis des vorherigen Stereobildpaares auf das aktuelle übertragen werden. War das Ergebnis des vorherigen Bildpaares das der feinsten Auflösung, dann muss für das aktuelle Paar nichts mehr berechnet werden. Die Disparitätsmatrix aus dem vorangegangenem Schritt kann als Ergebnis zurückgegeben werden. Andernfalls wird das Ergebnis des letzten Bildpaares als Zwischenergebnis verwendet und auf der nächst folgenden Stufe weiter verbessert.

Der dynamische Ansatz nutzt somit die redundanten Bildinformationen aus, um die Berechnungszeit des Anytime Block Matching Algorithmus zu verkürzen, was insgesamt zu einem effizienteren Vorgehen führt.

4.4.4 Fusionsansatz

Der Fusionsansatz erweitert den dynamischen Ansatz unter zu Hilfenahme der 3D-Rekonstruktion und weiterer Sensoren, die es ermöglichen, die Eigenbewegung des mobilen Roboters und damit des Kamerasystems, zu messen. Eine Aufstellung einiger dafür relevanter Sensoren ist in Tabelle 1-2 zu finden. Mit ihnen ist es möglich die Eigenbewegung für alle sechs Freiheitsgrade zu bestimmen. Folgende Bewegungen sind zu unterscheiden:

- Rotationsbewegungen
  - Rollen
  - Gieren
  - Neigen
- Translationsbewegungen
  - Bewegung entlang der x-Achse
  - Bewegung entlang der y-Achse
  - Bewegung entlang der z-Achse

Jetzt gilt es zu klären, was für Auswirkungen diese Bewegungen auf die Stereobildpaare haben. Eine Veränderung der Position des Roboters und damit des Kamerasystems führt zu einer Änderung der relativen Position zwischen dem Kamerasystem und den Punkten, die durch die 3D-Rekostruktion ermittelt worden sind. Abhängig von der Art der Bewegung wird die Menge aller Punkte horizontal (Gieren) oder vertikal (Neigen) verschoben oder gedreht (Rollen). Durch Translation bewegen sich die Punkte entlang der Fluchtlinien [32]. Um die Veränderung durch die Eigenbewegungen in den Anytime Block Matching Algorithmus einbeziehen zu können muss zunächst eine 3D-Rekonstruktion erfolgen (siehe Kapitel 2.1). Anschließend müssen die Koordinaten der rekonstruierten Punktmenge bezüglich der Eigenbewegung des Kamerasystems transformiert werden. Dies erfolgt mittels eines Kalman Filters [33]. Schließlich müssen die neuen Koordinaten in eine Disparitätsmatrix zurücktransformiert werden. Das wird durch die Umkehrung der 3D-Rekonstruktion erreicht. Die so erhaltene Disparitätsmatrix wird letztlich für die Initialisierung des Anytime Block Matching Algorithmus für das nachfolgende Stereobildpaar verwendet.

4.5 3D-Rekonstruktion

Die 3D-Rekonstruktion im Anytime Block Matching Algorithmus erfolgt genauso, wie die 3D-Rekonstruktion im klassischen BM Algorithmus (siehe Kapitel 2.1). Liefert die Korrespondenzanalyse eine Disparitätsmatrix einer gröberen Ebene aus der Gaußpyramide, kann die Rekonstruktion auf der groben Disparitätsmatrix erfolgen. Jedem Disparitätsvektor müssen dann aber mehrere Pixel gleicher Entfernung zugeordnet werden. Die Laufzeit ist abhängig von der Größe der Disparitätsmatrix. Da die maximale Größe bekannt ist, kann die WCET für die 3D-Rekonstruktion bestimmt werden.

5 Implementierung

In den vorangegangenen Kapiteln wurden die Konzepte zur Lösung der gestellten Aufgabe erläutert. Dieses Kapitel beschreibt die konkrete technische Realisierung des statischen Ansatzes. Die Implementierung erfolgte unter Linux in Hinblick auf eine spätere Portierung nach RTLinux [29].

Das hat den Vorteil, dass sich Fehler in der Software nicht unmittelbar auf die Systemstabilität auswirken. Da RTLinux Threads direkt im Kernel Modus laufen, verwenden sie den Systemspeicher, was bei einer fehlerhaften Adressierung häufig zu Systemabstürzen führt. Die Entwicklung unter Linux dagegen hat bei solchen Fehlern im schlimmsten Fall einen Programmabsturz zur Folge. Des Weiteren lassen sich Programme unter Linux besser debuggen, da bessere Kontrollmöglichkeiten bei der Ausführung von Linux Programmen zur Verfügung stehen als bei RTLinux Threads.

Zur Realisierung dieses Vorhabens, wurde das Hauptaugenmerk darauf gelegt, keine systemabhängigen Funktionen zu verwenden und Arbeitsspeicher nicht dynamisch anzufordern. Ausnahmen für eventuelle debug Operationen wurden in entsprechenden defines eingebettet.

Als Programmiersprache kam *C++* zum Einsatz. Objektorientierte Konzepte wurden auf Grund der dadurch einfacheren Modularisierung der Komponenten in Hinblick auf eine spätere Weiterentwicklung verwendet. Da der Einsatz von *new* und *delete* Operatoren auf Grund fehlender Speichermanagements in Echtzeitsystemen nicht erfolgen kann, fanden alle Speicheranforderungen zum Zeitpunkt des Programmstarts durch Templateklassen statt.

Da der TAFT Scheduler nur unter RTLinux zur Verfügung steht, wurde die Unterbrechung des Main Parts durch Exceptions durch entsprechende Änderungen im Programmablauf simuliert. Damit kann gezeigt werden, wann die Ergebnisse mit welcher Qualität im Vergleich zum Standard BM Algortihmus vorliegen.

5.1 Bildaufnahme

Die Bildaufnahme ist wie im Konzept beschrieben realisiert worden. Es wurde die Klasse *stereograbber* implementiert, um die Bilder von den Kameras aufzunehmen. Dafür wird die Video4Linux API genutzt. Sie verwendet die gleichen Schnittstellen, die im RTLinux Treiber von Tobias Rudolph [30] Verwendung finden. Für das Anzeigen der Bilder unter Linux werden Routinen der Bibliothek *OpenGL* [31] benutzt. Insgesamt ist es möglich, alle 100 ms ein neues Bildpaar bereit gestellt zu bekommen.

5.2 Vorverarbeitung

Die Funktionen, die der Bildvorverarbeitung dienen, sind mit in die Klasse *image* integriert. Es sind Methoden, die unmittelbar auf das Bild angewendet werden und deshalb mit diesem verknüpft sind. Realisiert sind neben der effizienteren Implementierung des Binomialfilters, auch ein bezüglich der Geschwindigkeit verbessertes Medianfilter.

Die Sortierung einer Wertereihe im Medianfilter beansprucht den größten Teil seiner Laufzeit. Durch die Annahme, dass nur eine ungerade Anzahl von Werten gefiltert wird, was nicht im Widerspruch zum Konzept steht, kann das Sortieren durch ein anderes Verfahren ersetzt werden. Aus einer Wertereihe mit *n* Werten wird das Minimum und das Maximum gesucht und herausgestrichen. Dieser Vorgang wird $\frac{n}{2}-1$ mal wiederholt. Übrig bleibt der Median. Die Laufzeit liegt mit $O\left(\frac{n}{2}-1\right)$ unter der einer Sortierung beispielsweise mit Quicksort von $O(n \cdot \ln(n))$. Diese Optimierung ist von großem Vorteil, da das Medianfilter allein schon in der Pixelselektion zwei mal verwendet werden muss.

Des Weiteren ist in die Vorverarbeitung die Erstellung der Gaußpyramiden ausgelagert. Sie ist eine Methode der Klasse *stereoimage*, welche von *image* abgeleitet ist und die Aufgabe hat ein Bildpaar zu repräsentieren.

5.3 Korrespondenzanalyse

Eine weitere Methode der Klasse *stereoimage* ist der Anytime BM Algortihmus. Im Wesentlichen ist er wie in Kapitel 4.4.2 beschrieben, umgesetzt. Alle Parameter, die für die Korrespondenzanalyse relevant sind, sind in der Datei *configure.h* definiert. Das hat den Vorteil, Änderungen beispielsweise der Kameraparameter oder der Suchräume an nur einer Stelle vornehmen zu müssen.

5.4 3D-Rekonstruktion

Die Rekonstruktion einer 3D Tiefeninformation ist ebenfalls eine Methode der *stereoimage* Klasse. Es werden in einem Array 3D Koordinaten geliefert, die ihren Koordinatenursprung im Kamarasystem haben. Die Transformation dieser 3D Informationen in ein anderes Koordinatensystem ist damit leicht durchführbar, wenn die Kameraposition darin bekannt ist.

5.5 Entwickelte Klassen und Methoden

In diesem Abschnitt sind in tabellarischer Form alle entwickelten Klassen und ihre Methoden aufgeführt und beschrieben. Es dient als Referenz, damit nachfolgende Weiterentwicklungen problemlos fortgeführt werden können.

Klassen- /Methodenname	Parameterliste	Kurzbeschreibung
CImage		Kapselt Funktionen für das Arbeiten mit einem Bild
FillImage	Unsigned char value	Füllt ein Bild mit der Farbe *value*
GetImageData		Liefert einen Zeiger auf die Bilddaten
GetImageDataSize		Liefert die Bildgröße in Bytes
GetHeight		Liefert die Bildhöhe
GetWidth		Liefert die Bildbreite
Load	RB_FileFormat format,	Lädt ein Bild *name*, welches im For-

	const char* name	mat *format* vorliegt (unterstützt werden PGM und PPM Formate)
LowPassFilter		Realisiert einen Tiefpassfilter
QuarterImage	CImage *newImage	Viertelt das Bild
Save	RB_FileFormat format, const char* name	Speichert ein Bild unter *name* im Format *format* ab (unterstützt werden PGM und PPM Formate)
CMatrix		Kapselt Funktionen für das Arbeiten auf Speichermatrizen
DoubleSize	CMatrix *newMatrix	Verdoppelt matrixbreite und -höhe
FillValue	TypeOfElement value	Füllt den Speicherbereich mit *value*
GetDataSize		Liefert die Größe des Speicherbereiches
GetWidth		Liefert die Breite der Matrix
GetHeight		Liefert die Höhe der Matrix
MedianFast	TypeOfElement *values, unsigned no	Implementierung eines effizienten Medianfilters
MedianFilter	unsigned maskWidth, unsigned maskHeight	Das klassische Medianfilter
SetDimensions	Unsigned w, unsigned h	Setzt die neue Höhe und Breite
SetHeight	Unsigned h	Setzt die neue Höhe der Matrix
SetWidth	Unsigned w	Setzt die neue Breite der Matrix
CStereoImage		Von CImage abgeleitet kapselt es das SBV Verhalten
BlockMatching		Implementiert den Anytime Block Matching Algorithmus
ComputeMSE	Linker block, rechter block	Vergleicht zwei Blöcke anhand des MSE
GetLeftImage		Liefert das linke Bild des Stereobild-

		paares
GetRightImage		Liefert das rechte Bild des Stereobildpaares
LowPassFilter		Realisiert das Tiefpassfilter für beide Bilder
PixelSelection		Implementiert die Pixelselektion
QuarterImages	CstereoImage *quStereoImage	Viertelt beide Bilder
CUtil		
MakeImage	CMatrix *mat1, Cmatrix *mat2, Cimage *img);	Erzeugt anhand des x und y Anteiles der Disparitätsvektoren eine Grauwertabbildung einer Disparitätsmatrix

6 Ergebnisse und Messungen

In diesem Abschnitt werden Ergebnisse der Implementierung des Konzeptes gezeigt und erörtert. Für die Messungen wurden die Algorithmen auf einem SuSE Linux 8.1 System mit Kernelversion 2.4 auf einem Pentium III mit 450Mhz und 128MB RAM durchgeführt.

Zunächst sollen die Laufzeiten des Anytime Block Matching Algorithmus für verschiedene Detailstufen dargestellt und verglichen werden. Von dem dafür verwendeten Bildpaar ist das rechte in Abbildung 6-1 dargestellt. Da kein TAFT Scheduler für Linux zur Verfügung steht, wurde das Auslösen einer Exception manuell in den Programmcode eingefügt. Die ermittelten Laufzeiten sollten dennoch repräsentativ sein. Tabelle 6-1 fasst die verstrichene Laufzeit an verschiedenen Punkten des Anytime Block Matching Tasks und die Laufzeit des klassischen Block Matchings zusammen.

Abbildung 6-1: Refernzbild für Messungen

Haltepunkt	Laufzeit bis zum Haltepunkt in Sekunden
Nach der Vorverarbeitung	0,72
Nach der ersten (gröbsten) Stufe	0,77
Nach der zweiten Stufe	0,95
Nach der dritten Stufe	1,45
Nach der vierten Stufe	4,02
Klassisches Block Matching	5,09

Tabelle 6-1: Tasklaufzeiten

Die Laufzeit des klassischen Block Matching Algorithmus wurde für einen Vergleich mit 5,09 Sekunden gemessen. Durch die Umsetzung des statischen Ansatzes mit dem Pyramidenkonzept wurde ein deutlicher Geschwindigkeitsgewinn erreicht.

Die Disparitätsmatrizen sind das direkte Ergebnis der Korrespondenzanalyse, die durch den Anytime Block Matching Task umgesetzt wurde. Der klassische BM Algorithmus berechnet auch eine Disparitätsmatrix. Daher bietet es sich an, die beiden Algorithmen auf Basis der Disparitäten zu vergleichen. In Abbildung 6-2 und Abbildung 6-3 sind die Ergebnisse des Anytime Algorithmus auf verschiedenen Auflösungsstufen dargestellt.

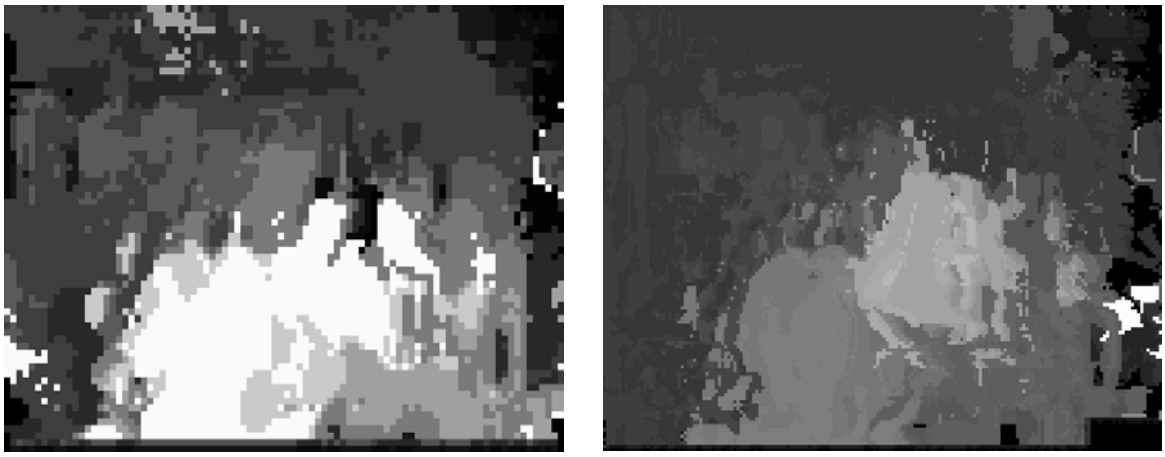

Abbildung 6-2: Ergebnisse des Anytime BM Algorithmus in verschiedenen Auflösungsstufen (a)

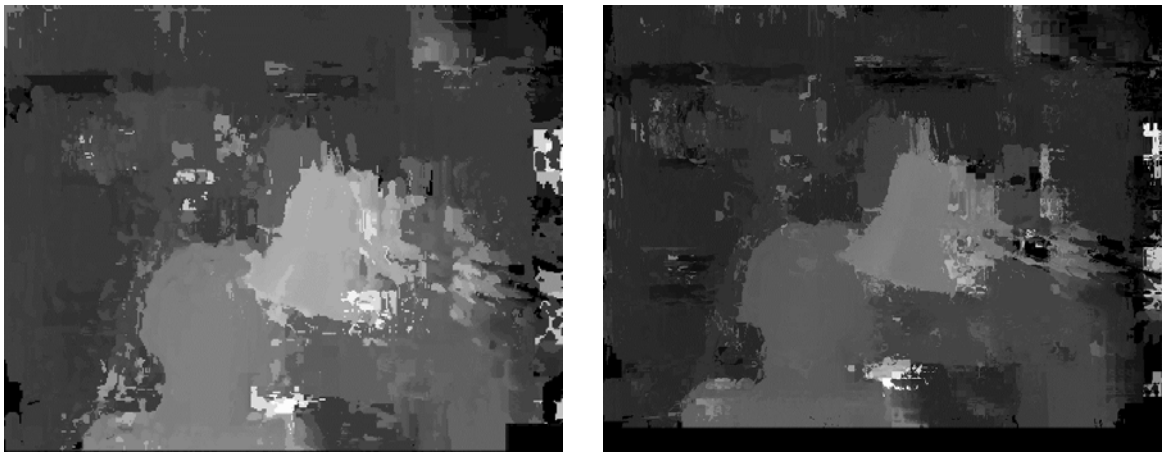

Abbildung 6-3: Ergebnisse des Anytime BM Algorithmus in verschiedenen Auflösungsstufen a)

Deutlich erkennbar ist die Herausbildung immer feinerer Details von links oben nach rechts unten. Bereits in nach der ersten Stufe können die nahe liegenden Objekte, Kopf und Lampenschirm, als Hindernisse erkannt werden. Zum Vergleich zeigt Abbildung 6-4 das Ergebnis des klassischen Block Matching Algorithmus. Erkennbar ist, das die Qualität in der feinsten Auflösungsstufe des Anytime Block Matching Algorithmus besser als im klassischen Ansatz ist.

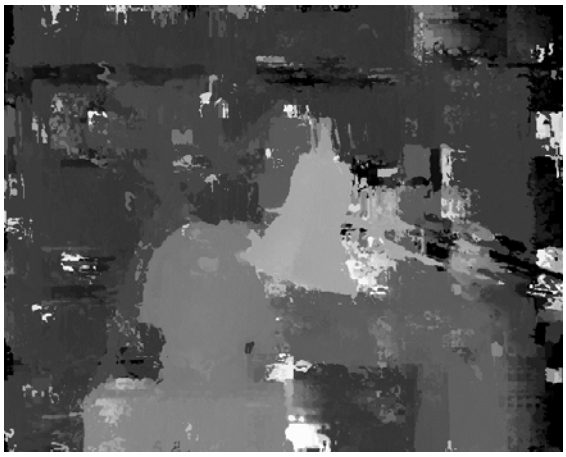

Abbildung 6-4: Ergebnis des klassischen BM Algorithmus

Für mögliche Applikationsszenarien ist es von Bedeutung zu wissen, in welchem Bereich Objekte durch das hier verwendete Kamerasystem erfasst werden können. Dazu wurden das Disparitätslimit d_{max} und das Auflösungslimit z_{max} betrachtet. Abbildung 6-5 stellt den Zusammenhang zwischen Disparität und Entfernung für den hier verwendeten Versuchsaufbau dar. Die Disparität wurde in Schritten von jeweils 12 Pixeln erhöht, da dies der Blockgröße entspricht und damit die Granularität festgelegt war. Erkennbar ist, dass bei einer Objektentfernung von etwa 15 m von der Kamera die minimale Verschiebung von einem Block auftritt. Damit ist die maximal zu erfassende Entfernung auf ca. 15 m begrenzt. In der Praxis ist aber auch diese Entfernung nicht möglich, da diese minimale Disparität von Rauschen nicht zu unterscheiden ist. Des Weiteren ist die Auflösung bei mehr als drei Metern relativ grob. Eine Veränderung der Disparität um einen Block führt zu einer Entfernungsveränderung von mehreren Metern. Wie bereits in Kapitel 4.4.2 beschrieben ist der Mindestobjektabstand durch das Disparitätslimit d_{max} begrenzt. Damit lässt sich ein Arbeitsbereich von einem halben bis drei Metern festlegen. Durch Vergröße-

rung des Abstandes zwischen den zwei Kameras wird eine Verschiebung des Arbeitsbereiches in weitere Entfernungen erreicht.

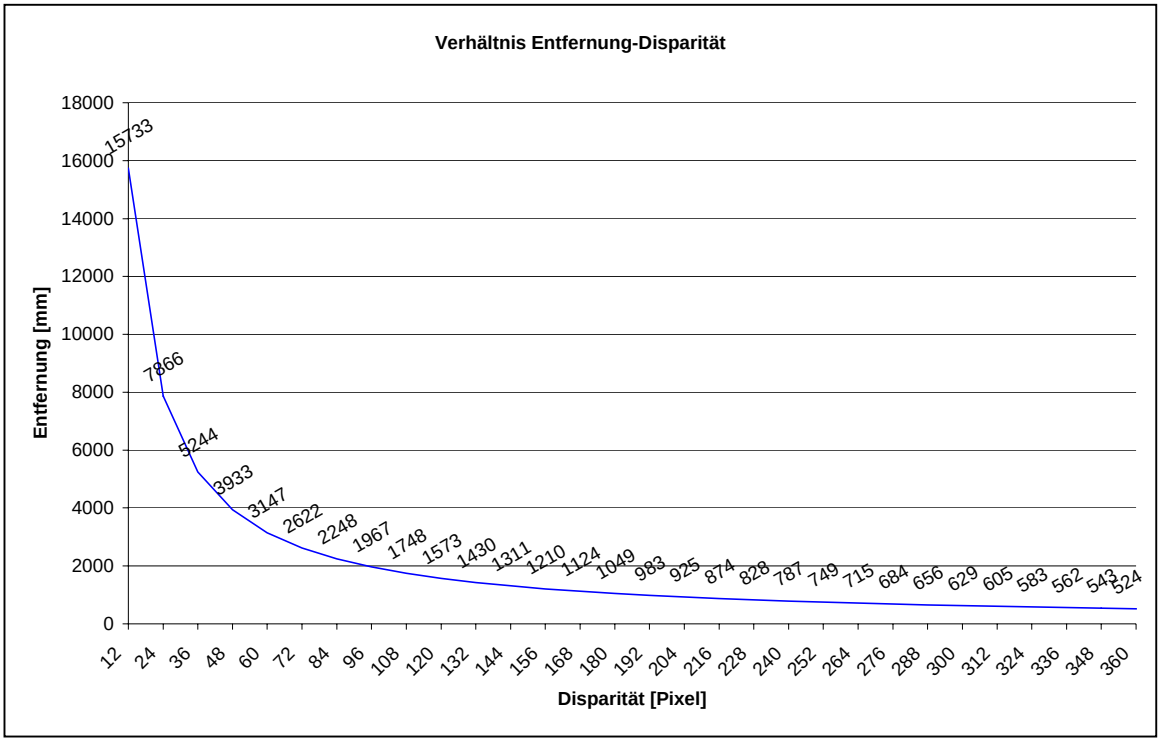

Abbildung 6-5: Verhältnis der Entfernung zum Disparitätswert

Abbildung 6-6 zeigt die zeitlichen Proportionen des Anytime Block Matching Algorithmus. Nach der Vorverarbeitung (VV) zum Zeitpunkt a, folgt die Berechnung der gröbsten Auflösungsstufe. Ab dem Zeitpunkt b liegt ein Ergebnis vor. Die verbleibenden Sekunden werden für eine Verbesserung des Ergebnisses jeweils zu den Zeitpunkten c, d und e verwendet. Der letzte Schritt ist eine Nachbearbeitung (NB) der Disparitätsmatrix, so das zum Zeitpunkt f das vollständige Ergebnis vorliegt. Wird die Nachbearbeitung in Form der Pixelselektion weggelassen, ist die Auflösung der Disparitätsmatrix sehr grob und für den

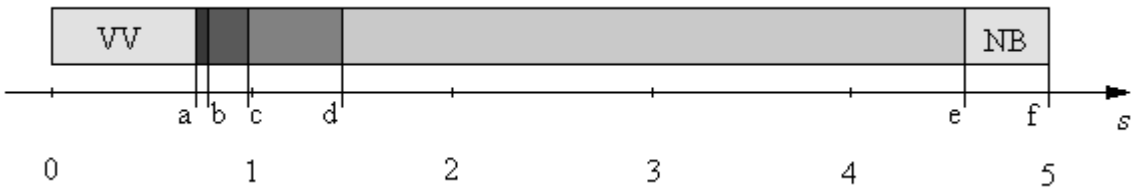

Abbildung 6-6: zeitliche Verteilung des Anytime BM Algorithmus

praktischen Einsatz nicht empfehlenswert. Wird der Algorithmus während der Berechnung einer Auflösungsstufe (zwischen b und e) abgebrochen, werden die noch fehlenden Teilergebnisse durch die Ergebnisse der vorherigen Stufe ergänzt.

Zum Abschluss soll noch ein Beispiel mit realen Daten gezeigt werden. Abbildung 6-7 zeigt das mit dem Versuchsaufbau aus Kapitel 4.1 aufgenommene Stereobildpaar, Abbildung x die berechnete Disparitätsmatrix. Deutlich erkennbare Probleme treten bei weißen Wand, sowie der Tür im Hintergrund auf. Die Pixel des Kartons sind trotzdem gut zugeordnet worden. Anhand von repräsentativen Pixeln wurde exemplarisch die Disparität nach Gleichung 2.7 in eine Entfernung vom Kamerasystem umgerechnet. Es wurde eine Disparität von 107 Pixeln (entspricht 725,46 µm) ermittelt. Das ergibt für den Versuchsaufbau eine Entfernung von 1,76 m. Der real gemessene Abstand betrug 1,70 m.

Abbildung 6-7: Reales Stereobildpaar

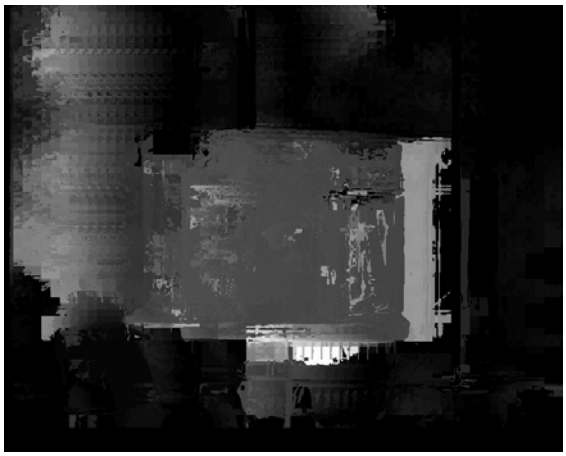

Abbildung 6-8: Disparitätskarte des Bildpaares aus Abbildung 6-7

Es konnte gezeigt werden, dass der ausgewählte Block Matching Algorithmus als Anytime Algorithmus umgesetzt werden konnte. Der TAFT Scheduler realisiert mit diesem Anytime Block Matching Algorithmus, dass Ergebnisse rechtzeitig und möglichst immer vorliegen. Des Weiteren ist konzeptionell beschrieben, wie der Algorithmus mittels der Verwendung von Redundanzen Ressourcen effizienter nutzen kann (Kapitel 4.4.3). Außerdem wurde gezeigt, wie eine weitere Verbesserung der Ergebnisse durch den Einsatz von Sensorfusion herbeigeführt werden kann (Kapitel 4.4.4).

7 Zusammenfassung und Ausblick

Die vorliegende Diplomarbeit beschreibt die geschickte Verschmelzung der zwei komplexen Gebiete Bildverarbeitung und Echtzeitsysteme. Dabei muss bedacht werden, das beide Bereiche eigentlich sich widersprüchliche Anforderungen stellen. Bildverarbeitung setzt eine hohe Berechnungszeit voraus, die unter Echtzeitbedingungen meist nicht gegeben ist. Die Idee der Kombinierung ist es, dass die Bildverarbeitung zunächst zu Gunsten der Geschwindigkeit auf qualitativ hochwertige Ergebnisse verzichtet. Diese können durch sukzessive Verbesserung später immer noch erreicht werden. Das Echtzeitsystem muss in der Lage sein, zu entscheiden, wie lange verbessert werden kann, um rechtzeitig ein Ergebnis liefern zu können. Die Bildverarbeitung muss darauf reagieren können, und das bestmögliche Ergebnis präsentieren.

Für die Echtzeitumgebung wurde aus diesem Grund ein TAFT Scheduler gewählt. Für die Bildverarbeitung wurde der Block Matching Algorithmus gewählt, weil er im Vergleich zu anderen Bildverarbeitungsmethoden eine relative niedrige Ausführungszeit hat. Eine sukzessive Verbesserung wird von ihm jedoch nicht unterstützt. Mittels des Konzeptes der Gaußpyramide konnte er entsprechend modifiziert werden.

Des Weiteren konnte durch eine sinnvolle Einschränkung des Suchraums die Ausführungszeit weiter gesenkt werden. Neben der Einschränkung des Suchraums auf eine einzelne Zeile, wurde ein maximaler Disparitätswert angegeben. Durch die Initialisierung mit Zwischenergebnissen konnte die Suche noch effektiver gestaltet werden.

Für Experimente wurde ein eigener Versuchsaufbau entwickelt. Dadurch ist es möglich, den implementierten Algorithmus an realen Daten zu testen. Tests mit Referenz- und realen Bildern produzierten sehr gute Ergebnisse. Sowohl die Ausführungszeit als auch die Qualität der Ergebnisse konnten im Vergleich zum klassischen Block Matching Verfahren gesteigert werden.

Die vorliegende Arbeit bietet zahlreiche Ansatzpunkte der Weiterentwicklung. Als nächster Schritt ist die Implementierung auf RTLinux geplant. Dank vorausschauender Pro-

grammierung sollte dies relativ leicht umsetzbar sein. Da das Hauptaugenmerk zunächst auf der Umsetzung des Block Matching Algorithmus lag, bleibt in der Implementierung noch Spielraum für künftige Optimierungen. Dies ist auch notwendig, da der Anytime Block Matching Algorithmus zwar in einer knappen Sekunde ein erstes Ergebnis liefern kann, aber die Gesamtlaufzeit von etwa fünf Sekunden für den allgemeinen Einsatz in der mobilen Robotik noch zu groß ist.

Danach kann über eine Umsetzung des dynamischen und schließlich des Fusionsansatzes nachgedacht werden. Dazu müssen die Stereokameras auf einen mobilen Roboter montiert werden, damit dynamische Änderungen der Szene durch Eigenbewegung auftreten.

Die jetzige Implementierung macht bereits eine teilweise Anwendung möglich. So ist der Einsatz auf einem Schreitroboter vorstellbar. Dieser erreicht im Gegensatz zu fahrenden Robotern nicht so hohe Geschwindigkeiten. Fünf Sekunden scheinen durchaus ausreichend für einen Schreitroboter, um rechtzeitig Hindernisse erfassen und darauf reagieren zu können.

8 Literaturverzeichnis

[1] K. Konolige, “Small vision systems: Hardware and implementation,” in Eighth International Symposium on Robotics Research, (Hayama, Japan), pp. 203-212, London, Springer, Oktober 1997.

[2] L. Matthies, A. Kelly, und T. Litwin, “Obstacle detection for unmanned ground vehicles: A progress report,” in International Symposium of Robotics Research, (München, Deutschland), Oktober 1995.

[3] R. Volpe, J. Balaram, T. Ohm, und R. Ivlev, “The rocky 7 mars rover prototype,” in International Conference on Intelligent Robots and Systems, Vol. 3, (Osaka, Japan), pp. 1558-1564, November 1996.

[4] C. Zhang, “A Survey on Stereo Vision for Mobile Robots,” Dept. Of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, USA

[5] Faugeras et al.: Real time correlation-based stereo: algorithm, implementations and applications. Rapport de recherche, no. 2013, Institut national de recherche en informatique et en automatique, 1993

[6] Dickmanns, E. D.; Graefe, V.: Dynamic monocular machine vision, Machine Vision and Applications, Springer International, Vol. 1, 1988

[7] Jähne, B.: „Digitale Bildverarbeitung“; 3. Auflage; Springer Verlag; 1993, ISBN 3-540-56926-X

[8] Klette, R.; Koschan A.; Schlüns, K.; „Computer Vision – Räumliche Information aus digitalen Bildern“; Vieweg Technik Verlag; 1996

[9] Liu, J. W. S.; Shih, W.-K.; Lin, K.-J.; Bettati, R.; Chung, J.-Y.: Imprecise Computations. Proceedings of the IEEE, vol. 82, no. 1, 1994, pp. 83-94

[10] Butazzo: „Hard Real-Time Computing Systems“, Kluwer Academic Publishers, 1997

[11] Krishna; Shin: „Real-Time Systems“, Mc Graw-Hill, 1997

[12] http://www.din-katalog.de/; Juni 2003

[13] Ihme, T.: „Steuerung von sechsbeinigen Laufrobotern unter dem Aspekt technischer Anwendungen“, Dissertation, Otto-von-Guericke-Universität Magdeburg, 2002

[14] Kopetz, H.: Real-Time Systems: Design Principles for Distributed Embedded Applications. In Serie: The Kluwer International Series in Engineering and Computer

Science; SECS 395 Real Time Systems. Kluwer Academic Publishers, Boston u.a., 1997

[15] Krishna, C. M.; Shin, K. G.: Real-Time Systems. In Serie: MCGraw-Hill Series in Computer Science, 1997

[16] Kim, K. H.; Liu, J.; Kim, M. H.: Deadline Handling in Real-Time Distributed Objects. Proceedings of the IEEE CS 3rd International Symposium on Object-oriented Real-time distributed Computing, ISORC 2000, Newport Beach, CA, März 2000, pp. 7-15

[17] Kaiser, J.; Nett, E.: Echtzeitverhalten in dynamischen, verteilten Systemen, GI Informatik Spektrum 21(6), Dezember 1998, S. 356-365

[18] Gergeleit, M.; Streich, H.: TaskPair-Scheduling with Optimistic Case Execution Times – An Example for an Adaptive Real-Time System. In: Second International Workshop on Object-oriented Real-Time Dependable Systems, Laguna Beach, CA., IEEE Computer Society Press, 1996

[19] Gergeleit, M.; Nett, E.; Fitzner, J.: On-line Prediction of Execution Times – A Basis for Adaptive Scheduling. In: Fourth International Workshop on Object-oriented Real-Time Dependable Systems, Santa Barbara, California, Februar 1999, ISBN 0-7695-0101-X, pp. 186-194

[20] Streich, H.: TaskPair-Scheduling: An Approach for Dynamic Real-Time Systems, Int. Journal of Mini & Microcomputers, Vol.17, No.2, pp 77-83, 1995

[21] Nett, E.; Gergeleit, M.: Preserving Real-Time Behaviour in Dynamic Distributed Systems, IASTED International Conference on Intelligent Information Systems, The Bahamas, 8.-10. Dezember 1997

[22] Seifart, M.: Analoge Schaltungen. Verlag Technik, Berlin, 1987, ISBN 3-341-00091-7

[23] Bracewell, R.: The Fourier transform and its applications. New York: McGraw Hill 1965

[24] Burt, P. J.: The pyramid as a structure for efficient computation. In: Rosenfeld, A. (Ed.): Multiresolution image processing and analysis. Springer Series in Information Sciences. Vol. 12. Berlin: Springer 1984

[25] Reuter, T.: HDTV standards conversion. Proc. IEEE-ASSP & EURASIP 5th Workshop on Multidimensional Signal Processing, Nordwijkerhouth, Niederlande, 1987, (Nachdruck Heinrich-Hertz-Institut Berlin)

[26] Becker L. B.; Gergeleit M.: Execution Environment for Dynamically Scheduling Real-Time Tasks, 22nd IEEE Real-Time Systems Symposium (RTSS 2001), WIP-Session, London, Dec. 3-6, 2001

[27] Schunk, B. G.; Horn, B. K. P.: Constraints on optical flow computation. Proc. Pattern Recognition and Image Processing Conf., Dallas, 1981, pp. 205-210

[29] Barabanov, M.; Yodaiken, V.: Real Time Linux, 3 März 1996 (white paper), http://fsmlabs.com/developers/white_papers/, Juni 2003

[30] Rudolph, T.: Kamerabasierte, echtzeitfähige Objekterkennung in der Teamrobotik, September 2002, Diplomarbeit, Otto-von-Guericke-Universität Magdeburg

[31] Wright, R. S.; Sweet M.: OpenGL superBible, Waite Group Press, Indianapolis, Ausgabe 2, 2000, ISBN 1-571-69164-2

[32] Raquel, F.; Vassallo, Schneebeli, Hans J.; Santos-Victor, José: Visual servoing and appearance for navigation. Robotics and Autonomous Systems 31 (2000)

[33] Neuburger, E.: Einführung in die Theorie des linearen Optimalfilters (Kalman-Filter), Oldenbourg Verlag München Wien, 1972, ISBN 3-486-39361-8

[34] eGRABBER-2 Manual, (c) Phytec, Ausgabe März 2002

Selbständigkeitserklärung

Ich versichere hiermit, die vorliegende Arbeit selbst verfasst, Zitate gekennzeichnet und keine anderen als die offen gelegten Quellen und Hilfsmittel benutzt zu haben.

Richard Bade