DIPLOMARBEIT

Optisch gesteuerte Werkzeugmaschine

ausgeführt zum Zwecke der Erlangung des akademischen Grades
eines Diplom-Ingenieurs unter der Leitung von

Professor: Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Kittl Burkhard
am Institut für Fertigungstechnik und Hochleistungslasertechnik E311
Betreuer: FH-Prof. DI Dr. Reinhard BUSCH

eingereicht an der Technischen Universität Wien
Fakultät für Maschinenwesen und Betriebswissenschaften

von

Bauer Christian
Matr.Nr. 0026581
Vertexgasse 16 1230 Wien

Wien, am 21.10.2010 _____

Kurzfassung

Ziel dieser Diplomarbeit war es, eine Werkzeugmaschine im Rahmen eines mit 70.000,- Euro dotierten Forschungsprojekts der Fachhochschule Wels, Abteilung Automatisierungstechnik mit ausschließlich optischen Hilfsmitteln (Kameras) anzusteuern, basierend auf einer unmittelbaren Umsetzung der Maschinengreifarmbewegungen durch entsprechende Handbewegungen des Bearbeiters eines Werkstückes in Echtzeit. Dieses Forschungsprojekt stellte zumindest in Europa eine komplette Neuentwicklung dar, da die bisherige Entwicklung von Robotern und Werkzeugmaschinen ausschließlich in der starren Programmierung der Maschine für eine spezielle und unveränderliche Anwendung mündete. Der große Nachteil dieser bisherigen Forschungen liegt in der für jede neue Anwendung jeweils erforderlichen aufwendigen Neuprogrammierung, was mit hohen zusätzlichen Kosten- und Zeitaufwand verbunden ist. Das Ziel des ggst. Forschungsprojekt war daher, unterschiedliche Werkstücke und Anwendungsmodelle bearbeiten zu können, ohne jedes Mal einen hohen Programmieraufwand für die Bearbeitungsänderung in Kauf nehmen zu müssen. Beim konkreten Forschungsprojekt musste der gesamte Aufbau und die Konfiguration der Kameras, das einwandfreie Zusammenspiel mit der Werkzeugmaschine und dem jeweiligen Werkstück und deren Programmierung entwickelt werden. Zusätzlich musste auch das Konzept für die gesamte Umsetzung ausgearbeitet werden, worin auch die Planung zur Fortführung und Weiterentwicklung dieses Forschungsprojektes beinhaltet ist. Ausgehend von den Platzproblemen, die aus dem geringen verfügbaren Platz zwischen Maschine und Werkstück resultierten und Problemen im Hinblick auf die Lösung einer möglichen Werkzeugbearbeitbarkeit von sämtlichen Seiten, musste u.a. auch die Befestigungsart des Materials auf einem Drehtisch ersonnen werden. Genauso umfangreich waren die Schwierigkeiten mit der vorhandenen Werkzeugmaschine, die erstens nicht in jede gewünschte Lage geführt werden konnte und in einigen Fällen ausserdem umständliche Zusatzbewegungen benötigte, um bestimmte Positionen zu erreichen.
Auch die Lösung des Problems, eine möglichst gute und fehlerfreie Ansteuerung zu erreichen und Überschneidungen von Mensch und **Koordinatengeber** zu vermeiden, war erforderlich. Da von Beginn an eine gute Tiefenauflösung bzw. eine genaue Koordinatenübertragung gefordert war, war die Verwendung mehrerer Kameras unabdingbar. Auch auf eine hinreichend gute Kontrastierung zwischen Hintergrund, Gewand und **Koordinatengeber** musste geachtet werden. Mit dem im ggst. Forschungsprojekt gewählten Lösungsansatz konnte das obige Ziel erreicht werden.

Abstract

The goal of this expensive 70,000 Euro research project of the Fachhochschule Wels, Automation Engineering Department, was to control a machine tool using only optical methods (optical cameras) in a real time implementation. This research project is unique in Europe and represents a new application of controlling a machine directly with a signal hand (coordinate pointer). Previously it was possible only through rigid programming, whereby there was no option to modify the program during the work. The major disadvantage of this kind of programming is that much time and money is required to change code or to program new code. The intention of this research project was to allow different work elements to be adjusted without changing the program. This project includes work on the implementation of the whole structure, the configuration of the cameras, and the development of an optimal interaction between the machine tool and the work piece through programming. In addition it was necessary to find a concept for converting the entire research project including future work and the project continuation. Starting with problems concerning the space between workpart and machine, there were problems to turn the workpart correctly to process it, as well as difficulties fixing the workpiece on the turn table. Additionally there was a problem moving the machine correctly to the special point required to tool the workpart, because the machine tool itself often acted as a physical barrier. There were extensive problems steering the machine as good as the hand could be moved with the coordinate-"pointer" on it. Sometimes this resulted in additional movements of the machine tool or, in some instances, it was completely impossible to control the machine. The movement of the machine should run precisely, free from error and without overlapping between the picture of the pointer and the picture of the person. Being precise was a major challenge and a high definition picture was needed to increase the resolution of the recording the movement. Therefore it was necessary to use more than one or two cameras (for stereovision). To reduce the possibility of finding a wrong blob was also important. This could happen when you took a white shirt into the field of vision of the camera. On the other hand it is a good approach to check the contrast at its highest point. This thesis describes a solution of sorting out this problem.

Danksagung

Der meiste Dank gebührt Prof. (FH) Jean D. Hallewell Haslwanter MSc BSc aus der EDV-Abteilung der Fachhochschule Wels, die sich viel Zeit genommen hat und immer ein offenes Ohr für programmiertechnische Probleme hatte. Die Hilfestellungen waren dabei immer sehr profund und anschaulich. Sehr bedanken möchte ich mich auch beim betreuenden Professor an der Fachhochschule Wels, Herrn Prof. (FH) Dipl.-Ing. Dr. Reinhard Busch, aber auch bei Herrn Prof. (FH) Dipl.-Ing. Kurt Niel, die das Projekt und mich, auch in schwierigen Zeiten, unterstützt haben, und jederzeit ihre Hilfe angeboten. Die technische Unterstützung durch den technischen Mitarbeiter der Abteilung Automatisierungstechnik an der Fachhochschule Wels, Herr Ing. Becherstorfer, war ebenfalls äußerst hilfreich. Die für die Fertigung der Stereokameraträgerkonsole notwendige Unterstützung wurde dankenswerterweise von Herrn Meindl geleistet. Bedanken möchte ich mich auch bei meinen Eltern für deren Unterstützung.

Inhaltsverzeichnis

Abkürzungen

CAD	Computer Aided Design
CCD	Charge-coupled Device
CMOS	Complementary Metal Oxide Semiconductor
FPGA	Field Programmable Gate Array
IR	InfraRed
LUT	Look Up Table
NC	Numerical Control
ROI	Region of Interest
RGB	RGB-Farbraum, Rot, Grün, Blau

1 Einleitung

1.1 Vorwort

Heutzutage ist es nicht mehr möglich, sich der optischen Bildverarbeitung zu entziehen, sei es bei Robotern oder bei der Kontrolle von Qualtitätsmerkmalen von Maschinenteilen bis zur Verwendung in der Überwachungsbranche. Auch zur Gesichtserkennung an Flughäfen oder bei Zutrittskontrollstellen bei besonders gesicherten Bereichen in behördlichen und privaten Organisationen (z. B. Iriskontrolle) wird diese Technologie in nicht allzu langer Zeit ihren Einzug halten. Vor allem die rasante Entwicklung immer leistungsfähigerer Kameras und Rechnersysteme, die in Echtzeit dutzende von Bildern verarbeiten können, ist absolut erstaunlich. Immer höhere Pixelraten und bessere Auflösungen der einzelnen Geräte machen diese zur ersten Wahl, wenn es um die Kontrolle von Maßen geht. Die Vorteile der optischen Bearbeitung liegen auch in der Tatsache, dass mit optischen Verfahren auch mehrere Merkmale gleichzeitig verarbeitet werden können. Auch im "3D-Bereich" werden die verwendeten Systeme immer ausgeklügelter. Mittels Stereokameras wird es ermöglicht, auch Tiefen zu messen. Immer kompaktere und kompliziertere Systeme, bei denen Kameras direkt mit einer Recheneinheit, Speicherplätzen und weiteren Rechnerkomponenten verbunden sind, werden von der Industrie geliefert. Die Arbeit mit sehr hochwertigen und dadurch aber auch recht kostspieligen Komponenten stellt jedenfalls große Entwicklungsmöglichkeiten im Bereich der Bildverarbeitung dar. Gegenstand dieser Diplomarbeit sind daher auch die aktuellsten Entwicklungen in diesem Bereich, wie z. B. die superschnellen Kameras und die mindestens genauso notwendigen, dazu passenden Rechnereinheiten.

1.2 Ausgangslage

Ausgangssituation der ggst. Forschungsarbeit stellt die Notwendigkeit der Entwicklung eines durch **Bildbearbeitung** und mittels Datenhandschuh gesteuerten Roboters zur maßgeschneiderten Bearbeitung von größeren technischen Modellen dar. Noch müssen größere Prototypen in langwieriger Handarbeit hergestellt werden, wodurch viel Zeit und Geld aufgewendet werden muss, was überdies die Modellentwicklungsmöglichkeiten aufgrund der sich daraus ergebenden eingeschränkten Effizienz reduziert und damit auch die diesbezügliche Forschung hemmt.Weiters können nur leichtbearbeitbare Werkstoffe für die Modellbearbeitung verwendet werden. Bisherige Verfahren laufen über den "CAD" - Entwurf, die Programmierung der NC-Maschinen, Fertigung des Werkstückes, Vermessung und Beurteilung, Nachbearbeitung in "CAD",die mit einer neuerlichen Programmierung verbunden ist, was naturgemäß äußerst aufwändig ist. Im Gegensatz zu

Systemen mit Infarotkameras dürfen bei einer optischen Steuerung mittels Bildverarbeitung keine Situationen auftreten, bei denen der **Koordinatengeber** durch andere Objekte verdeckt wird. Mit der optischen Erfassung ist eine direkte, unmittelbare und exakte Steuerung der Werkzeugmaschine möglich. Dies stellt eine völlig neue Art der Interaktion zwischen Mensch und Maschine dar.

1.3 Zielsetzung

Um eine massive Verbesserung dieser vor Projektsbeginn noch iterativen und sehr aufwändigen Entwicklung von Prototypen, wie in 1.2 beschrieben, zu erreichen, sollte über die intuitive Mensch-Maschine-Schnittstelle eine Beschleunigung, Effizienzsteigerung, Kostenersparnis und Erweiterung der Einsatzmöglichkeiten in der Entwicklung und Fertigung erreicht werden. Die Zielsetzung der ggst. Forschungsarbeit war daher die erstmalige Entwicklung eines mittels **Bildbearbeitung** und Datenhandschuh gesteuerten Roboters in Echtzeit zur maßgeschneiderten Bearbeitung von größeren technischen Modellen ohne der Notwendigkeit einer jeweils erforderlichen neuerlichen Programmierung des Roboters zur Durchführung jedes neuen Arbeitsschrittes, wie es bei den derzeit maschinellen Robotern der Fall ist, die für jede neue Aufgabe aufwendig neu programmiert werden müssen. Es sollte daher ein System zur Herstellung großer Modelle ($\geq 1m^3$) in der Prototypenfertigung entwickelt werden. Die Ansteuerung der Werkzeugmaschine sollte dabei optisch über einen **Koordinatengeber** unmittelbar durch entsprechende zielgerichtete Handbewegungen erfolgen, die eine unmittelbare und der Handbewegungen entsprechende Bearbeitung des Werkstücks durch den Roboterarm ermöglicht. Zur Bearbeitung sollten vorerst nur leicht zu bearbeitende Werkstoffe wie z.B. Holz oder Ähnliches zum Zug kommen. Für den Einsatz war dabei eine 6-achsige Handling-Maschine vorgesehen, die aufgrund ihrer hohen Achsbeschleunigungswerte als Werkzeugmaschine für die Bearbeitung von Modellen geeignet ist. Diese Werkzeugmaschine sollte das Modell exakt in gleicher Weise, wie von den Bewegungen der Hand vorgegeben, in Echtzeit bearbeiten. Die Ansteuerungen sollten dabei zuerst nur in drei Linearbewegungen, später auch mit drei rotativen Freiheitsgraden erfolgen. Die optische Nachverfolgung der Hand sollte hierbei unmittelbar vor sich gehen, da es sonst zu größeren Problemen mit dem Arbeitsfluss kommen könnte. Die Steuerung der Werkzeugmaschine sollte darüber hinaus direkt auf deren Antriebssysteme einwirken. Die hier beschriebene Zielsetzung ist für die über das ggst. Projekt hinausgehende Weiterentwicklung gedacht, für deren Verwirklichung an den Einsatz weiterer Diplomanden gedacht ist. Ziel des ggst. Forschungsprojekts war die Programmierung der Bildbearbeitung für die Trackingkameras. Dabei muss beachtet werden, dass pro Bild nur wenige Millisekunden zur Verfügung stehen. Daher muss ein effektiver, möglichst schneller Algorithmus die Bearbeitungszeit so kurz wie möglich halten. Ebenfalls wichtig sind Einstell- und Justierroutinen, die die gesamten Einstellungen der Vorrichtungen gewährleisten müssen. Es sollte möglich sein, die Werkzeugmaschine mittels einer Hand (direkt oder indirekt mit Koordinatengeber) direkt anzusteuern und zu bewegen. Dies sollte in allen drei translatorischen und allen drei rotatorischen Freiheitsgraden zu bewerkstelligen sein.

1.4 Schriftkonventionen

Schriftart von Stichworten, die näher beschrieben werden

`Schriftart bei Anführung von Programmcode`

- Formatierung bei Auflistung verschiedener Punkte

2 Recherche

2.1 Literatur Recherche

2.1.1 Systeme der Objektverfolgung mittels Bildverarbeitung

In [11] [00724860] wird mit einer Stereokamera ein zylindrisch geformtes Objekt optisch verfolgt. Um das Objekt zu bewegen, wird eine Roboterhand mit drei Fingern verwendet. Zur Vermessung

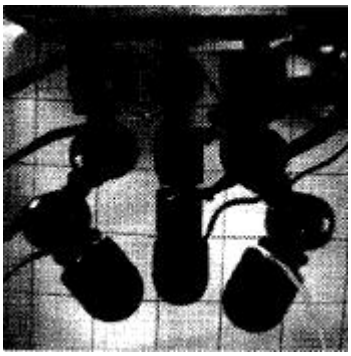


Figure 2.1: Roboterhand mit drei Fingern

der Lage des bekannten Gegenstandes wird mit einem Projektor ein Muster auf das Objekt projiziert. Zusätzlich werden die Lagekoordinaten der Roboterhand verwendet, welche auch die Position des Gegenstandes angeben. Dies hat den Vorteil, dass die Koordinaten auch bei teilweiser Sichtbehinderung der Kameras ermittelt werden können.

Dies ist sicherlich ein Vorteil dieser Anordnung, kann aber nicht für diese Diplomarbeit verwendet werden, da die Ansteuerung über eine menschliche Hand (dies ist eine grundlegende Voraussetzung dieser Arbeit) erfolgt, deren momentane Lage unbekannt ist. Mit einem "Least-Squares-Algorithmus" wird ausgehend von einer bekannten Ausgangslage der Hand die neue Position für das System in diesem Bericht [11] errechnet. Die dabei erreichte Bildwiederholungsrate der Stereokameras liegt bei 33 msek/Bild, was für die Erreichung der Zielsetzung des ggst. Forschungsprojekts aufgrund der zu langsamen Bildverarbeitung nicht adäquat wäre.

Mit vier Kameras, zwei als Stereokameras und die restlichen als Verfolgungskameras (Tracking Kameras) werden Gesichtszüge der Menschen und deren Hände in [19] [00840627] nachverfolgt

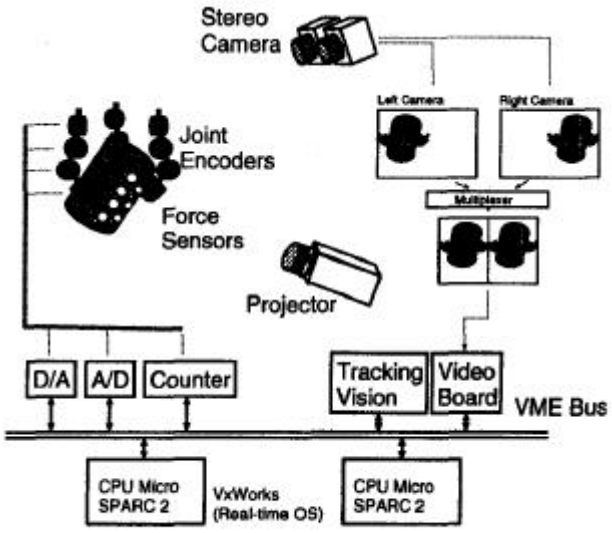


Figure 6: Hardware structure

Figure 2.2: Anordnung der Komponenten

und auch Gesten gedeutet. Dabei wird jedoch mit einer Hautfarbendetektion gearbeitet. Dies ist

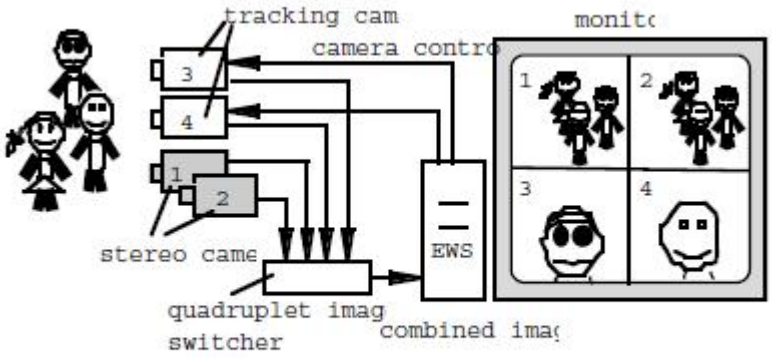


Figure 1. System configuration

Figure 2.3: Anordnung der Komponenten

jedoch für dieses Forschungsprojekt nicht sinnvoll einsetzbar, da wahrscheinlich nur Grauwertkameras eingesetzt werden (Kostenfaktor!/ Datenvolumen!). Die Verwendung von Farb-Kameras würde die Bildverarbeitungsgeschwindigkeit ausserdem zu sehr verringern. Bei dieser Anordnung ([19]) wird allerdings von einer Standard Hautfarbe ausgegangen und der Bereich in der Nähe dieser definiert (hellere oder dunklere Hautstellen, Farbabstand). Bei diesem Verfahren ([19]) können auch zwei Objekte gleichzeitig verfolgt werden (bei dem System der Diplomarbeit sollte es nur notwendig sein, ein Objekt zu verfolgen, und zwar den **Koordinatengeber**), wobei die Funktion auch bei sich im Bild überlappenden Personen nicht beeinträchtigt werden soll). Der Basisabstand zwischen den beiden Stereokameras beträgt dabei 100 mm, was sich sicher als zu

gering für die Diplomarbeit herausstellen wird, da das Ergebnis besser ist je weiter die beiden Kameras auseinanderstehen. Ab einem bestimmten Abstand werden allerdings die beiden Bilder, die ja möglichst gut miteinander korrelieren sollten, zu unterschiedlich, um noch eine ausreichende Tiefeninformation herauslesen zu können.

In dem für die Diplomarbeit verwendeten Modell wird es wahrscheinlich am Besten sein, vom Hintergrund (eigentlich Untergrund, da die beste Sicht mit geringst möglichen Überlappungen der verschiedenen Objekte beim Filmen der Kameras von oben herunter gegeben ist) ein Bild aufzunehmen und die Pixelwerte des immer gleich bleibenden Hintergrundes während der laufenden Routine durch den Prozessor vom Gesamtbild zu subtrahieren. Der Vorteil ist, das alle statischen Störungen damit behoben werden können. Ähnlich dieser Idee arbeitet ein adaptives Modell für die Hintergrunderkennung wie z.B. [10][00958058] mit Farberkennung und einer Tiefenerkennung. Die notwendige Berechnung kann dabei auch in Echtzeit ablaufen und ist nicht so fehleranfällig wie die bekannten Methoden. Um eine noch bessere Fehlerfreiheit zu erreichen, werden mehrere Systeme miteinander kombiniert. Dazu zählen eine dynamische Adaption des Hintergrundes in einem "Gauss-Mischungsmodell" und kombinieren raum-,tiefen und helligkeitsoptimierte Farben. Eine derartige komplexe Hintergrunderkennung, wie oben beschrieben, wäre

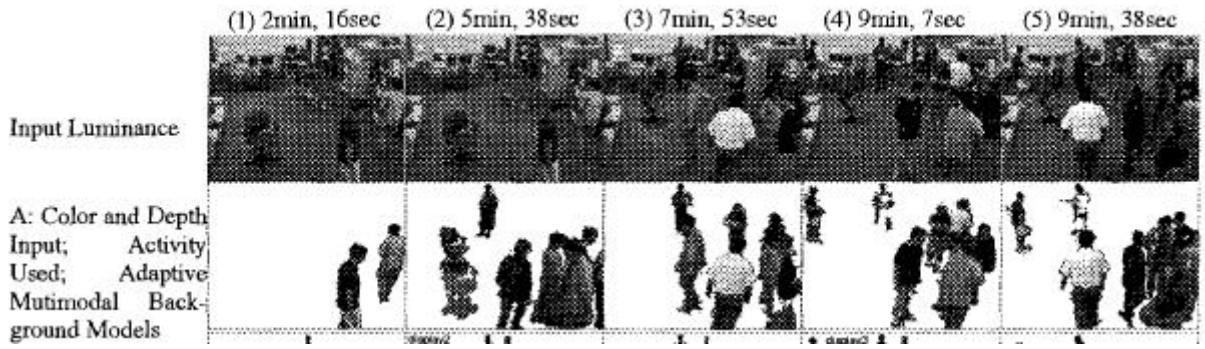


Figure 2.4: Wirkungsweise des Hintergrundberechnungsmodells

wegen ihrer geringeren Schnelligkeit für das Projekt jedoch nicht zielführend. Eine einfache Hintergrundsubtraktion hingegen erreicht eine ausreichend Fehlerfreiheit und ist zudem aufgrund ihrer geringeren Komplexität schneller als das obige Modell.

Ein der Diplomarbeit sehr ähnliches Verfahren wird in [9][cameratrackingball] dargestellt. Auch in diesem Fall wird mit drei Kameras (zwei Stereokameras und einer Kamera, die einen größeren Bildausschnitt darstellt) gearbeitet, allerdings wird ein Ball mit Suchkreisen gesucht, die den Suchbereich bei verlorenem Ballobjekt vergrößern. Außerdem werden die Stereokameras mit einem "**Tracking Device**" bewegt, welches die Kameras so führt, dass der Ball immer in Bildmitte zu finden ist. Diese Technik wird jedoch auch wahrscheinlich bei dem ggst. Forschungsprojekt, in der der **Koordinatengeber** möglichst immer in der Mitte der Bilder der Stereokameras zu finden sein sollte, angewendet. In einem ersten Schritt wird der Ball vorraussichtlich mit einem groben Suchalgorithmus gesucht, der eine hohe Rechengeschwindigkeit sicherstellt. Hier unterscheidet sich das Programm [9] vom ggst. Forschungsprojekt, da zwar auch mit adaptiven Bildausschnitten gearbeitet wird, jedoch sind diese nicht kreisförmig. Die Programmierung des gesamten Projektes [9] erfolgt in Assembler. Ein Anwendungsgebiet des fertigen Systems ist die

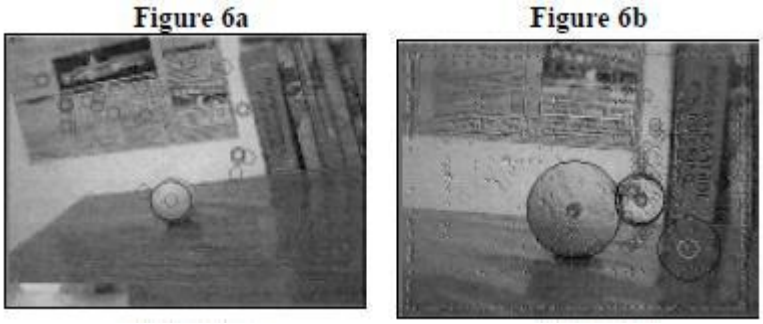


Figure 2.5: Ballsuche mit Suchkreisen

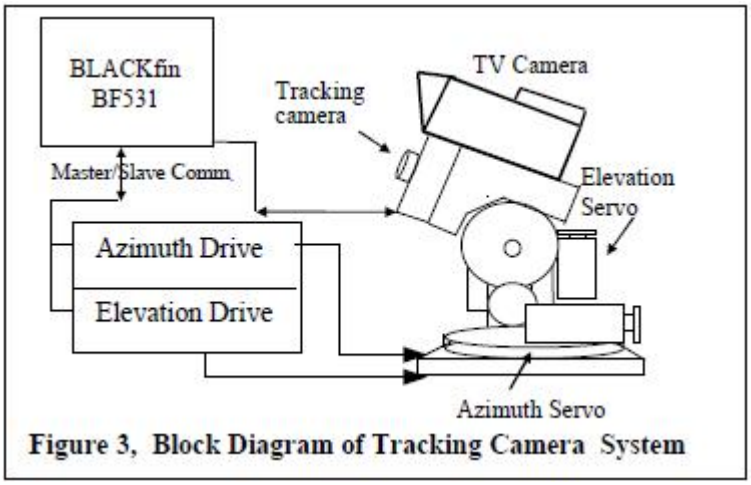


Figure 2.6: schematischer Aufbau

Ballverfolgung bei einer Fernsehübertragung eines Ballspieles.

Bei [15] [20][01048209,01232666] werden vier Kameras verwendet, neben der Stereokamera kommt noch eine IR- und eine RGB-Kamera zum Einsatz, um Hände oder in der Hand gehaltene Objekte zu verfolgen. Hierbei erfolgt die Suche der Farb-Kamera nach Hautfarben. Mit der IR-Kamera hingegen können warme Objekte (z.B. Hände) zur Verfolgung eingesetzt werden. Zielvorgabe des ggst. Forschungsprojekts war jedoch die Bilderfassung der Kameras auf ausschließlich optischer Basis. Personen oder Objekte im Hintergrund können mit der Infrarotkamera allerdings besser herausgefiltert werden. Die Fehlerrate der beschriebenen Testdurchläufe beim Tracking der Hand lag bei rund 6%.
Mit mehreren Stereokameras, wie bei [16][01222176], welche in einem Raum eingebaut sind, werden Menschen mit den Kameras erfasst und verfolgt. Gleichzeitig können Gesten der sich im Raum befindlichen Menschen gefilmt und gedeutet werden. Hierbei sind vier Stereokamerapaare in einem 4,5x3,6 m großem Raum über Kopfhöhe befestigt. In der Höhe wird dieser Raum in Einheiten von 20 cm unterteilt. Diese Technikanwendung ist mit der Vorgangsweise, wie sie in der ggst. Diplomarbeit in weiterer Folge beschrieben wird, mit Ausnahme der Gegebenheiten, dass keine Gesten, sondern nur die Bewegung des **Koordinatengebers** gemessen wird und keine Unterteilung erfolgt und ein viel kleinerer Arbeitsraum (in dem die Hand bewegt wird) zur Verfügung steht, ident.

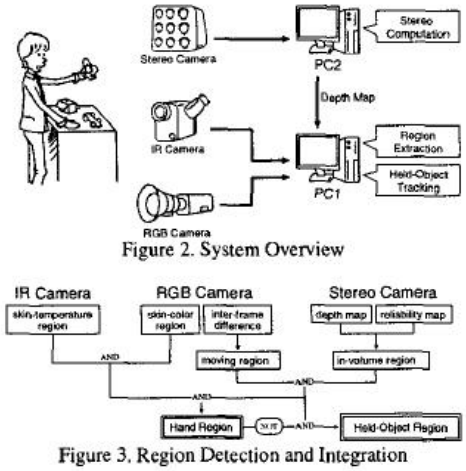


Figure 2.7: Projektaufbau

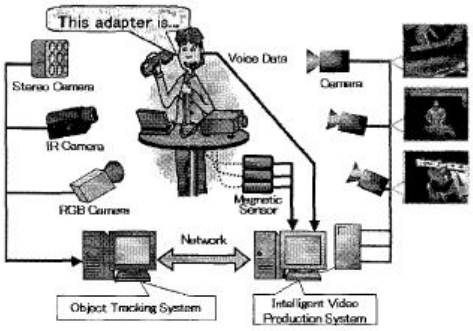


Figure 2.8: schematische Anordnung des Aufbaus

Ohne zusätzlichem Handschuh oder Ähnlichem ist in [8] [01248819] nur die nackte Hand direkt für die Steuerung einer Roboterhand in Verwendung. Dabei sind verschiedene Bilder mit verschiedenen Handkonfigurationen zum Testen benutzt worden. Auch eine Fingerspitzenextraktion zur Erkennung der Fingerrichtung wird verwendet. Die Hand- und die Fingerbewegung wird dabei verfolgt und die Bewegungsdaten an einen PUMA- Roboter geschickt. Als Hintergrund der Hand wird ein schwarzes Brett verwendet (im Rahmen der ggst. Diplomarbeit wahrscheinlich ein schwarzer Teppich am Boden). Mit Hilfe eines Triangulationsverfahren der beiden Bilder einer Stereokamera wird die Position der Hand festgestellt. Dieses Verfahren arbeitet mit einfachen trigonometrischen Dreiecksberechnungen, bei dem mithilfe des bekannten Abstands zwischen den beiden Stereokameraachsen und der Position eines Objektes auf den beiden Kamerabildern (unterscheiden sich gering) die Entfernung zum Objekt berechnet werden kann. Ausserdem wird mit einem kleineren Suchbild versucht, den Rechenaufwand möglichst gering zu halten. Diese Technik ist auch Basis der ggst. Forschungsprojektarbeit. Auch hier wird, nachdem der Mittelpunkt eines **Koordinatengebers** ermittelt wurde, der Bildschirmausschnitt so klein wie möglich gewählt. Sollte sich der **Koordinatengeber** dabei aus dem Bildausschnitt bewegen, so wird dieser solange

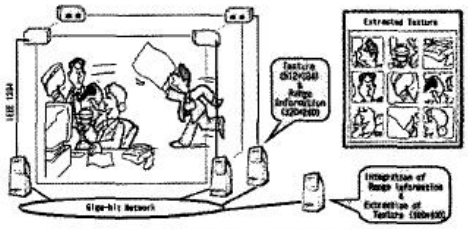


Fig. 1: Concept of Ubiquitous Stereo Vision.

Figure 2.9: Konzept des Aufbaus

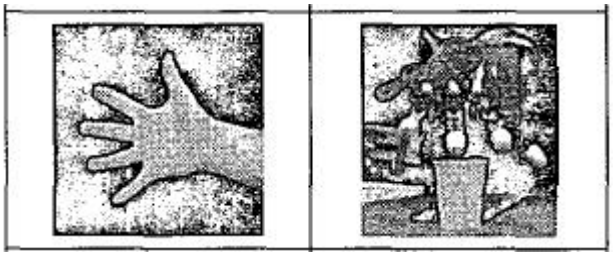


Figure 2.10: ausgestreckte Hand und Roboterhand

vergrößert, bis der Algorithmus den nächsten Mittelpunkt gefunden hat.

Zur besseren Verfolbarkeit des **Koordinatengebers** im ggst. Projekt könnte ein Gegenstand bekannter Form (weißer Ball oder Dreikugelobjekt) auf einem langen schwarzen Handschuh befestigt werden, damit so wenig Störungen wie möglich die Erkennung beeinflussen.

Ein Teil der Diplomarbeit von Herrn Gelhart umfasste die Berechnung der Tiefe des **Koordinatengebers** mittels einem Triangulationsverfahren. Dazu wurden die beiden Bilder der Stereokameras verwendet, um aus der Differenz der beiden Bilder, die jeweils verschoben zueinander sind, die "z" - Position (Bildkoordinatensystem "x" und "y" in der Bildebene, "z" in der Tiefe) zu ermitteln. Bei einem anderen System wird mit einer Suchroutine [14] [01048005] nach dem verlorengegangenen Objekt gesucht. Der hier beschriebene Aufbau entspricht dabei in weiten Teilen dem in der Diplomarbeit zu verwendenden System. Ein "3D"-Objekterkennungsalgorithmus ist dabei für die Winkelausgabe eines Objektes in Verwendung. Zur Lokalisation des Objektes im Stereokamerabild wird ein "3D" Objekttrackingalgorithmus angewandt. Damit ist eine Erkennung eines bekannten Gegenstandes in sechs Freiheitsgraden möglich.

Die gleiche Vorgangsweise wurde auch für [21] [01242105] ausgewählt: hier wird auch mit Stereokameras versucht, die Position und die Orientierung eines bestimmten Objektes zu berechnen. Das System besteht aus einem Aufbau mit "n"-Kameras und einem "Kalmann"-Filter.

Durch die Vielzahl an Kameras ist es möglich, eine optimierte Erfassung des zu suchenden Objekts zu erreichen. Über die Erkennung der Eckpunkte des bekannten Objektes und in Kombination mit einem "3D-"CAD"-Modell wird die Position festgelegt. Dies ist nicht für das ggst. Projekt anwendbar, da die zu erkennende Form immer ident aussieht (egal von welcher Kameraposition), da es sich um ein einfach geformtes Objekt handelt.
In [5] [00194847] werden zwei unterschiedliche Systemaufbauten gezeigt: ein Aufbau mit zwei Stereokameras in einer Ebene, die nebeneinander bzw. übereinander angeordnet sind. Es werden

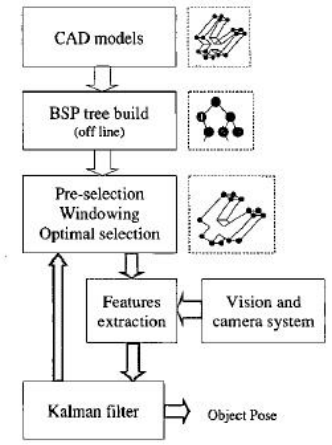


Fig. 3. Functional chart of the estimation procedure.

Figure 2.11: 3D Berechnungssystem

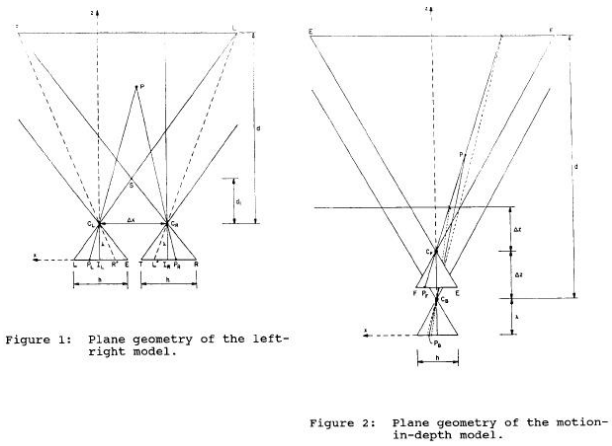


Figure 1: Plane geometry of the left-right model.

Figure 2: Plane geometry of the motion-in-depth model.

Figure 2.12: zwei verschiedene Systemaufbauten

die erreichbare Auflösung und Fehlergrenzen der beiden Methoden untersucht.

Zur Feststellung der Koordinaten der Hand müssten **Koordinatengeber** drei Kugeln (oder Ähnliches) herangezogen werden, dann kann die Lage im Raum berechnet werden. Ein Problem bei der **Bildbearbeitung** ist immer die Verwechslung der Hand mit anderen Objekten durch die Bildverarbeitungssoftware, wenn diese in den Bildbereich gelangen (z. B. ein anderer Körperteil). Zum Beispiel kann es trotz des den ganzen Unterarm bedeckenden Handschuhs zu einer Fehlerkennung kommen, wenn die Bedienperson ein kurzes Hemd trägt, da der Rechner z. B die helle Haut nicht vom **Koordinatengeber** (z. B. weiße Kugeln) unterscheiden kann. Eine weitere Arbeit über die Richtungserkennung einer Hand findet sich in [24] [01333934]. Dabei wird mit vier Stereokameras gearbeitet, die in den Ecken eines Raumes befestigt sind. Die Position, die Haltung und Richtung der jeweiligen beteiligten Person ist in [24] unerheblich, enauso wenig, wie die unterschiedliche Beleuchtung oder auch Kleidungsfarbe keine Rolle spielen sollen. Das Ziel war, eine beliebige Bewegung von einem ausgestreckten etwas zeigendem Arm zu unterscheiden. Außerdem dürfen,

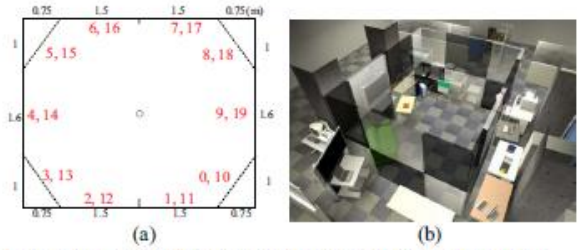


Fig. 4 20 panels dividing walls of experimental room.

Figure 2.13: Position der Kameras und Raumbild

für die einwandfreie Erkennung des Armes (kein Unterschied zwischen rechtem oder linkem Arm, obwohl dieser festgestellt wird), 30 cm um diesen keine Störobjekte sein.

Die Wände des abgeschotteten Raumes wurden in zehn Einheiten horizontal und zwei Einheiten vertikal eingeteilt, damit jedes dieser Unterteilungen die Deutungsrichtung des Armes erkennen kann.

Auch der Stereokameraaufbau, allerdings zur Erkennung des Schreibweges eines Schreibgerätes auf einem Tablett mit einem farbigen Marker, wird in [12] [01508853] näher beschrieben. Im Gegensatz zu einem normalen Tablett wird hier ein Schreibgerät auf einem normalen Blatt Papier verwendet, dessen Schrift zusätzlich optisch digitalisiert wird. Dieses Verfahren ist sinnvoll, wenn kleine Änderungen vorgenommen werden sollen, da eine bessere Orientierung bei einer vor sich liegenden Skizze gegeben ist. Weiters können beliebige Schreibgeräte verwendet werden (mit den entsprechenden Markern), auch Airbrushpistolen zählen dazu. Das Zentrum der Markerfläche wird als Schreibpunkt verwendet. Das heißt, auch wenn keine Spitze zu sehen ist, kann die Schrift digitalisiert werden. Der Basisabstand beträgt 6,5 cm (einem Menschen nachempfunden) und der

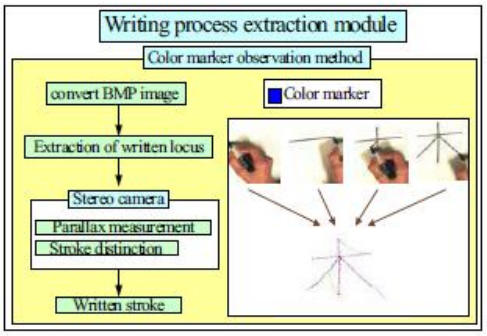


Figure 3: System configuration of vide tablet by stereo camera

Figure 2.14: Programmübersicht

Abstand von der Kamera zum Schreibgerät ca. 30 cm. Um Schreibbewegungen und Bewegungen, bei denen nicht geschrieben wird, voneinander zu trennen, wird mit den Stereokameras die Distanz des Stiftmarkers bei einer Bildwiederholungsrate von 15 Bildern pro Sekunde gemessen. Der Abstand des Stiftes variiert zwischen schreiben und einer Schreibunterbrechung nur um rund 1 cm, dies muss die Tiefenauflösung unterscheiden können. Auch beim ggst. Forschungsprojekt

muss eine gute Tiefenauflösung vorhanden sein. Der Abstand zwischen den beiden Stereokameras sollte ca. 200 mm und der Objektabstand größer als 1500 mm sein.
Mit zwei oder drei Kameras wird mittels Rundumsicht (bei verdeckten Kanten) bei [13] [00201659] versucht, einen Überblick über die Verformungen eines Trägers (I-Profil oder ähnliches) zu ermitteln. Die Genauigkeit geht dabei in den Subpixelbereich mit weniger als 2,4 Prozent Fehler-

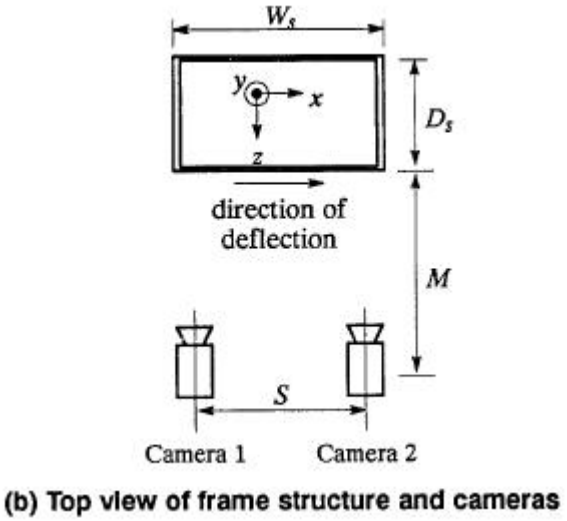


Figure 2.15: Versuchsanordnung

abweichung. Die "schwarz-weiß" Stereokameras werden dazu verwendet, um Strukturparameter unter verschiedenartigen nichtzerstörenden Belastungen annehmen zu können. Das Verfahren der Stereokameraabstandsmessung soll eine Hilfestellung zur Systemidentifikation bei großen Strukturen bieten. Bei der Bilderkennung runder Kreise wurden folgende Parameter verwendet: Radius, Mittenposition, Vorder- und Hintergrundintensitäten. Das nichtlineare Optimierungsproblem wurde mit einer "Least Squares" - Berechnung gelöst, um die besten Werte der Parameter zu erhalten. Ähnlich dazu wird wahrscheinlich auch im ggst. Diplomarbeitsprojekt ein "Least Squares" - Algorithmus zur Anwendung kommen, allerdings, um eine Funktion mit Hilfe mehrerer Messpunkte (tiefenabhängige Fläche, Position, "Feret" - Durchmesser) zu berechnen, die sich optimal an die gemessenen Punkte annähert. Diese Entzerrungsfunktion sollte dann für die Entzerrung eingesetzt werden.

Im vorliegenden Diplomarbeitsprojekt soll ein Stereokamerapaar verwendet werden, bei dem beide Kameras auf einem Gestell befestigt sind, welches wiederum auf einem Tracking Device montiert sind. Beide Kameraachsen sind dabei nicht parallel, sondern in einem Winkel zu einander gerichtet. Der Winkel sollte am Gestell verändert werden können, jedoch ist danach eine neue Vermessung und eine Adaptierung der Berechnung notwendig. Der eingestellte Winkel (der während des Durchlaufs der Erkennungsroutine statisch ist) wird dabei so groß sein, dass sich die optischen Kameraachsen in einem Abstand von ca. 1800 mm treffen (Arbeitsmittenebene). Damit die Einstellungen einfacher durchgeführt werden können, werden wahrscheinlich parallel zu den optischen Kameraachsen Laser montiert werden, um mit Hilfe dieser die Kameras schneller einzurichten. Das "**Tracking Device**" sollte dabei fest auf einem Träger angeschraubt sein. Bei [18] [00669258] wird ein auf einem mobilen Roboter befestigtes Stereokamerapaar dazu verwendet, Abstände von Gegenständen, die auf den Bildern der Kameras aufscheinen, zu messen,

dies jedoch ohne Rekonstruktion in "3D". Es ist also keine Kalibration zur Gewinnung der in-

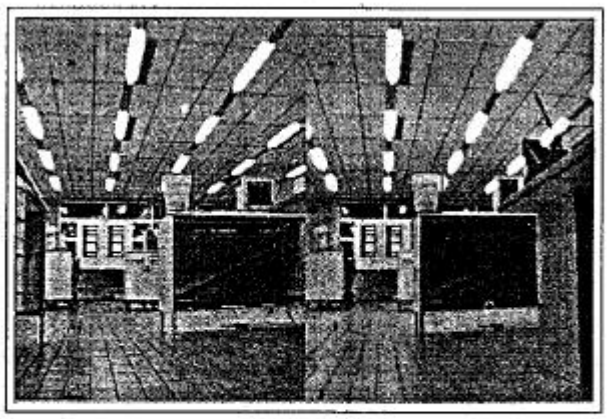


Figure 4: The target on the left corner of the table is focused by the stereo cameras. Its left and right images are in a symmetrical position with respect to the right edge of the left image frame. The two images of the target are also horizontally aligned.

Figure 2.16: Bilder zweier Stereokameras

trinsischen oder extrinsischen Parameter erforderlich. Um eine Beweglichkeit der Kameras zu gewährleisten, sind diese auf einem "**Pan-Tilt-Device**" befestigt. Bei schnellen Bewegungen des **Koordinatengebers** kann es vorkommen, dass sich die entsprechend den Sollvorgaben wie der Koordinatengeber zu bewegenden Stereokameras nicht ausreichend rasch bewegen können. Statt einer "3D"-Rekonstruktion wird die Tiefe mittels eines LUTs (oder eines "2D-Graphs") berechnet. Sollte sich also ein Objekt im Bild befinden, so werden die beiden Stereokameras symmetrisch auf dieses ausgerichtet.
Ähnliche Probleme sind auch bei [17] [01248146] festzustellen. Hier sind zwei Kameras auf einer Befestigung am Kopf montiert. Um hochfrequente Störungen zu beheben, wird mit einem sogenannten "Sobel-lokal-Operator" vorgegangen. In einem nächsten Schritt wird eine Farb Segmentierung vorgenommen. Danach werden die Koordinaten und Segmentgrößen im rechten und im linken Bild berechnet und verglichen. Mit diesen Koordinaten und Segmenten werden die epipo-

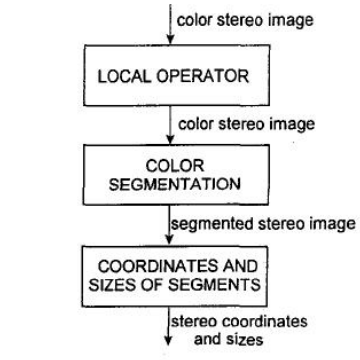


Fig. 4. Flowchart of stereo image processing

Figure 2.17: Ablaufplan der Berechnung

laren "3D" Koordinaten berechnet. Im Rahmen des vorliegenden Diplomarbeitsprojekts soll der **Koordinatengeber** auch über ein **Tracking Device** verfolgt werden. Dieses wird wahrscheinlich, wie schon weiter oben erklärt wurde, fest angeschraubt und lässt nur eine Schwenk- und Kippbewegung der Stereokameras zu. Die "3D" Berechnung sollte über eine einfache Triangulation erfolgen (genaueres bitte in der Diplomarbeit des Herrn Gelhart nachzulesen).

2.1.2 Koordinatengeber

Ein Koordinatengeber ist die Vorrichtung, mit der die momentane Position einer Hand (zum Ansteuern der Maschine) angegeben wird. Hierbei könnte es sich um einen einfachen schwarzen Handschuh handeln (wahrscheinlich wie bei dem in der Diplomarbeit verwendeten System), worauf ein weißer Tischtennisball (oder ein Objekt mit drei Kugeln, oder ähnliches) montiert ist. Sollte es jedoch aufgrund komplexerer Bewegungen (inklusive Drehungen der Hand) erforderlich werden, sämtliche sechs Freiheitsgrade auszulesen, ist der Einsatz eines räumlich eindeutigen **Koordinatengeber** notwendig.
Als Beispiel vorhandener Trackingsysteme siehe [25] [IRtracking-stereocam-Handkonst]. Hierbei ist fest zu stellen, dass für die räumliche Erkennung eines **Koordinatengebers** eine komplexere Form gewählt werden muss, um im Raum präzise erfasst werden zu können. Die Messung der

Figure 2.18: IR- Koordinatengeber

Koordinaten erfolgt hier ([25])mit einem optischen Sensor.
Ohne jegliche Kamera hingegen kommt der Aufbau [26] [cyberglove-datasheet] aus. Hier erfolgt die Steuerung über einen Datenhandschuh. Der Handschuh ist mit 22 Sensoren ausgestattet, die die momentane Haltung angeben. Die Ausgabe der Hand- und Fingerposition erfolgt hier mit biegesensitiven Sensoren, die den Winkel jedes Gelenkes genau angeben. Jeder Sensor ist sehr dünn, was einen leichten und beweglichen Handschuh ermöglicht.
Um bei der Diplomarbeit ohne einen schwarzen Handschuh arbeiten zu können, müsste eine aufwendige Farbauswertung durchgeführt werden, um eine sichere Erkennung der Hautfarbe durch den Prozessor sicher zu stellen. Aufwendig ist dabei nicht die Erarbeitung des Programms, sondern die gegenüber der Verwendung einer Grauwertkamera benötigte, sehr hohe Datenmenge (ca. die drei fache Datenmenge). Daher wird das nächste Beispiel nur aus Gründen der Vollständigkeit angeführt. Mit einem ausgestreckten Finger, wie bei [23] [00711925], wird versucht, einen Laserpointer oder ein anderes Zeigegerät zu ersetzen. Dabei wird auch mit einem beweglichen Stereokameraaufbau (aufgrund der notwendigen hohen Auflösung) gearbeitet, die den Finger verfolgen. Es finden sich zwei unterschiedliche Arten von sogenannten "Fingerpointern" in Anwendung: 1. In einem Fall erfasst das Kameraobjektiv ausschließlich die Fingerzeigerichtung 2. Im anderen Fall wird auch die Ausrichtung vom Auges zur Fingerspitze zusätzlich erfasst.

Figure 2.19: Datenhandschuh

Figure 2. This figure illustrates the operation of our free-hand pointer. The speaker is pointing his index finger to the projection screen (under the word "3D"). The IIS head on the left takes the stereo image pairs to be processed. According to the projection site determined from the stereo image pairs, the computer then controls its cursor to the word "3D".

Figure 2.20: Fingerpointer

Die zweite Methode ist dabei die einfacher zu realisierende von beiden. Um eine möglichst hohe Auflösung auch in einem größeren Raum zu erhalten, ist der Bildausschnitt eher klein. Allerdings werden die Kameras analog zur Bewegung des Bildes der Fingerspitze mit bewegt. Im ersten Bildbearbeitungsschritt erfolgt die Binärisierung der beiden Bilder, wobei der Trennungslevel automatisch berechnet wird. Danach wird mit einem morphologischen Algorithmus die Fingerspitze im gesamten Bild global gesucht. Bei der Erfassung des Fingers durch die Kamera wird die Position mit einer Prädiktion lokal vorausberechnet. Erst, wenn die Fingerspitze außerhalb des Bildbereichs liegt, oder verloren gegangen ist, wird mit einem globalen Algorithmus gesucht. Die Ausrichtung der selektierten Fingerspitze erfolgt mit einem einfachen Algorithmus mithilfe der Annahme, dass der Finger zylindrisch ist.

2.1.3 Verfahren für die Kalibrierung, Kollisionserkennung und Entzerrung

Eine einfache analytische Kalibration der extrinsischen Parameter der aufgenommenen Bilder ist in [3] [01266651] zu sehen. Zuerst unterteilt eine Kalibrationsgleichung rotatorische und translatorische Parameter. Die Kalibrationsgleichung benötigt nur rotatorische Anteile und keine genauen Positionsangaben. Eine "Vier-Punkt-Kalibrationsprozedur" braucht drei Punkte auf einer Linie und einen weiteren, beliebig im Raum angeordneten Punkt.Diese Prozedur führt zu

vier möglichen Lösungen. Zusätzliche Berechnungsschritte werden notwendig, um falsche Lösungen auszusortieren. Sobald die echten rotatorischen Parameter identifiziert sind, werden die translatorischen Parameter in analytischer Form ermittelt. Die absolute Positionsinformation liegt in weiterer Folge in einfacher und expliziter Form vor.
Ein Roboter, der mit zwei Kameras in einem Stereoaufbau gesteuert wird, wird in [1] [KollerkenngfRoboter] erläutert. Dabei wird hier auch schon eine Kollisionserkennung herangezogen, wodurch eine Roboter-Werkstück-Kollision vermieden werden soll. Diese Kollisionsvermeidungsroutine

Abbildung 2.1: Die Fertigungszelle mit eingesetzten Bildverarbeitungskomponenten.

Figure 2.21: Fertigungszelle Aufbau

wird eine Hauptrolle bei zukünftigen Arbeiten an diesem Projekt beanspruchen, da sonst keine sichere Bearbeitung eines Werkstückes möglich sein wird. Dies Themenbereich ist allerdings nicht Teil der ggst. Diplomarbeit, auch wenn darauf und auf weitere, damit zusammenhängende Aufgabenstellungen am Beginn dieser Arbeit hingewiesen wird.
Um die Bilder eines Stereokamerapaares, welches sich auf einer drehenden Neige- und Schwenk-Plattform befindet, auswerten zu können, wird dieses mit [22] [01041382] berechnet. Die Be-

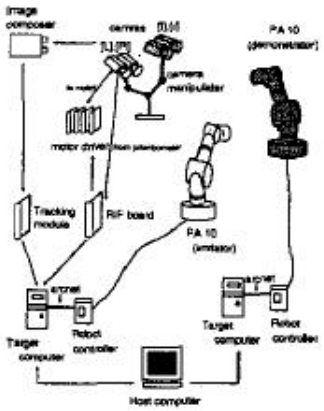


Figure 8: Experimental setup

Figure 2.22: Aufbau der Versuchsanordnung

wegungen gehen dabei so vor sich, dass keine Änderung des optischen Zentrums und damit der epipolaren Gleichung zustande kommt. Die sphärische Projektion ist die Randbedingung.

[7] [00071365] Mit der epipolaren Geometrie werden die Beziehungen zwischen den optischen Feldern auf den Bildern berechnet und zur Angleichung der Kamerabilder verwendet, um eine Tiefeninformation zu erhalten.

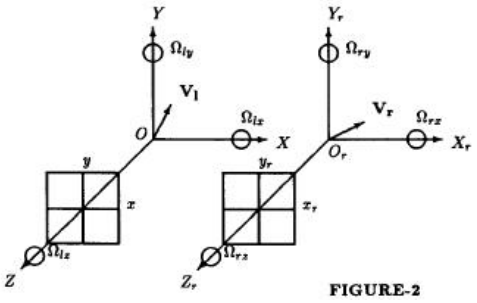


Figure 2.23: Berechnung bei parallelen Achsen

3 Realisierung

3.1 Grober Überblick

Zu Beginn dieser Diplomarbeit waren nur einige wenige Voraussetzungen zu beachten. Zum Beispiel wurde festgelegt, dass die Ansteuerung der Maschine ausschließlich optisch zu erfolgen habe. Als anzusteuernde Maschine lag eine Blechbearbeitungsmaschine vor, welche zwei Ständer mit jeweils 6 Freiheitsgraden und zwei fahrbare Schlitten mit jeweils 6 Freiheitsgraden und zwei Greifarmen aufwies. Ein Schlitten und beide Greifarme wurden abmontiert und entfernt. Anstelle des Greifarms ist die Verwendung einer Frässpindel in weiterer Zukunft vorgesehen. Die diesbezügliche Umsetzung und Anwendung ist jedoch nicht im Umfang dieser Arbeit enthalten. Wie genau die Ausführung des optischen Aufbaus ausgestaltet sein sollte, war nicht vorgegeben. Wie sich im Laufe der Arbeit herauskristallisierte, erwies sich die Ansteuerung für die x- und z- Achse der Maschine und die Steuerung des Tracking Devices als sinnvoll. Die Ansteuerung der y, A-, B- und C-Achse der Maschine wurde im Zuge der Diplomarbeit des Herrn Gelhart ausgearbeitet (siehe die Diplomarbeit von Hrn. Gelhart).

Die x,y und z Achsen sind hierbei die translatorischen Freiheitsgrade und die Achsen A, B und C die rotatorischen (näheres siehe [2]). Durch die doch recht hohen Anforderungen an die Genauigkeit genügte es nicht, nur eine Kamera (oder ein Stereokamerapaar) zu verwenden, da sich die Auflösung über den gesamten Arbeitsbereich als zu gering für eine gute Tiefenberechnung herausstellte. Aus diesem Grund ist das System mit drei optischen, baugleichen Kameras entstanden, wobei eine Kamera den gesamten Arbeitsbereich mit einer relativ geringen Auflösung abdeckt um die Berechnung zu beschleunigen. Mit den sich daraus ergebenden Koordinaten wird die Position des **Tracking Device**s angesteuert. Die auf diesem Gerät befestigten Stereokameras halten den **Koordinatengeber** dabei immer in Bildmitte. Um die Berechnungen durchführen zu können, war eine Entscheidung zwischen einer "Smartcam" (Kamera und Recheneinheit in einem Gehäuse) oder einer Karte, die man in einen PC einbauen kann, zu treffen. Obwohl es schon recht schnelle "Smartcams" am Markt gibt, war die Tatsache, das die Konfiguration (z.B. erweitern der Anschlüsse...) und die Komponenten des Systems nicht veränderbar sind, Grund genug, um zwei "PCI Express Einbaukarten" von Matrox anzuschaffen. Die auf diesen Platinen montierten Berechnungseinheiten sind sehr schnell und extra für graphische Berechnungen ausgelegt (Näheres siehe Kapitel 3.7). Zusätzlich mit einem "Dongle" für die Lizenz können Berechnungen aber auch an den Host (PC) ausgelagert werden. Diese Vorgehensweise würde allerdings eine Multitaskingfähigkeit des Programms voraussetzen, welche sich durchaus noch in weiterer Folge realisieren ließe. Um den verwendeten PC noch zu beschleunigen, wurde ein zweiter Prozessor

eingebaut. Zur Programmierung (siehe Kapitel 4) wurde das im Umfang von "Visual Studio 2005" enthaltene Programmiersprache "C++" verwendet. Diese Entscheidung rührte daher, dass diese Hochsprache eine schnelle echtzeitfähige Programmierung ermöglichte. Passend zu den bestellten "Framegrabbern" wurde für die **Bildbearbeitung** auch eine Software der Firma Matrox verwendet (Version "Mil 8.0", die gegen Ende der Arbeit durch die ausgereifte, besser angepasste und aktuellere Version "Mil 9.0" ersetzt wurde, was einigen zusätzlichen Programmieraufwand aufgrund der notwendigen Anpassung des Programms an die neue Version verursachte).

3.2 Vorüberlegungen

Um einen Überblick über den Projektumfang zu erhalten, werden in diesem Kapitel Grundüberlegungen betreffend des benötigten Datenumfangs angestellt, das für die Umsetzung dieses Projekt erforderlich ist. Dabei wird weit über die eigentlichen Arbeitsgrenzen dieser Arbeit hinausgegangen. Der gesamte Arbeitsbereich dieses Projektes umfasst die Programmierung der Bildverarbeitung. Im Detail gehören dazu die **Bildbearbeitung**, die Entzerrung der Tiefe, die Erstellung des Bildzuschnittsalgorithmussses und einiger Einstellroutinen (Laserpointereinstellung, Berechnung der Koeffizienten für die Entzerrung, Änderung der "Inifile-Datei", Aufnahme und Abspeicherung der Entzerrungspunkte, Erarbeitung der "**Tracking-Device**" Einstellung, erstellen eines "Logfiles"). Um die Beziehung zwischen den notwendigen Daten und den Arbeitsblöcken (gewählte Abgrenzungen zwischen verschiedenen Arbeitseinheiten) darzulegen, werden diese weiter unten angeführt. Die verwendeten Eingänge und Ausgänge sind dabei die Daten, die über die gewählten Grenzen zwischen den einzelnen Blöcken herein- oder herausgelangen müssen und in weiterer Folge zur Beschreibung des Systems notwendig sind. Die benötigten EINHEITEN sind dabei Variablen, welche bekannt sein müssen, um den Block zu beschreiben, oder um damit arbeiten zu können. Die zu erfüllenden Voraussetzungen geben dabei die Randbedingungen an, die eingehalten werden sollten.

1. Allgemein: Erläuterung des Projektes: Mit dem zu erstellenden Programm soll es möglich sein, eine Maschine zur Erstellung von Prototypwerkstücken aus weichen Materialien mittels einer, über drei digitale Kameras, erkannten Hand mit Zusatzmarkierungen intuitiv anzusteuern, welche in drei Linearachsen geführt werden kann, und in alle drei Schwenkrichtungen (mit Einschränkungen) drehbar ist. Dazu befindet sich ein Arbeitsbereich in der Nähe der Maschine. Die Kameras zur Erkennung der Handposition bestehen dabei aus einer **Trackingkamera** (die die Aufgabe hat, den gesamten Arbeitsbereich abzubilden und die Hand in diesem aufzufinden) und zwei Stereokameras, die für die genaue Koordinaten- und Winkelberechnung der Hand notwendig sind. Diese sind auf einem kipp- und drehbaren **"Tracking-Device"** montiert, das mit Hilfe der aus der Trackingberechnung ermittelten Koordinaten der Hand folgt. Die Berechnung erfolgt mittels eines schnellen "Framegrabbersystem" (zwei oder drei Karten mit integriertem Prozessor, Speicher und festverdrahteten Recheneinheiten und eventuell einem "FPGA") und einer "Workstation" mit zwei Prozessoren.

   Mit den ermittelten Koordinateninformationen der **Bildverarbeitung** wird nun, unterbrochen durch einige sicherheitstechnisch notwendige Berechnungen (Kollisionen, Schnittgeschwindigkeiten, Kamerabewegungsgeschwindigkeiten, Werkzeugbeschränkungen, Maschinenbeschränkungen,...), die Maschine über eine "Sercos"-Schnittstelle angesteuert, wobei die notwendige Regelung schon in der Maschinensteuerung implementiert ist. Das Werkstück

befindet sich während der gesamten Bearbeitung fest auf einem Drehtisch montiert, der über den Bearbeiter angesteuert werden kann(z.B. mit einer Handsteuerung, oder über virtuelle "Icons", die am Rande des Bildbereiches außerhalb des **Arbeitsbereiches** zu finden sind) und Rückgabewerte an das Programm liefert, um Kollisionen zu vermeiden.

2. Bildverarbeitung: Eingänge: Eine Trackingkamera, die Bildinformationen zur Lokalisierung einer bewegten Hand verwendet und den gesamtem Arbeitsbereich (ca. 400*400 Pixel, Monochrom) filmt.

   Zwei Stereokameras, die Bildinformationen zur genauen Positions- und Winkelermittlung verwenden und einen kleinen Bildausschnitt des gesamten Arbeitsbereiches (ca. 1000*1000 Pixel, Monochrom) abbilden.

   Ausgänge: Die Koordinaten und Winkel der bewegten Hand, um die "3D"-Position festlegen zu können, ausserdem die Geschwindigkeiten der Hand und eventuell (falls der Aufwand nicht zu hoch ist) die Beschleunigungen der Hand.

   Zu erfüllende Voraussetzungen:

   - Die Implementierung muss schnell und möglichst effizient sein, daher ist diese auch möglichst einfach gehalten. Bildfrequenzen von ca. 100 Bilder pro Sekunde sollen erreicht und in Echtzeit berechnet werden (diese geschieht auf einem eigenen Prozessor auf einer Framegrabbereinheit).
   - Mittels Trackingkamera soll bei recht grober Auflösung über den gesamten **Arbeitsbereich** die Position der Hand erfasst werden. Mit den Positionskoordinaten wird das Trackinggerät (welches vorher mittels Initialisierungsprogramm auf den gesamten **Arbeitsbereich** eingestellt wurde) so gesteuert, dass sich die Hand im Bild der Stereokameras ("Prinzip menschliches Sehen") befindet, die auf dem dreh- und schwenkbaren Gerät montiert sind. Diese Kameras nehmen nur einen kleinen Teil des gesamten **Arbeitsbereiches** auf, aber mit erhöhter Auflösung, um eine genaue Tiefeninformation berechnen zu können.
   - Um die Handerkennung für die Berechnung zu erleichtern, werden auf dem Handschuh kleine Koordinatensysteme montiert, die, um den Kontrast zu verstärken, eine größtmögliche Helligkeit (z. B. weiße Farbe) aufweisen sollten. Aus den gleichen Gründen sollte der Handschuh oder Ähnliches möglichst in schwarz gehalten sein).
   - Erkennung, ob keine Hand im Bildbereich , oder ob diese verdeckt ist.
   - Berechnung der Geschwindigkeits- und Beschleunigungsvektoren

   benötigte EINHEITEN: Stereobildkoordinaten, Trackingkoordinaten, Geschwindigkeits- und Beschleunigungsvektoren.

3. Maschine: Eingänge: Alle korrigierten (nach Berechnung der Korrektur, möglicher Geschwindigkeit, der Kamerafahrbereiche...) Koordinaten- und Winkel, inklusive der Geschwindigkeit für die Maschine.

   Ausgänge: Aktuelle Maschinenkoordinaten- und Winkel (eventuell momentaner Stromverbrauch der Motoren für Überlastungsschutz).

   Zu erfüllende Voraussetzungen:

   - Maximale Maschinengeschwindigkeit ausnützen
   - Maximale Maschinenbeschleunigung ausnützen

- Maschinenabmessungen zu Verfügung stellen
- Maschinenleistung, Maschinendrehzahl (Maximalwerte) zur Verfügung stellen
- Sonstige notwendige Maschinenkenndaten zu Verfügung stellen

benötigte EINHEITEN: Stellt Maschinenkoordinaten, Winkel, Maschinenvektoren (Geschw., Beschl.), aktuelle Maschinenmotorströme, aktuelle Maschinenspindeldrehzahlen zur Verfügung. Außerdem liegt eine statische Datenbank vor mit: Maschinenmotorenleistungen, Maschinenspindeldrehzahlen, Maschinenfahrbereiche und Maschinenabmessungen (der einzelnen zueinander beweglichen Teile).

4. Werkstück: Eingänge: Alle neuen Handkoordinaten- und Winkel, Werkzeugkenndaten, Drehtischkenndaten und die maximal mögliche Schnitttiefe.

   Ausgänge: Momentane maximale Schnitttiefe mit aktueller Werkstückgeometrie, abzüglich der alten Werkstückgeometrie, Material des Werkstückes, E-Modul und weitere Kenndaten für die Berechnung der Vorschubgeschwindigkeit und der Schnitttiefe.

   Zu erfüllende Voraussetzungen:

   - Berechnung der momentanen Schnitttiefe (Speicherung der notwendigen Daten)
   - Tabelle oder Auflistung (Datenbank) aller möglichen Rohteile inklusive Materialdaten, Abmessungen, Dichte (welches Gewicht hält der Drehteller des Drehtisches aus?)und daraus errechnete Maximalabmessungen mit der Möglichkeit der benutzerdefinierten Erweiterung. Auch eine Momentenberechnung (Werkzeug auf Werkstück, oder Drehtisch mit großer Sicherheit)zur Abschätzung, ob eine Bearbeitung möglich ist (maximale Bearbeitungskräfte, Beschränkungen der Stromzufuhr in Abhängigkeit der Position).
   - Implementierung eines Objektes, das die momentanen Geometriedaten abspeichern kann ("3D"-Array an diskreten Punkten oder Ähnliches).

   Benötigte EINHEITEN: Statische Datenbank mit den folgenden Daten: Abmessungen von Rohwerkstücken, Materialien, Werkstoffkenndaten ("E-Modul", Dichte und Ähnliches).

5. Werkzeug: Eingänge: Materialkenndaten der Werkstücke (Änderung der möglichen Schnitttiefen; ist eine Bearbeitung überhaupt mit diesem Werkzeug möglich? Wenn dies nicht der Fall ist, dann sollte durch die Grenzbereichszuordnung ein rotes Licht leuchten).

   Ausgänge: Werkzeugkenndaten (maximale Schnitttiefe, Werkzeugkontur, Abmessungen des Werkzeuges, maximale Drehzahlen, bearbeitbare Materialien,...). Ist vielleicht auch eine Datenbank erforderlich?

   Zu erfüllende Voraussetzungen:

   - Datenbank mit den jeweiligen Kenndaten eines Werkzeuges, welche für eine Bearbeitung notwendig sind. Abhängigkeiten von den unterschiedlichen Materialien.

   Benötigte EINHEITEN: Benötigt wird eine statische Datenbank. Enthalten sind die Werkzeugkonturfunktion, die maximal zulässige Arbeitstiefe (oder die maximal nutzbaren Schneidflächen eines Werkzeuges), das Werkzeugmaterial oder die Materialien, die bearbeitbar sind.

6. Drehtisch: Eingänge: momentane Drehtischposition (Winkel, Zahl für Position, 90°-Teilung).

   Ausgänge: Drehtischkenndaten (Abmessungen, Höhe, Tischbreite, Verdrehdauer, Winkelgeschwindigkeit , um Kollisionen zu vermeiden...), das maximale Aufnahmegewicht und Aufnahmemoment.

   Zu erfüllende Voraussetzungen:

   - Erfassung der momentanen Drehtischdaten, bei unklaren Daten aktivieren eines "Notaus".
   - Speicherung der allgemeinen Drehtischdaten in einer Tabelle (Datenbank), die nicht für jeden änderbar ist (nur bei Drehtischänderungen).

   Benötigte EINHEITEN: Drehtischdatenbank oder Tabelle mit den Daten. Hierbei handelt es sich um die Abmessungen des Tisches, die maximalen Gewichte und die maximalen Kräfte.

7. Korrekturberechnungen, Fehleranalyse Schutzzonen

   Eingänge: Neu mit den momentanen Handkoordinaten berechnete Werkzeuggeometrie, Drehtischabmessungen, aktuelle Handkoordinaten, Winkelgeschwindigkeiten und Beschleunigungen, neueste Maschinenkoordinaten, Winkel (Geschwindigkeiten u. Beschleunigungen sind eventuell notwendig, da die Schutzzonengröße in Abhängigkeit der Maschinengeschwindigkeit ist), Maschinenabmessungen, Änderung der Schutzzonen bei notwendigen Umschwenkbewegungen bei Unerreichbarkeit einer Position.

   Ausgänge: Kollisionsobjekte, oder eventuell auch schon eine implementierte Kollisionsberechnung aller bekannten und momentan aktuellen Kollisionsobjekte (dann gibt der Ausgang nur "ja" oder "nein" an, je nachdem, ob eine Kollision erfolgt, welches Objekt betroffen ist, welches Kollisionsausmaß gegeben ist, ...)

   Zu erfüllende Voraussetzungen:

   - Berechnung der dynamischen Kollisionsobjekte aufgrund der momentanen Geschwindigkeiten, Koordinaten, und eventuell auch der Beschleunigungen.
   - Auslesen von statischen Kollisionsobjekten aus den jeweiligen Datenbanken (Tabellen).
   - Kollisionsberechnungen unter Berücksichtigung der Sinnhaftigkeit (um z.B. die Effizienz zu steigern , könnten eine Anfrage erstellt werden, ob das Werkstück mit dem Drehtisch kollidieren kann oder, dass bei offensichtlichem Fehlen einer Kollision keine weitere Detailrechnung erfolgen sollte).
   - Geschwindigkeits- und eventuell beschleunigungsabhängige Schutzzonen (fährt das Werkzeug direkt auf das Werkstück zu, oder fährt es nur in tangentialer Richtung schnell?). Je schneller das Werkzeug auf das Werkstück zufährt, desto größer sollte auch die Schutzzone sein. Wie steht es um die Sicherheit? Z.B. die Trägheitsverhältnisse der Maschine, dabei sollte der Bremsweg in Abhängigkeit zur Geschwindigkeit berechnet werden.

   Vorschubgeschwindigkeit, Schnitttiefe

   Eingänge: Aktuelle Schnitttiefe, Handgeschwindigkeiten, Beschleunigungen, momentane Motorströme, Materialdaten, alle Werkzeugkenndaten, maximale Motorleistungen (Diagrammdaten?) und die maximalen Drehzahlen der Spindel (Drehzahlleistungsdiagramm,...).

Ausgänge: Maximal mögliche Schnitttiefe, Vorschubgeschwindigkeit Zu erfüllende Voraussetzungen:

- Berechnung der maximalen Schnittgeschwindigkeit und Vorschubgeschwindigkeit bei maximaler Drehzahl der Spindel.

Umschwenkbewegung

Eingänge: Aktuellste Maschinenkoordinaten und Winkel.

Ausgänge: Sollte ein Umschwenken der Maschine notwendig sein, so müssen Aufgrund von Unerreichbarkeiten oder einem Signal der Grenzbereichsberechnung, dass die Bearbeitung nicht möglich ist, Bewegungen gesperrt werden.

Zu erfüllende Voraussetzungen:

- Position der Maschine erkennen, Umschwenkbewegung einleiten, Signal an die Grenzbereichsberechnung senden (in diesem Fall sollte ein gelbes Licht leuchten und dies anzeigen, oder bei zu großer Differenz zwischen Hand- und Maschinenkoordinaten wird ein "Notaus" ausgelöst).

Geschwindigkeitsdifferenz

Eingänge: Geschwindigkeit der Maschine, und der Hand und die Koordinaten beider Bewegungen.

Ausgänge: Geschwindigkeitsdifferenz, Koordinatenabstand (somit Fehler der Örtlichkeit).

Zu erfüllende Voraussetzungen:

- Berechnung der Geschwindigkeitsdifferenz.
- Die Berechnung der Koordinatendifferenz sollte der Fehleranalyse zugeführt werden.

Grenzbereichsberechnung

Eingänge: Umschwenkbewegung, Vorschubgeschwindigkeit und Schnitttiefe, Kollisionssignal, Kollisionstiefe, Maschinengeschwindigkeit, beteiligte Kollisionsobjekte und das Geschwindigkeitsdifferenzergebnis.

Ausgänge: Arbeitsstatus (grün, gelb, rot) soll physisch angezeigt werden, Rückgabe an die Werkzeuggeometrie. Bei einer Kollision sind die Geometrie zu bearbeiten, die Koordinaten zu ändern und ein verlangsamen der Maschinengeschwindigkeit notwendig (aktuelle Handkoordinaten so abändern dass eine Bearbeitung möglich ist).

Zu erfüllende Voraussetzungen:

- Verwalten aller Fehlermeldungen.
- Abschätzen der Fehler .
- Korrekturaufträge an verschiedene Unterobjekte.
- Ausgabe der Endkoordinaten an die Maschine (Koordinaten, Winkel, Geschwindigkeiten, Beschleunigungen).

Benötigte EINHEITEN:

Werkstückgeometrie:
Berechnet die neue Werkstückgeometrie aufgrund der neuen Handkoordinaten und die

Schnitttiefe aufgrund der alten Geometrie und des Kamerafahrweges. Als Eingang wird auch ein Rückgabewert der Kollisionsrechnung verwendet (maximal befahrbare Koordinaten, ohne eine Kollision mit dem Werkstück zu verursachen).

Vorschubgeschwindigkeit, Schnitttiefe:
Dies wird mit den aktuellen Maschinenleistungen, den Kennlinien (Datenbank), anderen Materialkennwerten, den Werkzeugkennwerten, der momentanen Handgeschwindigkeit und der Maschinenspindeldrehzahl berechnet (Maximum in Datenbanken; und eventuell die momentane Drehzahl, damit keine allzu großen Drehzahlsprünge bei starker Variation der Schnitttiefe entstehen, was anzunehmen ist). Diese gibt die durch Schnitttiefe und die Vorschubgeschwindigkeit beschränkten Handkoordinaten aus. Alle Fehlermeldungen werden ebenfalls angezeigt.

Positionsdifferenz (alt):
Diese berechnet die Abweichungen der Maschine zur Position mit Hilfe der momentanen und der alten Handkoordinaten in Kombination mit den aktuellsten Maschinenkoordinaten. Weiters wird hierbei errechnet, ob es möglich ist, diese im nächsten Schritt zu kompensieren. Sie gibt eine Fehlermeldung aus, wenn dies nicht im nächsten Schritt möglich sein sollte (die Variable ist einstellbar, ab welchem Fehler oder ab wie vielen Berechnungsschritten ein Fehler ausgegeben wird. Warnleuchten werden erst GELB, in der nächsten Stufe kommt ein Notaus). Die hier errechnete Positionsdifferenz wird mit alten Koordinaten berechnet und kann daher nur als Kontrolle der vorher berechneten Sollkoordinaten dienen (ist die Maschine in die gewünschte Position gefahren oder nicht, wenn nicht, warum nicht, da schon in der Berechnung der Maschinenkoordinaten auf maximale Geschwindigkeiten und Beschleunigungen auf die Koordinaten geachtet worden ist).

Schutzzonen, Kollisionsrechnung:
In diesem Modul werden alle vorhandenen starren oder beweglichen Körper mit "Schutzhüllen" umgeben, die abhängig von der Geschwindigkeit, mit der der Greifarm bewegt wird, aber auch abhängig von den absoluten Koordinaten, sowie dem Verdrehwinkel der jeweiligen Schutzzonen (Maschinenarm, Drehtisch...) ausgestaltet wird. Dabei muss berechnet und kontrolliert werden, ob etwaige Kollisionen möglich sind. Trifft dies zu, dann werden die beteiligten Objekte in einer Fehlerausgangsmatrix aufgelistet (je nachdem, wie viele an Kollisionen beteiligt sind) und die maximal möglichen Koordinaten und Geschwindigkeiten berechnet.

Umschwenkbewegungen:
Aufgrund allgemeiner Unzulänglichkeiten der für das Forschungsprojekt herangezogenen Werkzeugmaschine (Maschinenfahrwegsbegrenzungen, Erreichbarkeitsprobleme etc.) ist es manchmal notwendig, die Lage der Maschine zu verändern, um bei der gleichen Koordinate nach dem Verfahren weiterarbeiten zu können. Zu diesem Zweck werden die Handkoordinaten und Winkel zur Berechnung herangezogen, um notwendige Umschwenkbewegungspunkte zu lokalisieren und auszuführen. Die Ansteuerung derartiger Koordinaten zieht eine Fehlermeldung nach sich. Gleichzeitig werden mögliche Koordinaten angegeben.

Gewichtsberechnung:
Hier wird auf die maximalen, durch den Drehtisch unterstützten Gewichtskräfte Rücksicht genommen. Auf Basis der Art des Werkstückmaterials, der Werkstückmaße und der Drehtischdaten erfolgt bei Überschreitung eine Fehlermeldung.

Graphische Berechnungen:
Dieses Modul stellt auf Wunsch nahezu alle errechneten Daten in einem bestimmten Anzeige-

format dar. Etwaige Diagramme oder Ähnliches sollen auch graphisch ausgegeben werden.

- Virtuelle Anwendersteuerung Eingänge: Handpositionskoordinaten der Trackingkamera, eventuell Handgeschwindigkeit.

Ausgänge: Verschiedene Steuerungsaufgaben (virtuelle Eingabe von Bearbeitungsminima über Handposition, sonstige Hilfsaufgaben, umschalten des **Arbeitsbereiches** bei größeren Werkstücken (fest eingespannt, nicht mit Drehtisch drehbar), Oberflächengüte verbessern (über ein Icon aktivieren).

Zu erfüllende Voraussetzungen:

- Feststellung, wenn die Hand im virtuellen Steuerungsbereich ist.
- Erkennung eines Ansteuerungsbefehls.
- Ausführen des Kommandos.
- Maschine im virtuellen Bearbeitungsmodus in eine Sicherheitsposition fahren.
- Virtuelle Icons im Bildbereich zur Verfügung stellen.

8. graphische Ausgabe Eingänge: Alle Koordinaten inklusive Winkel, Handbild, eventuell virtuelle Werkstückgeometrie, Handkoordinaten visualisieren (eventuell Auswertungen der Bearbeitungsgüte, Exaktheit des Fahrwegs prüfen, Oberflächengüte statistisch erfassen), Fehlerausgabe, Geschwindigkeitsdifferenzen, Fehler graphisch anzeigen und anzeigen von Diagrammen. Vor der Bearbeitung sollte die Möglichkeit zur Einstellung der zu speichernden Daten für eine Analyse oder Ähnliches berücksichtigt werden, darüber hinaus auch eine Zusammenfassung aller Daten (Motorströme, Handbewegung...).

   Ausgänge: Graphische Darstellung (ein- und ausschaltbar, um die Rechenleistung nicht zu reduzieren)

   - Speichern von Daten für Statistiken.
   - Berechnung der charakteristischen Daten.
   - Graphische Ausgabe.

3.3 Basisüberlegungen zu den Faktoren der Systemgestaltung

Die das gesamte Projekt behandelnde Grundplanung wurde am Beginn der Arbeit erstellt. Mittels einer Prioritätsliste wurden die verschiedenen Arbeitsbereiche eingeteilt und damit zeitlich gestaffelt. Dass diese grobe Planung nicht bis ins Detail mit dem endgültigen Ergebnis übereinstimmen kann, ist klar, doch es gibt einen guten Überblick über das Ausmaß des Arbeitsumfanges. Der Arbeitsumfang erstreckt sich über die erste Priorität der **Bildverarbeitung** bis zu Teilen der Berechnung (Priorität 2) und der Maschine (auch Priorität 2).

Entsprechend den Angaben im Bild 3.1, sind nach der Bildverarbeitung folgende Variablen ausgehend:

- XH (x-Wert in Bildebene)
- YH (y-Wert in Bildebene)

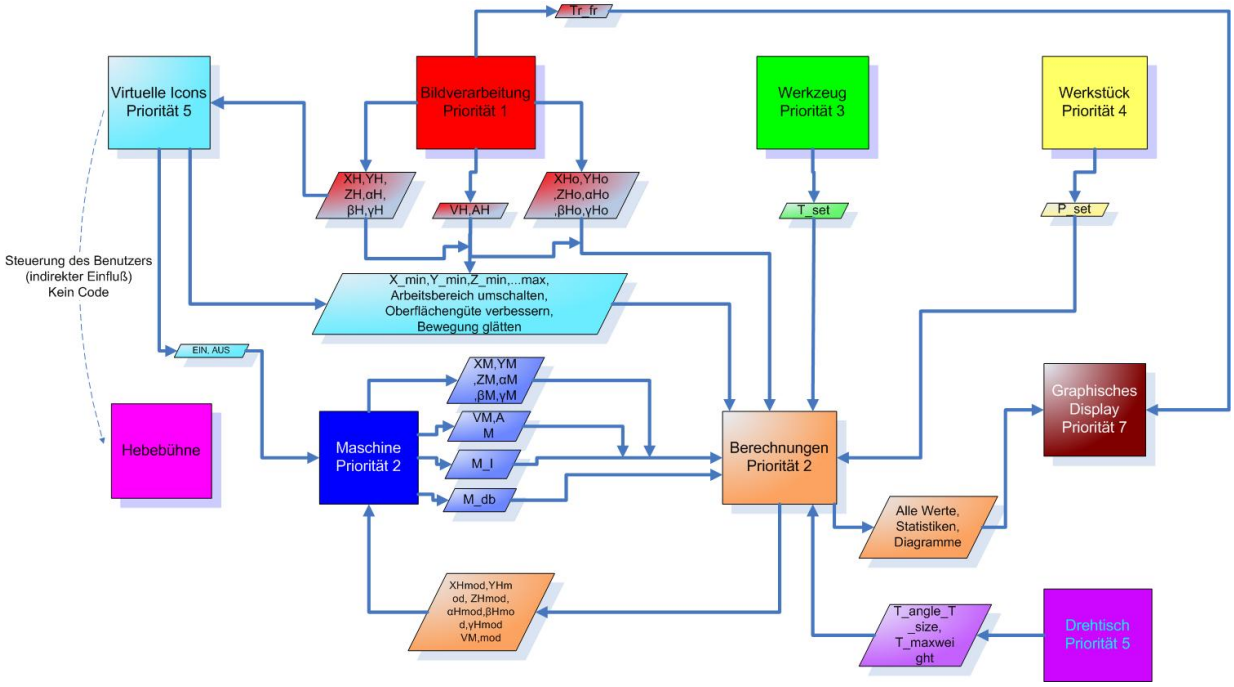


Figure 3.1: Programmierdiagramm Übersicht

- ZH (z-Wert; Höhe normal auf die Bildebene)
- αH (Winkel α)
- βH (Winkel β)
- γH (Winkel γ)
- VH (Volumen des virtuellen Werkstückes mit erkannten gefahrenen Wegen berechnet)
- AH (Fläche des erkannten **Blobs**)
- XH_0 (x-Wert alt)
- YH_0 (y-Wert alt)
- ZH_0 (z-Wert alt)
- αH_0 (Winkel α alt)
- βH_0 (Winkel β alt)
- γH_0 (Winkel γ alt).

Sämtliche oben angeführten Variablen werden für die Berechnungen herangezogen. Zum graphischen Display wird das momentane Trackingbild gesendet (Tr_fr). Auf diesem graphischen Display sollen alle relevanten Daten während des Bearbeitungsvorganges angezeigt werden. Dazu gehören Schnittgeschwindigkeiten, Drehzahlen des Werkzeuges, Positionen des Werkzeuges, mögliche Schnitttiefe, momentane Schnitttiefe, errechnete Geometrie des Werkstückes .... Aus den Berechnungen (Priorität 2) werden wiederum die modifizierten Daten

- XHmod (modifizierte x-Werte in Bildebene)
- YHmod (modifizierte y-Werte in Bildebene)
- ZHmod (modifizierte z-Werte; Höhe normal auf die Bildebene)
- αHmod (Winkel α modifiziert)
- βHmod (Winkel β modifiziert)
- γHmod (Winkel γ modifiziert)
- und VHmod (modifiziertes Volumen des geometrisch beschriebenen Wertstückes)

berechnet und zur Maschine gesendet. Diese Modifikation rührt daher, dass im Rahmen der **Bildbearbeitung** auch Daten geliefert werden, die nicht mit der Maschine ausführbar sind. Sollte z.B. die Schnitttiefe überschritten werden, so muss dieser Fehler bei den Berechnungen aufgedeckt werden, und der zurückgelegte Weg des **Koordinatengebers** korrigiert werden. Auch bei der Gefahr einer Kollision zwischen der Maschine und dem Werkzeug müssen die Fahrwege geändert werden. Die Ergebnisse der Berechnung werden auch zum graphischen Display geschickt, und zwar in Form von Diagrammen, Statistiken und anderen Daten. Von der Maschine ausgehende Daten sind:

- XM (x-Wert der Maschinenposition)
- YM (y-Wert der Maschinenposition)
- ZM (z-Wert der Maschinenposition)
- VM (Volumen des Werksstückes)
- M_size (Amessungen der Maschine, Kollisionsobjekte)
- M_pwr (Angabe der Leistung, die 1. gerade auf das Werkstück wirkt und 2. die maximale Leistung))
- und M_I (anliegender Strom).

Die letzten drei genannten Variablen werden in einer Datenbank gespeichert (M_db). Um Kollisionen zu verhindern, müssen alle Kollisionsobjekte der Maschine, des Werkstückes und des Werkzeuges (in weiterer Folge auch der Drehtisch) auf Überschneidung kontrolliert werden. Dies muss durch entsprechende Berechnungen abgeklärt werden, wobei für die Kollisionsobjekte möglichst einfache Geometrien zu Anwendung kommen sollten (Quader oder eventuell Polygone). Im Falle einer drohenden Überschneidung der geometrischen Formen müssen die Koordinaten (X, Y, Z sowie die Winkel α, β, γ) neu berechnet werden. Diese Daten werden dann direkt der Berechnung zugeführt und erneut korrigiert, um immer die korrekten Bewegungsmuster und damit die richtigen Werkstückgeometrien zu erreichen. Erst die endgültigen Koordinaten werden der Berechnung der Werkstückgeometrien zugeführt, da es sonst leicht zu Abweichungen zwischen den berechneten Geometrien und den vorhandenen kommen könnte, was wiederum zu großen Problemen bei der Bearbeitung mit Werkzeugen führen kann. Wenn die Abweichungen über einer gewissen Toleranz lägen, wären zu große Schnitttiefen, falsche Kollisionsobjekte, zu hohe Fahrgeschwindigkeiten und Ähnliches die Folge. Die Berechnung der Kollisionsvermeidung

bzw. die diesbezüglich notwendige Programmierung ist kein Teil der hier vorgestellten Arbeit. Dieser Forschungsbereich muss daher außerhalb des Rahmens der ggst. Diplomarbeit erarbeitet werden.

Der Block Werkzeug (Priorität 3) umfasst folgende Ausgangsvariablen:

- T_fkt (gibt die Formfunktion des Werkzeuges an, beschreibt also das Kollisionsobjekt)
- T_maxdeep (gibt die maximale Schnitttiefe in Abhängigkeit der Vorschubgeschwindigkeit an)
- T_mat (gibt das Werkzeugmaterial an, aus dem dann die Arbeitsgeschwindigkeit ermittelt wird: Hartmetall, Schnellarbeitsstahl...)

Der Block Werkstück (Priorität 4) enthält folgende Ausgangsvariablen:

- P_H (Höhe des zu bearbeitenden Materials)
- P_W (Breite des zu bearbeitenden Materials)
- P_L (Länge des zu bearbeitenden Materials)
- P_mat (Gibt das Material des Werkstückes an; wichtig für Bearbeitung, Schnitttiefe, Vorschubgeschwindigkeit und verwendetes Werkzeug).

Der nächste Block umfasst "Virtuelle Icons" (Priorität 5). Hierbei handelt es sich um Interaktionselemente, die außerhalb des Arbeitsbereiches auf einem Bildschirm dargestellt werden. Diese Interaktionselemente ("Icons") könnten oben, unten, rechts oder links des Arbeitsbereichs angezeigt werden. Ausgangsvariablen sind:

- X_min (Mindestkoordinate in x-Richtung)
- Y_min (Mindestkoordinate in y-Richtung)
- Z_min (Mindestkoordinate in z-Richtung)
- X_max (Maximalkoordinate in x-Richtung)
- Y_max (Maximalkoordinate in y-Richtung)
- Z_max (Maximalkoordinate in z-Richtung)
- EIN (Maschine in Betrieb; Koordinaten im Arbeitsbereich)
- AUS (Maschine ausgeschaltet, Koordinaten im "Icon"-Bereich oder außerhalb des Arbeitsbereichs)
- Arbeitsbereich umschalten: da der Arbeitsbereich eingeschränkt ist, aber breitere Werkstücke kein Problem darstellen, kann mittels Anklickens eines dafür vorgesehenen "Icons" ("anklicken" heißt in dem Fall z.B. mit der Hand in die Kamera winken, siehe "Eye-Toy" oder Ähnliches) in den danebenliegenden Werkstückbereich gewechselt werden, um diesen zu bearbeiten; Oberflächengüte verbessern und Bewegung glätten(da die Steuerung mit der Hand nicht immer ruhig und zitterfrei möglich ist, so können Funktionen (Polynome), wenn gewünscht, in die Fahrwegberechnung eingreifen, und eine bessere Oberfläche gewährleisten.

Eingangsvariablen werden von der Bildverarbeitung gesendet. Also x, y, z und alle drei Winkel. Dies dient der Berechnung, ob der **Koordinatengeber** schon im Bereich der "Icons" ist. Sollte das der Fall sein, so können die verschiedenen Schalter betätigt werden.

Der Drehtisch (Priorität 5) hat folgende Ausgangsvariablen:

- T_angle (momentaner Winkel des Drehtisches zur Maschine)
- T_size (gibt die Größe des Kollisionsobjekts "Drehtisch" an)
- T_maxweight (zulässiges Maximalgewicht des Drehtisches)

Auf diesem Drehtisch wird das Werkstück befestigt. Die Spannvorrichtung zur Befestigung des Werkstückes funktioniert nach dem Prinzip einer Spannzange. Dabei sollten die Spannbacken nicht zu viel Platz einnehmen, da sich dies negativ auf die Größe des einzuspannenden Werkstückes auswirkt. Jeweils zwei dieser Spannzangen werden im rechten Winkel zueinander befestigt. Zusätzlich könnte man auch "Spikes" auf die Spannbacken montieren, die bei weicheren Materialien (Holz, Kunststoffe ...) eine festere Befestigung erlauben, um die Sicherheit zu erhöhen. Diese Spannvorrichtung sollte zusätzlich auf den Drehtisch angeschraubt sein. Um den jeweiligen Drehtischwinkelwinkel im Verhältnis zur Maschine angeben zu können, muss ein Messfühler am Drehtisch befestigt sein. Bei einer maximalen Drehtischdrehbarkeit von jeweils nur 90° bietet sich alternativ eine Erfassung mit Schaltern an. In diesem Fall geben diese Kontakte den jeweiligen "Ist"-Zustand an.

Die Eingänge für das graphische Display (Priorität 5) betreffen sämtliche Daten, die während des Betriebes angezeigt werden sollen. Dazu gehören verschiedenste Diagramme (Maschinen-, Drehtischdaten, die Daten des verwendeten Werkzeuges, des Werkstückes, ...) und Statistiken, sowie die graphische Darstellung der momentanen Werkstückgeometrie. Die Schnitttiefe kann, wenn die Form des Ausgangsmaterials bekannt ist, einfach aus dem Schnittvolumen des Kollisionsobjekts "Werkzeug" und der aktuellen Werkstückgeometrie berechnet werden (das Kollisionsobjekt muss relativ genau vermessen sein, da es sonst zu starken Abweichungen zwischen wahrer Form und berechneter Form kommen kann). Nachteil hierbei ist, dass nicht von einer beliebigen Form ausgegangen werden kann, da es nicht möglich ist, diese zu erfassen. Auch wenn dies möglich wäre, so müsste die Werkstückgeometrie zu Beginn erst aufwändig in einer dreidimensionalen Form eingegeben werden. Eher denkbar wäre hier eine Speicherung des erstellten Modells, um in späterer Folge eine Weiterarbeit zu ermöglichen. Um die jeweilige Position im **Arbeitsbereich** jederzeit im Auge behalten zu können, könnte eine Verbesserung der diesbezüglichen Handhabung auch durch die zusätzliche Ausgabe der Kamerabilder erreicht werden.

Als letzten Punkt wurde zunächst noch die Verwendung einer Hebebühne zu überlegt. Angedacht wurde die Verwendung einer verschiebbaren Stehfläche in der vertikalen Ebene, um die Tätigkeit im Arbeitsbereich zu erleichtern. Diese Vorrichtung hätte allerdings keinen direkten Einfluss auf die Berechnung gehabt, da eine rein händische Bedienbarkeit der Hebebühne per Joystick ausreichen hätte können. Letztendlich hat sich die Überlegung dieses Aufbaus als nicht besonders hilfreich herausgestellt, da auch ohne diese Hebebühne der gesamte **Arbeitsbereich** gut zu erreichen ist.

3.3.1 Grundaufbau Bildverarbeitung

Nach der Kamerainitialisierung ist es mittels eines kleinen Tools der Firma Mikrotron möglich, die Kameraeinstellungen zu variieren. Im Normalfall werden keinerlei Einstellungen vorgenommen, außer die Umweltbedingungen haben sich geändert (Lichteinfall, zusätzliches künstliches Licht, Änderung der Position der Kameras ...). Die wichtigsten Einstellungen sind dabei: Helligkeit des Bildes, Kontrast, Bildausschnitt, Bildwiederholungsrate, Bildschärfe .... Ansonsten können die gespeicherten alten Einstellungen (Kameraprofile) schnell wieder aufgerufen werden. Die Daten werden in eine **.dcf-Datei** mittels der "Mikrotron" Software in den Speicher geschrieben. Die dabei verwendete Kamera ist nur die Trackingkamera. Die beiden Stereokameras und deren Einstellungen finden sich in der Diplomarbeit von DI FH Gelhart [6] zu finden. Wie das Tool aussieht, und welche Parameter veränderbar sind, können sie in [2] Seite 40 nachlesen. Nachdem auch die **ROI** und damit die Bildgröße (Tr_fr_size) festgelegt ist, kann mit der eigentlichen **Bildverarbeitungsroutine** begonnen werden. In diesem Programm wird ein heller Ball (weiß) vom **Trackingbild** verfolgt, während der Hintergrund möglichst dunkel ist. Wie diese Routine genau aussieht, kann im Kapitel Bildverarbeitung nachvollzogen werden. Das **"Trackingdevice"** wurde auch schon zu Beginn initialisiert und ein Profil geladen ("Trackingdevice" wird die Maschine bezeichnet, die die beiden Stereokameras dahingehend steuert, dass diese den Ball jederzeit, unabhängig von seiner jeweiligen Position, im Bild halten; im ggst. Fall handelt es sich um ein Gerät der Firma Schunk: PW 070 POWERCUBE WRIST). Das Ergebnis der "**Blob**"-Suche sind Koordinaten in X- und Y. Die Daten der Höhe (Z- Komponente) des **Koordinatengebers** werden, wie schon erwähnt, durch ein Programm von DI FH Gelhart berechnet. Nachdem auch die Initialisierung des **Tracking Devices** abgeschlossen ist (wird genauso wie die **ROI**- Einstellungen zu Beginn des Programmes ausgeführt), kann dieses mittels der x- und y-Koordinaten der **Bildverarbeitung** angesteuert werden. In Berechnung neu werden die aktuellen Koordinaten (XH, YH, ZH) und Winkel (αH, βH, γH) mit den für die Trackingberechnung sowie mit den ermittelten Daten für die Stereoberechnung ausgegeben. Die Daten werden auch direkt für die Grenzbereichszuordnung und Schutzzonenberechnung verwendet. Nach der Berechnung der aktuellen Koordinaten werden diese auch zur Berechnung der Werkstückgeometrie herangezogen. Die Daten des vorangegangenen Rechenschrittes werden gespeichert, damit diese zusammen mit den aktuellen Koordinaten für die Vektorenberechnung verwendet werden können. Im Anschluss an die Berechnung der Vektoren wird das Volumen ausgegeben. Zur Berechnung der Schnitttiefe kann das aktuelle Werkstückvolumen verwendet werden. Für die Ermittlung einer Positionsdifferenz werden die alten Koordinaten der **Bildverarbeitung** und das Volumen verwendet. Zur Berechnung der graphischen Ausgabe wird das berechnete Werkstückvolumen herangezogen.

3.3.2 Grundaufbau Berechnung

Voraussetzung für die Festlegung einer Werkstückgeometrie ist die Ermittlung der Werkstückausgangsmaße (P_H, P_W, P_L). Zusätzlich muss die Formfunktion (beschreibt die Form des Werkzeuges) T_fkt aus einer Datenbank abrufbar sein. Als Ergebnis der Berechnung der Werkstückgeometrie sind einerseits die regulierte Schnitttiefe (P_deep), andererseits die sich daraus ergebende Geometrie (P_geo_n) zu nennen.

Zur Berechnung der maximalen Vorschubgeschwindigkeit und Schnitttiefe werden die Materialdaten (P_mat) des Werkstückes benötigt. Weiters sind zur Berechnung auch die Daten aus der Werkstückdatenbank notwendig. Diese teilen sich auf in T_maxdeep, die maximale Schnitttiefe

in Abhängigkeit vom verwendeten Werkstückmaterial, in (P_deep), Schnitttiefe des Werkzeugmaterials, sowie in die Maschinendaten auf. Unter die Maschinendaten fallen der momentan anliegende Strom (M_I) und die Maschinenleistungsdatenbank (M_pwr_db). In der Maschinenleistungsdatenbank sind die Maximalwerte der Leistungsaufnahme und des Maximalstromes für jeden Maschinenantrieb gespeichert, die bei Annäherung dieser Werte sofort zur Grenzbereichszuordnung geschickt werden, um z.B. ein gelbes Licht, und bei Überschreitung ein rotes Licht (inklusive einer Maschinenabschaltung) auszugeben. Zusätzlich zu den Warnleuchten könnte auch eine automatische Korrektur der Schnitttiefe oder der Vorschubgeschwindigkeit das Problem lösen. Als Eingangsdaten werden überdies das momentane Volumen(VH) und die momentane Fläche (AH) verwendet. Aus dem Block Vorschubgeschwindigkeit/ Schnitttiefe werden alle modifizierten Koordinaten und Winkel, sowie die Variablen, E_PT_mat (Materialkennwerte), E_T_maxdeep (maximale Schnitttiefe aufgrund der verfügbaren Leistung) und E_M_Pwr (vorhandene Leistungsaufnahme) an die Grenzbereichszuordnung gesendet.

Zur Berechnung der Schutzzonen und der Kollisionsrechnung werden folgende Daten benötigt: VH, AH, P_geo_n, die Koordinaten und Winkel direkt aus der Bilderkennung, coll_object_turn, M_size_db (Abmessungen der einzelnen Maschinenteile), T_angle (Winkelstellung der Werkstückbefestigung des Drehtisches) und T_size_db (Abmessungen des Drehtisches). Coll_object_turn ist dabei ein Kollisionsobjekt, das den "Greifarm" davor schützen soll, bei Umschwenkbewegungen am Werkstück anzustreifen oder bei unerreichbaren Positionen, die erst durch eine Änderung der Maschinenposition erreichbar sind, entsprechend korrektiv einzugreifen. Wichtig ist es aber auch, Positionen zu erkennen, die aufgrund der Struktur des Werkstückes nicht erreichbar und daher nicht bearbeitbar sind. Ausgehend vom Block Schutzzonen und Kollisionsberechnung sind eventuelle Kollisionswarnungen (E_Coll) und die durch verhinderte Kollisionen geänderten Koordinaten und Winkel erforderlich.
Um eine Gewichtsüberlastung des Drehtisches zu verhindern, gibt es noch einen Block "Gewichtsberechnung". Dafür werden die Abmessungen des Ausgangsmaterials benötigt (P_H, P_L, P_W). Auch die Art des Materials (P_mat) sowie die maximale Tragkraft des Drehtisches (TI_maxweight) muss gegeben sein. Der berechnete Wert (E_P_maxweight) findet in den Grenzbereichszuordnungen Eingang.
Der Block "Positionsdifferenz" berechnet aus den im Kamerabild gefundenen Koordinaten und Winkel, aus den Koordinaten und Winkeln des vorangegangenen Bildes und aus den Maschinenkoordinaten und Winkeln eine Koordinatendifferenz für die Weiterverarbeitung (errechnen von Vektoren ...). E_M_H_deviation_o ist die Ausgangsvariable, die in die Grenzbereichszuordnung einfließt.

Im Block Umschwenkbewegung wird, wie schon weiter oben erwähnt, darauf geachtet, dass es zu keinen Kollisionen der Maschine mit dem Werkstück kommt, sei es, weil die Position unmöglich zu erreichen ist, oder weil die Maschine erst umschwenken muss, um eine gewünschte Positionierung (kollisionsfrei) zu finden. Eingehend in diesen Block sind nur die Koordinaten und Winkel der Bildverarbeitung, wobei die endgültige Berechnung erst in der Grenzbereichszuordnung durchgeführt werden kann, da die Koordinaten sich durch Modifikation noch erheblich ändern können. Ausgang des Blocks ist E_M_turn, die in Grenzbereichszuordnung Eingang finden Der Block "Graphische Berechnungen" erstellt aus den verschiedenen gesammelten Daten Diagramme, Statistiken etc. Eingang finden hier unter Anderem die Grenzbereichszuordnung, die Warnleuchten (grün, gelb, rot) und Daten der virtuellen "Icons".

Der größte Block, die Grenzbereichszuordnung, weist im Ausgang die endgültigen Koordinaten und Winkel, die sich nach Berücksichtigung der einzelnen Modifikationen (durch Leistung, Werkstückgeometrie, Maschinengeometrie und Werkzeugkennwerten) ergeben, auf. Weitere Ausgänge sind

die drei Warnleuchtvariablen, die den momentanen Zustand der Werkstückbearbeitung und das Endvolumen (VM) anzeigen.

3.4 Bewegungsraumermittlung

Um die menschlichen Handbewegungsmuster besser zu erfassen, wurden entsprechende Messungen durchgeführt. Im Rahmen dieser Messungen wurden die Bewegungstoleranzen der Armbewegungen ermittelt (z. B. wie sind die Bewegungsmuster einer ausgestreckten Hand bei horizontalen Bewegungen in möglichst gleichem Abstand zum Untergrund?). Bei folgenden Bewegungsmustern wurden Testvermessungen vorgenommen:

1. sitzend, ohne seitliche Bewegung des Körpers, statisch

2. sitzend, mit seitlicher Bewegung des Körpers, dynamisch

3. stehend, ohne Schritte, statisch

4. stehend, mit Schritten, dynamisch.

Es wurde mit einer gewöhnlichen Webcam gearbeitet. Für die Erkennung der Bewegungspunkte der Hand wurde ein orangefarbener Tischtennisball verwendet, der auf einem gewöhnlichen Handschuh befestigt war. Mit einem "Matlab"-Programm zur Ballkoordinatenvermessung wurden die jeweiligen Bewegungspunkte erfasst (Erfassung des orangenen Balls im Bild mithilfe einer Farberkennungssoftware).Die maximal verwendete Bildzahl pro Sekunde war 30. Die Ermittlung der Bewegungstoleranzen und des Handbewegungsmusters erfolgte mit einer "Framerate" von 15 Bildern pro Sekunde. Die in den Diagrammen eingezeichneten Knotenpunkte bilden daher die jeweilige Positionsbestimmung ab, die von den Kameras erfasst werden. Die Schwankungsbreite gibt daher, ausgehend von einer horizontalen Geraden, die maximale Abweichung an. Die dargestellten Diagramme zeigen nur einen Ausschnitt der erstellten Diagramme, Darstellung sämtlicher Diagramme aufgrund ihres Umfangs nicht darstellbar wäre. Aus diesem Grund wurde im anschließenden Punkt nur ein Handbewegungsmuster beispielshaft beschrieben.

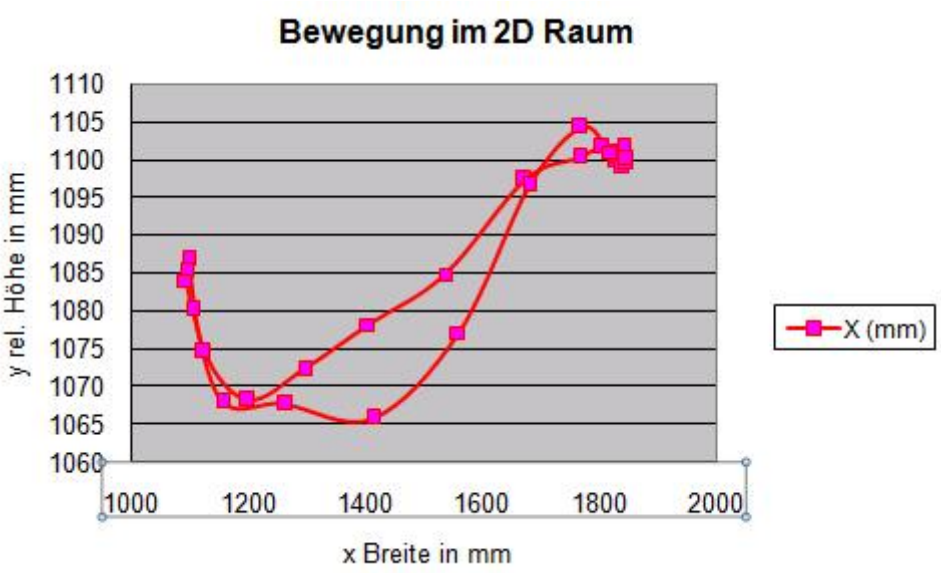


Figure 3.2: Bewegungstoleranzmessung des Armes, still stehend, x über y; max. Schwankungsbreite 38,5mm

Zu Punkt 3 siehe 3.2

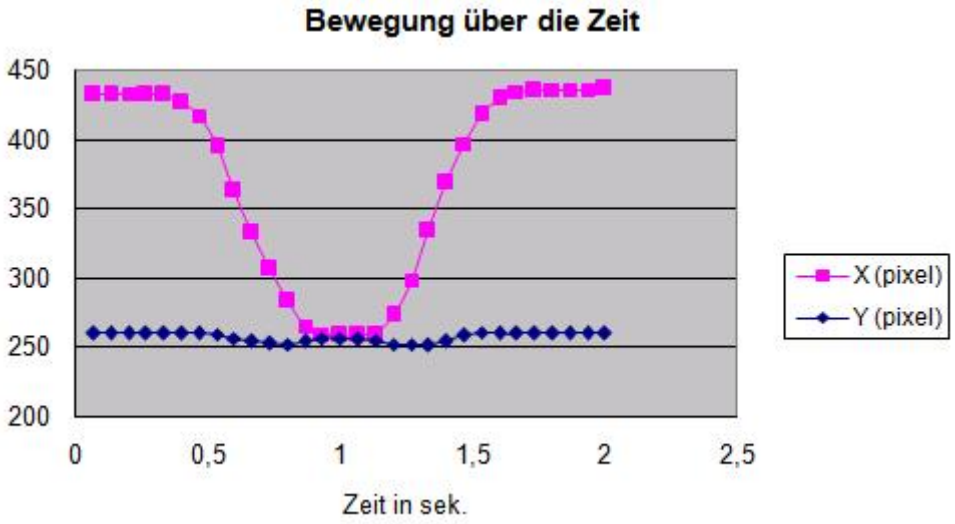


Figure 3.3: Bewegungstoleranzmessung des Armes, still stehend, x und y über der Zeit

Auch die maximalen Handgeschwindigkeiten (z. B., wie schnell erfolgt das Handbewegungsmuster "ausstrecken des Arms"?) wurden untersucht. Die maximale Handgeschwindigkeit wurde dem Handgeschwindigkeitsdiagramm entnommen. Dabei wurde die maximale Steigung zwischen zwei Knotenpunkten berechnet. Die größte auftretende Geschwindigkeit lag bei 15 m/s. Zur weiteren Berechnung wurden 10 m/s angenommen, da dies einem Standardhandbewegungsmuster entspricht. Zuletzt wurde noch kontrolliert, wie exakt eine menschliche Hand auf einer bestimmten Position gehalten werden kann (Messungsanordnung an einer sitzenden Person).

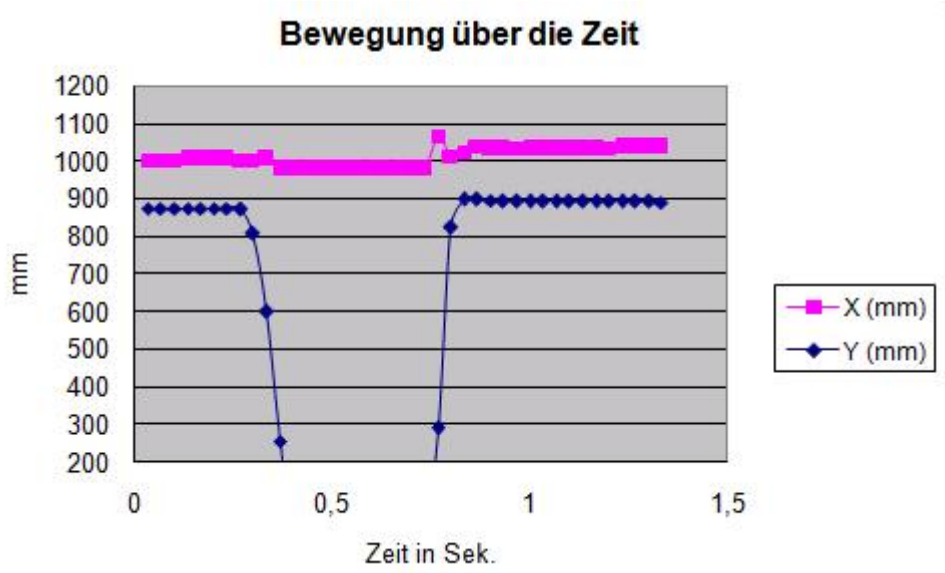


Figure 3.4: Handgeschwindigkeiten beim Handheben, Hand rauf, x und y über die Zeit (nichtaufgenommene Handkoordinaten der Kennlinie befinden sich außerhalb des Kamerabereiches).

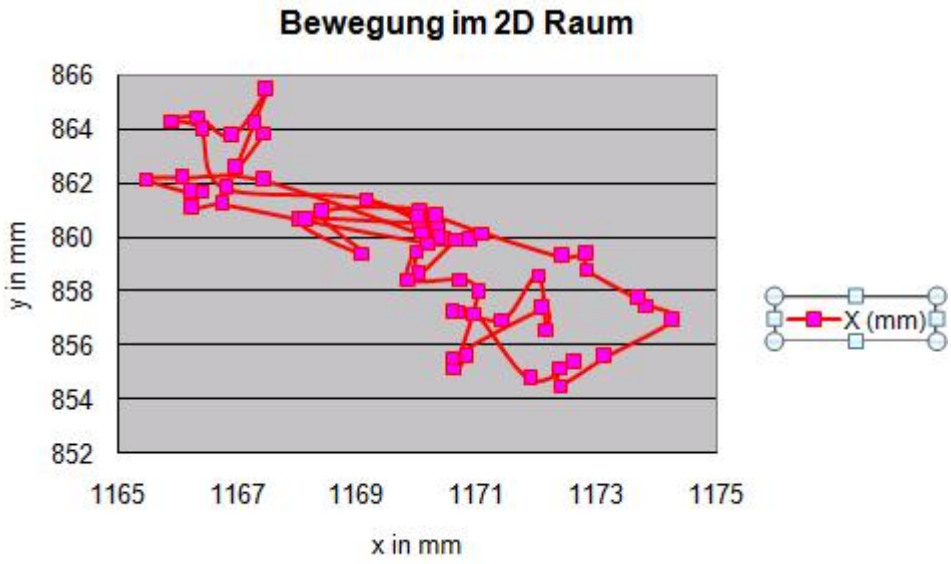


Figure 3.5: Handhaltung der Position im Sitzen

Um die beste Position zur Eingabe des **Koordinatengebers** herauszufinden, wurden verschiedene Positionen getestet.

3.5 Kameraauswahl

Da eines der Ziele des ggst. Diplomarbeitsprojekts in der optischen Kamerasteuerung lag, waren Kameras notwendig, die mit Hilfe einer Bildbearbeitungssoftware in der Lage waren, einem Bildpunkt zu folgen. In die engere Wahl kamen drei Kameratypen: das Produkt der Firma Basler, Modell 501k erreicht bei einer Auflösung von 1280 x 1024 Pixel eine Bildwiederholungsrate von 74 Bildern pro Sekunde. Diese Spezifikationen entsprachen nicht den Anforderungen des ggst. Forschungsprojekts. Dies war einer der Gründe, weshalb diese Kamera nicht in die Auswahl gelangte.

Figure 3.6: Basler 501k

Eine andere Kamera, von Dalsa, ist die 1M150-SA, die eine Bildwiederholungsrate von 150 Bildern pro Sekunde schafft, was an der etwas geringeren Auflösung (1024x1024) liegt. Aufgrund der geringeren Auflösung und der etwas kleineren Pixelgröße (wichtig für die Lichtempfindlichkeit) wurde auch diese Kamera nicht angeschafft. Der ausgewählte Kameratyp (CMOS) ist ein Produkt

Figure 3.7: Dalsa 1M150-SA

der Firma Mikrotron. Er erreicht bei voller Auflösung (1280x1024) eine Bildwiederholungsrate von ca. 100 Bildern pro Sekunde. Da die Datenrate über einen Base Camera Link ©aber "nur" 132 Mbyte/Sek beträgt, können maximal ca. 95 Bilder pro Sekunde abgerufen werden. Das bedeutet eine suboptimale Datenübertragungsrate des Camera Link® Protokolls selber,

was die Gesamtsystemgeschwindigkeit ebenfalls entsprechend reduziert. Dies hat jedoch keine Auswirkungen auf die Berechnungsgeschwindigkeit, da diese bei ca. 60 Bildern pro Sekunde liegt. Die Bilder der Kameras sind monochromatisch. Farb-Kameras hätten nur einen weiteren Anstieg der Datenrate und weitere Bildbearbeitungsvorgänge notwendig gemacht. Auf den jeweiligen Kameras sind Laser montiert (siehe Abbildung weiter unten), die zur einfacheren Einstellung und Justierung der Kamerasysteme eingesetzt werden. Die Laser sind dabei so ausgerichtet, dass sie möglichst parallel zur Kameraachse angeordnet sind. Die Einstellung der Laserrichtung erfolgt dabei mit Hilfe eines eigenen Algorithmusses, der auch Abweichungen der Lasereinstellung feststellt. Die eigentliche Justierung und Einstellung erfolgt händisch direkt am Laser.

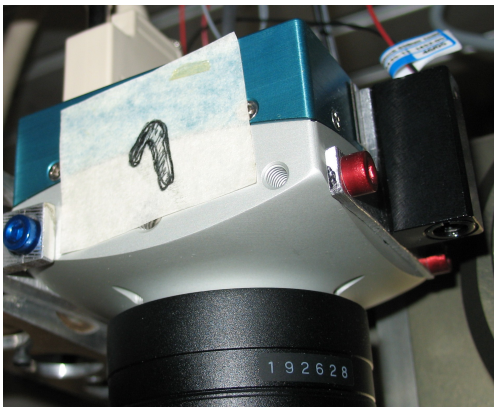


Figure 3.8: Kamera der Firma Mikrotron inklusive Laser

Um sicher zu stellen, dass der **Koordinatengeber** immer im Bildbereich der Stereokameras bleibt, ist eine hohe Bildwiederholungsrate nötig. Im Rahmen einer eigenen Berechnung (siehe den vorigen Abschnitt "Bewegungsraumermittlung") wurde eine durchschnittliche maximale Handgeschwindigkeit von 10 m/Sek. angenommen (dies kann bei schnelleren Bewegungen durchaus vorkommen, z. B. beim schnellen nachbearbeiten bei gleichzeitig geringer Schnitttiefe). Der Bildausschnitt in der **Arbeitsmittenebene** der Stereokameras wurde mit einer Größe von 250 mal 300 mm berechnet. Um eine maximale Wegstrecke der Hand von 100 mm pro Bild berücksichtigen zu können, wäre eine "Framerate" von mindestens 100 erforderlich. Diese Mindestanforderung war auch notwendig, um möglichst viele Koordinatenpunkte und damit eine möglichst optimierte Erfassung der Bewegungsmuster, sowie ein möglichst homogenes und stetiges Bewegungsmuster sicher zu stellen. Damit ist auch ein einmaliges Aussetzen des Bildfindungsalgorithmus besser beherrschbar. Ein großer Vorteil der Kameras der Firma Mikrotron ist, dass diese bei einem kleineren Bildausschnitt auch eine höhere Bildwiederholungsrate als 100 erreichen kann.
Zur Erkennung des **Koordinatengebers** wurde aus Gründen der Beschleunigung der Umsetzung von der Bewegungserfassung der Handbewegung bis zur Werkmaschinensteuerung, wie bereits oben unter Pkt. 3.4 näher ausgeführt, ohne Farbkriterien gearbeitet. Aus diesem Grund ist die Wahl der Kamera auf eine Grauwert-Kamera gefallen. Außerdem wurde keine sogenannte "Smartcam" ausgewählt, die schon ein Gehäuse inklusive Berechnungseinheit und Speicher beinhaltet, da die Prozessorleistung von Kameramodellen mit eingebautem Prozessor bei nicht ausreichender Rechenleistung nicht nachrüstbar ist, was sich für die ggst. Forschungsarbeit aufgrund der fehlenden Flexibilität als nicht sinnvoll erwies.
Probleme hinsichtlich der ausreichenden Beleuchtung sind im Fall einer CCD-Kamera weniger gegeben, da dieser Chip lichtempfindlicher gegenüber dem gewählten Kameramodell ist. Die Wahl fiel schlussendlich jedoch auf ein Kameramodell mit geringerer Lichtempfindlichkeit (Näheres siehe

	KAMERA:									SCHNITTSTELLEN:				
										analog:		digital:		
(Fa.)Typenbez.	Auflsg H	Auflsg. B	Framerate	bit	Farbe	s/w	Basis (mm):	Pixelgr. (mm)	Shutter (us)	LVDS	CameraLink	Firewire	RS232	USB1.1
Bi-i V301	1024	1280	27,5	8	j	j	100	0,0067				n	j	j
Bi-i v2 Stereo	1024	1280	27,5	8	j	j	80	0,0067				n	j	j
VC4038	480	640	63	8	j	j	100	0,021	5			n	j	n
VC2048	480	640	110	10	j	j	100	0,021	18			n	j	n
VC4065	582	782	55	10	j	j	100	0,0162	18			n	j	n
DALSA DS21001M0150	1024	1024	150	8	n	j	x	0,0106			Base	n	n	n
Mikrotron MC1302	1024	1280	100	8	j	j	x	0,012	4		Base	n	n	n
Mikrotron MC1303	1024	1280	100	8	j	j	x	0,012	4		Base	n	n	n
Mikrotron MC1310	1024	1280	500	8	j	j	x	0,012	4		FullExt	n	n	n
Mikrotron MC1311	1024	1280	500	8	j	j	x	0,012	4		FullExt	n	n	n
Photonfocus MV-D640	480	640	200	8	j	j	x	0,0099	40		Base	n	n	n
COHU 7800	1014	1280	30	8	j	j	x	0,007	25		Base	n	n	n
Basler A504k/kc	1024	1280	500	8	j	n	x	0,012			Full	n	n	n
Basler A504k/kc	1024	1280	500	8	n	j	x	0,012			Full	n	n	n
PUL TM-6710 CL	494	659	120	8	n	j	x	0,009	83,3		Base	n	n	n
Pulnix TM-6710	480	640	120	8	n	j	x	0,009			n	n	RS644	n
Roper Scientific Megaplus ES:310	480	640	125	8			x	0,009						

Figure 3.9: Kameraauswahl Teil 1

	RECHNER:						CHIP:				WEITERES:	
	CPU (Mhz)				SPEICHER:							
Ethernet TCP/IP	DSP fixed	DSP float	Bandbreite (Mpixel):	Leistung	RAM (Mbyte)	Flash (Mbyte	Chiptyp	Breite (mm)	Höhe (mm)	Trigger	Preis	Dynamik (dB):
j	1000	250	40		64	4	CMOS	8,8	6,6			
j	600	150	40				CMOS	8,8	6,6			
j	400				32	4						
j	150				16	2						
j	150				16	2						
n			2*80				CMOS				5000	48 lin 120 linlog
n			85				CMOS	12,29	15,36		5600	59
n			85				CMOS	12,29	15,36		6000	59
n			85				CMOS	12,29	15,36		7400	59
n			85				CMOS	12,29	15,36		8000	59
n			66				CMOS	6,34	4,75		1061	60
n			40				CMOS					60
n			67				CMOS				9980	
n			67				CMOS				9480	
n							CCD				2180	45
n			50				CCD					
							CCD					

Figure 3.10: Kameraauswahl Teil 2

weiter oben im Text unter Pkt. 3.5), das einen "CMOS"-Chip besitzt. Diese Auswahlentscheidung wurde aufgrund der Variabilität des Bildausschnitts des Kamerabildes während des Betriebes getroffen, da von diesem Feature oftmals Gebrauch gemacht wird.

3.6 Kameravorberechnungen

Abgeänderte Linsengleichung[wikipedia]: Berechnung zur Brennweitenbestimmung der jeweiligen Kamera:

$$f = \frac{g * B}{G + B} \tag{3.1}$$

Berechnung des größten Winkels des **Tracking Device**s in Neige- und Schwenkrichtung:

$$\alpha = \arctan(\frac{b_1 * 180}{g_1 * \pi})$$

f . . . Brennweite in mm
α. . . maximaler Öffnungswinkel in Grad g_1. . . **Abbildungsoberebenenabstand** in mm

Figure 3.11: Stereokamera mit Objektiv

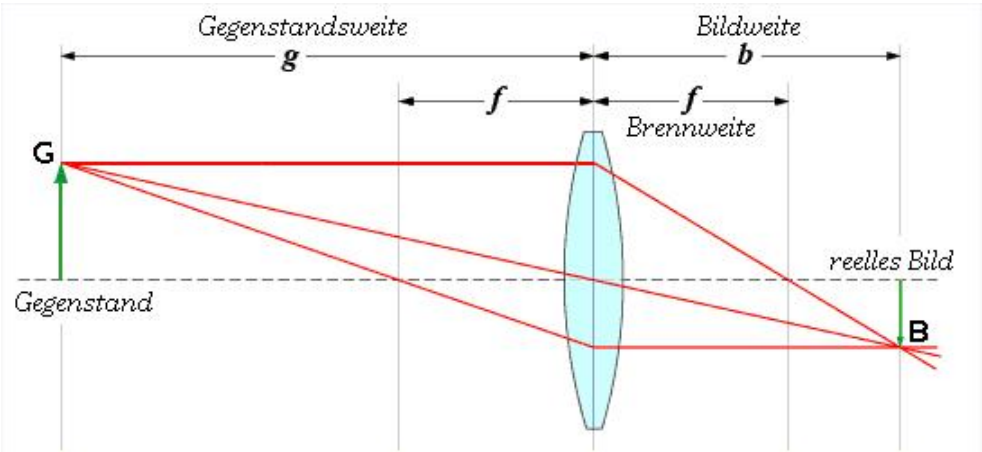


Figure 3.12: Linsengleichung

g_2... **Abbildungsunterebenenabstand** in mm

Berechnung der schnellsten Handbewegung
b_1 ... Halbe Bildbreite in mm

$$H_b = \frac{H_w}{2 * b_1} * 2 * \alpha$$

H_b... maximale Handbewegung in Grad
H_w... Handreichweite in mm

$$W_g = \frac{H_b}{t}$$

t... Handausstreckzeit in Sekunden
W_g... notwendige Winkelgeschwindigkeit in Grad/Sekunde

Berechnung des Auflösungsverlustes durch den Öffnungswinkel der Kamera:

$$oe_1 = \tan(\alpha) * (g_2 - g_1)$$

oe_1 ... Öffnungswinkelverlust in mm
B_{Bild}... Breite des Bildes in mm (y- Achse, Gegenstandsbreite)

Tiefenauflösung bei parallelen Kameras:

$$A_T = \frac{p * g_1}{B_K * f}$$

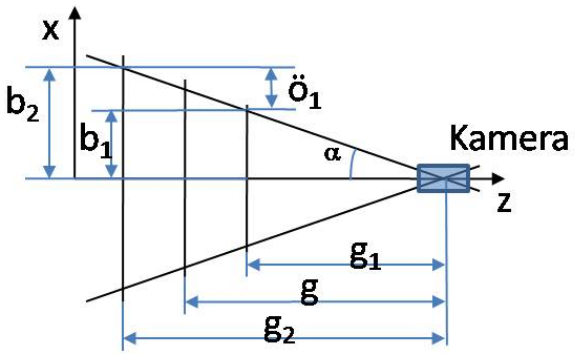
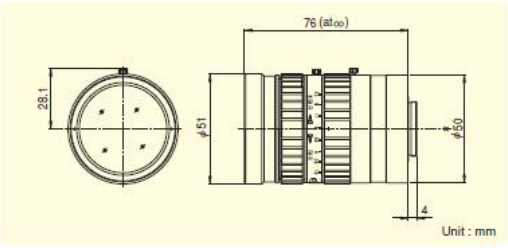


Figure 3.13: Kameranebenrechnungen

p. . . Pixelgröße in mm
B_K. . . Basisabstand der beiden Stereokamerasachsen in mm

Die Brennweite der Stereokameras wurde mit 75 mm ermittelt. Wichtig für eine schnelle Bewegung mit dem **Tracking Device** war ein möglichst geringes Gewicht der Objektive. Auch eine Möglichkeit zur Fixierung der Blenden- und Linseneinstellungen genauso wie ein möglichst großer Linsendurchmesser (besserer Lichteinfall, keine Randbildungen durch das Objektiv bei voller Auflösung) waren wichtige Auswahlkriterien. Die beiden angekauften Objektive mit der Bezeichnung "CF75HA-1" der Firma Fujinon hatten auch einen relativ geringen Lichtabschwächungsfaktor. Die Blendeneinstellung liegt für beide Stereokameras bei 5,6. Je mehr die Blende geschlossen

- High-resolution design, providing support for up to 1.5 megapixel camera resolution.
- Wide-aperture (F1.8) design achieves clear images under low light intensity, despite the long focal distance.
- Low-distortion design achieving accurate image input.
- Robust enclosure resistant to vibrations and shocks. Equipped with locking knobs for the iris and the focus.



Focal Length (mm)		75
Iris Range		F1.8 ~ F22
Operation	Focus	Manual
	Iris	Manual
Angle Of View (H×V)	1"	9°45′ × 7°19′
	2/3"	6°43′ × 5°02′
	1/2"	4°53′ × 3°40′
Focusing Range (From Front Of The Lens) (m)		∞ ~ 0.9 *2
Object Dimensions at M.O.D. (H×V) (mm)	1"	147 × 111
	2/3"	101 × 76
	1/2"	74 × 55
Back Focal Distance (in air) (mm)		24.43
Exit Pupil Position (From Image Plane) (mm)		−52
Filter Thread (mm)		M49 × 0.75
Mount		C
Mass (g)		295

Remarks
- With Metal Mount
- With Locking Knob for Iris and Focus

*2 Using an extension tube longer than 5mm will increase the M.O.D. to 0.5m.

Figure 3.14: Kameraobjektive der Stereokameras

werden kann, desto besser ist die Tiefenschärfe. Dabei ist aber auch zu bedenken, dass der Lichteinfall bei einer hohen "Framerate" geringer ist, was eine stärkere Öffnung der Blende erfordert. Durch die Verwendung einer starken Beleuchtung wurde hier Abhilfe geschaffen.

9 mm wurde zunächst als erforderliche Brennweite für die Trackingbildkamera ermittelt. Im Rahmen einer Feinabstimmung stellte sich eine Brennweite von 8 mm als optimale Variante heraus. Hierfür wurde ein Produkt der Firma Pentax verwendet. Da die Trackingkamera nur einen kleinen Bildausschnitt in der Mitte abbildet, wurde ein geringerer Linsendurchmesser gewählt. Die Blendeneinstellung des Trackingkameraobjektives liegt bei 4.

3.7 Bildentzerrung

3.7.1 Kamerabildentzerrung

Die Kamerabildentzerrung erfolgt mit Hilfe der Programmiersoftwareversion "Matrox Mil 8.0". Dabei wird eine Punktmatrix zunächst unter das aufgenommene Kamerabild unterlegt und die Punktabstände mittels eines "Blob" - Suchalgorithmus eingelesen, um die vorhandene kissenförmige Verzerrung zu entzerren. Die zur Erstellung einer Punktmatrix notwendigen Daten und Abmessungen können dem Handbuch [4], Seite 144 entnommen werden.

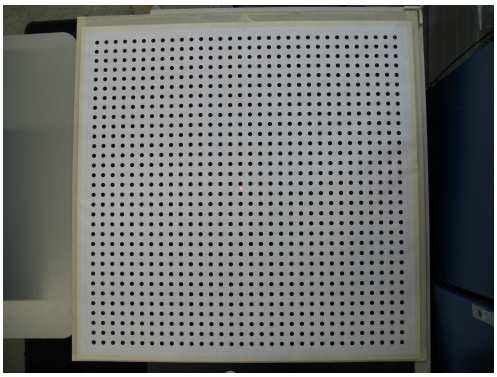


Figure 3.15: Verzerrtes Kamerabild

Die vorher verzerrte Bildansicht (siehe 3.15) kann durch die oben beschriebene Entzerrung entzerrt wiedergegeben werden. Alle Punkte der Matrix sollten nach diesem Vorgang geradlinig aufgereiht, und in einem rechten Winkel zueinander stehen. Diese Umrechnung kann in einer Datei gespeichert werden und ist dann für diese Kamera verwendbar. Bei Verwendung eines neuen Objektivs oder im Fall ähnlicher Änderungen ist eine erneute Kalibration unumgänglich, da die ursprüngliche Entzerrungsberechnung (**Kalibrationsdatei**) im Fall derartiger Veränderungen kein entzerrtes Bild mehr garantieren kann.

3.7.2 Tiefenbildentzerrung

Eine weitere Verzerrung tritt auf, wenn ein Gegenstand weiter weg oder näher hin zu einer Kamera bewegt wird. Auch wenn der Gegenstand genau in der Mitte bewegt wird, ändert sich die Größe der Darstellung im Kamerabild. Wird das zu suchende Objekt außerhalb der Mitte bewegt, zeigt sich ebenfalls eine nach aussen hin zunehmende Nichtlinearität (siehe 3.16). In diesem Diagramm ist die x Achse in mm und die y Achse in Pixel. Um diese Nichtlinearität aus dem Kamerabild

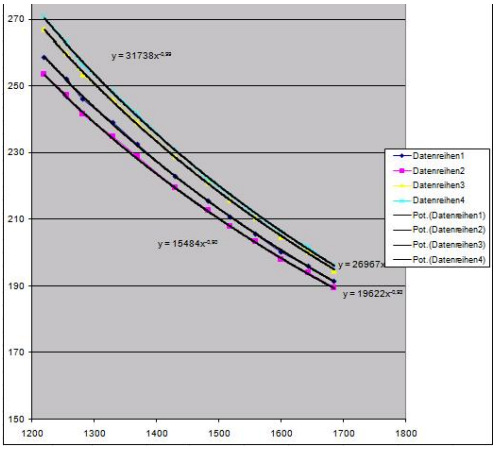


Figure 3.16: Tiefenverzerrung

herausrechnen zu können, werden mit Hilfe einer Entzerrungsmatrix Bildpunkte aufgenommen. Diese in verschiedenen Tiefenabständen zur Kamera ermittelten Punktewolken lassen sich mit Hilfe eines Blobsuchalgorithmus berechnen. Im vorliegenden Fall geschieht dies mit einer "5-Punkte-Matrix", wobei ein Punkt in der Mitte und vier weitere Punkte in der jeweiligen Ecke des quadratischen Feldes angeordnet werden 3.17).

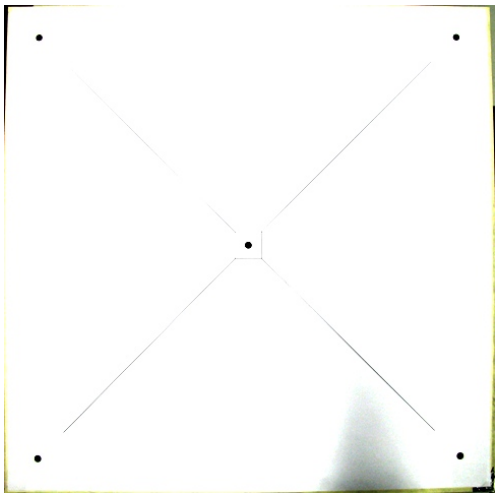


Figure 3.17: Entzerrungsgitter

Der Punkt in der Mitte dient dazu, eventuelle Ungenauigkeiten zu korrigieren, die aus einer Verschiebung der Messfelder in den verschiedenen Höhenlagen resultieren. Damit ist sicher gestellt, dass die Mittellinie immer in der Mitte liegt. Für die Annäherung einer Funktion (aus der sich leicht die unterschiedlichen Verschiebungen der jeweiligen Tiefe ergeben) durch einen "Least Squares"-Algorithmus werden die Punkte herangezogen, die sich immer unverändert in der gleichen Ecke befinden. Bei vier Eckpunkten ergeben sich somit vier Funktionen, die sich optimal an die sich ergebenden Koordinaten annähern. In diesem Zusammenhang ist auch zu bedenken, dass ein Gegenstand mit zunehmender Nähe zum Kameraobjektiv im Kamerabild auch entsprechend vergrößert dargestellt wird. Im Umkehrschluss dazu erscheint dieser Gegenstand mit zunehmender Entfernung vom Objektiv im Kamerabild wiederum auch entsprechend verklein-

ert. Der daraus resultierende Schluss bedeutet, dass die Erfassungsgenauigkeit mit zunehmender Einbeziehung von Abstandspunkten in die Berechnung gleichfalls zunimmt.

3.8 Aufbau (des gesamten Systems, Rechner etc.)

In diesem Abschnitt der ggst. Forschungsarbeit werden die für den Aufbau des Systems notwendigen Vorrichtungen und Geräte näher beschrieben.

3.8.1 Übersicht

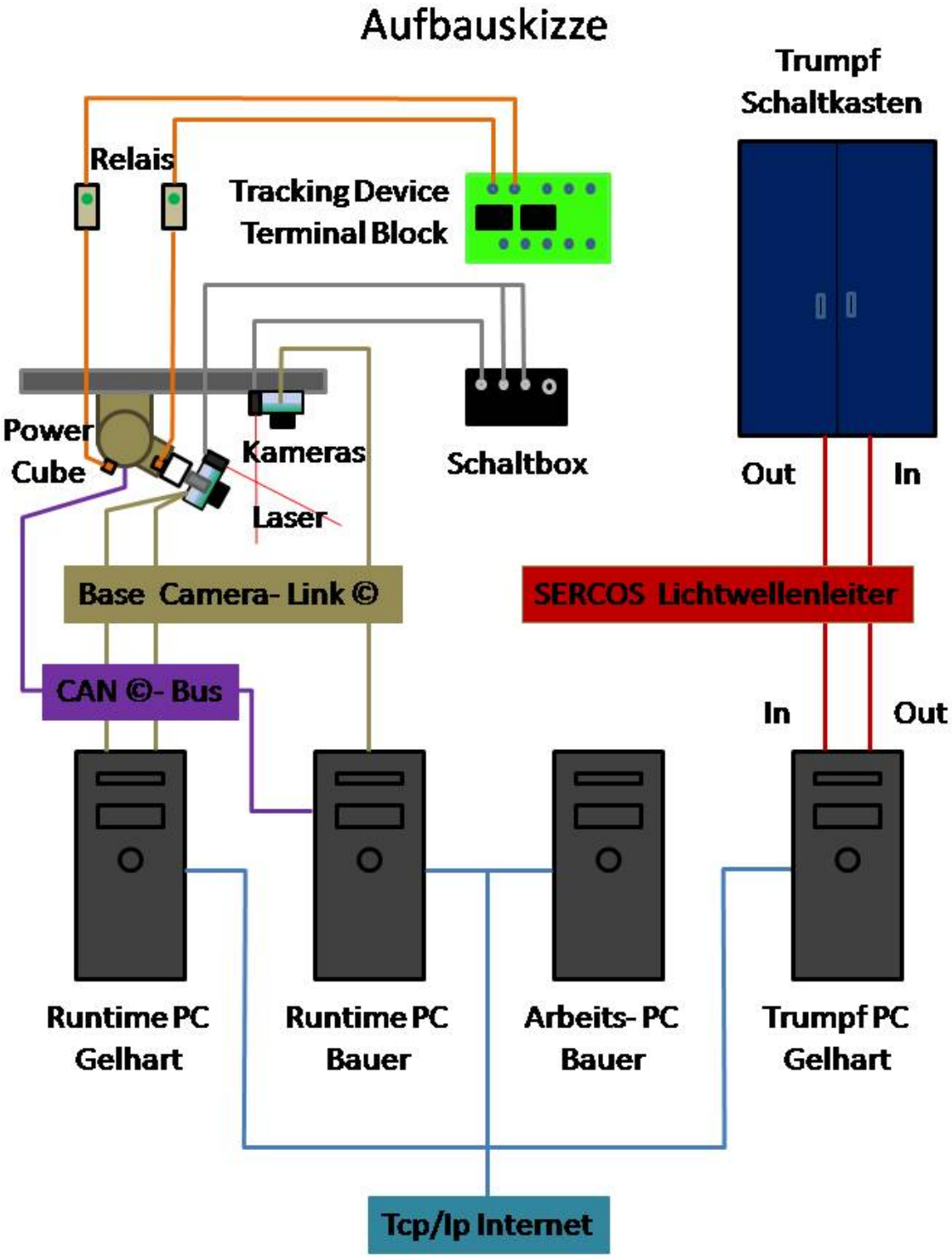


Figure 3.18: Skizze des gesamten Aufbaus

Im Bild 3.18 sind alle Signalverbindungen des gesamten Aufbaus schematisch dargestellt. Die Spannungsversorgungen wurden hier nicht berücksichtigt. Die orangen Kabel stellen die Verbindungen zwischen dem "**Tracking Device** Terminal Block" und den induktiv arbeitenden Endschaltern dar (ebenfalls in oranger Farbe).

Figure 3.19: induktiver Endschalter

Bei den Endschaltern handelt es sich um Schließvorrichtungen, die die Funktion haben, sich bei Vorhandensein eines metallischen Gegenstands in der Nähe des Sensorfeldes zu schließen, ansonsten geöffnet zu bleiben. Um den Bewegungsbereich der Dreh- Schwenkeinheit einzuschränken, wurden Aluminiumstreifen entsprechend zurechtgeschnitten und -gebogen und am **Tracking Device** befestigt, um eine Kollision mit dem Hauptträger und der Stereokameraträgerkonsole zu vermeiden. In Bild 3.20 dient der Aluminiumstreifen der Einschränkung der Kippbeweglichkeit

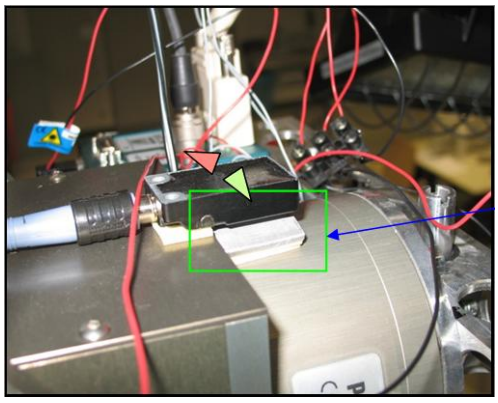


Figure 3.20: induktiver Endschalter Kippmodul

des **Tracking Device**s (er befindet sich direkt unter dem Sensor). In Bild 3.21 befindet sich

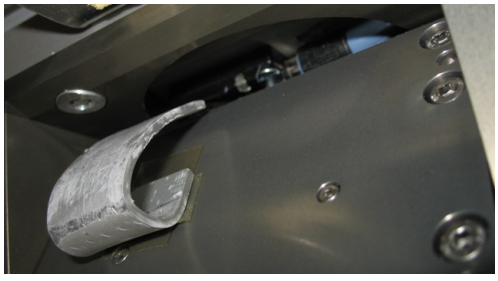


Figure 3.21: induktiver Endschalter Schwenkmodul

der für die Schwenkbewegungsbegrenzung montierte Aluminiumstreifen. Sollte nun der "**Power Cube**", durch einen Fehler in eine Endlage des Dreh- oder Schwenkbereiches gefahren sein, so wird sofort die Stromversorgung unterbrochen. Dies erfolgt durch zwei in Serie (in der Skizze 3.18 nicht ersichtlich) geschaltete Relais (siehe Aufbauskizze 3.18, grau- grün). Um jedoch wieder aus dem Endlagenbereich fahren zu können, wurde ein Überbrückungsknopf in die Schaltbox (siehe weiter unten, Schaltbox 3.36, schwarzer Knopf) eingebaut, der den Stromkreis wieder schließt. Die Stromversorgung (24V) des **Tracking Device** Terminal Block und der Schaltbox erfolgt dabei direkt über den Schaltkasten der Trumpfmaschine.

Direkt auf jeder der drei Kameras ist ein Laser mittels einer kleinen Bodenplatte angeschraubt. Die drei Laserstrahlen sind so eingestellt, dass sie möglichst parallel zur optischen Achse der jeweiligen Kamera verlaufen. Diese Einstellung kann mit einer kleinen Einstellroutine, die im Programmablauf vorgesehen ist, justiert werden. Dabei wird der Laserpunkt im Bild der Kamera erfasst und dann der Laser händisch (mit dünnen Blechunterlegeplättchen, siehe 3.22 darunter) eingestellt. Für die Feineinstellung der Laserstrahlrichtung können auch die beiden Wurmschrauben, die sich im Gehäuse befinden, verwendet werden. Die Funktion des verwendeten Programmes ist durch den gleichen Blobsuchalgorithmus gewährleistet der auch in der Hauptroutine (Echtzeitmessung) für die Suche des **Koordinatengebers** zuständig ist. Nach jeder neuen Positionierung des Lasers kann die Routine erneut gestartet werden, die die Abweichung des momentanen Laserpunktes von der genauen Bildmitte ermittelt. Um die Laserstrahlen in

Figure 3.22: Positionierungslaser

der Arbeitsmittenebene zu fokussieren und damit die kleinste Ausdehnung des Laserpunktes zu erreichen, wird die Linse mittels einer Fokussierhülse (im Lieferumfang des Lasers enthalten) in des Gehäuse hinein- oder herausgeschraubt. Die Spannungsversorgung der einzelnen Laser erfolgt parallel (nicht in der Aufbauskizze ersichtlich), damit jeder Laser einzeln über einen kleinen Schalter auf der Schaltbox ein- oder ausgeschalten werden kann.

Die Kameras (näheres siehe weiter unten) sind über Datenleitungen direkt mit den beiden "Framegrabbern" verbunden.

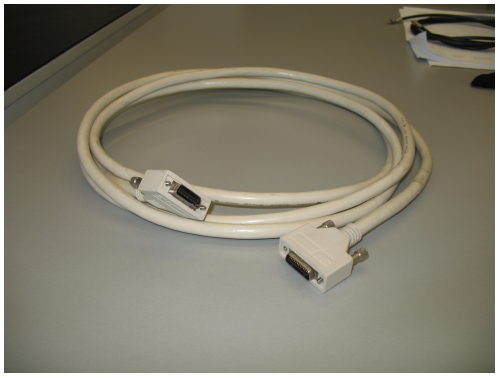


Figure 3.23: Base Camera Link ©Kabel

Die verwendeten Kabel sind "Universal Camera Link" ©Kabel mit einer maximalen Datenrate von 132 Mbyte/Sek. Eine Kamera ist dabei mit dem "Runtime" PC Bauer verbunden (und damit an einem Eingang mit dem "Odyssey Xpro+ Framegrabber"),welche zur Trackingberechnung zur Steuerung des "Power Cubes" benötigt wird. Kamera zwei und drei sind mit dem "Runtime" PC von DI FH Gelhart verbunden (an den beiden möglichen Eingängen des zweiten "Odyssey Xpro+ Framegrabbers"). Diese beiden Kameras sorgen für die Tiefenerkennung mit Hilfe eines Triangulationsverfahrens.

Um das **Tracking Device** ansteuern zu können, wird ein schneller "CAN Bus" verwendet (violett gestaltet), der am "Runtime" PC Bauer angeschlossen ist.

Zur direkten Ansteuerung Maschine der Firma Trumpf (siehe nächstes Kapitel) wird eine sehr schnelle Lichtwellenleiteranbindung verwendet, die mit dem Trumpf PC von DI FH Gelhart verbunden ist. Das verwendete Protokoll und damit die digitale Schnittstelle mit dem Namen "SERCOS" verläuft in einer geschlossenen Ringtopologie.

Da es nicht möglich war, beide "Framegrabber"-Einheiten in einem Computer zu installieren, wurde dies auf zwei Rechner aufgeteilt ("Runtime" PC- DI FH Gelhart, "Runtime" PC- Bauer). Zur Verbindung dieser beiden Einheiten in einem PC wurde auch ein OLPIC- Board (siehe Kapitel "Framegrabber"-Einheit) angeschafft, jedoch nicht verwendet. Um trotzdem eine Datenübermittlung zwischen den beiden "Runtime" PC zu ermöglichen, wurden alle verwendeten Computer über ein Netzwerk miteinander verbunden ("Tcp/ip Protokoll"). Die eigentliche Übertragung der Daten erfolgt über eine Software ("Twin Cat"), die die erforderlichen Bibliotheken der "C++"-Programmiersprache zur Programmierung der für das ggst. Forschungsprojekt benötigten Entwicklung einer maßgeschneiderten Software zur Verfügung stellte. Die auf dem "Runtime" PC-DI FH Gelhart" berechneten Tiefendaten des **Koordinatengebers** werden über dieses Netzwerk an den "Runtime" PC Bauer gesendet und im Programm als z- Komponente in die Berechnungen in Echtzeit miteinbezogen. Die sich ergebenden Daten (x,y und z- Koordinaten des **Koordinatengebers**) werden wiederum an den Trumpf PC DI FH Gelhart geschickt. Dieser Rechner schickt dann mit Hilfe von "Twin Cat" die übers Netzwerk erhaltenen Koordinaten über einen Lichtwellenleiter an die Maschine der Fa. Trumpf.

3.8.2 Die Maschine

Bei der anzusteuernden Maschine handelt es sich um einen Prototypen einer Blechbearbeitungsmaschine der Firma Trumpf mit zwei fahrbaren Stehern, auf denen jeweils eine Greifvorrichtung

befestigt ist. Auf einem Stehgestell, der in alle drei Richtungen linear fahren kann (x-, y- und

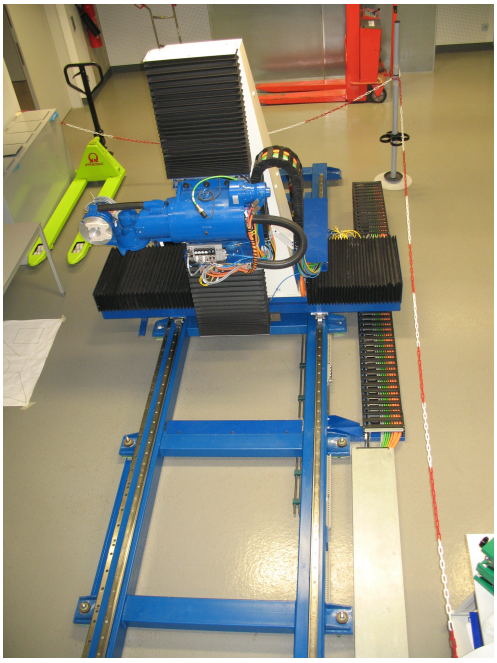


Figure 3.24: Werkzeugbearbeitungsmaschine der Firma Trumpf

z-Achse), ist ein Dreharm montiert, der in drei Richtungen verdrehbar ist (um die x-, y- und um die z-Achse). Durch die Konstruktion ist es jedoch nicht möglich, in allen Drehrichtungen eine volle Umdrehung auszuführen, da dies mit Endanschlägen verhindert wird. Aus diesem Grunde ist nicht jeder Punkt ohne zusätzliche Verdrehungen direkt zu erreichen. Auch muss berücksichtigt werden, dass bestimmte Bewegungen aufgrund der Maschinenausführung nicht ausführbar sind. Diese Problematik wird vor allem von der nicht ausreichend großen Auskragweite verursacht, da die Werkzeugmaschine in erster Linie für die Blechbearbeitung konzipiert wurde. Grundsätzlich ist die Maschine jedoch in jede beliebige Richtung (6-achsig) bewegbar bzw. drehbar. Hersteller der Linearbewegungsantriebe (x, y) ist die Firma Siemens, währenddessen

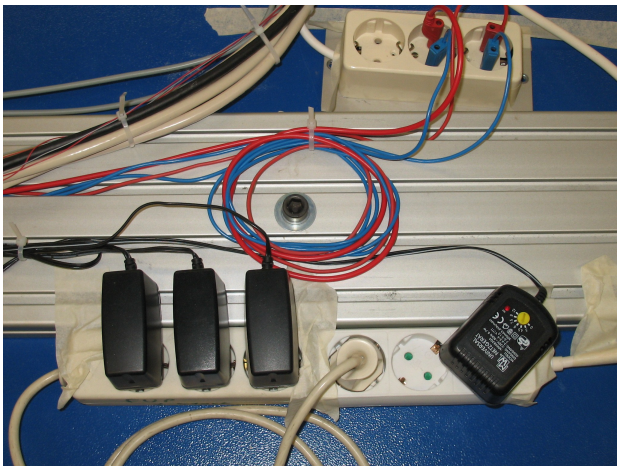


Figure 3.25: Ansicht von oben der Träger und der Verkabelung (auf dem Maschinenkühlaggregat befestigt)

die Drehbewegungsantriebe von der Firma Wittenstein hergestellt wurden. Für die Aufgaben dieses Diplomarbeitsprojekts wurde nur ein Stehgestell benötigt, das zweite wurde an das andere Ende der Maschine gefahren. Auf dem Verbindungsstück, wo vorher ein Greifarm montiert war, sollte eine Frässpindel befestigt werden, die einen Messerkopf trägt. Dieser Fräsaufsatz sollte in weiterer Folge für die Bearbeitung von Werkstücken herangezogen werden.

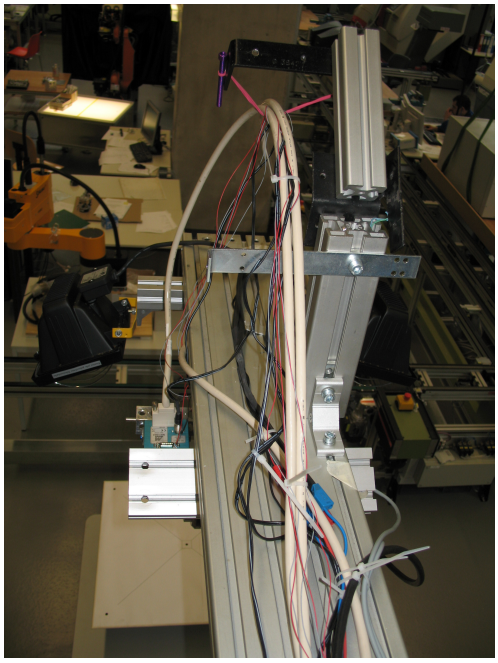


Figure 3.26: Aluminiumträger für Kamera- und Beleuchtungsbefestigung

Das "**Pan- Tilt device**" und die Halogenscheinwerfer werden an einem auskragenden Träger befestigt (Aluminiumprofil in der Abbildung 3.26), der auf dem Kühlaggregat der Maschine liegt. Die Befestigung erfolgt dabei mit einer Schraube, die sich in der Mitte des Kühlapperates befindet. Um die Konstruktion nicht allzu sehr mit der ungleichen Belastung zu beschweren, wurde an der gegenüberliegenden Seite ein Gegengewicht befestigt. Um eine Übersicht über den kompletten Maschinenaufbau zu gewinnen, wurde mit Hilfe von "Catia" eine "3- D"-Konstruktionszeichnung angefertigt. Mit dieser Zeichnung wurden Bewegungsgrenzen und Aufstellungsoptionen getestet. Auch die Prüfung der verschiedenen Bearbeitungsmöglichkeiten erfolgte auf Basis dieser Konstruktionszeichnung. Hierbei wurde unter anderem auch festgestellt, dass die Größe der bearbeitbaren Werkstücke nicht im ursprünglich gewünschten Umfang realisierbar war, da sich die Maschine für die Bearbeitung von sehr großen Werkstücken als ungeeignet erwies.

3.8.3 "Framegrabber"-Einheit

Den wichtigsten Bestandteil des gesamten Aufbaus stellen die "Framegrabber" dar. Deren Aufgabe ist es, den Großteil der auftretenden Rechenlast zu verarbeiten, und die Bilder von sämtlichen drei Kameras zwischenzuspeichern. Aus diesem Grund wurden zwei hochqualitative Einheiten angeschafft, die den höchsten Ansprüchen Rechnung tragen, die der Markt derzeit anbietet. Es handelt sich dabei um zwei Stück des Produkts "Matrox Odyssey XPro+ Framegrabber" (30.43).

Auf jedem dieser Recheneinheiten sind ein "Processingboard" (für die Berechnung, Zwischenspeicherung von Bilddaten) und eine eigene "Framegrabber"-Einheit montiert. Als Recheneinheit fungiert der Prozessor der Marke "G4 Power PC" mit einer Taktrate von 1,4 Ghz. Zusätzlich befinden sich auf der Platine die Produkte "Oasis ASIC" (Pixel Accelerator) und ein "FPGA". Der "FPGA" ist ebenfalls programmierbar, womit die Leistung verbessert werden kann, da die Effizienz durch Berechnungen direkt mit diesem Produkt steigt.
Auf dem 3.27 sind auch die beiden Anschlüsse für je eine Kamera ersichtlich. Dabei handelt es sich um eine "Camera Link ©Base" Verbindung. Betreffend weiterer Informationen zum "Camera Link" ©Protokoll wird an das Internet verwiesen.

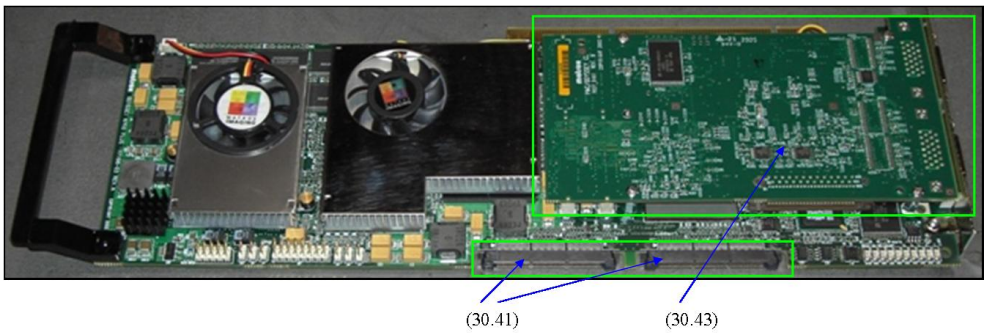


Figure 3.27: Matrox Odyssey Xpro+ Framegrabbereinheit

Für eine möglichst effiziente Erweiterung können weitere "Matrox Odyssey Xpro+" Boards über eine spezielle Schnittstelle (30.41) angeschlossen werden, ohne den PCI Bus zusätzlich zu belasten. Hierzu wird ein spezielles Framegrabberverbindungsboard ("OLPIC"="Odyssey Link Port InterConnect board") benötigt.

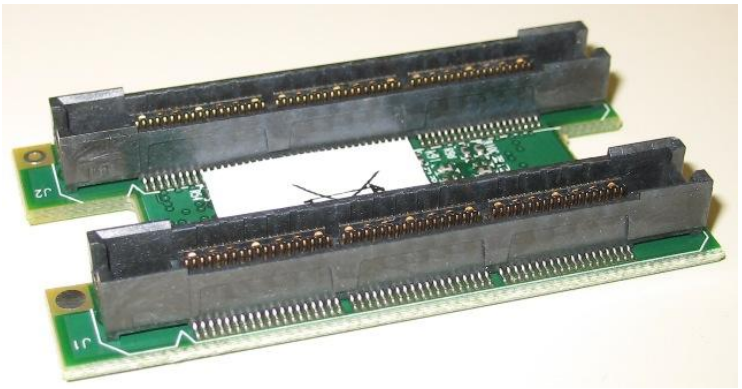


Figure 3.28: OLPIC

3.8.4 Computer

Hierbei handelt es sich um einen Computer der Marke "Dell Precision 650", der für die Bildverarbeitung der Trackingkamera benötigt wird. Der Arbeitsspeicher dieser Rechnereinheit hat 1 GB RAM. Weiters sind zwei Prozessoren der Marke "Intel Xeon 3,6 Ghz" eingebaut. In diesem Rechner ist eine "Framegrabber"-Einheit der Marke "Matrox Odyssey Xpro +" an einem

"PCI-X"-Steckplatz montiert. In der Abbildung (3.29) sind Anschlüsse der Graphikkarte (30.10) und (30.11) zu sehen. Der "CAN-Bus" PCI Kartenanschluss (33.10), die beiden "Camera Link ©Base" - Verbindungsstellen (30.44) und (30.45) finden sich ebenfalls in dieser Abbildung. (31.2) ist ein "Camera Link" ©Kabel mit einer Länge von 10m, wobei auch drei 3m lange Kabel bestellt worden sind.

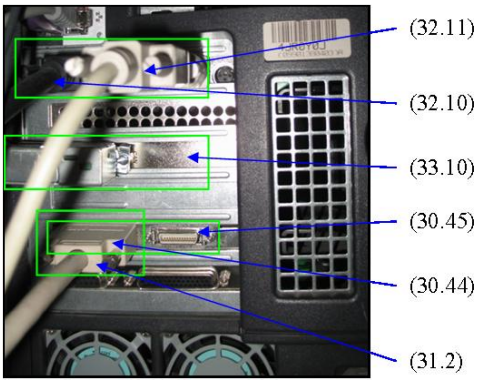


Figure 3.29: Trackingbildverarbeitungs-PC

Der zweite, für das ggst. Projekt in Verwendung stehende PC ist ein Gerät der Marke "Dell Precision 530", der als Arbeitscomputer genutzt wurde, um den Trackingbildverarbeitungs-PC nicht mit zusätzlichen administrativen Vorgängen zu überfrachten. Die Rechnereinheit besitzt einen 1,4 Ghz Prozessor mit einem Arbeitsspeicher von 500 MB RAM.

3.8.5 Dreh-, Schwenkeinheit

Gleichzusetzen mit den Bezeichnungen **Tracking Device** oder Power Cube. Um die beiden Stereokameras oder besser gesagt deren relativ kleine Bildausschnitte immer zentral zu dem sich bewegenden **Koordinatengeber** entlang zu führen, mussten diese beiden Kameras auf einer **Dreh- Schwenkeinheit** befestigt werden. Zu diesem Zweck wurde die Dreh- und Schwenkeinheit der Marke "Acuity Research PTU 46-7,5"

Figure 3.30: Dreh-, Schwenkeinheit Acuity PTU 46-17.5

zu Testzwecken von der Firma CVIS für eine Woche ausgeliehen. Die dabei gewonnenen Erkenntnisse mündeten in der Notwendigkeit der Bestellung eines Gerätes mit einem erweiterten Schwenkbereich, um eine erhöhte Kamerabeweglichkeit zu erreichen.

- PTU-D46-17 (CVIS):
  Auflösung: 0,0514 °
  Drehgeschwindigkeit: 300 °/s
  Schwenkgeschwindigkeit: 300 °/s
  Max. Gewicht: 1,81 kg

  Darüber hinaus war die Belastbarkeit dieser **Dreh- Schwenkeinheit** mit 1,8 kg sehr eingeschränkt. Aber auch der Verlust an Synchronität während schnellerer Fahrbewegungen und das festgestellte große mechanische Spiel der einzelnen Teile, was eine unzureichende Exaktheit der Kamerabewegungen zur Folge hatte, entsprach nicht den geforderten Qualitätskriterien für die Anwendung dieser "PTU" - Einheit im Rahmen des ggst. Forschungsprojekts. Überdies lag das Gesamtgewicht der beiden Kameras und des Stereokameraträgers bereits im Stillstand an der Belastungsgrenze der Dreh- und Schwenkeinheit. Aus Qualitätsgründen wurde daher der teureren Einheit der Marke "Powercube PW 070" bei diesem konkreten Gerätebeschaffungsvorgang der Vorzug gegeben.

- PowerCube PW070 (Schunk):
  Auflösung: 0,04°
  Drehgeschwindigkeit: 248 °/s
  Schwenkgeschwindigkeit: 356 °/s
  Max. Gewicht: (Abh. von Lastverteilg.) - kg

Figure 3.31: PowerCube 070

Zur exakten Ansteuerung dient hierbei ein "Harmonic Drive", das eine hohe Präzision bei schneller und kraftvoller Bewegung bietet. Nähere Details können der kurzen auszugsweisen Auflistung weiter oben entnommen werden.

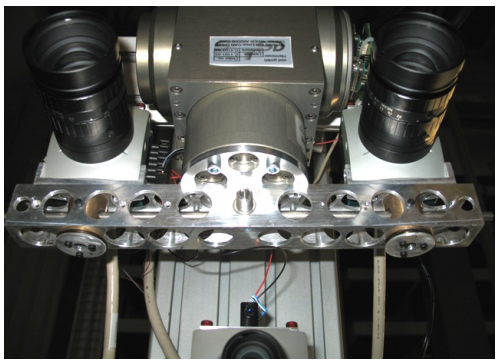


Figure 3.32: Pan Tiltdevice

3.8.6 Kamera

Die drei Kameras wurden so positioniert, dass diese den **Koordinatengeber** direkt von oben verfolgen können. Der Abstand zwischen der **Arbeitsmittenebene** und den Kamerachips beträgt 1800 mm (siehe Abbildung 3.33, schwarze Linie). Um diesen Abstand möglichst exakt zu erreichen, wurde ein Tisch auf höhenverstellbare Stützen gestellt. Die Tischhöhe entspricht zwar exakt den Ausmaßen des geforderten **Arbeitsabstand**s, stellt jedoch trotzdem nur eine Zwischenlösung dar, da er für die notwendige Nutzung des gesamten Arbeitsbereichs ungeeignet ist. Zur Sicherstellung eines besseren Kontrasts wird ein schwarzer Teppich verwendet, damit der

Koordinatengeber von den Kameras auch ohne Tischkontrast sicher erkannt werden kann. Die beiden Stereokameras sind über die Stereokameraträgerkonsole mit dem "**Pan-Tilt Device**" verbunden, damit der Fokus während der Kamerafahrbewegung immer genau auf den **Koordinatengeber** ausgerichtet ist.

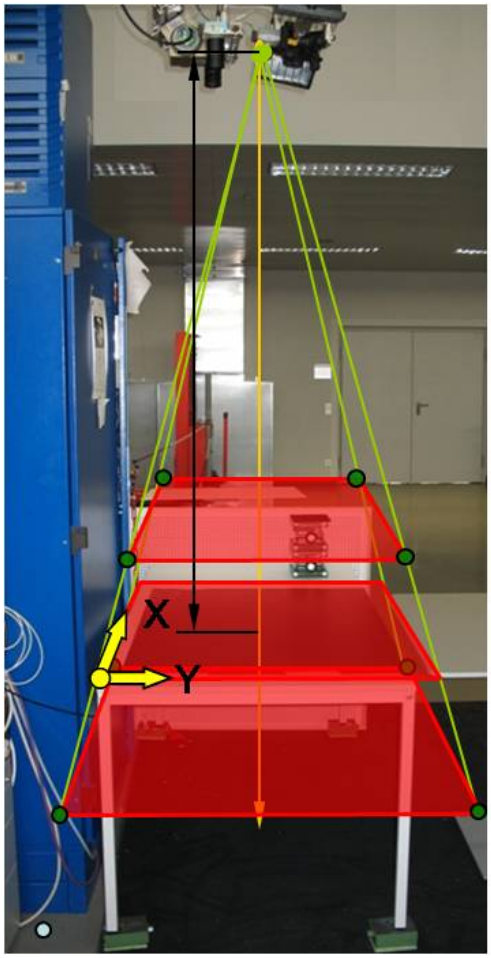


Figure 3.33: Arbeitsraum der Trackingbildkamera

In der Abbildung 3.33 ist das **Trackingbildkoordinatensystem** gelb eingezeichnet. Rot eingezeichnet sind die drei Ebenen "**Abbildungsoberebene**", "**Abbildungsmittenebene**" und "**Abbildungsunterebene**" zu sehen. Weitere Angaben sind in [2] zu finden.

3.8.7 Beleuchtung

Aufgrund der hohen Bildwiederholungraten ist die Belichtungszeit zwischen den Einzelbildern ungenügend. Um dieses Manko auszugleichen, werden zwei sehr leuchtstarke 500 Watt Scheinwerfer eingesetzt, die den Arbeitsbereich beidseitig von oben herab beleuchten. Dies vermindert gleichzeitig den Einfluss, den die Sonneneinstrahlung auf die Gesamthelligkeit ausübt. Ohne Scheinwerfer wäre es notwendig, die in ihrer Helligkeit stark schwankende Sonneneinstrahlung auszugleichen. Um zusätzlich noch eine hohe Tiefenschärfe zu erreichen, muss die Blende auch relativ stark geschlossen werden, was die auf den Chip auftreffende Helligkeit weiter verringert.

Ausserdem ist zu berücksichtigen, dass ein "CMOS"-Kamera-Chip gegenüber einem "CCD"-Kamera-Chip eine geringere Lichtempfindlichkeit aufweist.

Figure 3.34: Beleuchtung

3.8.8 "Tracking Device Terminal Block"

Hierbei handelt es sich um eine Platine, auf der sich die Sicherung und die Stromanschlüsse für den "Power Cube" befinden. Weitere Informationen finden sie in [2], Seite 76.

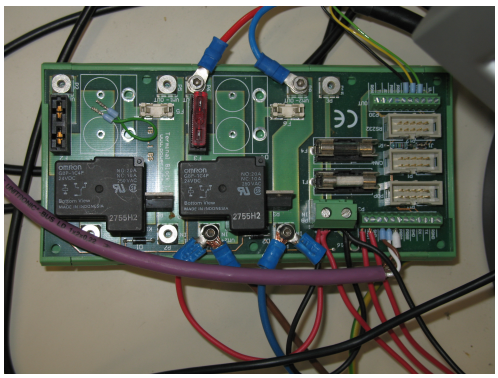


Figure 3.35: Tracking device Terminal Block

3.8.9 Schaltbox

Dies Ist jene Einheit, womit die drei Positionierungslaser der drei Kameras ein- und ausgeschaltet werden können. In dieser Schaltbox befindet sich auch der Überbrückungsschalter, der bei

Auslösung eines der Endschalter durch das "**Tracking Device**" betätigt werden muss. Weiteres ist [2] Seite 68 entnehmen.

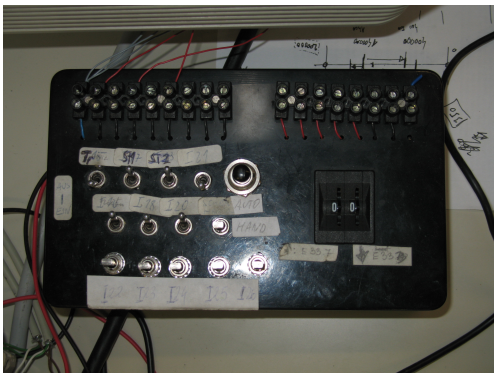


Figure 3.36: Schaltbox

3.9 Stereokameraträgerkonsole

3.9.1 Voraussetzungen

Zur Verbindung der beiden Stereokameras wurde eine Stereokameraträgerkonsole entworfen. Die Aufgabe dieser Konsole war es, einen möglichst symmetrischen Aufbau der beiden Kameras zu gewährleisten. Auch sollte der Basisabstand (der Abstand der beiden Kameraachsen zueinander) verstellbar sein.

Anforderungen an die Stereokameraträgerkonsole:

- Leichtes Gewicht (je leichter das Gewicht, desto schnellere Bewegungen waren erzielbar)
- Basisabstand variierbar
- Symmetrische Verstellbarkeit des Kamerawinkels beider Kameras
- Möglichst tragfähige und kompakte Ausführung (das Gewicht der Kameras verursacht entsprechend hohe Trägheitskräfte)
- Schnelle Montage, schneller Umbau

3.9.2 Konstruktion

Zur Berechnung der Tragfähigkeit der Konstruktion wurde das Flächenträgheitsmoment an der dünnsten Trägerstelle ausgerechnet. Die dynamisch wirkenden Kräfte wurden mit Berücksichtigung des anteiligen Trägergewichts, der maximalen Geschwindigkeit und der Beschleunigung des **Power Cube**s unter der Annahme, dass sich das Kameragewicht in der äussersten Position befindet, berechnet. Um das Gewicht der Kamerakonsole zu verringern, wurden in der rechten und der linken Seite des Trägergerüsts Löcher gebohrt. Oben und unten wurden die Löcher gefräst, um den Basisabstand veränderbar zu können. Dazu müssen die beiden Drehachsenhülsen

umgesteckt werden.
Rechts unten im 3.37 kann man die bereits eingebaute Hülse sehen, die seitlich mittels kleiner Schauben mit dem Träger verbunden wird. Danach werden die Drehachsen in die Drehachsenhülsen gesteckt, und das Drehachsenverbindungsgestänge in die eine schon eingebaute Hülse geschoben, bis der Mittelteil an der Hülse ansteht. In weiterer Folge kann die zweite Hülse eingebaut werden (inklusiver Drehachse) und das Gestänge miteinander verbunden werden. Dann wird noch die Drehachsenverstellschraube1 eingebaut und mit der zweiten Drehachsenverstellschraube2 zusammengefügt.

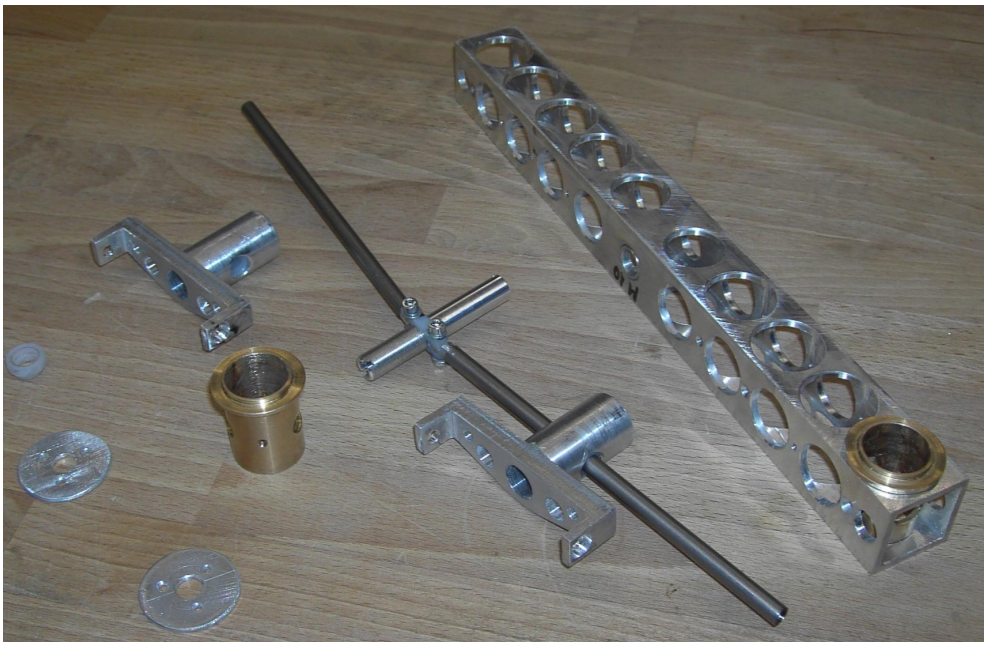


Figure 3.37: Träger zerlegt

4 Programmierung

4.1 Variablenkonventionen

Klassen, Funktionen, Strukturen werden mit einem großen Anfangsbuchstaben geschrieben:

C_Bsp_dat ... Klassendefinitionen
m_... Member einer Klasse
F_Bsp_dat()... Funktionsbezeichnungen (ohne Klassenzugehörigkeit)
Str_Bsp_dat... Strukturbezeichnungen

Instanzen einer Klasse werden klein geschrieben, ohne sonstige Merkmale.
Konstante Variablen (mit "define" deklarierte Variablen oder "const"):

BSP_DAT... mit "define" deklarierte Variablen werden durchgehend groß geschrieben
ini_... zeigt eine konstante Variable an, die jedoch über eine Datei geändert werden kann.
const_... zeigt eine Variable des Typs "const" an

Normale Variablen werden klein geschrieben (falls mehrere Merkmale zutreffen, werden diese in der Reihenfolge der Auflistung kombiniert; siehe Kombinationsmerkmale (KM)):

p_... zeigt einen "pointer" an (KM)
st_... zeigt eine statische Variable an (KM)
a_... zeigt ein "array" an (KM)
b_... zeigt eine Variable des Typs "bool" an
n_... zeigt eine Variable des Typs "int (short)" an
w_... zeigt eine Variable des Typs "unsigned int (short)" an
l_... zeigt eine Variable des Typs "long" an
dw_... zeigt eine Variable des Typs "unsigned long" an
dwd_... zeigt eine Variable des Typs "DWORD" an (Window.h)
f_... zeigt eine Variable des Typs "float" an
df_... zeigt eine Variable des Typs "unsigned float" an
d_... zeigt eine Variable des Typs "double" an
c_... zeigt eine Variable des Typs "char" an
s_... zeigt eine Variable des Typs "string" an
sz_... zeigt eine Variable des Typs "string" mit "0" abgeschlossen an

v_... zeigt eine Variable des Typs "void" an

Funktionen der MIL MATROX 8.0 Bibliothek:
Mil... Variable mit dieser Bezeichnung zeigen eine Variable der "Mil Library" an.

4.2 Programmierumgebung

Für die Programmierung wurde die Programmiersprache "C++" in Windows XP verwendet, die in der Software "Visual Studio 2005" von Microsoft beinhaltet ist. Das Projekt läuft in einer "Win32" -Anwendung, basierend auf einer Konsolenanwendung. Als Graphikbibliothek wurde die Software "MIL 8.0" der Firma Matrox verwendet ([4]). Damit konnten alle für die Bearbeitung notwendigen Funktionen verwendet werden. Von einfacheren arithmetischen Berechnungen bis zu komplexeren Kantenfindungsalgorithmen war alles vorhanden. Mit dem Tool der Firma Mikrotron (siehe [2] Seite 40) konnten die verschiedenen Variablen der drei Kameras eingestellt werden. Dieses Werkzeug muss bei jeder Veränderung der Kameraeinstellungen herangezogen und damit neu aufgerufen werden. In Zukunft sollten diese Einstellungen jedoch auch direkt über das "C++"-Programm zu steuern sein. Zum Einrichten der "Framegrabber" wurden mehrere Softwareprogramme mitgeliefert (siehe [2], Seite 36 bis 40). Diese ermöglichten unter anderem, die Auslastung aller Recheneinheiten des Framegrabbers anzuzeigen. Um das **"Tracking Device"** ansteuern zu können, wurde eine "C++"-Bibliothek mitgeliefert. Die Anwendung des mitgelieferten "Amtec Power Cube Demo"-Programms kann in [2], Seite 43 nachgelesen werden. Dieses Tool wurde zur Einstellung des "Power Cubes" verwendet. Vor allem bei Problemen, wie z. B. bei Kontakt der Werkzeugmaschine mit deren Eigenschutzvorrichtungen (Endschalter) oder Fehlern bei der Erkennung des "Cubes" lieferte das Werkzeug gute Dienste. Weitere Beschreibungen von verwendeten Programmen sind in [2] ab Seite 36 nachzulesen.

4.3 Programmbeschreibung

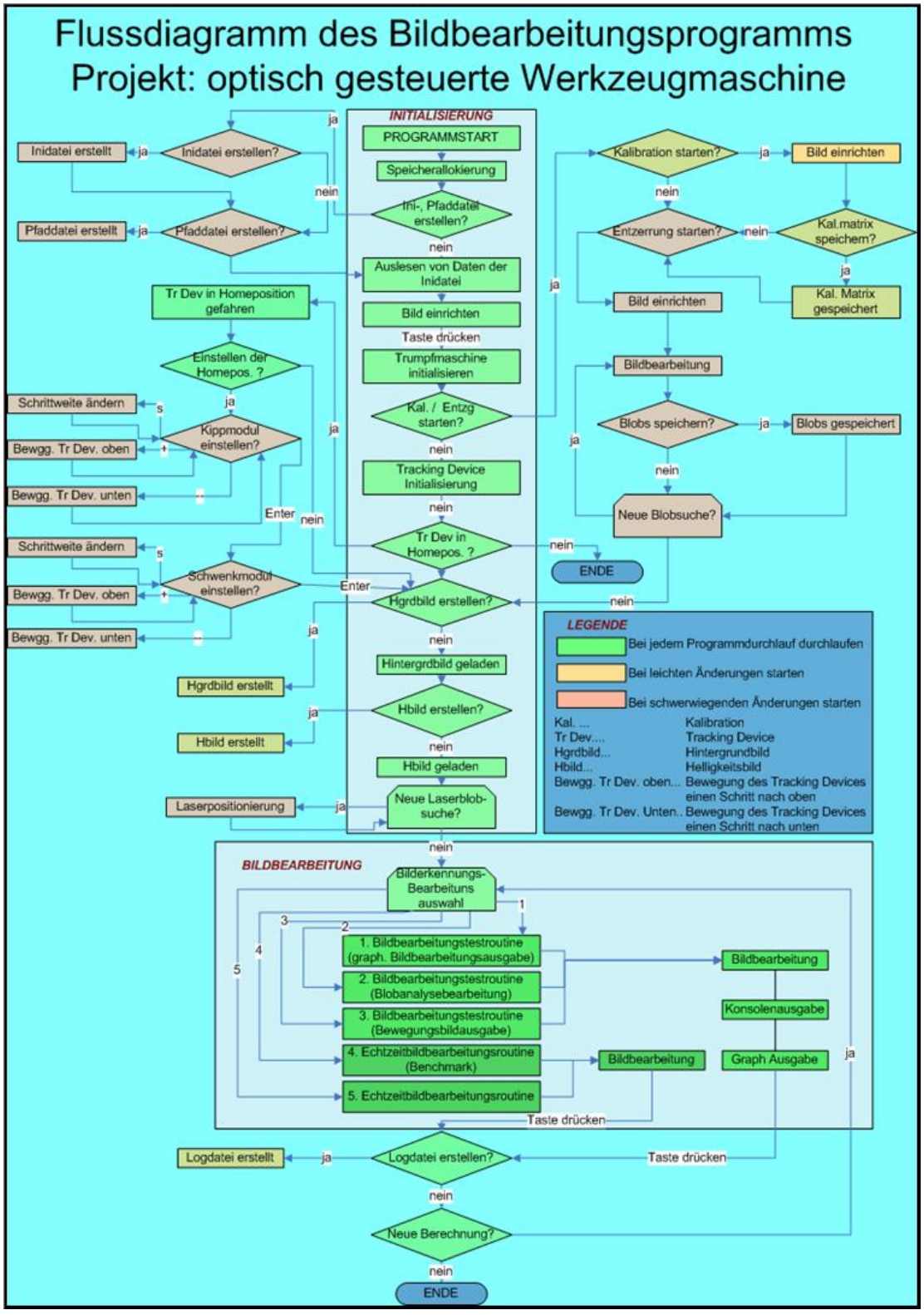


Figure 4.1: Ablaufplan der Bildbearbeitungsroutine

Als nächster Punkt erfolgt die Kameraaufnahme der **Kalibrationsmatrix**, wenn dies erforderlich ist. Diese dient dazu, das Bild von Bildverzerrungen, wie z. B. dem Kissen- oder anderen Kameraeffekten mit optischen Verfälschungen zu befreien. In weiterer Folge wird die Vermessung zur Entzerrungsberechnung durchgeführt. Dabei wird eine **Entzerrungsmatrix** ins Bild gelegt und in regelmäßigen Abständen aufgenommen, um eine Entzerrungsfunktion zu errechnen. Mit dieser wird die Verzerrung des **Koordinatengebers**, in Abhängigkeit zur dessen Entfernung , im Bild kompensiert. Danach kann die "Homeposition" des Kipp- und des Drehmoduls des "**Tracking Devices**" eingestellt werden. Soll die Erstellung eines neuen **Hintergrundbilds** oder eines neuen **Helligkeitsbilds** erforderlich sein, so kann dies mithilfe des Bildbearbeitungsprogramms erledigt werden. Auch die Durchführung der Einstellungen zur Positionierung der Laser ermöglicht dieses Programm. Vor dem Start der eigentlichen Echtzeiterkennung ist es überdies notwendig, sich zwischen fünf verschiedenen Bildbearbeitungsroutinen zu entscheiden (Genaueres siehe [2], Seite 13). Nach Abbruch der Bildbearbeitungsroutine kann die Aufzeichnung der Daten in einer Datei (Dataarray.dat, Variablenbezeichnungen und Erklärungen siehe [2] Seite 55) gespeichert werden.

4.3.1 Programmablauf

Die Ablaufprozedur mit den Programmein- und -ausgabewerten wird in einem Konsolenfenster am Computerbildschirm angezeigt, wie im Anschluss illustriert, wobei die Bezeichnung "ANr." für Ausgabenummer steht (aber nur der laufenden Nummerierung dient) und "Konsolenausgabe" für den Text (oder zumindest dem angedeuteten Text), der im Konsolenfenster sichtbar ist.

ANr.$	"Konsolenausgabe"
Hilfestellung bei der Konsolenausgabe	
1$	"Programmstart"
Zeigt an, dass das **Bildbearbeitungsprogramm** ordnungsgemäß gestartet wurde (diese Routine sollte in der Konsolenausgabe an erster Stelle aufscheinen). Sollten Fehlermeldungen wie z. B. "Datei Inifile.dat konnte nicht geöffnet werden" auftreten, sagt dies nur aus, dass die Datei "Inifile" nicht existiert oder nicht lesbar ist. Sollten weitere Probleme im Programmverlauf auftreten, erstellen Sie eine neue Datei "Inifile", um dann das Programm wieder neu starten zu können.	

Table 4.1: Konsolenausgabe1

2$	"Erstellen einer neuen Inidatei/Pfaddatei mit Standardeinstellungsdaten (j/n)"
Abfrage, ob der Anwender eine neue Initialisierungs- oder Pfaddatei erstellen möchte.	
2.1$	"Neuerstellung der Initialisierungs- Datei ? (j/n)"
Für den Fall, dass keine Initialisierungsdatei vorhanden ist, oder wenn alle Parameter mit Standardwerten (die in der Datei "Inistart.h" stehen -> siehe Kap. 5.1 VNr. 15 Pathlog.dat [2]) beschrieben werden sollen, erstellt diese Datei eine neue Initialisierungsdatei. Eine Neuerstellung kann auch sinnvoll sein, wenn eine Initialisierungsdatei umbenannt oder gelöscht wurde, um für einen bestimmten Verwendungszweck zur Verfügung zu stehen und bei Bedarf einfach wieder in "Inifile.dat" umbenannt werden muss, um die aktuelle Datei zu ersetzen.	
2.2$	"Neuerstellung der Standardverzeichnis- Datei? (j/n)"
Dieser Prgrammabschnitt erstellt eine neue "PathLog.dat"- Datei. Darin sind alle selbst erstellten Dateien und deren Verzeichnisstrukturen abgespeichert.	
3$	"Ändern eines Parameterwertes der Initialisierungs- Datei ? (j/n)"
Diese Datei listet sämtliche Parameterwerte, die in der Initialisierungs Datei stehen, auf. Diese Auflistung lässt eine Reihe von Textfeldern frei, um Parameter auch nachträglich in das ggst. Programm hinzufügen zu können. Das bedeutet, dass Parameter der Initialisierungsdatei, die umgestellt werden sollen, mit Ausnahme der berechneten Konstanten und einigen weiteren Ausnahmen, die nicht direkt geändert werden können, da diese wiederum von anderen Parametern abhängig sind, umgeschrieben werden können (sollte ein Parameter, der von anderen Parametern abhängig ist, ausgewählt werden, wird nach der Eingabe des Wertes ein kleines Unterprogramm aufgerufen, um die Eingabe zu steuern). Die abgeänderten Werte werden unmittelbar in die Initialisierungsdatei geschrieben, um für den weiteren Programmverlauf verwendet zu werden. Daher sollte eine derartige Änderung vor Umsetzung gut überlegt werden.	
4$	"Einrichten des Bildes (Stoppen mit einer bel. Taste)"
Mit Hilfe dieser Funktion ist es bis zum nächsten Tastendruck möglich, das momentan dargestellte "Live"-Kamerabild einzustellen. Z. B. ist es möglich, die Kontraststärke fein abzustimmen, wenn das Bild zu dunkel erscheinen sollte. Zur Gewährleistung einer optimalen "Blob"-Erkennung ist die Verwendung eines strahlenden weißen Materials als **Koordinatengeber** sinnvoll, da die Helligkeitskorrektur in diesem Fall nicht zum Einsatz kommen muss. Für die bestmögliche Erkennung der Signalhand ist als Kontrast dazu ein möglichst statischer und dunkler Hintergrund wichtig, um die Hintergrundberechnung möglichst einwandfrei durchführen zu können. Da eine Änderung der Kameraeinstellungen mit Hilfe des Bildverarbeitungsprogramms aufgrund programmiertechnischer Schwierigkeiten noch nicht möglich ist, kann das "Kamera-Tool" der Firma Mikrotron alternativ dazu für diese Bearbeitungsschritte aufgerufen werden. Näheres dazu siehe die Übersicht in [2] Seite 29 oder auch Kap. 5.3.2 [2].	
5$	"Entzerrungs - und Kalibrationsroutine starten? (j/n)"
Diese Routine eröffnet die Möglichkeit, die Trackingkamera zu kalibrieren (siehe Kap. 2.2.4 [2]) und zu entzerren (Entzerrungsvorgang siehe Kap. 2.2.5 [2]).	
5.1$	"Kalibrieren des Bildes (j) oder alte Kalibrationsmatrix aufrufen (n)...?"
Hierbei wird eine neue Kalibrationsmatrix erstellt, die zur Kalibrierung der Kamerabilder und zum Ausgleich von optischen Verzerrungen genutzt wird. Wenn eine andere Taste als 'j' betätigt wird, wird eine schon in einem vorigen Programmdurchlauf erstellte Kalibrationsmatrix herangezogen, was kein Problem darstellt, solange keine gravierende Änderungen am Aufbau (siehe Kap. 3.3 [2]) vorgenommen wurden.	

Table 4.2: Konsolenausgabe2

5.1.1$	"Einrichten der richtigen Bildposition (Stoppen mit einer bel. Taste)"
Hierzu muss ein Kalibrationsgitter verwendet werden und in den Bildbereich gelegt werden. Die Ausrichtung sollte möglichst waagrecht zur Arbeitsmitte erfolgen, die Kanten symmetrisch und parallel zueinander angeordnet sein. Solange keine weitere Taste betätigt wird, können die Positions- und eventuell auch die Kameraeinstellungen über das "Kamera-Tool" der Firma Mikrotron vorgenommen werden. Vor dem Einrichten des Kamerabilds ist die Schaffung guter Kontrastverhältnisse zwischen dem Hintergrund und den Matrizenpunkten erforderlich (der Hintergrund sollte einen satten weißen bzw. schwarzen Farbton aufweisen und die Matrizenpunkte den jeweils gegenteiligen Farbton). Die hier verwendeten Kalibrationsmatrizen haben einen weißen Hintergrund mit schwarzen Punkten, was jedoch einfach durch einen Einbau einer Invertierungsfunktion in das Programm geändert werden kann. Um die nächsten Bildbearbeitungsschritte, aber ggf. auch die Umstellung der Kameraparameter nach einem Neustart des Bildverarbeitungsprogramms und deren Wirkung auf die **Bildbearbeitung** vor der Kalibration besser nachvollziehen zu können, werden diese im Graphikfenster ausgegeben.	
5.2$	"Geben Sie nun die Blobmatrix ins Bild, um die Entzerrung zu messen (j/n)"
Um anschließend mit der Entzerrungsroutine starten zu können, ist die Taste 'j' zu betätigen. Vor dieser Tätigkeit ist eine Entzerrungsmatrix (gleiche Bedingungen wie bei der Kalibration) in den Abbildungsbereich zu legen. Wichtig ist hierbei wieder eine genaue Positionierung, gute Kontraste, sowie eine waagrechte Ausrichtung zur Arbeitsmitte. Im Anschluss daran wird die gleiche **Bildbearbeitungssoftware**, wie schon im vorigen Punkt beschrieben, gestartet. Im Gegensatz zur Kalibration wird nun eine "Blobanalyse" gestartet, die versucht, aus dem Entzerrungsgitter die "Blobs" herauszufiltern. Die gefundenen "Blobs" werden dann im Graphikfenster angezeigt, und die Koordinaten sowie die "Blob"-Anzahl ausgegeben.	
5.2.1$	"Blobkoordinaten speichern? (j/n)"
Im Fall einer fehlerfreien Feststellung sämtlicher Gitter-"Blobs" können die ermittelten Blobkoordinaten der jeweiligen Blobs in die Datei "Blobkoordinaten.dat" transferiert werden (die "Blob"-Anzahl hängt vom Entzerrungsgitter und vom Entzerrungsverfahren ab; Standard sind 5 Blobs).	
5.2.2$	"Bitte geben sie den momentanen Abstand(auf...)"
Bevor die Koordinaten in die Datei geschrieben werden können, muss noch der genaue Abstand des Entzerrungsgitters zum Kamerachipmittelpunkt angegeben werden (oder von einem anderen markanten Punkt, dessen Abstand zur Kamera genau bekannt ist). Dabei ist darauf zu achten, auch die verlangten Kommastellen in die Datei einzugeben.	
5.2.3$	"Bitte geben sie den momentanen Winkel der Kalibriergitterebene ein ..."
Hier ist zusätzlich der momentane Winkel des Entzerrungsgitters anzugeben, den dieses mit der Arbeitsmittenebene einschließt (dieser Winkel stellt einen veralteten Parameter dar; hierbei ist immer die Zahl '0' einzugeben). Mit diesen Angaben können die Blobkoordinaten endgültig in die Datei geschrieben werden.	

Table 4.3: Konsolenausgabe3

5.2.4$	"Verschieben des Entzerrungsgitter von der Kameraposition weg, um ..."
Um den Vorgang bei Punkt 5.2 neu zu starten, können sie hier die Taste 'j' betätigen. Davor ist das Kalibrationsgitter möglichst in der Achse der Trackingkamera in Richtung Trackingkamera zu bewegen (der Abstand zur vorherigen Position ist nicht von Bedeutung, allerdings sollten dieser im Bereich von ca. 70 mm liegen, jedoch sollten Kameraaufnahmen über den gesamten Arbeitsraum erstellt werden (bzw. über den gesamten Arbeitsraum hinaus), um die Funktionen, die mit diesen Messungen errechnet werden, so genau wie möglich zu ermitteln. Je geringer der Abstand zwischen den einzelnen Messungen ist, desto mehr Messungen müssen durchgeführt werden (bei einem Arbeitsraum von ca. 900 mm entspricht das mehr als 12 Messungen). Der Mindeststandard umfasst 3 Messungen, um eine Entzerrungsfunktion zu approximieren.	
6$	"Bibliotheks-, Versionsnummer: ..."
Hierbei wird die momentan aktuell verwendete Versionsnummer der "*.dll" herangezogen, die für das "**Tracking Device**" in Verwendung ist. Die darauf folgenden Zeilen stellen nur eine Information für den Anwender dar, wobei die Baudrate noch nicht richtig ausgegeben wird, was der Funktionalität des Programms keinen Abbruch tut.	
7$	"Tracking Device in Homeposition fahren? (j/n)"
Diese Abfrage muss (!) bei jedem Bildverarbeitungsprogrammdurchlauf positiv beantwortet werden, da hier das "**Tracking Device**" in die Position gefahren wird, die der Ausgangsposition für die **Bildbearbeitung** entspricht (diese liegt im Idealfall genau in der Mitte des Arbeitsbereiches). Die im Titel bezeichnete "Homeposition" ist hier nicht als "**Tracking Device**- Homeposition" des "**Tracking Device**s" zu sehen, die für die Kalibrierung des Gerätes vorgenommen werden muss, und auch vor dem Betätigen einer weiteren Taste umzusetzen ist. Dies geschieht zwar automatisch, doch muss der "User" solange warten, bis das Gerät wieder völlig zum Stillstand gekommen ist, da das "Device" in weiterer Folge nicht mehr angesteuert werden kann.	
7.1$	"Homeposition einstellen? (j/n)"
Für den Fall,dass das "**Tracking Device**" (und damit auch die Stereokameraposition (siehe Kap. 2.2.3 [2]) eingestellt werden soll, können sie mit dem Drücken der 'j' Taste auf der Tastatur eine Einstellroutine starten, über die sich die Ausgangsposition ("Homeposition") auf kurzem Wege einstellen lässt.	
7.2$	"Geben sie ein '+' (runter) oder ein '-' (rauf) pro Schritt...."
Durch Betätigen der Taste '+' oder '-' kann der "Power Cube" in der jeweils angegebenen Achse bewegt werden. Falls die Schrittweite geändert werden soll (und damit der jeweilige Drehwinkel entsprechend der Zahl der betätigten Bewegungstasten '+', '-'), so kann stattdessen ein 's' auf der Tastatur eingegeben werden worauf die Aufforderung auf der Konsole aufscheint, und sie einen neuen Wert eingeben können, mit dem sie das Gerät neu positionieren können.	
7.2.1$	"Geben sie eine neue Schrittweite ein [Bereich der Einstellmöglichkeit]..."
Der eingetippte Wert wird in Radianten angegeben, wodurch Vorsicht bei größeren Werten geboten ist, da das Gerät schnell außerhalb des Endlagenbereichs gebracht werden kann. Diese Einstellung kann während der Einstellung des Drehmoduls, als auch während der Änderung des Kippmoduls neu gesetzt werden. Denkbar ist zuerst eine grobe Schrittweite für die Annäherung an die gewünschte Position, um dann eine feinere Schrittweite für die Feinjustierung wählen zu können (siehe Kap. 2.2.2 "Homeposition" des "**Tracking Device**s" einstellen [2]). Zu erwähnen wäre noch, dass die Bewegungstasten nach der Eingabe der neuen Schrittweite wieder aktiv sind und der Einstellvorgang fortgeführt werden kann. Das Klicken während der Einstellung deutet auf die Bremsenaktivität hin.	

Table 4.4: Konsolenausgabe4

7.3$	"Kippmodul einstellen:"
Hierbei kann der Kippmodul des "Power Cubes" eingestellt werden, wobei hier immer die Schrittweite nach 7.2.1$ verändert werden kann. Die Beendigung der Einstellung des Kippmoduls kann durch die Betätigung einer anderen beliebigen Taste ausser der beiden Bewegungstasten oder der Taste 's' für die Schrittweiteneinstellung erfolgen.	
7.3.1$	"Wollen sie die neu eingestellte Kippmodul -Homeposition abspeichern?..."
Hier besteht die Möglichkeit, durch betätigen der 'j' Taste die eben eingestellte Kippmodulposition in die Initialisierungs- Datei zu schreiben, um die Einstellungen beim nächsten Programmaufruf nicht mehr neu einstellen zu müssen.	
7.4$	"Drehmodul einstellen:"
Genau wie beim Einstellen des Kippmoduls ist hier die Möglichkeit gegeben, den Drehmodul einzustellen. Das Beenden der Einstellung des Drehmoduls erfolgt durch Drücken einer anderen beliebigen Taste mit Ausnahme der beiden Bewegungstasten oder der Taste 's' für die Schrittweiteneinstellung.	
7.4.1$	"Wollen sie die neu eingestellte Kippmodul- Homeposition abspeichern?.."
Auch hier kann die Drehmodulposition in die Initialisierungs- Datei geschrieben werden. Der Einstellvorgang wird durch das Drücken einer beliebigen Taste beendet.	
8$	"Falls ein neues Hintergrundbild erstellt werden soll, bitte 'j' drücken..."
Bei Betätigen der Tastaturtaste 'j' wird ein neues Bild aufgenommen, wobei wichtig ist, dass sich keine, auch im späteren Verlauf, bewegliche Objekte im Bildausschnitt befinden (im Speziellen ist hier der **Koordinatengeber** gemeint). Dabei sollten keine weiteren Veränderungen im Abbildungsbereich der Kamera vorgenommen werden (keine Kameraeinstellungsänderungen, Objektverschiebungen oder Ähnliches). Wenn sich die Einstellungen seit dem letzten Berechnungsdurchgang nur geringfügig oder gar nicht geändert haben, ist es möglich, durch Drücken einer anderen beliebigen Taste, mit Ausnahme der Taste 'j', ein altes gespeichertes Hintergrundbild aufzurufen. Im Regelfall ist es jedoch sinnvoller, ein aktuelles Bild zu erstellen, da konstante Umgebungseinflüsse, z. B. bedingt durch schwankende Sonneneinstrahlung, Verschiebungen von Hintergrundobjekten, etc. meist nicht gegeben sind. Eine genauere Beschreibung der Verwendung des Hintergrundbildes in der **Bildbearbeitung** findet sich in Kapitel 4.4.	
8.1$	"Hintergrundbild kalibriert"
Diese Anzeige gibt die Entzerrung des Hintergrundbilds mittels einer in Pkt. 5$ (5.2.4$) erstellten Kalibrationsmatrix entzerrt worden ist, was auch im Graphikfenster ersichtlich ist. Auch das Helligkeitsbild wird kalibriert, dies wird jedoch nicht explizit dargestellt.	
9$	"Falls ein neues Helligkeitsbild erstellt werden soll, bitte 'j' drücken...."
Für die Erstellung eines aktuellen Helligkeitsbilds ist die Taste 'j' zu betätigen (hierbei handelt es sich grundsätzlich um die gleichen Bedingungen wie bei der Erstellung eines Hintergrundbildes). Der einzige Unterschied zur Erstellung eines Hintergrundbildes ist, dass der **Koordinatengeber** in diesem Fall in den Arbeitsbereich gehalten werden sollte, um die maximale Helligkeit, die während der Kameraaufnahme auf Basis der momentanen Kameraeinstellungen besteht, gemessen werden kann. Für das Laden eines gespeicherten Helligkeitsbildes gelten hier die gleichen Voraussetzungen wie bei dem vorher beschriebenen Hintergrundbild. Die genaue Helligkeitsberechnung kann dem Kapitel 4.4 entnommen werden.	

Table 4.5: Konsolenausgabe5

10$	"Laserblobkoordinaten werden berechnet und ausgegeben, wenn ...."
Diese Routine ist dazu gedacht, den Positionierungslaserpunkt für die jeweils zugehörige Kamera mittig einzustellen (Genaueres siehe Kap. 2.2.2 [2]). Da jedoch einen gewisser Abstand zwischen der optischen Kameraachse und dem Laserstrahl existiert (Parameter LASER_TRACK_CAM_OFFSET_MM {92}), muss dieser bei der Berechnung berücksichtigt werden. Der Laserpunkt befindet sich aus diesem Grund nicht in der Mitte des Kamerabildausschnitts. Die Routine sucht in jedem Fall einen "Laserblob" im momentanen Bildausschnitt, wobei der erkannte "Blob", grün eingezeichnet, im Graphikfenster ausgegeben wird und die Abweichungen beider Richtungen, sowie die "Blobdaten" auf der Konsole ausgegeben werden (es ist nur erforderlich, das Positionierungslasergerät, das auf der das momentane Bild liefernden Kamera befestigt ist, einzuschalten). Um den Laserpunkt des Positionierungslasers einzustellen, wird dieser nun solange verschoben, bis die Abweichungen sich verringern, die für die waagrechten und senkrechten Richtungen auf der Konsole ausgegeben werden. Nach Bestätigung durch Betätigen einer beliebigen Taste wird eine neue Messung durchgeführt. Dieser Vorgang sollte so oft wiederholt werden, bis sich die Abweichungen in einem gewissen Toleranzbereich befinden (+/- 1 Pixel; Aufgrund der etwas ungenauen und umständlichen Justierungsmöglichkeit der Positionierungslaser ist dies aber schwer zu erreichen). Sobald keine weiteren Justierungen mehr vorgesehen sind, oder, wenn die Laserpositionierungen der Kameras schon in einem vorhergegangenen Durchlauf durchgeführt worden sind, kann der Vorgang durch Eingabe eines 'j' abgeschlossen werden. Um die Erkennung zu optimieren, ist es sinnvoll, während der Laserpositionserkennung die Zusatzbeleuchtung abzuschalten (wesentlich bessere Erkennung des "Laserblobs", keine fälschlich erkannten "Blobs"). Probleme bei der Erkennung des Laserpunktes der Trackingkamera können dadurch auftreten, dass, bedingt durch deren relativ grobe Auflösung, die im Bildausschnitt dargestellte Laserfläche zu klein ist (weniger als 2 Pixel, was der minimalen erkennbaren "Blobfläche" entspricht), um korrekt erkannt zu werden. Durch ein leichtes Verstellen des Laserfokus (siehe Positionierungslaser) kann die Fläche vergrößert und somit die Erkennung gewährleistet werden.	
10.1$	"Speichern der ermittelten Laserpositions- Abweichungen von der Bildmitte ..."
Dieser Vorgang schreibt die eingestellten Laserpositionierungsdaten direkt in die Initialisierungsdatei.	
11$	"Auswahl der Bilderkennungsbearbeitung (1-5) :..."
Diese Bildschirmmaske eröffnet dem Anwender die Möglichkeit, zwischen drei verschiedenen Bildbearbeitungsroutinen und einer Echtzeitbildbearbeitungsroutine (mit einer dazugehörigen "Benchmark" Rechnung) zu wählen.	
1...	Hierbei wird eine Bildbearbeitungstestroutine mit graphischer Ausgabe aller Bildbearbeitungsschritte gestartet, wobei die "Blob"-Kontur und der Feretdurchmesser jedes "Frames" eingeblendet wird. Alle anderen Bildbearbeitungsschritte werden einzeln je "Frame" ausgegeben, um die Darstellungszeit verlängern zu können. Die Wartezeit zwischen den einzelnen Bildern kann mit dem GRAPH_SLOW_TIME_MS 273 Parameter (siehe ANHANG Kap. 9.7.13 [2]) umgestellt werden.
2...	Diese Routine startet ausschließlich eine Bildbearbeitungstestroutine mit graphischer Ausgabe der "Blobsuche" und "Blobfindung". Zusätzlich werden hier auch einige Detailausgaben des festgestellten "Blobs" ausgegeben Auch hier kann die Anzeigedauer der einzelnen Bilder eingestellt werden, indem der Parameter GRAPH_FAST_TIME_MS 274 (siehe ANHANG Kap. 9.7.13 [2]) entsprechend geändert wird.

Table 4.6: Konsolenausgabe6

3...	Dieses Programm startet eine Bildbearbeitungstestroutine mit graphischer Ausgabe der für die Bewegungsberechnung notwendigen Bilder. Auch hier kann die Anzeigedauer der einzelnen Bilder durch entsprechende Änderung des Parameters "GRAPH_FAST_TIME_MS {274}" eingestellt werden.
4...	In diesem Fall wird die Ermittlung einer "Benchmark"-Rechnung gestartet, um die maximale Bildverarbeitungsrate zu berechnen. Im Rahmen dieser Routine sollte die Signalhand im Bild in gleicher Weise wie während des Echtbetriebs bewegt werden. Bei diesem Vorgang ist ein Klickgeräusch des "**Tracking Device**s" wahrzunehmen. Die Ursache dafür liegt darin, dass diese Routine zwar nicht in mechanische Maschinenbewegungen umgesetzt, jedoch "Blob"-Koordinaten an die Werkzeugmaschine geschickt werden und daher die Bremse gelöst wird. Da anschließend jedoch keine weiteren "Blob"-Koordinaten an das Gerät geschickt werden, wird die Bremse wieder geschlossen, was diese Klickgeräusche zur Folge hat. Bei starker Abweichung der gemessenen Bildverarbeitungsrate von der möglichen Rate der Echtzeitbildbearbeitungsroutine kann es bei der Bewegung des "**Tracking Device**s" zu einem Rütteln kommen. Zur Behebung dieses Problems kann ein neuer "Benchmarktestdurchgang" durchgeführt werden.
5...	Hierbei wird die eigentliche Echtzeitbildbearbeitungsroutine gestartet, um das "**Tracking Device**" zu steuern, das die Aufgabe hat, die Handbewegungen des Operators während der Werkstückbearbeitung im gleichen Maß und unmittelbar in Echtzeit nachzuvollziehen. Bei dieser Arbeit ist es sehr wichtig, dass die Hand mit dem **Koordinatengeber** immer im Kamerasichtfeld bleibt und nicht verdeckt wird, da dies sonst eine Fehlermeldung zur Folge hat (siehe Anhang Kap. 9.5 und Kap. 9.6 [2]). Um eine Kontrolle über die momentane Lage des **Koordinatengebers** im Fall einer Fehlermeldung wieder zu erlangen, wären die beiden Laser der Stereokameras im Betriebszustand des Systems einzuschalten. Bei zu großer Abweichung der Position der Laserpunkte von der Signalhand kann es zu Problemen mit der Stereobildberechnung kommen, sodass keine Koordinaten in y-Richtung des "Trumpfkoordinatensystems" der "Trumpfmaschine" weitergegeben werden können, was zu Unterbrechungen in der Höhenbewegung führt. Sollten Bewegungen außerhalb des Arbeitsbereiches durchgeführt werden, wird zuerst eine Warnung ausgegeben. Bei kompletter Entfernung des **Koordinatengebers** aus dem Abbildungsbereich der Trackingkamera erscheint eine Fehlermeldung der "Bloberkennung".

Table 4.7: Konsolenausgabe7

4.4 Bildbearbeitung

Als erster Schritt wird der Hintergrundlutpuffer "MilLutBackground" mit einer rampenförmigen Funktion gefüllt.

```
//LUT für Hintergrundberechnung füllen
    MgenLutRamp(MilLutBackground,ini_st_n_Pic_Lut_Startindex_1,
       ini_st_n_Pic_Lut_Startvalue_1,ini_st_n_Pic_Bgrtoleranz-1,
       ini_st_n_Pic_Lut_Endvalue_1);
```

```
MgenLutRamp(MilLutBackground,ini_st_n_Pic_Bgrtoleranz,
   ini_st_n_DigDepht-1,ini_st_n_DigDepht-1,ini_st_n_DigDepht-1);
```

Zu Beginn der Berechnung wird ein Hintergrundbild nach Bestätigung des Operators aufgenommen (dies muss nicht bei jedem Berechnungsdurchgang erfolgen, da der Hintergrund auch unverändert sein kann). Ein Hintergrundbild ist nichts Anderes, als das Bild der Trackingkamera, die den Hintergrund (ohne **Koordinatengeber**) abbildet.

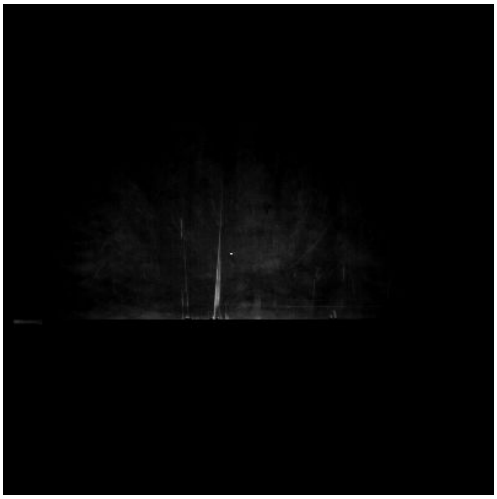


Figure 4.2: Hintergrundbild

Als zweites Bild kann auch ein **Helligkeitsbild** aufgenommen werden. Bei diesem Bild handelt es sich um das gleiche Bild wie das Hintergrundbild, aber diesmal wird auch der **Koordinatengeber** einbezogen. Mit Hilfe dieses Bildes werden die maximalen und die minimalen Pixelgrauwerte ausgelesen.

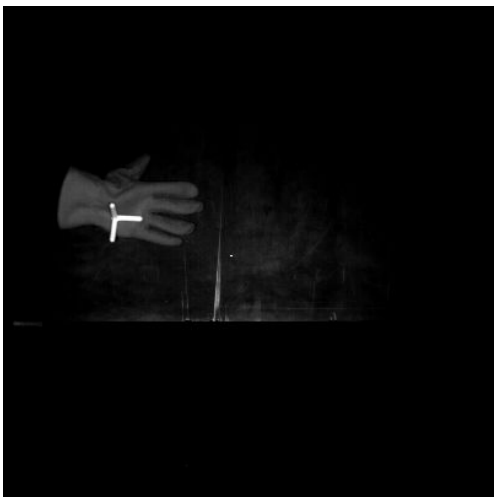


Figure 4.3: Helligkeitsbild

Um den Helligkeitsverlauf des Helligkeitsbildes ("MilImageIntensity") zu erhalten, wird der maximale und der minimale Grauwert des Bildes mittels "MimFindExtreme()" erhoben. Wenn der

kleinste Helligkeitswert kleiner ist als der Grenzwert ("ini_st_n_Pic_Intensity_Treshhold"), so wird der minimale im Bild gemessene Wert ("st_a_n_MinMaxval[0]") mit dem Standardwert gleichgesetzt. Sollte der kleinste Helligkeitswert größer oder gleich dem Grenzwert sein, so wird dieser um einen angenommenen Wert ("ini_st_n_Pic_Intensity_Dist_Equal") verkleinert.

```
//Helligkeitsanpassung
  MimFindExtreme(MilImageIntensity,MilMinMax,M_MAX_VALUE );
  MimGetResult(MilMinMax,M_VALUE,st_a_n_MinMaxval);

  if (st_a_n_MinMaxval[0]<ini_st_n_Pic_Intensity_Treshhold)
  {
    printf("Bild␣sehr␣dunkel,Kameraeinstellungen␣aendern,␣Beleuchtung␣
       hinzufuegen\n");
    st_a_n_MinMaxval[0]=ini_st_n_Pic_Intensity_Min_Val;
  }
  else
  {
    st_a_n_MinMaxval[0]-=ini_st_n_Pic_Intensity_Dist_Equal;
  }
```

Als Nächstes wird mittels der Funktion ("MimHistogramEqualize()") das **Helligkeitsbild** besser kontrastiert, um eine erleichterte "Blobsuche" zu bewirken. Mit der Funktion "MimLutMap()" wird die Verteilung des "MilLutIntensity" nun zuerst auf das Hintergrundbild ("MilImageBackground") und dann auf das Helligkeitsbild ("MilImageIntensity") gemappt.

```
    //Aufziehen der Bildhelligkeit auf das gesamte Bild und speichern
       in einen Lut speicher
    MimHistogramEqualize(MilImageIntensity,MilLutIntensity,M_UNIFORM,
       M_UNIFORM,st_a_n_MinMaxval[0],255);
    //mappen des Luts auf einen Bildspeicher
    MimLutMap(MilImageBackground,MilImageBackground,MilLutIntensity);
    MimLutMap(MilImageIntensity,MilImageIntensity,MilLutIntensity);
```

Mit "MdigProcess()" wird die Hauptberechnungsfunktion ("F_l_Trackingfunction"= Echtzeitberechnungsfunktion) gestartet und auch wieder gestoppt.

```
    MdigProcess(MilMikro1,MilImageMovChild,ini_st_n_DigProcBufferNr,
       M_START,M_DEFAULT ,F_l_Trackingfunction,&UserHookData);
      printf("Druecken␣Sie␣eine␣Taste,␣um␣die␣Berechnung␣zu␣beenden␣\n"
         );
      getch();
    MdigProcess(MilMikro1,MilImageMovChild,ini_st_n_DigProcBufferNr,
       M_STOP,M_DEFAULT,F_l_Trackingfunction,&UserHookData);
    MappTimer(M_TIMER_READ,&st_d_TimerCount);
```

Der LUT-Speicher "MilLutIntensity" wird auf den Speicher "UserHookDataPtr->MilImageMovProcChild" gemappt. Der mit Hilfe des Helligkeitsbildes erstellte "LUT" wird auf dem aktuellen Bild abgebildet. Dabei wird eine bereits erstellte Tabelle verwendet, über die Daten eines Bildes mittels Zuweisung in ein anderes Bild transferiert werden. Jedem Bildpunkt im alten Bild wird ein Zahlenwert im neuen Bild zugewiesen. Näheres siehe [4] ab Seite 567.

```
    //Mappen des Helligkeitsluts über das eben gegrabbte Bild (eventuell
       auf neuen Speicher mappen ist schneller)
    MimLutMap(UserHookDataPtr->MilImageMovProcChild,UserHookDataPtr->
       MilImageMovProcChild,UserHookDataPtr->MilLutIntensity);
```

Um ein **Bewegungsbild** zu erhalten, müssen zwei aufeinanderfolgende Bilder voneinander subtrahiert werden. Ein **Bewegungsbild** ist ein Bild, das entsteht, wenn zwei direkt hintereinander aufgenommene Kamerabilder übereinander gelegt und von einander abgezogen werden (wobei es keine negativen Werte gibt). Bei zu schneller Bewegung des **Koordinatengebers** kann es zu keiner Überschneidung der "Blobs" kommen. Ist die Geschwindigkeit hingegen niedriger und die Bildwiederholungsrate hoch, entsteht ein **Bewegungsbild** mit einem neuen "Blob".

```
MimArith(UserHookDataPtr->MilImageMov1Child,UserHookDataPtr->
    MilImageMov2Child,UserHookDataPtr->MilArithBufferMovFChild,
    M_SUB_ABS );
```

Um ein Bild zu erhalten, das den **Koordinatengeber** besonders gut hervorhebt, wird das Hintergrundbild vom aktuellen Bild abgezogen. Mit der Funktion "MimArith" entsteht durch eine absolute Subtraktion des **Hintergrundbildes** vom momentan aktuellen Bild ein Bild mit nur positiven Pixelhelligkeitswerten. Das bedeutet, dass jeder Grauwert eines Pixels (0-255) des aktuellen Bildes absolut vom Hintergrundbild subtrahiert wird. Das Ergebnis der Subtraktion ist immer positiv, da keine negativen Ergebnisse zugelassen werden, was mit dem Zusatz absolut bestimmt wird. Damit wird die Gefahr verringert, dass bei der "Bloberkennung" Fehler entstehen. Dadurch reduziert sich die Fehleranfälligkeit des Blobfindungsalgorithmuses, da keine Hintergrundfehler mehr auftreten, solange sich der Hintergrund nicht stark ändert. Das Ergebnis dieser mathematischen Operation ist vor allem dann gut anwendbar, wenn nur ein kleiner Ausschnitt des momentanen Bildes sich ändert. Dabei werden alle Bildpunkte, die denen im Hintergrund gleichen, schwarz. Aus diesem Grund wurde die Kamera auch oberhalb des Operators montiert, um eine Hintergrundänderung durch andere Personen oder Fremdobjekte so weit wie möglich zu reduzieren.

Figure 4.4: Hintergrund abgezogen

Mit "MimLutMap" wird ein LUT über das Bild gelegt werden, um den Kontrast zwischen den einzelnen Bildpunkten zu verbessern. "MimArith" wird noch einmal verwendet, um diesmal das aktuelle und das berechnete Bild bitweise UND zu verknüpfen.

```
//START HINTERGRUNDBERECHNUNG
//Bild vom Hintergrund subtrahieren
MimArith(UserHookDataPtr->MilImageBackgroundMovChild,UserHookDataPtr->
    MilImageMovProcChild,UserHookDataPtr->MilArithBufferMovChild,
```

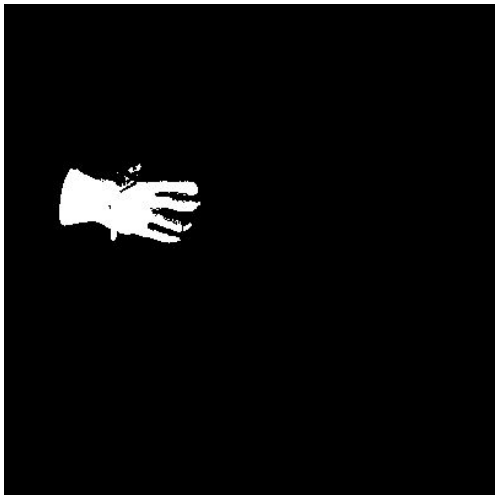


Figure 4.5: Lut gemappt

```
    M_SUB_ABS );
MimLutMap(UserHookDataPtr->MilArithBufferMovChild,UserHookDataPtr->
    MilLutImageBackgroundChild,UserHookDataPtr->MilLutBackground);

//Bitweise UND Verknüpfung
MimArith(UserHookDataPtr->MilImageMovProcChild,UserHookDataPtr->
    MilLutImageBackgroundChild,UserHookDataPtr->MilArithBufferMovChild,
    M_AND);
```

Um die Fehlerfreiheit zu erhöhen, wird mit "MimClip" gearbeitet. Diese Funktion bewirkt die Aufhellung sämtlicher hellen Pixel und eine Verdunklung der dunklen Pixel. Dies wird erreicht, indem die im Bild vorhandenen Pixelwerte ab einer bestimmten Helligkeit als weiß und dunkle Pixelwerte als schwarz definiert werden. Damit steigt der Kontrast des Bildes, womit auch die "Bloberkennung" besser funktioniert.

```
//Bild clippen mit den berechneten Clippinggrenzen (nach clippen der
    Grenzen eventuell mit binärem Puffer weiterrechnen, schneller!!)
    MimClip(UserHookDataPtr->MilArithBufferMovChild,UserHookDataPtr->
        MilArithBufferMovChild,M_OUT_RANGE,ini_st_n_Pic_Minclip,
        ini_st_n_Pic_Maxclip,0,255);
```

Die Funktion "MimDilate" verringert die "Blobs" um eine änderbare Anzahl an Pixel und die Funktion "MimErode" erhöht diese Zahl. Genauer gesagt rechnet "MimDilate" Pixel am Umfang der einzelnen "Blobs" hinzu, wobei unverbundene "Blobanteile" miteinander verschmelzen. Danach werden dem neu enstandenen "Blob" die gleiche Anzahl Pixel vom Umfang abgezogen, wodurch eine neue Kontur entsteht. Dieser Vorgang dient dazu, die von der Bildverarbeitungssoftware z.B. aufgrund von dunklen Bildpunkten durch Bildschatten in den "Blobs" fälschlicherweise auf mehrere "Blobs" aufgeteilten "Blob" durch diese Funktionen soweit wieder miteinander zu verbinden, dass die Bildbearbeitungssoftware wieder den ursprünglichen "Blob" erkennt .

```
    //Fehlerstellen ausmärzen (Parameter in Abhängigkeit der Auflösung
        ändern!!! Rechenaufwand!
    MimDilate(UserHookDataPtr->MilArithBufferMovChild,UserHookDataPtr->
        MilArithBackBufferMovChild,ini_st_n_Pic_Iteration,M_GRAYSCALE);
```

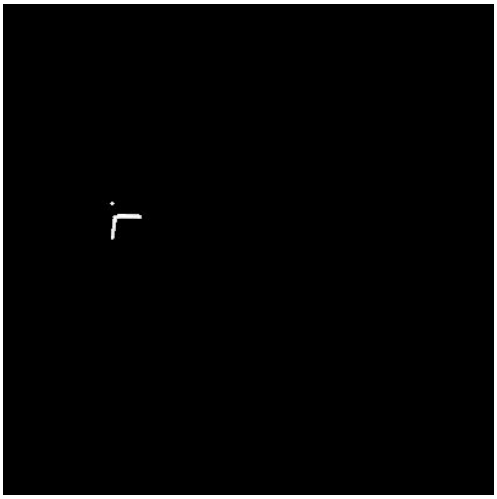


Figure 4.6: clipping des Bildes

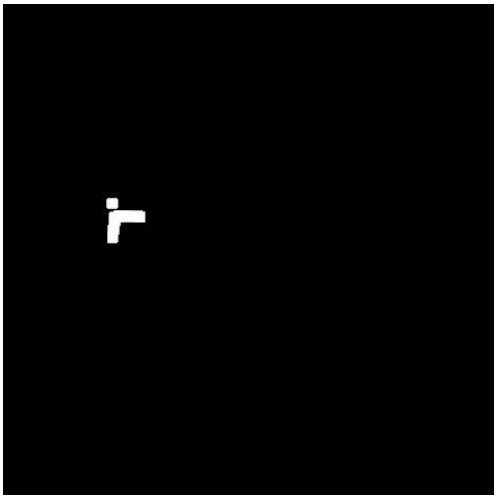


Figure 4.7: erode

```
MimErode(UserHookDataPtr->MilArithBackBufferMovChild,UserHookDataPtr
    ->MilArithBufferMovChild,ini_st_n_Pic_Iteration,M_GRAYSCALE);
```

Mit "MblobSelectFeature" werden die Kriterien ausgewählt, die nötig sind, um "Blobs" detektieren zu können. Beim ggst. Forschungsprojekt werden drei "Blobs" verwendet. Zuerst werden "Blobs" in einem bestimmten Flächengrößenbereich gesucht. Dazu muss das Kriterium "M_AREA" in die Berechnung Eingang finden. Das bedeutet, dass alle "Blobs", die nicht in den gewünschten Bereich fallen, ausgesondert werden. Dasselbe gilt auch für die anderen beiden Parameter "M_FERET_MAX_DIAMETER" und "M_MOMENT_CENTRAL_X1_Y1". Der erste Parameter sortiert die gefundenen "Blobs" nach dem maximal in einen "Blob" einschreibbaren Radius, dem sogenannten "Feretdurchmesser" (nähere Informationen können sie auch in [2] nachschlagen). Der zweite Parameter berechnet das Moment erster Ordnung, ausgehend vom Schwerpunkt des "Blobs". Dieser Parameter ist unabhängig von der Lage des "Blobs" im Bild.

```
//START BLOBERKENNUNG
//Auswahl der Blobs
```

```
MblobSelectFeature(UserHookDataPtr->MilBlobList, M_AREA);
MblobSelectFeature(UserHookDataPtr->MilBlobList, M_FERET_MAX_DIAMETER );
MblobSelectFeature(UserHookDataPtr->MilBlobList, M_MOMENT_CENTRAL_X1_Y1 )
    ;
```

Mit der Funktion "MblobCalculate()" werden die "Blobs" berechnet. Für die erkannten "Blobs" werden alle gewünschten Merkmale berechnet (siehe auch "MBlobselectFeature") und anschließend aussortiert. Sämtliche, nicht dem Grenzwert entsprechende "Blobs" werden aus dem "Blobspeicher" gelöscht (diese "Blobs" werden mit Hilfe der Funktion "MBlobselect()" aussortiert). Dieser Vorgang wird solange wiederholt, bis nur mehr ein einziger erkannter Blob übrigbleibt. Wenn nur noch zwei sehr ähnliche "Blobs" vorhanden sind, bleibt kein "Blob" übrig, da beide von dem Algorithmus aussortiert werden. Nach jedem Durchlauf werden die Grenzen enger gesetzt, um die restlichen "Blobs" zu selektieren. Dieser Iterative Ansatz wird insgesamt zweimal wiederholt, einmal für das Hintergrundbild und einmal für das **Bewegungsbild**. Diese zwei unterschiedlich entstandenen Bilder ergänzen sich.

```
//Berechnung der Blobs
MblobCalculate(UserHookDataPtr->MilArithBufferMovChild,
    UserHookDataPtr->MilArithBackBufferMovChild,UserHookDataPtr->
    MilBlobList,UserHookDataPtr->MilBlobBuffer);

//Als Flag beim speichern der gesamt gefundenen Blobanzahl (für
    Einstellung der Parameter)
MblobGetNumber(UserHookDataPtr->MilBlobBuffer,&UserHookDataPtr->
    a_l_BlobNumber[n_m]);
UserHookDataPtr->a_l_BlobNumber[n_m]=0;

st_d_Min_Blob_Area_chg=st_d_Min_Blob_Area;
st_d_Max_Blob_Area_chg=st_d_Max_Blob_Area;
st_d_Min_Feret_r_chg=st_n_Min_Feret_r;
st_d_Max_Feret_r_chg=st_n_Max_Feret_r;

do
{
  MblobSelect(UserHookDataPtr->MilBlobBuffer, M_INCLUDE,
      M_MOMENT_CENTRAL_X1_Y1 , M_IN_RANGE  ,
      ini_st_d_Pic_Blob_Max_Mom_Low, ini_st_d_Pic_Blob_Max_Mom_High);
  MblobSelect(UserHookDataPtr->MilBlobBuffer, M_INCLUDE,
      M_MOMENT_CENTRAL_X1_Y1 , M_IN_RANGE  ,
      ini_st_d_Pic_Blob_Min_Mom_Low, ini_st_d_Pic_Blob_Min_Mom_High);
  MblobSelect(UserHookDataPtr->MilBlobBuffer, M_DELETE, M_AREA ,
      M_OUT_RANGE ,st_d_Min_Blob_Area_chg, st_d_Max_Blob_Area_chg);
  MblobSelect(UserHookDataPtr->MilBlobBuffer, M_DELETE,
      M_FERET_MAX_DIAMETER , M_OUT_RANGE , int(st_d_Min_Feret_r_chg),
       int(st_d_Max_Feret_r_chg));
```

4.5 Bildverarbeitung

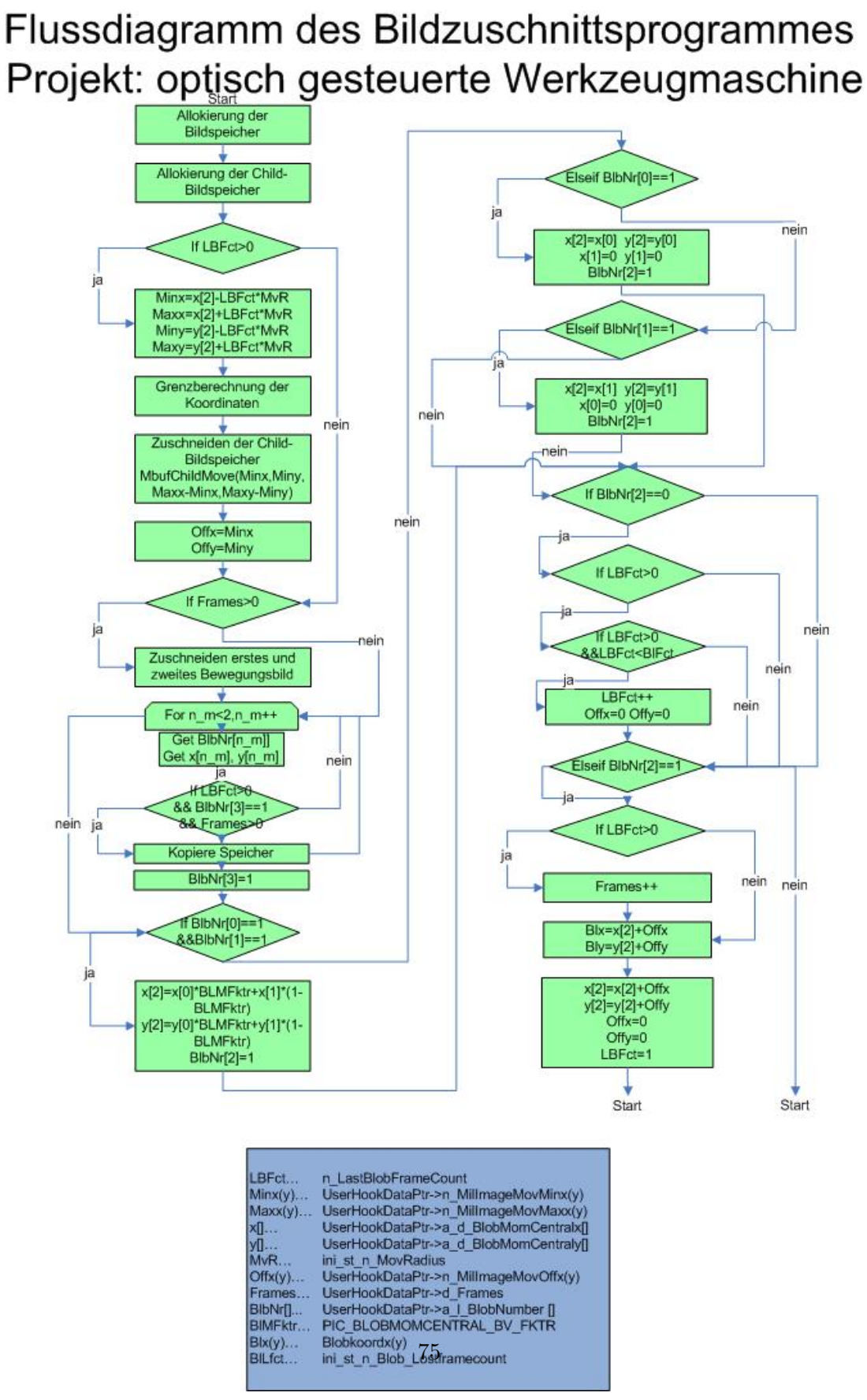


Figure 4.8: Flussdiagramm

Am Anfang des Programmes werden zuerst die Speicheranteile allokiert und bereitgestellt. Der nächste Schritt stellt die Erstellung von sogenannten "Child-Speichern" dar. Diese Speicher werden über den jeweiligen normalen Speicher gelegt, die ohne Änderung des Hauptspeichers schnell verschoben oder zugeschnitten werden können.

In Bild 4.9 ist das Abbild des **Koordinatengebers** dargestellt (weißer Kreis mit blauem Mittelpunkt).

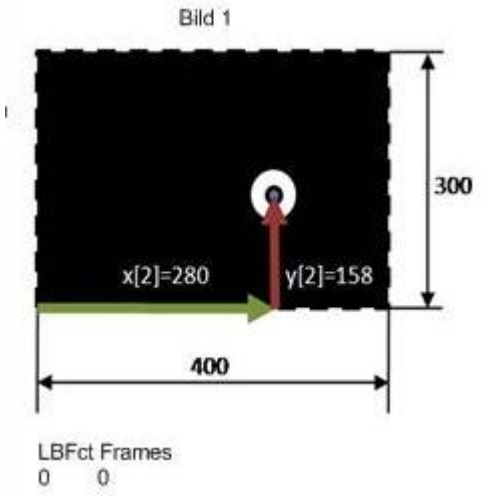


Figure 4.9: Ausgangslage des Bildzuschnittsprogramms

Die Position des **Koordinatengebers** wird hierbei willkürlich angenommen. Das gezeigte Kamerabild weist eine Größe von 400 x 400 (400 x 300 sind auf dem Bild dargestellt, was jedoch keine Auswirkungen auf die Erklährungen hat) Pixel auf, wobei der Rand des Gesamtbildes durch eine strichlierte Linie illustriert wird. Dies stellt also den Zustand nach Beginn eines Schleifendurchlaufes dar, bei dem die Suche nach dem **Koordinatengeber** noch im gesamten Bild durchgeführt wird. Da dies jedoch aufgrund der dafür benötigten erhöhten Rechenleistung gegenüber der Durchsuchung eines kleineren Bildausschnitts mehr Zeit in Anspruch nimmt, verkleinert dieses Programm diesen Bildausschnitt in Abhängigkeit zum Ausmaß des bereits erkannten **Koordinatengebers**.

Die Abfrage, ob "n_LastBlobFrameCount" größer als Null ist, ist nur im ersten Durchlauf des Programms relevant, da diese Variable sonst 1 ist. Die Variable dient der Vermeidung des Bildzuschnitts, solange noch keine Position eines "Blobs" bekannt ist. "UserHookDataPtr->n_MilimageMovMinx", "UserHookDataPtr->n_MilimageMovMiny", "UserHookDataPtr->n_MilimageMovMaxx" und "UserHookDataPtr->n_MilimageMovMaxy" werden zur Berechnung der Eckpunkte des ausgeschnittenen Bildes herangezogen. "ini_str_n_MovRadius" gibt an, um wie viele Pixel das Bild senkrecht und horizontal vergrößert wird, wenn kein "Blob" gefunden wird (Bild 4.12).

Dieser Vorgang wird solange aufrecht erhalten, bis ein neues Bild des "Blobsuchalgorithmus" gefunden wurde. Die Grenzüberschreitung des Bildspeichers wird mittels Grenzberechnung behoben. Wenn jedoch der äussere Rand nach vier- oder fünfmaliger Erweiterung erreicht ist, beginnt der Suchalgorithmus wieder den gesamten Bildbereich nach "Blobs" abzusuchen (abhängig von der Funktion "ini_str_n_MovRadius" (gibt den Umfang das Suchbereiches an) und dem Bildbereich des von der Bildbearbeitungssoftware zuletzt erkannten "Blobs"). Wenn die Funktion

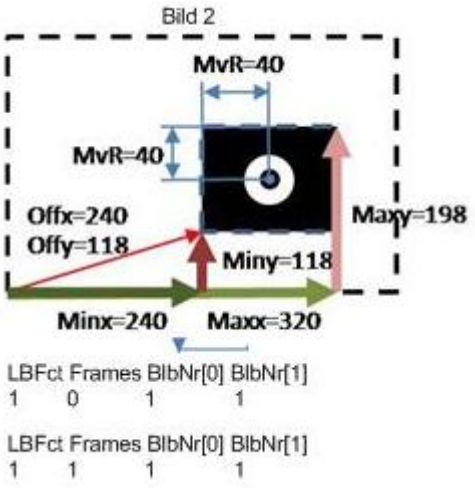


Figure 4.10: Erster Bildzuschnitt

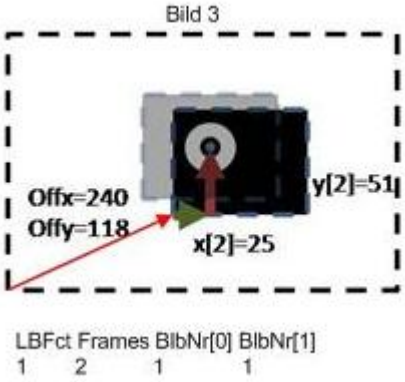


Figure 4.11: Neue Suche des Blobs mit einem kleinen Bildausschnitt

"UserHookDataPtr->d_Frames" größer als Null ist, was frühestens nach dem zweiten Durchlauf dieses Programmteils der Fall sein kann, werden die beiden Bewegungsbilder zugeschnitten. Ein Bewegungsbild ist ein unter Standardbedingungen aufgenommenes Bild, wobei jedoch zwei aufeinander folgende Bilder auch einen "Blob" als Ergebnis aufweisen, sofern die beiden Bilder nicht zu sehr differieren und keine Überschneidungen mehr auftreten.

Die folgende "For"-Schleife stellt den eigentlichen "Blobsuchprogrammteil" dar. Hier gibt es zwei Durchläufe, einen das **Hintergrundbild** betreffend, der andere das **Bewegungsbild**. Daraus erhält man, im besten Falle, bei jedem Durchgang der Schleife "Blobkoordinaten" und die "Blobnummer" ("UserHookDataPtr->a_l_BlobNumber[]"). Die Koordinatenspeicher für x ("UserHookDataPtr->a_d_BlobMomCentralx[]") und y ("UserHookDataPtr->a_d_BlobMomCentraly[]") werden mit den Koordinatendaten aus dem "Blobsuchalgorithmus" gespeist. Dabei erhält "UserHookDataPtr->a_d_BlobMomCentralx[0]" die Koordinaten des Hintergrundsuchprogramms und "UserHookDataPtr->a_d_BlobMomCentralx[1]" die Koordinaten des Bewegungsbildprogramms. Diese Koordinaten werden immer ausgehend vom **Bildkoordinatensystem** angegeben. Das bedeutet, dass das Koordinatensystem nicht vom Koordinatennullpunkt des

"Parentspeichers" ("Parentspeicher" ist der Ausgangsbildspeicher, aus dem dann "Childspeicher" herausgeschnitten werden) ausgeht, sondern sich in Abhängigkeit des ausgewählten Bildzuschnittes in dessen Nullpunkt befindet. Somit läuft mit der "For"-Schleife das erste und zweite Bild durch den gleichen "Blobsuchalgorithmus", jedoch mit unterschiedlich gewonnenen Bildern.

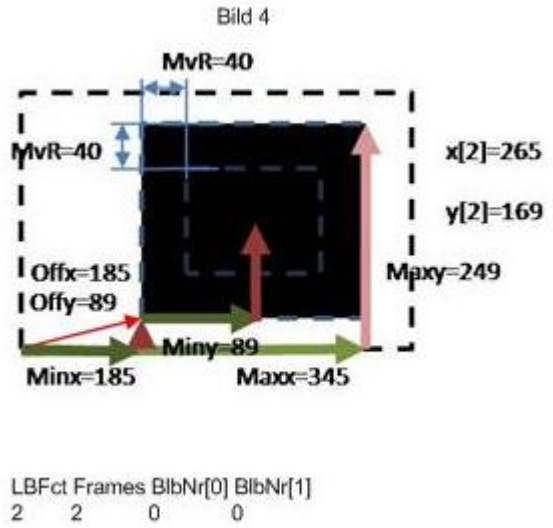


Figure 4.12: Erweiterung der Blobsuchfläche um den Blobsuchradius

"UserHookDataPtr->a_l_BlobNumber[]" ist eine Speichervariable, die abspeichert, ob der "Blobsuchalgorithmus" erfolgreich abgeschlossen wurde. Erfolgt dies bei der Berechnung des "Hintergrundblobs", entspricht "UserHookDataPtr->a_l_BlobNumber[0]" gleich 1. Dasselbe gilt für den Durchgang der Bewegungsbildberechnung für "UserHookDataPtr->a_l_BlobNumber[1]". Nachdem die "For"-Schleife fertig abgearbeitet wurde, kommt es nun zur Zusammenfassung der "Blob" - Koordinaten. Dies geschieht in Abhängigkeit von den entdeckten "Blobs". Bei Feststellung eines "Blobs" während der zwei Durchläufe werden die Koordinaten über einen "PIC_BLOBMOMCENTRAL_BV_FKTR" zusammengerechnet. Mit diesem Faktor können die Anteile eines Verfahrens höher gewichtet werden, als die Anteile des anderen Durchlaufs. Bei einem Faktor von 0,33 stammt zum Beispiel ein Drittel vom "Hintergrundblobsuchverfahren", der Rest entfällt auf das Bewegungsbildverfahren. Wenn bei einem Verfahren kein "Blob" entdeckt wird, werden nur die vorhandenen Koordinaten des jeweils anderen Verfahrens herangezogen. Wenn "UserHookDataPtr->a_l_BlobNumber[3]" gleich 1 ist, "UserHookDataPtr->d_Frames" und "n_LastBlobFrameCount" größer 1 ist, dann wird der **Bewegungsbild**speicher statt des Hintergrundspeichers zur "Blobberechnung" herangezogen. Wenn "UserHookDataPtr->a_l_BlobNumber[2]" gleich Null , "n_LastBlobFrameCount" größer Null und "n_LastBlobFrameCount" kleiner als "ini_st_n_Blob_Lostframecount" ist, dann wird "n_LastBlobFrameCount" um eins erhöht. Dadurch wird "n_LastBlobFrameCount" (Variable zur Speicherung der Anzahl der Durchläufe der Funktion ohne entdecktem "Blob") solange zahlenmäßig akkumuliert, bis wieder ein "Blob" festgestellt und "n_LastBlobFrameCount" auf Null zurückgesetzt wird. Die beiden Offset-Variablen werden beide auf Null gesetzt. Wenn "UserHookDataPtr->a_l_BlobNumber[2]" gleich eins ist, was bedeutet, dass ein "Blob" detektiert wurde und "n_LastBlobFrameCount" größer Null ist, dann wird die Variable "UserHookDataPtr->d_Frames" um eins erhöht. Diese Zahl gibt die Anzahl der Durchläufe an, in denen ein "Blob" festgestellt wurde. Um die Koordinaten für den nächsten Schleifendurchlauf zu erhalten, werden diese um den "Offset" der jeweiligen Koordinatenrichtung

(x oder y) aufsummiert. Danach werden die Offsetwerte selbst auf Null gesetzt. Die Variable "n_LastBlobFrameCount" wird auf eins umgeschrieben, die nur beim ersten Programmdurchlauf den Wert Null hat. Nach diesem letzten Schritt wird das Programm neu ausgeführt. Wie in Bild 4.11 ersichtlich, wurde ein neuer "Blob" entdeckt. Der neue Koordinatenursprung liegt jetzt am Eckpunkt des zugeschnittenen Bildes, weswegen die Verwendung der Offsetvariablen für die Speicherung dieser Koordinatenverschiebung notwendig werden. Transparent gezeichnet ist der neu zugeschnittene Bildteil, wobei die Funktion "ini_str_n_MovRadius" den Abstand zu den neu gefundenen Koordinaten darstellt.

4.6 Entzerrungsroutine

Mit zunehmendem Abstand zwischen der Kamera und dem **Arbeitsbereich** nimmt auch der Bildausschnitt an Größe zu. Diese Bildveränderung erfolgt nicht linear, weswegen die Durchführung einer Bildentzerrung erforderlich wird. Zuerst werden die Veränderungen der einzelnen x- und y- Werte in Richtung Bildmitte neu berechnet (Näheres siehe Kapitel 4.7) und die halbe Bildauflösung infolgedessen vom Pixelwert abgezogen. Um die Darstellung der vier Funktionen im gleichen Quadranten zu erreichen, werden die Radien absolut angegeben.

$$x_{alt} = x_{Bild} - \frac{x_{Bildaufloesung}}{2}$$

$$y_{alt} = y_{Bild} - \frac{y_{Bildaufloesung}}{2}$$

$$y = \sqrt{y_{alt}^2 + x_{alt}^2}$$

Dazu werden mit Hilfe eines **Entzerrungsgitters** Kamerabilder in verschiedenen Abständen (Kamera- Gitter) gemessen. Beim **Entzerrungsgitter** sind fünf Punkte gegeben, wobei die äußeren vier Punkte zur Berechnung der Entzerrungsfunktion herangezogen werden und der einzelne Punkt in der Mitte des Gitters zur Korrektur der Lage in der Mitte dient (es sind auch andere Formen möglich, z. B. eine Anordnung mit einer größeren Punkteanzahl. Es muss die Bildbearbeitungssoftware hierbei jedoch dementsprechend auf die neue Konstellation adaptiert werden). Wenn das **Entzerrungsgitter** nicht mittig befestigt ist, kann dieser Fehler daher mit dem Punkt, der ja in der Mitte des Gitters liegt, korrigiert werden. Der Abstand zwischen den einzelnen Bildern sollte ca. 10 cm betragen (in der Tiefe=z-Richtung). Dabei ist es wichtig, dass die Aufnahmen des **Entzerrungsgitters** den gesamten Arbeitsbereich abdecken (beim geringsten Gitterabstand sollte das Entzerrungsgitter im gesamten Kamerabild zu sehen sein, beim weitesten Gitterabstand sollte es außerhalb der unteren Bildebene liegen). Nachdem sämtliche Punkte gemessen und in der Datei "Blobkoordinaten.dat" abgespeichert sind, kann mit der Berechnung der Entzerrungsfunktionen begonnen werden. Hierbei wird eine Funktion errechnet, deren Verlauf sich soweit wie möglich den Koordinaten der gemessenen Punkte annähert. Die diesbezügliche Funktion lautet "

$$y = a * z^b \tag{4.1}$$

". Hierbei entsprechen "a" und "b" den beiden Unbekannten, die für jede neue Positionierung des **Arbeitsbereichs** berechnet werden müssen. Die Berechnung erfolgt nach einem "Least Squares-Algorithmus" von "Numerical Recipes" (einer Internetseite für numerische Berechnungen, diese werden als Funktionen in Programmiersprache zur Verfügung gestellt). Nachdem die Unbekannten berechnet sind, kann die notwendige Entzerrung mit Hilfe dieser ermittelten Werte

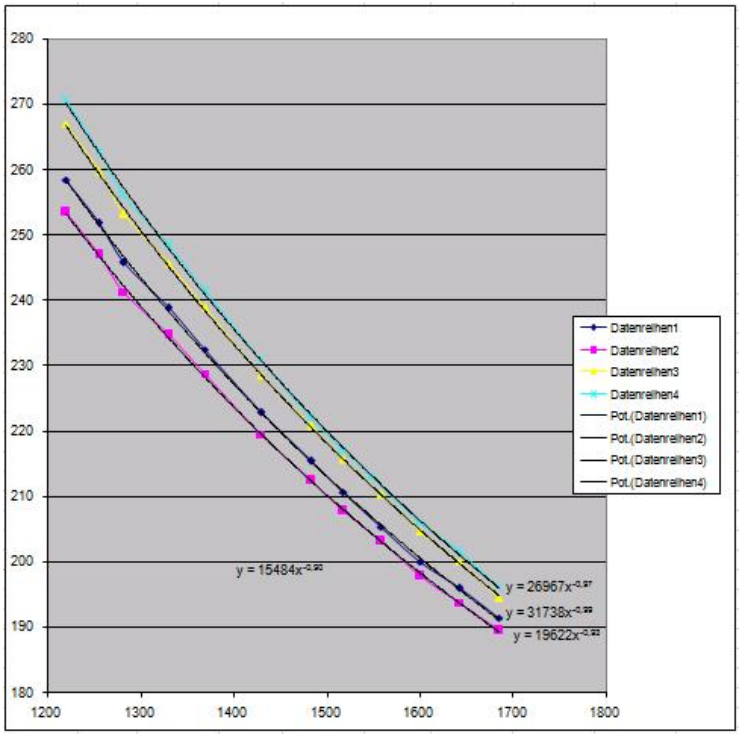


Figure 4.13: Verzerrungsdiagramm x-Achse mm in z- Richtung, y-Achse Pixel

in Abhängigkeit von der Position des **Koordinatengebers** im **Arbeitsbereich** in z-Richtung berechnet werden.

4.6.1 Kalibrationsroutine, Entzerrungsroutine

Diese Funktion kalibriert den Kamerabildausschnitt in dem mittels einer vordefinierten Kalibrationsmatrix die Kameraverzerrung heraus berechnet wird. Zu diesem Zweck ist es notwendig, ein Kalibrationsbild herzustellen (Die Matrix besteht dabei aus schwarzen Punkten auf einem weißen Hintergrund; die Abmessungstoleranzen sind in [4] angegeben).

```
//Hauptentzerrungsroutine
int C_Perspective::m_F_Perspectiven(void MPTYPE *Str_Perspk_HookDataPtr)
```

Ob eine neue Kalibration vorgenommen werden soll, wird dem Anwender bei jedem Bildbearbeitungsprogrammsdurchlauf zur Wahl gestellt. Sollte keine neue Kalibration erfolgen, wird eine alte Kalibration mit "McalRestore()" geladen (wenn diese vorhanden und schon in einem früheren Durchgang erstellt wurde).

```
printf("Kalibrieren␣des␣Bildes␣(j)␣oder␣alte␣Kalibrationsmatrix␣aufrufen␣
    (n)?\n");

  if ('j'==getch())
  {
    n_Calibration=1;
  }
```

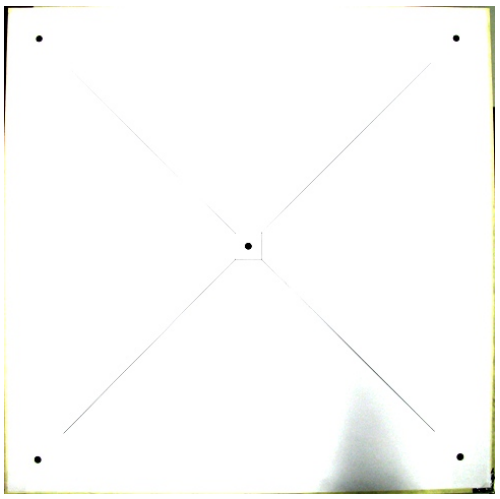


Figure 4.14: Entzerrungsgitter1

```
else //wenn nicht neue Kalibrationsmatrix erstellt werden soll, schauen
    , ob alte vorhanden, wenn ja, laden
{
  if (!McalRestore(FILE_CAL_PERSPEC_MIL1,Str_Perspk_UserHookDataPtr->
      MilSystem ,M_DEFAULT ,&Str_Perspk_UserHookDataPtr->MilCalibration
      ))
  {
    baerr.m_F_n_Err(ERR_PERSPEC_CAL_LOAD,0,__FILE__,__LINE__,1);
    exit(1);
  }
}
```

Mit "MdigGrab()" wird ein neues Kamerabild aufgenommen, um die Kalibrationsmatrix im Bild exakt zu positionieren. Die Funktion "MimClip" dient dazu, den Kontrast des Bildes zu verbessern, indem helle Pixelwerte noch mehr erhellt werden. Die "MbufCopy()" ist nur zur graphischen Ausgabe des Bildes.

```
//Grabben eines neuen Bildes
MdigGrab(Str_Perspk_UserHookDataPtr->MilMikro1,ModifiedBufferId);

//Bild clippen mit den berechneten Clippinggrenzen (um die Kontraste zu
    verstärken)
MimClip(ModifiedBufferId,ModifiedBufferIdCal,M_OUT_RANGE,w_Clipmin,
   w_Clipmax,0,255);
MbufCopy(ModifiedBufferIdCal,Str_Perspk_UserHookDataPtr->MilImageDisp);
//printf("00 \n");
//getch();
```

Um die Grauwertbilder in "Schwarz-Weiß-Bilder" umzurechnen, wird die Funktion "MimBinarize()" verwendet. Die Trennung zwischen schwarz und weiß erfolgt über einen Trennungslevel.

```
//Bildbearbeitung binärisieren
  MimBinarize(ModifiedBufferIdCal,Str_Perspk_UserHookDataPtr->
     MilImageMovProc,M_LESS_OR_EQUAL,w_Treshhold,M_NULL);
```

Zur weiteren Verarbeitung wird das Bild negiert (invertiert).

```
//Bild negiert dazurechnen
  MimArith(Str_Perspk_UserHookDataPtr->MilImageMovProc,M_NULL,
     Str_Perspk_UserHookDataPtr->MilImageMovProc,M_NOT);
```

Zur Kalibrationsberechnung mit der Funktion "McalGrid()" muss der x- und y- Abstand, sowie die Anzahl der schwarzen Punkte in die x- und y- Achse des **Kalibrationsgitters** eingegeben werden. Die Routine "McalSave()" speichert diese Daten.

```
//Kalibrierung des Bildes mit Kalibrierungsgitter als Motiv
    McalGrid(Str_Perspk_UserHookDataPtr->MilCalibration,
       Str_Perspk_UserHookDataPtr->MilImageMovProc,
       ini_d_Cal_Calc_Track_Grid_Offset_x_Pix,
       ini_d_Cal_Calc_Track_Grid_Offset_y_Pix,CAL_GRID_OFFSET_Z,long(
       ini_d_Cal_Calc_Track_Grid_Row_Nr),long(
       ini_d_Cal_Calc_Track_Grid_Col_Nr),
       ini_d_Cal_Calc_Track_Grid_Row_Space_Pix,
       ini_d_Cal_Calc_Track_Grid_Col_Space_Pix,M_LINEAR_INTERPOLATION ,
       M_CIRCLE_GRID);
    McalSave(FILE_CAL_PERSPEC_MIL1,Str_Perspk_UserHookDataPtr->
       MilCalibration,M_DEFAULT);
```

Nachdem die Kalibration beendet ist, können die **Entzerrungsgitter** für die Durchführung der Entzerrung vermessen werden. Zu diesem Zweck wird die Entzerrungsmatrix in den Kamerabildbereich gelegt und justiert (siehe Kalibrationsmatrix). Die Entzerrungsmatrix umfasst im Gegensatz zur Kalibrationsmatrix wesentlich weniger Vermessungspunkte. Das **Standardentzerrungsgitter** weist 5 Punkte auf, 4 davon in jeder Ecke und ein Punkt in der Mitte. Diese Vorgabe kann allerdings auch verändert werden, indem mehr Punkte in das Gitter eingezeichnet werden. Dabei muss das Programm über die Art der Änderung und die neuen Verhältnisse benachrichtigt werden.
Nachdem das Bild erneut invertiert und die definierten "Blobeigenschaften" für die "Blobsuche" ausgewählt ist (Fläche, Feretdurchmesser ...), werden die entdeckten "Blobs" graphisch auf dem Bildschirm dargestellt. Die Darstellung zeigt eventuelle Fehler bei der Identifizierung der "Blobs" auf. Im Fall einer vollständigen Erkennung der "Blobs" werden diese in der Datei "Blobkoordinaten.dat" abgespeichert. Danach wird das Gitter einen Schritt weiter entlang der Kameraachse in Gegenrichtung von der Kamera wegbewegt, um ein neues Bild aufzunehmen zu können. Dieser Vorgang wird bis zur vollständigen Bearbeitung des gesamten **Arbeitsbereichs** wiederholt (das bedeutet, dass im gesamten Arbeitsbereich Entzerrungsbilder aufgenommen werden). Die Abstände der **Entzerrungsgitter**lagen wären dabei möglichst konstant zu halten.

4.7 Entzerrungsberechnungsroutine

Hierbei handelt es sich um eine Routine zur Berechnung der Entzerrung, in der auch die Ergebnisse der Entzerrungsroutine Eingang finden.

```
//Berechnung der Funktion für die Bildentzerrung (Berechnung der
   Approximationsfunktion START)
int C_Perspective::m_F_n_pot_Fkt_Approx(char *File_input)
```

Mit der Funktion "m_F_n_Testequal()" kann ein Berechnungstestlauf vorgenommen werden. Dazu werden Funktionswerte mittels eines Zufallsgenerators berechnet, die geringfügig von der jeweiligen zu berechnenden Funktion abweichen. Ziel dieses Testlaufs ist die Berechnung einer Funktion, die den gegebenen Funktionswerten im Durchschnitt am Nächsten kommt.

```
//Testlauf der Berechnung mit einer verrauschten Funktion (Random)
printf("Wollen␣sie␣vor␣der␣Berechnung␣einen␣Testlauf␣der␣Berechnung␣
    durchfuehren?\n");
if(getch()=='j')
{
  m_F_n_Testequal();
}
```

Die Funktion "m_F_n_AreaFeretequal()" berechnet die Flächen- und Feretfunktion mit einem "Least Squares Algorithmus".

```
//Berechnung der Flächengrenzanpassungsfunktion
int C_Perspective::m_F_n_AreaFerretequal(char * Filenamepar,double *
    p_d_Blobarea,double * p_d_Blobferet)
```

Die Bezeichnung des Berechnungsalgorithmusses für die "Least Squares" Berechnung heißt "mrqmin()". Den genauen Berechnungsvorgang kann man sich auch bei "numerical recipes[1]" im Internet durchlesen.

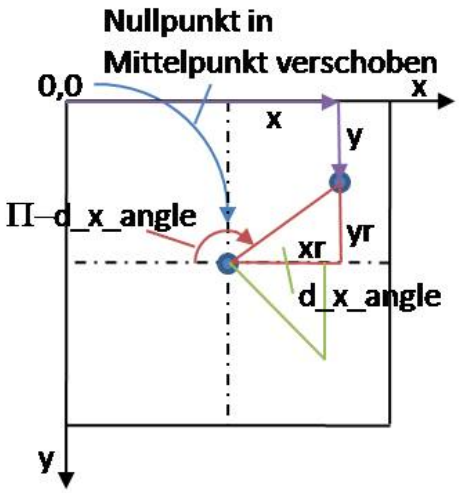


Figure 4.15: Quadrantenberechnung

Um ein zylindrisches System zu erhalten, wird das Koordinatensystem zuerst vom Bildnullpunkt (linke obere Ecke, siehe Bild) ausgehend auf die Mittelachse verschoben. Die Koordinatenbezeichnung wird von "x" zu "xr" und "y" zu "yr" abgeändert.

```
//Iterationsschleife für Funktionsberechnung (Durchlauf so oft wie
    Funktionen approximiert werden, ohne Mittelachse)
for (int m_w_bl=0;m_w_bl<(PERSPEC_MINBLOBNR*(PERSPEC_FKTDIV_LENGTH+1));
    m_w_bl++)
{
```

[1] http://www.nr.com/

```
  if(m_w_bl<(1+(PERSPEC_FKTDIV_LENGTH+1)*n_count) && m_w_bl!=(((
      PERSPEC_MINBLOBNR-1)/2)*(PERSPEC_FKTDIV_LENGTH+1)))
  {
    for(int i=0;i<NPT;i++)
    {
    //Umrechnen der gemessenen Blobkoordinaten, drehen um die optische
       Kameraachse (Radiusberechnung)
    //Abstandsberechnung von der Mittelachse
          xr[i]=m_pp_d_Fktdataarray[m_w_bl-((PERSPEC_FKTDIV_LENGTH)*
             n_count)][i]-m_ini_n_Cam_Track_Width_Pixel/2;
          yr[i]=m_pp_d_Fktdataarray[m_w_bl-((PERSPEC_FKTDIV_LENGTH)*
             n_count)][m_w_Datasets+i]-m_ini_n_Cam_Track_Width_Pixel/2;
             //bei beiden Koordinaten Width genommen , da eventuelle
             virtuelle Icons Auflösungsverhältnis ändern würden
```

Der Winkel des radialen Vektors wird zunächst mit absoluten Werten, unabhängig vom Vorzeichen der Zahlen, berechnet. In weiterer Folge wird dieser Winkel mit der Funktion "m_F_n_Quadr()" in den erforderlichen Quadranten umgerechnet.

```
//Berechnung des Winkels des transformierten Koordinatenvektors (in rad)
d_x_yangle=atan(abs(yr[i])/abs(xr[i]));//Berechnet den Winkel schon im
   richtigen Quadranten
//Berechnung des gemittelten Winkels aller Vektorenwinkel, die der
   gleichen Entzerrungsfunktion angehören
m_F_n_Quadr(&xr[i],&yr[i],&d_x_yangle);
//Aufsummierung aller Winkel einer Funktion
m_x_ysqangle[m_w_bl-((PERSPEC_FKTDIV_LENGTH)*n_count)]+=d_x_yangle;
```

Mit Hilfe der Variable "m_x_ysqangle()" werden alle aufsummierten Winkel gespeichert (diese müssen miteinander ident sein).

```
//Füllen der Arrays zur Least Squares Berechnung mit den gemessenen und
   umgerechneten Blobkoordinaten
x[i]=m_d_Xposition[i];
y[i]=sqrt(xr[i]*xr[i]+yr[i]*yr[i]);
m_x_yrad[m_w_bl-((PERSPEC_FKTDIV_LENGTH)*n_count)]=y[i];
```

Die Entfernungen zwischen den aufgenommenen Bildern der **Entzerrungsmatrix** werden nun in die Routine "m_d_Xposition" eingeschrieben. In der y-Variable werden hierbei die jeweiligen Längen des Radius festgehalten. Diese werden auch unter der Funktion "m_x_yrad" abgespeichert.

```
//Damit keine Funktion für die Mittelachse berechnet wird
else if(m_w_bl==((PERSPEC_MINBLOBNR-1)/2)*(PERSPEC_FKTDIV_LENGTH+1))
{
  m_w_bl+=PERSPEC_FKTDIV_LENGTH;
  n_count++;
  continue;
}
```

Da die **Entzerrungsmatrix** im Normalfall fünf "Blobs" aufweist und ein Punkt genau in der Mittelachse des Kamerabildes platziert ist (und daher auch kein Vektor berechenbar ist), muss diese Ausnahme im Programm übersprungen werden.

```
else if(m_w_bl>=(1+(PERSPEC_FKTDIV_LENGTH+1)*n_count))
{
```

```
  if(m_w_bl==(1+(PERSPEC_FKTDIV_LENGTH+1)*n_count))
  {
```

Zur Berechnung der restlichen Funktionen der Punkte der Entzerrungsmatrix werden diese entsprechend den obigen Beschreibungen behandelt.

```
mrqmin(x,y,sig,gues,ia,covar,alpha,chisq,m_F_fnlin,alamda);
```

Hierbei werden die beiden Variablen "a" und "b" wieder mit der Funktion "mrqmin()" berechnet, die die Entzerrungsfunktion bestimmen.

```
//Mittelung der berechneten Blobkoorinatenverhältnisse in Bildebene
m_x_ysqangle[n_count]/=NPT;
```

Um den Durchschnittswinkel zu ermitteln, wird die Variable durch die Anzahl der Berechnungsdurchgänge dividiert.

```
//Entzerrungsberechnungen der Blobkoordinaten
int C_Perspective::m_F_n_Equalisation(double *p_d_Blbkoordx, double *
    p_d_Blbkoordy,double *p_d_hight)
```

Mit der Funktion "m_F_n_Equalisation()" wird die Entzerrung während der Echtzeitberechnung der "Blobs" berechnet.

```
m_F_n_Koord2rkoord(p_d_Blbkoordx,p_d_Blbkoordy,&p_d_xyangle,&
    p_d_Blbkoordrad);
```

Diese Funktion rechnet die "Blobkoordinaten" in Zylinderkoordinaten um, ermittelt den Vektorradius und dessen Winkel.

```
//Berechnung der Winkellage relativ zu dem Funktionswinkel der
    Entzerrungsfunktion
for(int n=0;n<(PERSPEC_MINBLOBNR-1);n++)
{
  if(m_a_d_ab[0][n]==p_d_xyangle)
  {
    d_yend=m_a_d_ab[2][n]*(pow(*p_d_hight,-m_a_d_ab[3][n]));
    d_yend=m_F_n_Radipol(&p_d_Blbkoordrad,&d_yend,&d_angleratio,n);
```

Wenn "m_a_d_ab[0][n]" gleich dem Winkel "p_d_xyangle" ist, dann wird "d_yend" wie folgt berechnet: zuerst werden die Variablen "a" und "b" in die Funktionsgleichung eingesetzt und danach der Radius interpoliert.

```
double C_Perspective::m_F_n_Radipol(double *p_d_Blkrad,double *p_d_yend,
    double *p_d_angleratio,int n_cnt)
```

Diese Funktion dient der Interpolation des Radius und des Winkels, wie eben erwähnt.

```
if(*p_d_Blkrad>*p_d_yend)
{
  //Berechnung der gewichteten gemittelten Funktionsparameter für die
     Entzerrungsfunktion
  d_aend=m_a_d_ab[2][n_cnt];
  d_bend=m_a_d_ab[3][n_cnt];
}
```

Wenn der berechnete Radius größer als der maximale Radius ist, werden den Variablen "a" und "b" die höchsten Funktionswerte zugewiesen. Die Funktion "d_aend" stellt dabei den Endwert für "a" und "d_bend" den Endwert für "b" dar. Diese Werte werden dann zur Berechnung verwendet.

```
else if(*p_d_Blkrad>(*p_d_yend*a_d_ipolarray[0]))
{
  //Radiusinterpolationsverhältnis berechnen, um die Parameter
     auszurechnen
  d_radratio=((*p_d_Blkrad-*p_d_yend*a_d_ipolarray[0])/((*p_d_yend-(*
     p_d_yend*a_d_ipolarray[0]))));
  //Berechnung der gewichteten gemittelten Funktionsparameter für die
     Entzerrungsfunktion
  d_aend=m_a_d_ab[4][n_cnt]*(1-d_radratio)+m_a_d_ab[2][n_cnt]*d_radratio;
  d_bend=m_a_d_ab[5][n_cnt]*(1-d_radratio)+m_a_d_ab[3][n_cnt]*d_radratio;
}
```

Wenn der ermittelte Radius größer als z. B. das 0,77-fache des größten Radius ist, so wird zuerst das Radiusinterpolationsverhältnis berechnet. Dieses dient dazu, das Verhältnis zwischen dem neuen und dem berechneten Radius zu liefern, womit die Variablen "a" und "b" ermittelt werden können.

```
else
{
  //Zwischen welche Stützpunkte der Entzerrungsfunktionspunkte liegt der
     Blobkoordinatenvektor?
  int r=0;
  while((*p_d_Blkrad>(*p_d_yend*a_d_ipolarray[PERSPEC_FKTDIV_LENGTH-1-r])
     ) && r<(PERSPEC_FKTDIV_LENGTH-2)) r++;
  //Falls sich der Koordinatengeber innerhalb eines Toleranzbandes von
     der Mitte entfernt befindet,
  //wird keine Entzerrung vorgenommen, sondern die EingabeKoordinaten als
      Ausgabe übergeben.
  if(r==0)
  {
    *p_d_yend=-1;
  }
  else// if(*p_d_Blkrad>(*p_d_yend))
  {
    //Radiusinterpolationsverhältnis berechnen, um die Parameter
       auszurechnen
      d_radratio=((*p_d_Blkrad-*p_d_yend*a_d_ipolarray[
         PERSPEC_FKTDIV_LENGTH-1-r])/((*p_d_yend*a_d_ipolarray[
         PERSPEC_FKTDIV_LENGTH-2-r])-(*p_d_yend*a_d_ipolarray[
         PERSPEC_FKTDIV_LENGTH-1-r])));

    //Berechnung der gewichteten gemittelten Funktionsparameter für die
       Entzerrungsfunktion      d_aend=m_a_d_ab[4+2*(
       PERSPEC_FKTDIV_LENGTH-1-r)][n_cnt]*(1-d_radratio)+m_a_d_ab[4+2*(
       PERSPEC_FKTDIV_LENGTH-2-r)][n_cnt]*d_radratio;      d_bend=
       m_a_d_ab[4+1+2*(PERSPEC_FKTDIV_LENGTH-1-r)][n_cnt]*(1-d_radratio)
       +m_a_d_ab[4+1+2*(PERSPEC_FKTDIV_LENGTH-2-r)][n_cnt]*d_radratio;
    }
  }
```

Wenn beides nicht zutrifft, wird die Länge des Radius stattdessen mittels einer "While-Schleife" ermittelt. Dies geschieht mit "a_d_ipolarray". Dieses "Array" ist eine Variable, in der standardmäßig auf den sechs Speicherplätzen folgende Werte abgespeichert sind: 0,77; 0,54; 0,34; 0,18; 0,08; 0,03. Zwischen diesen Variablen wird der Radius interpoliert. Die Genauigkeit schwindet mit der Größe des gemessenen Radius. Nachdem das Interpolationsverhältnis "d_radratio" bekannt ist (Subtraktion der kleineren Stützstelle vom gemessenen Radius, dividiert durch den Abstand zwischen den beiden Stützstellen, die den gemessenen Radius einschließen), werden mit Hilfe dieser Kennzahl die Funktionen "d_aend" und "d_bend" berechnet. In dieser Graphik wird ver-

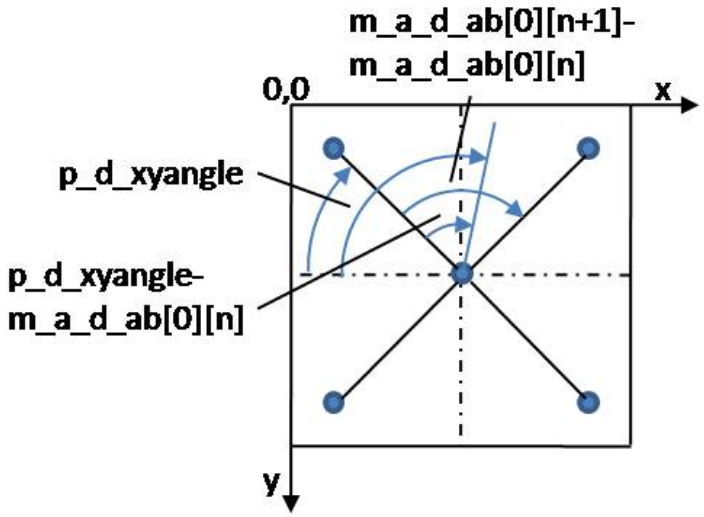


Figure 4.16: Winkel Interpolation

anschaulicht, wie die Interpolation des Winkels berechnet wird. Zur Berechnung des Verhältnisses des Winkels zwischen zwei Stützstellen und dem Winkel zwischen ersterem Winkel der Stützstellen und dem Winkel auf welchem sich der gefundene "Blob" befindet.

```
//Umrechnung der Koordinaten auf echte Koordinaten (in die
    Abbildungsoberebene)
*p_d_yend=d_aend*(pow(d_stdhight,-d_bend));
```

Mit den nun bekannten Parametern "d_aend" (Variable "a") und "d_bend" (Variable "b") kann der y-Wert "d_yend" berechnet werden.

```
//Routine, um den richtigen Winkel zu finden und dann zu interpolieren
else if(p_d_xyangle>m_a_d_ab[0][n] && p_d_xyangle<m_a_d_ab[0][n+1])
{
  //Winkelverhältnis zwischen zwei auf einanderfolgenden Winkeln
  d_angleratio=((p_d_xyangle-m_a_d_ab[0][n])/(m_a_d_ab[0][n+1]-m_a_d_ab
      [0][n]));
  //Einfache Lösung, hier wird der größte Wert einfach übernommen ->
      Ungenauigkeiten im äußeren Bewegungsbereich
  //(Bereich auch so nahezu ganz abgedeckt, nur Ecken nicht genau)
  //Besser: Einfache Extrapolation (oder Berechnung eines weiteren Wertes
       über LeastSquares, welcher weiter draussen als
  //der vermessene Blob des Kalibriergitters liegt!
  //Berechnung des gewichtet gemittelten Radius:
```

```
  d_y1=m_a_d_ab[2][n]*(pow(*p_d_hight,-m_a_d_ab[3][n]));
  d_y2=m_a_d_ab[2][n+1]*(pow(*p_d_hight,-m_a_d_ab[3][n+1]));
  //Berechnung des gewichtet gemittelten Radius:
  d_yend=d_y1*(1-d_angleratio)+d_y2*d_angleratio;
  d_yend=m_F_n_Radipol(&p_d_Blbkoordrad,&d_yend,&d_angleratio,n);
  break;
}
```

Mit der Parameterspeichervariable "m_a_d_ab[ ][ ]" wird das Winkelinterpolationsverhältnis "d_angleratio" ausgewertet. Die beiden Variablen "d_y1" und "d_y2" stellen die Ergebnisse der Entzerrungsfunktionsberechnung dar. Zur Ermittlung der Funktion "d_yend" wird der Radius mit Hilfe von "d_angleratio" berechnet und durch die Funktion "m_F_n_Radipol()" interpoliert.

```
else if(p_d_xyangle>m_a_d_ab[0][PERSPEC_MINBLOBNR-2] || p_d_xyangle<
    m_a_d_ab[0][0])
{
  d_lastangle=2*PI-m_a_d_ab[0][PERSPEC_MINBLOBNR-2];
  if(p_d_xyangle<m_a_d_ab[0][0])
  {
    d_angleratio=((d_lastangle+p_d_xyangle)/(d_lastangle+m_a_d_ab[0][0]))
        ;
  }
  else
  {
    d_angleratio=((d_lastangle-(2*PI-p_d_xyangle)/(d_lastangle+m_a_d_ab
        [0][0])));
  }
  d_y1=m_a_d_ab[2][PERSPEC_MINBLOBNR-2]*(pow(*p_d_hight,-m_a_d_ab[3][
     PERSPEC_MINBLOBNR-2]));
  d_y2=m_a_d_ab[2][0]*(pow(*p_d_hight,-m_a_d_ab[3][0]));
  //printf("%.4f %.4f",d_y1,d_y2);
  d_yend=d_y1*(1-d_angleratio)+d_y2*d_angleratio;
  d_yend=m_F_n_Radipol(&p_d_Blbkoordrad,&d_yend,&d_angleratio,n);
  break;
}
```

Wenn der Winkel größer als der letzte Funktionswinkel oder kleiner als der kleinste Funktionswinkel ist, dann wird der größte Winkel "d_lastangle" berechnet. Danach wird "d_angleratio" in Abhängigkeit zum Winkel berechnet. Mit "d_y1" und "d_y2" werden Funktionswerte der Entzerrungsfunktion berechnet, die anschließend zur Berechnung der Variable "d_yend" für die Interpolation verwendet wird. Mit Hilfe der Funktion "m_F_n_Radipol()" wird die Radiuslänge interpoliert.

In dieser Graphik ist die Aufteilung der fünf variablen "Blobpunkte" zu erkennen. Dargestellt sind die vier Winkel der vier äußeren Punkte um den Mittelpunkt. Zwischen diesen bekannten Winkeln wird interpoliert.

Die Unterteilung der Radien in sieben Bereiche dient der Interpolationsrasterung für den vorhandenen Radius (mit zunehmender Entfernung von der Mitte nehmen diese Bereiche einen immer größer werdenden Wert an). Zwischen diesem Raster wird interpoliert, abhängig davon, in welche Schranken der Radius hineinfällt (bei diesem Raster handelt es sich um bekannte Funktionswerte).

```
else if(p_d_xyangle>m_a_d_ab[0][PERSPEC_MINBLOBNR-2] || p_d_xyangle<
    m_a_d_ab[0][0])
```

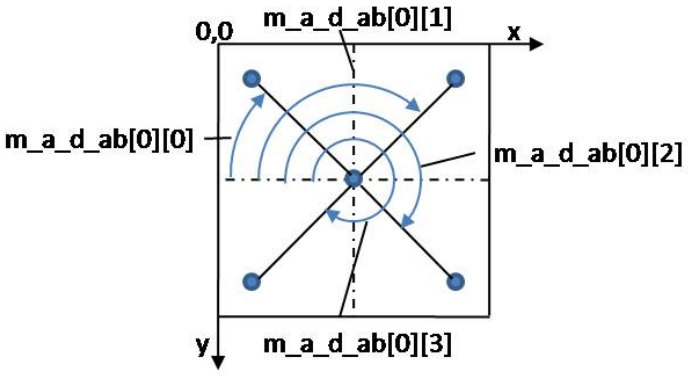


Figure 4.17: Positionen der vier Winkel

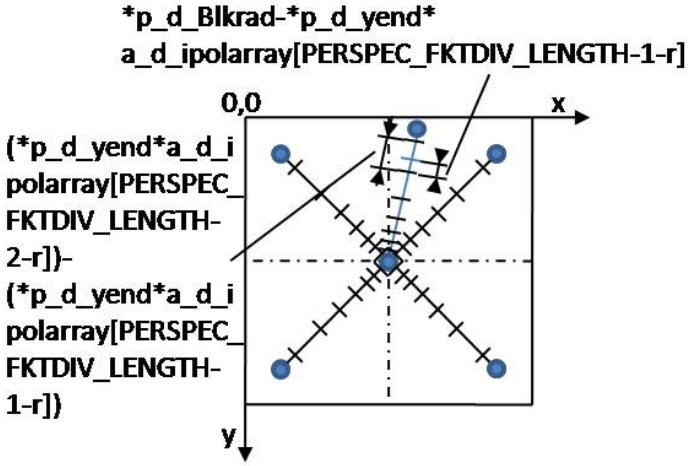


Figure 4.18: Radiusinterpolation

```
{
  d_lastangle=2*PI-m_a_d_ab[0][PERSPEC_MINBLOBNR-2];
  if(p_d_xyangle<m_a_d_ab[0][0])
  {
    d_angleratio=((d_lastangle+p_d_xyangle)/(d_lastangle+m_a_d_ab[0][0]))
        ;
  }
  else
  {
    d_angleratio=((d_lastangle-(2*PI-p_d_xyangle)/(d_lastangle+m_a_d_ab
        [0][0])));
  }
  d_y1=m_a_d_ab[2][PERSPEC_MINBLOBNR-2]*(pow(*p_d_hight,-m_a_d_ab[3][
     PERSPEC_MINBLOBNR-2]));
  d_y2=m_a_d_ab[2][0]*(pow(*p_d_hight,-m_a_d_ab[3][0]));
  //printf("%.4f %.4f",d_y1,d_y2);
  d_yend=d_y1*(1-d_angleratio)+d_y2*d_angleratio;
  d_yend=m_F_n_Radipol(&p_d_Blbkoordrad,&d_yend,&d_angleratio,n);
  break;
```

```
}
```

Wenn die Variable "d_yend" der Zahl minus eins entspricht, werden die "Blobkoordinaten" in die Bildbearbeitungssoftware übernommen, sofern die erkannten "Blobs" genau in der Bildmitte liegen. Mit der Funktion "m_F_n_Rkoord2koord()" werden die Zylinderkoordinaten wieder zurück auf normale Bildkoordinaten umgerechnet. Die Berechnung der endgültigen Koordinaten erfolgt dabei mit einem einfach strukturierten digitalen Filter.

$$x_{end} = x_{Koord.alt} * Dampval + x_{Koord.neu} * (1 - Dampval) \tag{4.2}$$

Dampval... Dämpfungsfaktor
Mit zunehmendem Ausmaß des Dämpfungsfaktors (auch Vergessensfaktor genannt) verlangsamt sich die Angleichung der ursprünglichen Koordinaten an die neuen Koordinaten, währenddessen mit zunehmender Verkleinerung des Faktors die Angleichung der Endkoordinaten an die neuen Koordinaten beschleunigt wird, wobei zu bedenken ist, dass stark von der Norm abweichende Koordinatenwerte bei geringerer Faktorgröße schlechter absorbiert werden als bei höherer Faktorgröße.

5 Ausblick, Zukunft

5.1 Stand der Dinge

Nach erfolgreicher Umsetzung der ggst. Projektstudie konnte diese Diplomarbeit nunmehr fertiggestellt werden. Das Hauptziel, die direkte und unmittelbare Ansteuerung der Werkzeugmaschine durch die Hand des Operators mit Handschuh in Echtzeit funktioniert und kann bedient werden. Zur Erreichung dieses Projektziels wurde zunächst eine Versuchsanordnung entwickelt, mit der die möglichen Geschwindigkeiten und Beschleunigungen einer menschlichen Hand vermessen werden konnten, um die Bildbearbeitungssoftware an die Gegebenheiten der menschlichen Handbewegungen entsprechend zielorientiert adaptieren zu können und in die Lage zu versetzen, diese Handbewegungen entsprechend den obigen Zielen ident in Maschinenarbeit umzusetzen. Gleichzeitig wurden dabei auch die maximal möglichen, notwendigen und sinnvollen Arbeitsbereiche des Operators im Stehen oder im Sitzen vermessen. Wichtig für die ggst. Forschungstätigkeit war auch die Erforschung der Auswirkungen der möglichen Positionen des Bedienungspersonals (z.B. sitzende oder stehende Position) auf die Bewegungen der Hand und damit auf die Art und Weise der Verarbeitung dieser Bewegungen durch die Bildbearbeitungssoftware.
Der nächste Projektschritt stellte die Entwicklung eines Versuchsaufbaus mit dem Ziel, eine exakte Erkennung und Positionsbestimmung des **Koordinatengebers** durch die Entwicklung einer speziellen Software und entsprechenden räumlichen und Ausstattungsgegebenheiten zu erreichen, dar. Als Nächstes folgte die Erarbeitung eines Bildzuschnittsalgorithmus, der eine Erhöhung der Rechnerkapazität zur Beschleunigung der Bildverarbeitung durch Reduktion des Suchprozesses vom gesamten Bildausschnitt auf einen kleinen Bildausschnitt zum Ziel hatte. Weiters wurde ein Programm zur Ausführbarkeit des Suchalgorithmus des **Koordinatengebers** entwickelt. Es wurde auch ein übersichtliches graphisches Modell (Catia) der gesamten Werkzeugmaschine einschließlich ihrer sämtlichen Komponenten erstellt, um einen Überblick über den gesamten Arbeitsaufbau der Anlage zu erhalten und diverse Anwendungen und die Raumverhältnisse an Hand dieses Modells testen zu können. Danach wurde eine für die optischen Berechnungen notwendige Datei erarbeitet, wie z.B. für die Berechnung der erforderlichen Kamerabildauflösung und der zu verwendenden Objektive. Des weiteren wurden eine Laserpositionierungseinheit, eine "Inifile" Schreib- und Leseeinheit und weitere Einstellroutinen erstellt. Bald wurde auch ein **Tracking Device** angeschafft, bei dem die Art der Ansteuerung entwickelt werden musste. Auch eine Justierungsroutinesoftware wurde für das Gerät wurde fertiggestellt. Die Bildbearbeitungsprogramme wurden während der gesamten Projektdauer entwickelt bzw. verbessert oder abgeändert . Gegen Ende der ggst. Forschungsarbeit erfolgte schließlich die Programmierung der fünf unterschiedlichen Bildbearbeitungsroutinen. Auch ein "Logfile" wurde erstellt, um sehr viele Para-

meter aufnehmen und in eine Datei schreiben zu können. Dies erwies sich als sehr nützlich, um Probleme aufzuspüren. Zuletzt wurde die Bildentzerrungroutine (und damit auch die Berechnung des "Least Squares Algorithmus") fertiggestellt, die während des gesamten Projektzeitraums entwickelt worden war. Die Arbeit der Stereoberechnung und der Ansteuerung der Achsen y-, A-, B-, C- (A, B, C sind die rotatorischen Freiheitsgrade) war Aufgabe von DI FH Gelhart, der parallel arbeitete. Das Endergebnis dieser Arbeit ist eine funktionsfähige Einheit, die nur durch eine sich bewegende Hand (in Handschuhen) gesteuert wird.

5.2 Probleme

- Trackinggeschwindigkeiten: Trotz schneller Komponenten kommt es bei der Steuerung der Maschine mit einem **Koordinatengeber** zu einer sichtbaren Verzögerung. Am Datenübertragungsweg zwischen Computer und Werkzeugmaschine ist das Beschleunigungspotential nach derzeitigem Stand der Technik erschöpft, da die maximal möglichen Zykluszeiten nicht noch mehr reduziert werden können. Für die Beschleunigung der Berechnungszeit pro Bild des Bildbearbeitungsprogrammes könnten die Prozessoren der zwei angeschlossenen Computer noch einbezogen werden (im Moment findet die gesamte Berechnung auf dem eigenen Prozessor des Framegrabber statt). Der dafür notwendige "Dongle" ist für einen Rechner verfügbar. Für die Aufteilung der Berechnung müsste allerdings das Programm "Multithreading"-konform programmiert werden.

- Koordinatengeber: Aufgrund der recht geringen Abmaße der drei Kugeln des **Koordinatengeber**s ist ab einer bestimmten Entfernung der Kamera zum **Koordinatengeber** kein Blob mehr zu erhalten. Diese Problematik wird durch die etwas unscharfe Optik der Trackingkamera verursacht, was zu einer Reduktion der entdeckten "Blobfläche" führt, da es keine scharfe Kontur im Randbereich gibt. Auch der Abstand zwischen den einzelnen Kugeln könnte sicherlich vergrößert werden, was auch eine exaktere Ausgabe der drei rotativen Achsen zur Folge hätte. Durch diese Einflüsse reduziert sich der Arbeitsbereich auf ca. 10 cm über der 1300mm Marke (Abstand Kamera- **Koordinatengeber**) und 10 cm darunter.

- Handschuh: Statt dem verwendeten Handschuh, der den Nachteil der starken Reflektion aufweist, sollte nach einem besser absorbierenden Stoff gesucht werden. Der Versuch, das Material einer Schürze zu verwenden, schlug fehl, da der Stoff zwar das Licht besser absorbierte, sich allerdings als zu unflexibel erwies (er wies zu starke Falten auf, auch schon bei geringer Positionsänderung).

- Tracking Device: Die Geschwindigkeit, mit der sich die Schwenk- und Kippvorrichtung bewegt, ist etwas langsam. Dadurch verzögert sich die Berechnung der Tiefe, da die Bildbearbeitungssoftware die Bilder zur "Blobsuche" nicht ausreichend schnell erhält. Abhilfe könnte eine Verkürzung der "Stepzeiten" schaffen (das ist die Zeit, die von einem Punkt zum nächsten Punkt vergeht), allerdings kommt es dann vermehrt zu Rüttelbewegungen der Schwenk- und Kippvorrichtung.

- Anfahren in die "Homeposition": Bei diesem Punkt des Bildbearbeitungsprogramms soll das **Tracking Device** in die "Homeposition" fahren. Allerdings ist es noch möglich, diese Fahrt der kleinen Maschine durch die Tasteneingabe "ja" in die Computertastatur zu unterbrechen, während das Gerät langsam in die "Homeposition" fährt, was eine vollständige

Rückkehr in die Ausgangsposition verhindert. Dies könnte mit einer Modulstatusabfrage, die eine Unterbrechung ausschließt, solange die Bremsen noch nicht wieder aktiviert sind, geändert werden.

- Drehbewegungen: derzeit sind Drehbewegungen nur bis ca. 45 Grad möglich und in der Ausführung noch nicht sehr ausgereift (starke Schwankungen und Rütteln etc.). Diese Problematik könnte durch einen weiter zu entwickelnden Algorithmus behoben werden. Näheres zum derzeit verwendeten Programm siehe [6].

5.3 Zukunft

- Spindel: Als nächsten Schritt wäre es sinnvoll, die Motorspindel mittels einer Befestigungsplatte auf der Maschine der Firma Trumpf zu befestigen und auszutesten (die Zeichnung einer derartige Befestigungsplatte wurde bereits angefertigt und mittels "Rapid Prototyping" hergestellt). Vorher wird es jedoch notwendig sein, ein Notaussystem zu installieren, um in Notsituationen die Maschine sofort zum Stillstand bringen zu können. Die passende Motorspindel wurde bereits angeschafft.

- graphische Oberfläche: Um eine leichtere und intuitivere Bedienung zu gewährleisten, sollte ein graphischen "Userinterface" implementiert werden. Damit wäre es leichter möglich, verschiedene Routinen in beliebiger Reihenfolge aufzurufen, ohne vorher das gesamte Programm neu starten zu müssen. Dies könnte zum Beispiel der Fall sein, wenn das Bild nach einer Echtzeitberechnung neu eingestellt werden müsste.

- "Modulflags": Da es bei besonders schnellen Bewegungen des **Tracking Device**s z. B., wenn es zu einer Fehlerkennung kommt, schnell zu einem Verlust der Synchronität führen kann, sollte während des Betriebes der Echtzeitberechnung eine diesbezügliche Abfrage stattfinden. Dies kann über ein kontinuierliches Auslesen der "Modulflags" und damit des Modulzustandes umgesetzt werden. Wenn ein solcher Zustand erkannt wird, dann sollte die Echtzeitberechnung gestoppt und das außer Betrieb gesetzte Modul wieder gestartet werden.

6 Fehlereinflüsse

Um ein Gefühl dafür zu bekommen, welche Fehler auftreten können, und wie stark die Auswirkungen auf die Genauigkeit der erhaltenen Koordinaten sind, werden die Fehlereinflüsse in diesem Kapitel behandelt.

6.1 Kamera

- Chipgenauigkeiten:
  - Chip- Gehäuse- Winkel:
    Erläuterung: Gibt an wie stark der Kamerachip von der Parallelität zwischen Chipkante und Gehäuse abweicht. Dies ist abhängig von den Fertigungstoleranzen.
    Fehlergröße: ist nicht direkt feststellbar, da das Gehäuse nicht für eine Öffnung vorgesehen ist.
    Mögliche Einflussnahme: dies kann über die Entzerrungskalibrierung herausgerechnet werden. Problematisch kann es nur werden, wenn die Kamerachips stark unterschiedlich zueinander geneigt sind, da die Entzerrung eventuell zu zwei nicht gleich großen Kamerabildern führt.
  - Chip- Gehäuse- Offset (in x- und y)
    Erläuterung: Durch die vorhandenen Fertigungstoleranzen ist es möglich, dass es eine leichte Verschiebung in x- und y- Richtung gibt.
    Fehlergröße: Im zehntel Millimeterbereich.
    Mögliche Einflussnahme: keine direkte Behebung möglich.
  - Chip- Gehäuse- Offset (in z):
    Erläuterung: Aufgrund der schon erwähnten Fertigungstoleranzen gibt es auch in der Tiefe (z- Richtung) zwischen Chipfläche und Gehäuse der beiden Stereokameras zueinander kleine Fehler.
    Fehlergröße: auch im zehntel Millimeterbereich. Sehr geringe Verbreiterung des Bildes bei der maximalen Verschiebung der beiden Kameras.
    Mögliche Einflussnahme: keine Änderung möglich, Fehler nicht leicht messbar.
  - Chip: Erläuterung: Aufgrund der oben genannten Fertigungstoleranzen sind auch Abweichungen der Pixelgrößen oder zumindest des gesamten Chips zu erwarten. Auch sind minimale Schwankungen in der Chipbreite zu erwarten. Schwankungen der einzelnen Pixelempfindlichkeiten sind auch zu berücksichtigen (oder auch unterschiedliche

Chipempfindlichkeiten gesamt).
Fehlergröße: Abhängig von der maximal erlaubten Schwankung der Chippositionierung.
Mögliche Einflussnahme: Eventuelle Fehler sind über die Kameraeinstellungen reduzierbar (Rauschen durch Beleuchtung reduzieren, maximale Empfindlichkeit ausnützen ...).

- Gehäusefertigungstoleranzen Erläuterung: Zusätzlich zu den Chiptoleranzen treten auch noch fertigungsbedingte Gehäusetoleranzen auf, die sich wie folgt aufteilen:
  - Rechtwinkeligkeit des Gehäuses zwischen den einzelnen Begrenzungsflächen , welche zur Montage vorgesehen sind.
  - Ebenheit der Montageflächen, auf denen die Kamera montiert wird.
  - Positionierung der Gewindebohrungen in Bezug aufs Gehäuse.
  - Positionierung der Gewindebohrungen einer Fläche zueinander.

  Fehlergröße: Gehäusefertigungstoleranzen liegen im zehntel Millimeterbereich.
  Mögliche Einflussnahme: Etwaige fertigungstechnische Nachbearbeitung der Montageflächen der Einspannvorrichtung.

6.2 Stereokameraträgerkonsole

- Montagegenauigkeit der Kamera
  - Montagegenauigkeit der Kamera auf der Drehachse Erläuterung: Da die Verbindung Kamera- Drehachse nur über ein normales Gewinde gewährleistet ist, ist es notwendig, den möglichen Versatz einer jeden Verschraubung in die Fehlerberechnung mit ein zu beziehen.
    Fehlergröße: Abhängig von den Gewindetoleranzen. Da jedoch eine seitliche Begrenzung (die beidseitigen "Laschen" der Drehachse) vorhanden ist, wird der maximal mögliche Versatz reduziert.
    Mögliche Einflussnahme: Mehrfache Anschraubmöglichkeiten (Ausgleichen der Gewindeungenauigkeiten mittels Flächenauflage), Feingewinde verwenden.
  - Montageungenauigkeiten der Drehachse auf das Trackingdevice Erläuterung: Die Verbindung Drehachse- **Trackingdevice** (Power Cube) ist wiederrum mit einem gewöhnlichen Gewinde befestigt.
    Fehlergröße: mehrere zehntel Millimeter
    Mögliche Einflussnahme: Verringerung der Ungenauigkeiten einer Verschraubung durch die Verwendung mehrerer Verschraubungen. Eventuell einen zusätzlichen Anschlag oder ähnliches vorsehen.
  - Auflageungenauigkeiten der Kamera Erläuterung: Beinhalten die Ungenauigkeiten der Flächen der gegeneinander verdrehbaren Teile zueinander (Drehachse und darauf befestigter Kamera).
    Fehlergröße: Abhängig von der Fertigungsgenauigkeit. Nicht näher angebbar.
    Mögliche Einflussnahme: Genaue Fertigung, genaue Bohrung, Maßhaltigkeit und Einhaltung der Lagetoleranzen.

- Genauigkeiten der Drehachsen- Bohrung für die Verdrehung der Kameras Erläuterung: Für eine genaue Positionierung der beiden Kameras zueinander , und damit eine möglichst epipolare Ausrichtung der Bilder zu erhalten, ist es notwendig, eine möglichst enge Passung der Drehachsen zu wählen und allerdings auch die Lage der Bohrung zueinander exakt zu fertigen. Auch muss die Rechtwinkeligkeit der Bohrung zur Oberfläche des Trägers gewährleistet sein.
  Fehlergröße: Abhängig von der Fertigungsgenauigkeit.
  Mögliche Einflussnahme: Möglichst exakte Fertigung der Bohrungen, nachbearbeiten dieser mit einer Reibahle, einzeln einpassen. Bohrung möglichst rechtwinklig bohren, gute Einspannung, vorher Träger schleifen (Ober- und Unterseite).
- Einhaltung der symmetrischen Anordnung der Kameras Erläuterung: Um der Annahme einer epipolaren und symmetrischen Kameraposition gerecht zu werden, um wiederum eine exakte Tiefeninformation zu erhalten, ist es notwendig, die für die Berechnung angenommenen Voraussetzungen bestmöglich zu erfüllen. Die Abweichungen von dieser idealen Vorstellung gehen direkt in die Fehleranalyse ein.
  Fehlergröße: Bei Berechnung des z- Abstandes des **Koordinatengebers** kann es zu Ungenauigkeiten führen.
  Mögliche Einflussnahme: Möglichst Exakte Fertigung der symmetrisch wirkenden Einstellvorrichtung (Stereokameraträgerkonsole) und Vermeidung von Spiel.

6.3 Trackingdevice

- Verfahr- und Positionierungsgenauigkeit des Montagetellers Erläuterung: Die Genauigkeit, mit welcher das Tracking- device verfahren kann, und eine Position anfahren kann (wiederholt, aus der gleichen Richtung).
  Fehlergröße: Die Wiederholgenauigkeit des Power Cubes PW90 liegt für Schwenk- und Neigebewegung bei 0,02°.
  Mögliche Einflussnahme: Keine direkte Genauigkeitserhöhung möglich, eventuell auf minimale Gewichtsbelastung achten.

6.4 Kalibriereinrichtung

- Abstandmessung des Kalibriergitters Erläuterung: Abstandsmessung, die notwendig ist, um eine möglichst genaue und akkurate Kalibrierung zu erhalten.
  Fehlergröße: Abhängig von der Genauigkeit der Kalibriervorrichtung.
  Mögliche Einflussnahme: Möglichst die Messgenauigkeiten ausreizen, präzise Messwerkzeuge.

- Genauigkeit der **Kalibriergitter**positionsfixierung Erläuterung: Sehr wichtig für die richtige Funktionsweise der Kalibrierung, da eine falsche Berechnung der Entzerrfunktion berechnet würde. Änderung der Höhe, oder ein seitlicher Versatz (eventuell Verdrehung), welcher in eine ungenaue Funktion resultieren würde.
  Fehlergröße: Abhängig von der Fertigungsgenauigkeit der Kalibriervorrichtung, wenn eine solche Vorrichtung überhaupt vorhanden ist.
  Mögliche Einflussnahme: Möglichst exakte Fertigung der Fahrbahn oder ähnlichem , auf welchem das Kalibriergitter bewegt wird.

- Genauigkeit der **Kalibriergitter**Winkelpositionierung Erläuterung: siehe Erklärung voriger Punkt.
  Fehlergröße: Abhängig der Genauigkeit der Vorrichtung.
  Mögliche Einflussnahme: Exakte Vorrichtung zum einstellen des Winkels (Anschläge, Skalen oder ähnliches).

- Genauigkeit des **Kalibriergitters** Erläuterung: Die Genauigkeit der einzelnen Gitterpunkte, der Anordnung und aber auch der Layouterstellung (viele große Abstände, ungenauer, große Punkte, kleine Punkte, Anzahl?).
  Fehlergröße: Abhängig von den Herstellungsgenauigkeiten, also Druckerfehler, Druckergenauigkeit und ähnliches.
  Mögliche Einflussnahme: Exaktes Drucken, eventuell spezielle Fertigung eines eigenen Kalibriergitters (fräsen, gravieren...).

6.5 Kalibrierung

- Entzerrungsungenauigkeiten
  - Entzerrungsungenauigkeiten bei **Entzerrungsmatrix**Erstellung der Stereokameras in Ausgangsposition Erläuterung: Die Erstellung einer **Entzerrungsmatrix** eines Bildes birgt viele Fehlerquellen in sich, wie z.B.: Falsche Kameraposition oder falsche Position des **Kalibriergitters** (x und y Position, z nicht so wichtig).
    Fehlergröße: Abhängig von der Kalibriervorrichtung.
    Mögliche Einflussnahme: Exakte Positionierung des **Kalibriergitters**, exakte Kalibriervorrichtung.
  - Entzerrungsungenauigkeiten bei Berechnung der Funktionen der Stereokameras Erläuterung: Die Erstellung einer **Entzerrungsmatrix** eines Bildes birgt viele Fehlerquellen in sich, wie z.B.:
    - Falsche Tiefeninformation des **Kalibriergitters** zum Wahren Abstand des Kamerabrennpunktes zur Kalibrierebene.
    - Falsche Winkelinformation des **Kalibriergitters** zum Wahren Winkel der Kameraachse zur Kalibriergitterebene.
    - Falsche oder ungenaue (Aufgrund schlechter Bilder, schlechter Kontraste oder ähnlichem)
    - Blob Erkennung, welche wiederum zu Ausreißern der Stützpunkte für die Berechnung der Entzerrungsfunktionen führen.
      Fehlergröße: Abhängig von **Kalibriergitter**, Kalibriervorrichtung...
      Mögliche Einflussnahme: Genaue Positionierung des **Kalibriergitters**, genaue Fertigung.
  - Berechungsungenauigkeiten der Funktionen der Stereokameras Erläuterung: Die Erstellung einer Funktion aufgrund von Bildinformationen, wobei hier folgende Fehlerstellen auftreten können:
    - Zu stark von den Wahren Werten abweichende Werte, starke Ausreißer und ähnlichem Schlechte Approximation der Funktion aufgrund schlechter Annäherungsalgorithmen Falsche Grundfunktion gewählt.

  * mehrfache Interpolation zu ungenau
    Fehlergröße: Abhängig von den Messergebnissen...
    Mögliche Einflussnahme: Überprüfung der Funktionen mit einer Messung der Entzerrungsqualität...

6.6 Umwelt

- Beleuchtungsschwankungen Erläuterung: Da die wichtigsten Eingangsgrößen optisch erfasst werden , ist eine robuste Bilderkennung wichtig. Probleme können auftreten , wenn eine starke Lichtquelle unvorhergesehen und nur partiell den Bildbereich erfasst.
  Fehlergröße: Möglicherweise das System zum Stillstand bringend, großer Einfluss auf gesamtes System.
  Mögliche Einflussnahme: Eine die ganze Zeit währende Beleuchtung verwenden, die die Auswirkungen anderer möglicher Lichtquellen reduziert.

- Temperaturschwankungen Erläuterung: Auch wenn nicht damit zu rechnen ist, sind doch auch bei nicht allzu großen Temperaturschwankungen schon mit Auswirkungen auf die empfindlichen Aufbauten zu erwarten.
  Fehlergröße: Eher geringe Auswirkungen.
  Mögliche Einflussnahme: Temperatur konstant halten, während der Bearbeitung. Neujustierung , wenn längere Zeit die pralle Sonne auf die Vorrichtung geschienen hat...

6.7 Detektion, Lageerkennung

- Trackingungenauigkeiten
  Erläuterung: Da die Tracking Steuerung mittels einer Bloberkennung arbeitet, sind hier auftretende Fehler am gravierendsten, und beeinflussen das System am stärksten. Mögliche Fehler sind unter anderem:
  - Falsche Bloberkennung mit der Folge einer falschen Schwerpunktsausgabe, die eventuell noch über die letzten Blobkoordinaten in Grenzen korrigiert werden kann, aber nicht berichtigt.
  - Erkennung zweier **Blobs**, wobei dies zwar möglich ist, aber eine Falschdetektion nur schwer erkannt werden kann. Bei Erkennung eventuell die Wahl des falschen **Blobs**...
  - Erkennung zu vieler **Blobs**, oder nach Begrenzung der Parameter, keinen mehr.
  - Kontinuierliche Falscherkennung eines angenommenen **Blobs**, dadurch auch kontinuierlich falscher Bildausschnitt, Abbruchbedingung.
    Fehlergröße: Sehr groß, da Falschkoordinaten das System stark beeinträchtigen können, oder zumindest in eine Notausposition führen können.
    Mögliche Einflussnahme: Möglichst robuste Bloberkennung, oftmalige Hintergrunderkennung während der Bearbeitung, Kameras für beste Kontraste einstellen...

- Stereoerkennungsungenauigkeiten Erläuterung: Mit Annahme einer richtigen Trackingerkennung , sind auch bei der Stereoerkennung (Tiefenerkennung, exakte Lageerkennung, Verdrehwinkeldetektion) einige Fehlerquellen vorhanden:
  - Falsche Erkennung des Schnittpunktes für Linearkoordinatenerkennung

- Falsche Winkeldetektion, aufgrund angrenzender Blobflächen
- Falsche Berechnung der Geraden , welche die Koordinatensystemrichtungen des Koordinatenkreuzes geben sollten.

Fehlergröße: Nicht unwesentlich, da wenn ein **Blob** erkannt wurde , noch der gesamte Stereobildbereich als Koordinatenausgangswerte gelten kann.
Mögliche Einflussnahme: Möglichst robuste Bloberkennung, Mittelung über letzte Werte, logische Überlegungen mit einbauen...

6.8 Zusammenfassung

Lange Zeit nahmen die ersten Versuche in Anspruch, sich einmal des Bewegungsraumes klar zu werden, um eine Maschine optisch anzusteuern. Dazu wurde zuerst in Normen gesucht, um herauszufinden, welche Reichweite ein Mensch hat, sei es nun mit den Händen, sitzend oder stehend, mit Schritten oder ohne usw.. Um die Angaben der Normen einerseits zu überprüfen, und andererseits auch weiter relevante Daten, die nicht in einer Norm abgeklärt wurden, zu bestimmen, wurden einige Versuche gemacht. Dabei wurde z.B. die Armreichweite im Sitzen, ohne Bewegung des restlichen Körpers gemessen. Oder aber auch die Armreichweite im Stehen, bei zusätzlicher Bewegung, wurden mittels einem einfachen Tischtennisball, welcher mit einer normalen Webcam aufgezeichnet wurde, gemessen. Das dafür notwendige kleine Programm wurde mit Matlab geschrieben. Um sich ein Bild zu machen, wie es mit den Platzverhältnissen der Maschine bestellt ist, wurde die Maschine virtuell in Catia gezeichnet. Auch zur Platzierung des Drehtisches wurde die 3D Konstruktion verwendet. Die Programmierung des Programmes zur Berechnung der Position des **Koordinatengebers** im Bild nahm die meiste Zeit in Anspruch. Neben der **Bildbearbeitung** mussten auch Routinen für die einzelnen Justierungen geschrieben werden. Dazu zählen eine Lasereinstellroutine, eine Kalibrationsroutine , eine Entzerrungsroutine ... Die eigentliche **Bildverarbeitungsroutine** kann in 5 verschiedenen Modi laufen gelassen werden, die sich durch verschiedene Ausgabe und Logfunktionen unterscheiden. Das größte Problem welches zu meistern war, war die von der Höhe abhängige Größenänderung des zu suchenden **Koordinatengebers**. Der Versuch, beide Framegrabber in einen PC einzubauen um die Gesamtgeschwindigkeit der Datenversendung zu erhöhen und die Kommunikation zu verbessern, schlug fehl. Aufgrund von Inkompatibilitäten war es nicht möglich, beide Einheiten in einem Computer laufen zu lassen.

7 Anhang

7.1 Klassen und Funktionsübersicht

- Main_PROC_DIGPROC_ODY_TRACKDEV_OPTI.cpp
  1390 Zeilen

  Notwendige externe Dateien:

  cpp-Dateien:
  Trackdev.cpp
  Header Dateien:
  Trackdev.h
  Mil.h
  TcAdsAPI.h
  TcAdsDef.h
  M5apiw32.h
  Bibliotheksdateien:
  TcAdsdll.lib
  M5apiw32.lib

  Kurze Programmerläuterung:
  Basierend auf der Funktion Main_PROC_DIGPROC_ODY.cpp, jedoch mit optimierter **Tracking- Deviceansteuerung** (static Variablen, Funktionen nicht als Klassen implementiert, Codeoptimierung), um dem Blob im Bild nachzufahren. Alle Echtzeitfunktionen sind direkt im Hauptprogramm implementiert, aber auch noch einmal als Klassenfunktion verwendbar (Trackdev.cpp, Trackdev.h). Allerdings aufgrund der Klassenaufrufe etwas langsamer. Auch ist eine Kameraeinstellfunktion direkt als Funktion eingebaut, die die Einstellung der Kamera über die serielle Schnittstelle gewährleisten soll. Dabei ist noch eine Initialisierungsroutine ausprogrammiert, die zum Einstellen des **Tracking Devices** verwendet wird (genaue Lageeinstellung, um eine senkrechte Stereokameraachsenlage (gemittelt beider Stereokameras) auf die **Arbeitsbereichsmittenebene** zu gewährleisten. Auch können mit dieser Routine alle allgemeinen Parameter eingestellt werden, sei es für das **Tracking Devices** selbst, oder für die Maschine, oder für die jeweiligen **Bildbearbeitungen**. Auch die Laserpositionierungsroutine ist implementiert. Die zur Berechnung notwendigen Variablen und Messergebnisse können in einer Logdatei abgespeichert werden. Diese Log- Datei

heißt DataArray.dat, wobei der Umfang der Speicherung eingestellt werden kann.

Probleme:
Speicherfreigabeprobleme. Uart Funktion noch nicht funktionsfähig. Initialisierungsspeicherprobleme.

- Trackdev.cpp (h)
  380 Zeilen (65 Zeilen)

  Notwendige externe Dateien:

  cpp-Dateien:
  Header Dateien:
  Trackdev.h
  Mil.h
  M5apiw32.h
  Inistart.h
  Ini.h
  Bibliotheksdateien:
  TcAdsdll.lib
  M5apiw32.lib

  Klasse:
  C_Trackingdevice

  Klassenfunktionen:
  m_F_n_Trackinit(float f_WorkDistPixel, float f_WorkRangePixelx,float f_WorkRangePixely,int *p_n_EDSdevnumber):

  f_WorkDistPixel: Gibt den Abstand der Trackingkamera in Pixel (von der Mittenebene) an.
  f_WorkRangePixelx: Gibt die Bildbreite des Trackingbildes in x- Richtung in Pixel an.
  f_WorkRangePixely: Gibt die Bildbreite des Trackingbildes in y- Richtung in Pixel an.
  *p_n_EDSdevnumber: Gerätenummer des Gerätes des Herstellers ESD.

  Dies ist eine Funktion zur Initialisierung des **Tracking Devices**. Gibt die Bibliotheksversionsnummer aus, und es erfolgt eine Aktivierung des Gerätes. Danach kommt ein resetten der beiden Module (Schwenk- und Neigeantrieb) und die Eingabe der Softwareendschalter, außerdem werden die maximalen Schwenk- Neigegeschwindigkeiten und Beschleunigungen eingestellt. Als nächstes kommt es zu einer Abfrage der Versionsnummern der Module und zu einem Speichern der Baudraten (Übertragungsgeschwindigkeiten) und einer Einstellung der Grundhomeposition. Der nächste Punkt ist das Setzen des maximalen Modulstromes für die Antriebe. Am Ende erfolgt noch die Einrichten der Homeposition (Ausgangsposition des Gerätes).

  m_F_n_Trackclose(void):

Schließen der Verbindung des **Tracking Device** und dem PC.

m_F_f_Trackangle(double *p_d_Value):
*p_d_Value: Pixelwert nach der Bloberkennung.

Umrechnung der Pixelkoordinaten des Bildes auf den Schwenkwinkel des Devices.

m_F_f_Trackanglemax(void):

Berechnung des maximal auftretenden Schwenk- oder Neigewinkels in Abhängigkeit des maximalen Bildausschnittes der Kamera des **Tracking Devices**.

m_F_dwd_Digfilter(DWORD *p_dwd_OldVal,DWORD *p_dwd_NewVal):

*p_dwd_OldVal: Enthält den alten Koordinatenwert der Bloberkennung.
*p_dwd_NewVal: Enthält den neuen Koordinatenwert der Bloberkennung.

Berechnung der Koordinatenwerte mittels eines einfachen digitalen Filters. Für DWORD Werte

m_F_d_Digfilter(double *p_d_OldVal_db,double *p_d_NewVal_db):

*p_d_OldVal_db: Enthält den alten Koordinatenwert der Bloberkennung.
*p_d_NewVal_db: Enthält den neuen Koordinatenwert der Bloberkennung.

Berechnung der Koordinatenwerte mittels eines einfachen digitalen Filters. Für double Werte.

m_F_n_Trackmov(double d_xval_old,double d_xval_new,double d_yval_old,double d_yval_new):
d_xval_old: Enthält den alten Koordinatenwert der Bloberkennung in x- Richtung.
d_xval_new: Enthält den neuen Koordinatenwert der Bloberkennung in x- Richtung.
d_yval_old: Enthält den alten Koordinatenwert der Bloberkennung in y- Richtung.
d_yval_new: Enthält den neuen Koordinatenwert der Bloberkennung in y- Richtung.

Berechnung neuer Koordinaten mittels einem einfachen digitalen Filter. Umrechnung der kartesischen Koordinaten in zylindrische (x- und y umrechnen in Radius und Winkel). Bewegung des **Tracking Devices** mit den neuen Koordinaten.

m_F_n_Trackdev_Pos_Setup(void):

Kurze Programmerläuterung:
Einstellung des **Tracking Devices** bezüglich der Homeposition. Genaues Nachjustieren mittels Positionseinstellroutine von Schwenk- und Neigeeinheit.

Probleme:
Speicherfreigabeprobleme.

- Init.cpp (h)
  1517 Zeilen (38 Zeilen)

Notwendige externe Dateien:

cpp-Dateien:
Header Dateien:
Trackdev.h
Mil.h
TcAdsAPI.h
TcAdsDef.h
M5apiw32.h
Inistart.h
Ini.h
Bibliotheksdateien:
TcAdsdll.lib
M5apiw32.lib

Klasse:
C_Init

Klassenfunktionen:

m_F_b_Nr2Parametername(int n_Parmeter_Number, char *p_c_Value_Type="0", char *p_c-_Data_Name="0"):

n_Parameter_Number: Gibt die Parameternummer an.
*p_c_Value_Type="0": Gibt die Art des Parameters an.
*p_c_Data_Name="0":Gibt den Namen des Parameters, der auch in der Inidatei gespeichert ist an.
Gibt für die Parameternummer den Namen und den Typ aus.

m_F_d_Laserpos(Str_HookDataStruct *Str_InitUserHookData, int n_LaserParts, float f_Offset,int n_CameraType,int n_SearchType):

*Str_InitUserHookData: Gibt die Strukturvariable des Str_HookDataStruct aus.
n_LaserParts: Gibt an, wie viele Laserpunkte im Kamerabild gefunden werden sollen.
f_Offset: Gibt die Höhen- Differenz zwischen Kameraachse und Laserachse an (in Pixel).
n_CameraType: Gibt den Kameratypen an (Trackingkamera oder Stereokamera, noch nicht funktionsfähig). n_SearchType: Switch für **Trackingdevice**Einstellungsroutine (normal 1).

m_F_n_Trackingpos(float a_f_Pic_size[2]):

nicht fertig.

m_F_n_Pic_size_Fkt(int n_Brennweite,float f_Picsize[2]):

n_Brennweite: Gibt die Brennweite einer Kamera in mm an.
f_Picsize[2]: Kamerabildgröße in x- und y- Richtung.

Berechnet die Bildgröße Aufgrund der Brennweite.

m_F_n_Create_Ini_File(int n_bitdpt,int n_digfrrate):
n_bitdpt: Gibt die Bittiefe eines Kamerabildes an.
n_digfrrate: Gibt die Bildwiederholungsrate der Kamera an.

Erstellen eines neuen Ini Files.

m_F_n_Change_Ini_File(void):
Ändern von Einträgen der Ini Datei.

m_F_n_Calc_allnew_Ini_File(int n_bitdpt,int n_digfrrate):
n_bitdpt: Gibt die Bittiefe eines Kamerabildes an.
n_digfrrate: Gibt die Bildwiederholungsrate der Kamera an.

Berechnet die zu berechnenden Parameter alle neu (z.B. falls Datei händisch umgeschrieben...).

m_F_n_ArrayInit(void):
Initialisieren der drei Stringarrays der Funktionen.

m_F_n_Create_Installpath_File(void):
Schreibt eine Datei mit allen Pfaden , wo sich die jeweiligen Dateien befinden.

m_F_n_Write_Ini_File_Exceptions(int n_Parameter_Change,void *p_v_Value_Write):
n_Parameter_Change: Gibt die Parameternummer an, um Parameter zu ändern.
*p_v_Value_Write: Gibt den aktuellen Parameterwert aus.
Ausnahmen beim umschreiben des Ini Files (einzige Schreibmöglichkeit von außen).

m_F_n_Parm_Proof(int n_Parm_Change_Nr):
n_Parm_Change_Nr: Gibt die Parameternummer an, die geändert werden soll. Kontrolle der eingegebenen Parameternummer.

m_F_n_Change_Parm_ParmNr(void):
Auswahl der Parameterzahl, welche ins Logfile geschrieben wird.

m_F_n_Change_Parm_Log_File(void):
Auswahl der Parameterzahlumschreibswitches.

m_F_n_Read_Ini_File(int n_Parameter_Read, int *p_n_Value_Read, char *p_c_Val_Type=""):
*p_n_Value_Read: Gibt den Wert des Parameters an.
n_Parameter_Read: Gibt den auszulesenden Parameter an.
*p_c_Val_Type="": Gibt den Typ des ausgelesenen Parameters an.
Parameter aus Ini- Datei lesen.

m_F_n_Read_Ini_File(int n_Parameter_Read, long *p_l_Value_Read, char *p_c_Val_Type=""):
*p_n_Value_Read: Gibt den Wert des Parameters an.
n_Parameter_Read: Gibt den auszulesenden Parameter an.
*p_c_Val_Type="": Gibt den Typ des ausgelesenen Parameters an.
Liest den Parameterwert aus der Ini- Datei. Überladene Funktion.

m_F_n_Read_Ini_File(int n_Parameter_Read, double *p_d_Value_Read, char *p_c_Val_Type=""):
*p_n_Value_Read: Gibt den Wert des Parameters an.
n_Parameter_Read: Gibt den auszulesenden Parameter an.
*p_c_Val_Type="": Gibt den Typ des ausgelesenen Parameters an.
Liest den Parameterwert aus der Ini- Datei. Überladene Funktion.

m_F_n_Read_Ini_File(int n_Parameter_Read, char *p_c_Value_Read, char *p_c_Val_Type=""):
*p_n_Value_Read: Gibt den Wert des Parameters an.
n_Parameter_Read: Gibt den auszulesenden Parameter an.
*p_c_Val_Type="": Gibt den Typ des ausgelesenen Parameters an.
Liest den Parameterwert aus der Ini- Datei. Überladene Funktion.

Kurze Programmerläuterung:
gesamte Initialisierung der Software, **Initialisierungsdatei** manipulieren (schreiben, lesen, erstellen...), **Trackingdevice**Position einrichten, Laserposition einrichten im Bildbereich.

Probleme:
Speicherfreigabeprobleme.

- Trumpf.cpp (h)
  214 Zeilen (51 Zeilen)

Notwendige externe Dateien:

cpp-Dateien:
Header Dateien:
Init.h
TcAdsAPI.h
TcAdsDef.h

Klasse:
C_Trumpf

Klassenfunktionen:

m_F_n_Trumpfinit():
Einstellen und Initialisieren der Kommunikation zwischen Computer und Trumpf- Maschine.

m_F_n_Trumpfclose():
Schließen der Kommunikation zwischen Computer und Trumpf- Maschine.

m_F_n_TrumpfmoveXZ(double **Data,double FrameC):
**Data: Enthält alle Daten, die zur Ansteuerung der Maschine benötigt werden.
FrameC: Gibt Anzahl der aufgenommenen Bilder an.
Verfahren der Trumpfmaschine.

DezToHex(char *DezValue):
*DezValue: Gibt die Zahl an, der in eine hexadezimalen Wert umgerechnet werden soll.
Nicht in Verwendung.

Umrechnung Dezimaler Werte in Hexadezimale Werte.

m_F_d_Digfilter(double *m_p_d_OldVal_db,double *m_p_d_NewVal_db):
*p_dwd_OldVal: Enthält den alten Koordinatenwert der Bloberkennung.
*p_dwd_NewVal: Enthält den neuen Koordinatenwert der Bloberkennung.
Einfacher Digitaler Filter, zur Glättung der Bewegungen.

m_F_dwd_Digfilter(DWORD m_dwd_OldVal,DWORD m_dwd_NewVal);
*p_dwd_OldVal: Enthält den alten Koordinatenwert der Bloberkennung.
*p_dwd_NewVal: Enthält den neuen Koordinatenwert der Bloberkennung.

Kurze Programmerläuterung:
Steuerung der Trumpfmaschine, Initialisierung der Maschinen- Rechner Schnittstelle. Bewegungsfunktion für die Ansteuerung der Maschine (X,Z), Berechnung eines digitalen Filters. Beenden der Maschinen- Rechner Kommunikation.

Probleme:
Speicherfreigabeprobleme.

- Initcalc.cpp (h)
  483 Zeilen (55 Zeilen)

Notwendige externe Dateien:

cpp-Dateien:
Header Dateien:
Init.h
Bauererr.h

Klasse:
C_Initcalc

Klassenfunktionen:

m_F_d_Dig_Depht(int n_bitdepth, bool b_camtype):
n_bitdepth: Gibt die Bittiefe des Bildes an.
b_camtype: Gibt den Kameratyp an.
Berechnet die Digitizer (Kamera) Bildtiefe (8bit, 10 bit....).

m_F_d_Handmm_Frame(int n_digframerate):
n_digframerate: Gibt die Bildwiederholungsrate der Kamera an.
Berechnet wie viele mm die Hand maximal pro Bild bewegt werden kann (Trackingkamera):

m_F_d_workspace(int n_dist,int n_focaldist,int n_chiplength,int n_campixel):
n_dist: Gibt den Abstand zwischen Kamerachip und **Arbeitsmittenebene** an.
n_focaldist: Gibt die Brennweite des Objektives an.
n_chiplength: Gibt die Chipabmessungen an.
n_campixel: Gibt die Kameraauflösung in Pixel an.
Berechnet den **Arbeitsbereich** über optische Gleichungen in Abhängigkeit der Kamera und Objektive.

m_F_d_Pixel_mm(bool b_camtype):
b_camtype: Gibt den Kameratypen an (Trackingkamera oder Stereokamera).
Gibt das Verhältnis Pixel zu mm an in Abhängigkeit der Kamera.

m_F_n_HandPixel_Frame(void):
Berechnet die maximale Handbewegung pro Frame in Pixel (Trackingkamera).

m_F_n_MovBorderPixel(void):
Berechnet die Pixelanzahl des Randbereiches.

m_F_n_MovRadius(void):
Berechnet den Bewegungsradius mit maximaler Handgeschwindigkeit plus Randbereich (für Bildzuschnittverfahren).

m_F_n_SichRandPixel(void):
Berechnet den Sicherheitsradius des gesamten Trackingbildausschnittes in Pixel.

m_F_n_MaxHandPixelVel(void):
Berechnet die maximale Handgeschwindigkeit in Pixel.

m_F_d_MaxHandPixelAcc(void):
Berechnet die maximale Handbeschleunigung in Pixel.

m_F_d_Laser_Pic_Offset_posPixelx(void):
Rechnet den Offset der Kameras Laserposition in Pixel um.

m_F_d_Workareadist_pixel(bool b_camtype):
b_camtype: Gibt an, um welchen Kameratypen es sich handelt (Trackingkamera oder Stereokamera).
Berechnet die jeweiligen Kameraabstände zur Bildmittenebene in Pixel.

m_F_n_MaxTrackStepTime(int n_digframerate):
n_digframerate: Gibt die Bildwiederholungsrate an.
Berechnet die jeweilige Schrittweite zwischen den einzelnen Blobkoordinatenpunkten (des **Trackingdevices**).

m_F_n_MaxTrackStepStartTime(int n_digframerate):
n_digframerate: Gibt die Bildwiederholungsrate an.
Berechnet die Startschrittweite des **Trackingdevices** aufgrund der ermittelten Minimalzeit.

m_F_d_CamPixSize(bool b_SizeDirection):
b_SizeDirection: Gibt an, ob die Breite oder die Höhe des Chips zur Berechnung herangezogen wird.
Berechnet die Pixelgröße in Abhängigkeit der Kameraauflösung und Chipgröße in x Richtung.

m_F_d_mm_to_Pixel(bool b_camtype, int n_valuenr, int n_writenr):
b_camtype: Null gilt für die Trackingkamera, und eins für die Stereokameras.
n_valuenr: Gibt die Nummer der Variable ein, die aus der Ini- Datei ausgelesen werden soll.
n_writenr: Gibt die Nummer der Variable ein, die in die Ini- Datei geschrieben werden soll.
Allgemeine Funktion zur Umrechnung mm in Pixel in Abhängigkeit der Kamera.

m_F_n_CalcFramerate(bool camtype) camtype: Null gilt für die Trackingkamera, und eins für die Stereokameras.
Berechnung der eingestellten Kameraframerate aufgrund der Pixelrate, die seriell ausgelesen werden muss.

m_F_d_Std_mm_to_Pixel(bool b_cam,double d_mmval):
b_cam: Gibt den Kameratypen an.
d_mmval: Gibt den Wert in mm an.
Allgemeine Umrechnung mm in Pixel.

Kurze Programmerläuterung:
Aufgabe: Berechnungen von Konstanten für das gesamte Projekt.

- Bauererr.cpp (h)
  404 Zeilen (28 Zeilen)

Notwendige externe Dateien:

cpp-Dateien:
Header Dateien:
Initstart.h
Iostream.h
Errdef.h

Klasse:
C_BauerErr

Klassenfunktionen:

m_F_n_Err(int n_ErrNr=0,int n_TrackErrNr=0, const char* const_p_c_File=NULL,int n_Line=0-
,bool ErrType=1):
n_ErrNr: Gibt die Fehlernummer an.
n_TrackErrNr: Gibt die Fehlernummer des Tracking- Devices an.
const_p_c_File: Gibt den Dateinamen der betroffenen Datei an.
n_Line: Gibt die Zeile an, in welcher der Fehler ausgelöst worden ist.
ErrType: Gibt den Fehlertypen an.
Ist eine Fehlerbehandlungsfunktion.

m_F_n_ErrOut(int n_ErrNr):
n_ErrNr: Gibt die Fehlernummer an.
Fehlermeldungsübersetzung der Fehlernummern.

Kurze Programmerläuterung:
Aufgabe: Fehlerbehandlung, Ausgabe von Fehlermeldungen.

Eidestattliche Erklärung

Ich erkläre hiermit, dass ich diese Diplomarbeit/Dissertation selbstständig ohne Hilfe Dritter und ohne Benutzung anderer als der angegebenen Quellen und Hilfsmittel verfasst habe. Alle den benutzten Quellen wörtlich oder sinngemäß entnommenen Stellen sind als solche einzeln kenntlich gemacht. Diese Arbeit ist bislang keiner anderen Prüfungsbehörde vorgelegt worden und auch nicht veröffentlicht worden. Ich bin mir bewusst, dass eine falsche Erklärung rechtliche Folgen haben wird.

7.2 Erklärungen

[Abbildungsmittenebene]	Ist die mittlere Abbildungsebene (1800mm Abstand zur Kamera), früher Tischebene
[Abbildungsoberebene]	Ist die oberste sich noch im Arbeitsbereich befindliche Ebene
[Abbildungsunterebene]	Ist die unterste sich noch im Arbeitsbereich befindliche Ebene
[Bewegungsbild]	Ist das Ergebnis zweier direkt hintereinander erstellten Bilder, deren Helligkeitswerte der einzelnen Pixel voneinander angezogen werden
[Bildbearbeitung]	Ist ein Algorithmus zur Änderung der Bildeigenschaften
[Blob]	ugs. Klacks, entspricht in diesem Fall dem Ausschnitt eines Bildes, dessen Pixel farblich zusammenpassen (in diesem Projekt weiß)
[Entzerrungsmatrix]	Ist eine Vorlage (Schablone) mit 5 Punkten (vier an dern Rändern und einem in der Mitte), um die Tiefenentzerrung zu berechnen.
[Kalibrationsmatrix]	Ist eine Schablone zur Kameralinsenentzerrung
[Koordinatengeber]	Dient zur direkten Ansteuerung der Maschine. Dabei wird ein Ball oder Ähnliches auf einen Handschuh geklebt um mit diesem über die Kameras und der optischen Bildverarbeitung die Maschine geführt
[Pan-Tilt-Device]	Gerät um Stereokameraträger zu drehen und zu schwenken (Schwenk-Neigegerät)
[Power Cube]	siehe Pan-Tilt-Device
[Tracking Device]	siehe Pan-Tilt-Device

Table 7.1: Erklärungen

List of Figures

Wissenschaftliche Literatur

[1] B. A. Echtzeit kollisionsvermeidung für einen industrieroboter durch 3d-sensorüberwachung. In *IEEE*, 1999. 16

[2] B. Christian. *Dokumentation Teil 1 Trackingkamerabildbearbeitung*. Bauer Christian, Austria, 2006. 18, 30, 54, 55, 56, 59, 61, 62, 64, 66, 67, 72

[3] W. F.Y. A simple and analytical procedure for calibrating extrinsic camera parameters. In *IEEE*, 2004. 15

[4] Matrox. *Matrox Imaging Library 8*. Matrox, Canada, 2005. 40, 59, 69, 80

[5] A. N. Resolution limitations and error analysis for stereo camera models. In *IEEE*, 1988. 9

[6] G. Rudolf. *XX*. Gelhart Rudolf, Austria, 2006. 30, 93

[7] C. C. und Chatterjee S. Depth from stereo image flow. In *IEEE*, 1989. 17

[8] I. I. und Chella A. und Dzindo H. und Macaluso I. Visual control of a robotic hand. In *IEEE*, 2003. 8

[9] R. G. und Dr. Aengus M. Intelligent tracking camera system. In *IEEE*, x. 6

[10] H. M. und Gordon G. und Woodfil J. Adaptive video background modeling using color and depth. In *IEEE*, 2001. 6

[11] H. K. und Hasegawa T. und Kiriki T. und Matsuoka T. Real time pose estimation of an objekt manipulated by multi- fingered hand using 3d stereo vision and tactile sensing. In *IEEE*, 1998. 4

[12] M. M. und Hayashi T. und Tominaga H. Video tablet based on stereo camera - human-friendly handwritten capturing system for educational use. In *IEEE*, 2005. 11

[13] S. A. und Hjelmstad K.D. und Huang T.S. Nondestructive evaluation of civil structures and materials using stereo camera measurements. In *IEEE*, 1992. 12

[14] S. Y. und Ishijama Y. und Tomita F. Hyper frame vision: A real time vision system for 6-dof object localisation. In *IEEE*, 2002. 9

[15] O. M. und Itoh M. und Nakamura Y. und Ohta Y. Tracking hands and objects for an intelligent video production system. In *IEEE*, 2002. 7

[16] I. Y. und Katsuhiko S. Concept of ubiquitous stereo vision and applications for human sensing. In *IEEE*, 2003. 7

[17] R. P. und Kramberger I. Six degrees of freedom stereovision inside-out tracking. In *IEEE*, 2003. 13

[18] X. M. und Lee C.M. und Li Z.Q. und Ma S.D. Depth assessment by using qualitative stereo-vision. In *IEEE*, 1997. 12

[19] H. H. und Mitsunori O. und Mamoru Y. und Yoshinori N. und Kazuhiko Y. Focus of attention for face and hand gesture recognition using multiple cameras. In *IEEE*, 1998. 4, 5

[20] I. M. und Ozeki M. und Nakamura Y. und Ohta Y. Simple and robust tracking of hands

and objects for video- based multimedia production. In *IEEE*, 2003. 7

[21] L. V. und Siciliano B. and V. L. Robust visual tracking using a fixed multi-camera system. In *IEEE*, 2003. 9

[22] Y. Y. und Tsuji Y. und Asada M. und Hosoda K. View-based imitation with rotation invariant pan- tilt stereo cameras. In *IEEE*, 2002. 16

[23] H. Y. und Yangz Y.S. und Cheny Y.S. und Hsiehz I.B. und Fuhz C.S. Free-hand pointer by use of an active stereo vision system. In *IEEE*, x. 14

[24] Y. Y. und Yoda I. und Sakaue K. Arm-pointing gesture interface using surrounded stereo cameras system. In *IEEE*, 2004. 10

[25] x. advanced realtime tracking. In *IEEE*, x. 14

[26] x. Immersion cyberglove. In *IEEE*, x. 14