Inbetriebnahme und Validierung eines Roboterkonstruktionssatzes mit Kamerasensorsystem

Bericht zum Laborpraktikum

von

Peter Knüppel

knueppel@cs.uni-magdeburg.de

17. August 2007

Betreuer:

Dipl.-Ing. Sebastian Zug

Otto-von-Guericke Universität Magdeburg
Fakultät für Informatik

Institut für Verteilte Systeme

Zusammenfassung: Der „VolksBot“ ist ein Roboter-Entwicklungskit vom Fraunhofer Institut für autonome intelligente Systeme. In diesem Report wird das Kit in seiner 3-rädrigen Outdoorausführung „VolksBot RT 3-wheeled“ mit der zugehörigen Kamera-Sensorkomponente AISVision vorgestellt und die aus praktischen Versuchen mit dem System gewonnenen Erfahrungen dargestellt. Schwerpunktmäßig wird die nötige Vorgehensweise bei der Inbetriebnahme des Systems geschildert sowie Einsatzmöglichkeiten und hardwaretechnische Grenzen des Kamera-Sensorsystems aufgezeigt. Hierzu wurden diverse Messungen an dem Sensor durchgeführt und dessen Eckparameter bestimmt. Schließlich wird eine mit Hilfe der grafischen Programmierumgebung IConnect, der Kernkomponente zur Entwicklung eigener Anwendungen für den VolksBot, erstellte Beispielapplikation vorgestellt.

Inhaltsverzeichnis

1 Einleitung

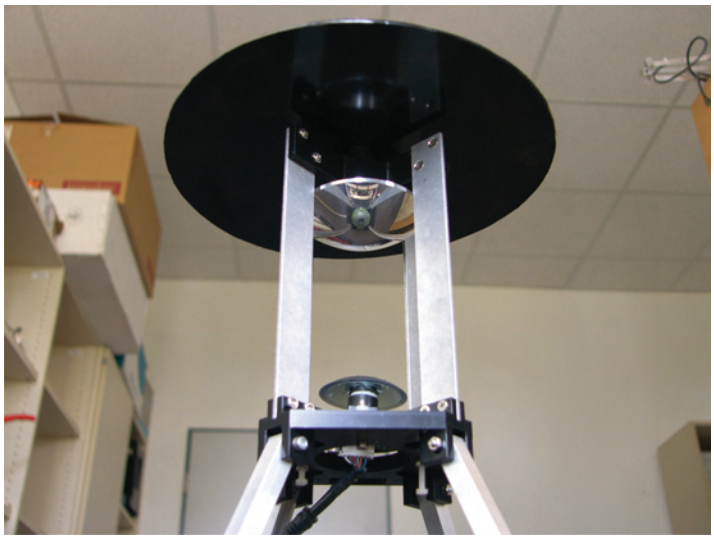

Abbildung 1.1: Foto des AISVision-Sensors bestehend aus Spiegel und Kameramodul

Der Roboterkonstruktionssatz „VolksBot RT 3-wheeled“ wurde vom Fraunhofer Institut für autonome intelligente Systeme für die schnelle Entwicklung von Roboterprototypen zu Lehr- und Forschungszwecken konzipiert. Um diesem Zweck auf möglichst universelle Weise gerecht zu werden, bietet das Fraunhofer Institut für autonome intelligente Systeme (AIS) passend zu dem Grundbausatz bestehend aus Fahrwerk, Motor, Motorcontroller und weiteren Komponenten verschiedenste auf den Grundbausatz abgestimmte Aktuatoren und Sensorsysteme an. Standardmäßig zum Roboterkit gehört das omnidirektionale Kamera-Sensorsystem AISVision, welches aus der nötigen Kamerahardware (Firewire-Kameramodul, Spiegel, Halterung – siehe Abbildung 1.1) sowie Software-Bildverarbeitungsmodulen für die Entwicklungsumgebung IConnect besteht. In IConnect erfolgt über die Kombination dieser und anderer Module per Drag and Drop die Entwicklung der eigentlichen Roboterapplikationen. Dabei werden Entwickler durch im Roboter-Kit enthaltene, grundlegende Dokumentationen unterstützt. In diesem Report wird ergänzend zu diesen Anleitungen dargestellt, welche Probleme bei der Einrichtung des Roboters mit dem Kamerasensors in der Praxis aufgetreten sind und wie diese im Einzelnen gelöst wurden. Zusätzlich werden durchgeführte Messungen und Versuche am Kamerasystem dokumentiert und Rückschlüsse auf die Verwendungsmöglichkeiten des Sensorsystems gezogen.

2 Hard- und Software: Inbetriebnahme und Überblick

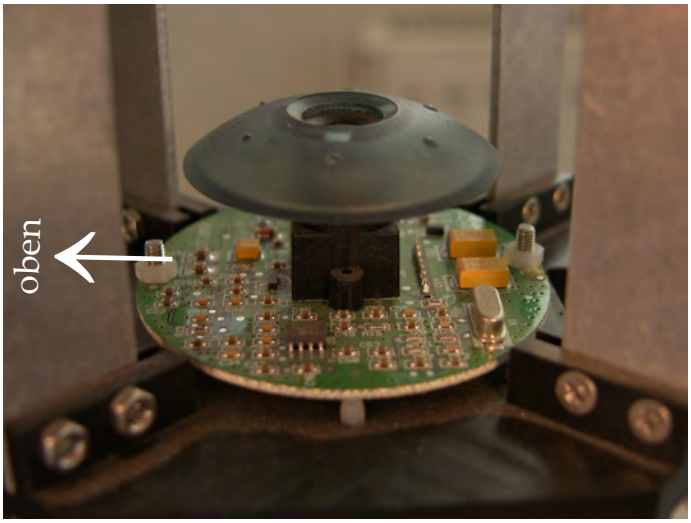

Abbildung 2.1: Nahaufnahme des Kameramoduls

Hardwareseitiger Kernbestandteil des VolksBot-Bausatzes ist ein vormontiertes, $40 \cdot 40\,\text{cm}$ großes Chassis, an dem lediglich noch Akkus und Notebook befestigt werden müssen. Weiterhin muss der Kamerasensor montiert werden, was bei vier verschiedenen Montagerichtungen wider erwarten nicht trivial ist: Es muss darauf geachtet werden, dass die oberste Bildzeile der Kamera in Richtung der Antriebsachse des Roboters zeigt (siehe Pfeil in Abbildung 2.1).

Die weitere Montage beschränkt sich auf die Verkabelung von Stromversorgung und Datenanschlüssen gemäß der dem VolksBot mitgelieferten Kurzanleitung [AIS04b]. Hier gestaltete sich wiederum lediglich der Anschluss der Kamera schwierig:

2.1 Anschluss der Kamera: Stromversorgung und Datenübertragung

Mit Strom versorgt wird der Motorcontroller und die Motoren des VolksBot von zwei in Reihe geschalteten 12 V Bleigel-Akkus. Während das Notebook durch seinen eigenen Akku versorgt wird, ist es nötig, der Kamera als dritten und letzten Verbraucher eine Betriebsspannung

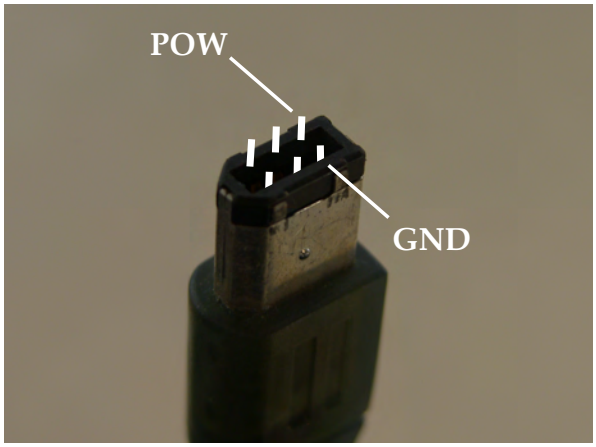

Abbildung 2.2: PINs zur Stromversorung an 6-poligem Firewire-Stecker (männlich, nicht Buchse)

von 12 V zur Verfügung zu stellen. Diese muss an den entsprechenden PINs des 6-poligen IEEE-1394 (Firewire-) Steckers der Kamera anliegen (siehe Abbildung 2.2).

Heutige Notebooks verfügen meist über eine Firewire-Buchse, jedoch nur in der 4-poligen Mini-Ausführung für batteriebetriebene Geräte, bei der die Pins zur Stromversorgung fehlen [Wef00]. Ein entsprechendes Adapterkabel war im Handel nicht erhältlich und wurde deshalb testweise selbst angefertigt. Mit ihm gelang die Stromversorgung problemlos; obwohl keine Abschirmung des Steckers vorgenommen wurde, wurde die Datenübertragung zur Kamera hin ebenfalls nicht beeinträchtigt.

In der Dokumentation zum VolksBot wird dagegen zur Verwendung einer PCMCIA-Firewirekarte mit 6-poligen Anschlüssen geraten, welche ihren Strom aus einer der 12 V Akkus des VolksBots bezieht. Hierzu muss ebenfalls zunächst ein Adapterkabel (5 mm Kabelschuh auf den entsprechenden Anschluss der PCMCIA-Karte) angefertigt werden.

Bei zwei getesteten Notebooks mit eingebautem Firewire-Anschluss war es jedoch nicht möglich, die Kamera über einen zusätzlichen PCMCIA-Firewire-Controller anzusteuern. Die Kamera zeigte keinerlei Funktion. Aus diesem Grund wurden exemplarisch einige weitere Firewire-Karten und Notebooks auf ihre Kompatibilität mit der Kamera hin untersucht – mit dem Ergebnis, dass die meisten Kombinationen nicht korrekt funktionierten. Getestet wurde jeweils mit zwei AISVision-Sensoren um eine Fehlfunktion der Sensoren ausschließen zu können. Die Tests fanden, falls nicht anders angegeben, unter dem Betriebssystem Microsoft Windows XP SP 2 statt. Als Treiber für die Kamera wurden jeweils der CMU Open-Source Firewire-Kameratreiber der in der Version 6.1.3 (mit dem VolksBot mitgeliefert) und in der zum Testzeitpunkt aktuellsten Version (6.4.4) verwendet [Bak06]. Folgende funktionierende Konfigurationen konnten ermittelt werden:

Notebook IBM Thinkpad R51: Keine Funktion mit eingebautem Firewire-Adapter (unter Suse Linux 10.2 verzerrtes, unbrauchbares Bild) – aber einwandfreie Funktion nur mit einer PCMCIA-Firewirekarte von SIIG („1394 FireWire Dual port CardBus Model

#NN2612“[1]), wenn die ins Notebook eingebaute Karte im BIOS deaktiviert wird.

Notebook IBM Thinkpad R60, Dell Latidude X1: Einwandfreie Funktion mit dem internen Firewire-Adapter bei Verwendung der CMU-Treiberversion 6.4.4. Beim R60 funktionierte zusätzlich auch die PCMCIA-Firewirekarte von SIIG.

Notebook IBM T42p: Einwandfreie Funktion mit der Firewirekarte von SIIG und dem mit dem VolksBot mitgelieferten Treiber.

Mit einer neueren PCMCIA-USB/Firewire Adapterkarte von Conrad Elektronic[2] ließ sich die Kamera dagegen unabhängig vom Notebook überhaupt nicht ansprechen. Unter der Linuxdistribution OpenSuse 10.2, bei der die Kamera als standardkomforme IEEE 394 Industriekamera problemlos erkannt wird, lieferte sie ebenfalls ein grünstichiges, stark verzerrtes und nicht weiter verwendbares Bild. Zum Testen wurde hierfür die Anwendung Coriander [Dou07] verwendet.

Zur Inbetriebnahme des Gesamtsystems ist daher unbedingt eine Recherche nach einer kompatiblen Kombination aus Notebook und Kamera einzuplanen. Ob eine bestimmte Kombination funktioniert, ist leicht mit Hilfe des mit dem Firewire-Kamera-Treiber mitgelieferten Testprogramms „Camera Demo“ [Bak06] feststellbar: Zeigt die Kamera hier ein stark verzerrtes Bild oder lässt sich überhaupt nicht ansprechen („Problem acquiring image“), funktionierte sie auch in den weiteren Applikationen nicht oder nicht korrekt.

2.2 Die Softwarekernkomponente: Die Entwicklungsumgebung IConnect

Bestandteil des Roboterkits sind verschiedene Applikationen, die eine schnelle Anwendungsentwicklung unterstützen sollen (siehe Abbildung 2.3). Kernkomponente ist die grafische Entwicklungsumgebung IConnect. In ihr erfolgt die Entwicklung per Drag-and-Drop durch die Anordnung und Verdrahtung von Modulen mit unterschiedlicher Funktionalität in einem Signalflussgraphen. Ein solcher Flussgraph ist dabei nicht nur eine Abstraktion bzw. Modell der Applikation, sondern stellt gleichzeitig die eigentliche Implementierung dar.

IConnect enthält eine Vielzahl von verkettbaren Modulen, die sich in die Rubriken Schnittstelle (z. B. ein RS232 Ein- und Ausgabemodul), Verarbeitung (z. B. Fourier-Transformation, Matrixberechnungen) oder Darstellung (Videoanzeige, Balkendiagramm, ...) einteilen lassen. Ebenfalls lassen sich über weitere Module „klassische“ GUI-Eingabeformulare erstellen, mit denen sich Anwendungen interaktiv steuern lassen. Auch eine große Anzahl an Modulen speziell für Bildverarbeitungsaufgaben (Filter, Segmentierung, morphologische Operatoren, ...) stehen zur Verfügung und sind allesamt recht einfach zu handhaben.

[1] http://www.siig.com/
[2] Conrad Electronic 32 bit CardBus 1384a & USB 2.0 4 Ports – Model 5-PC103-01A

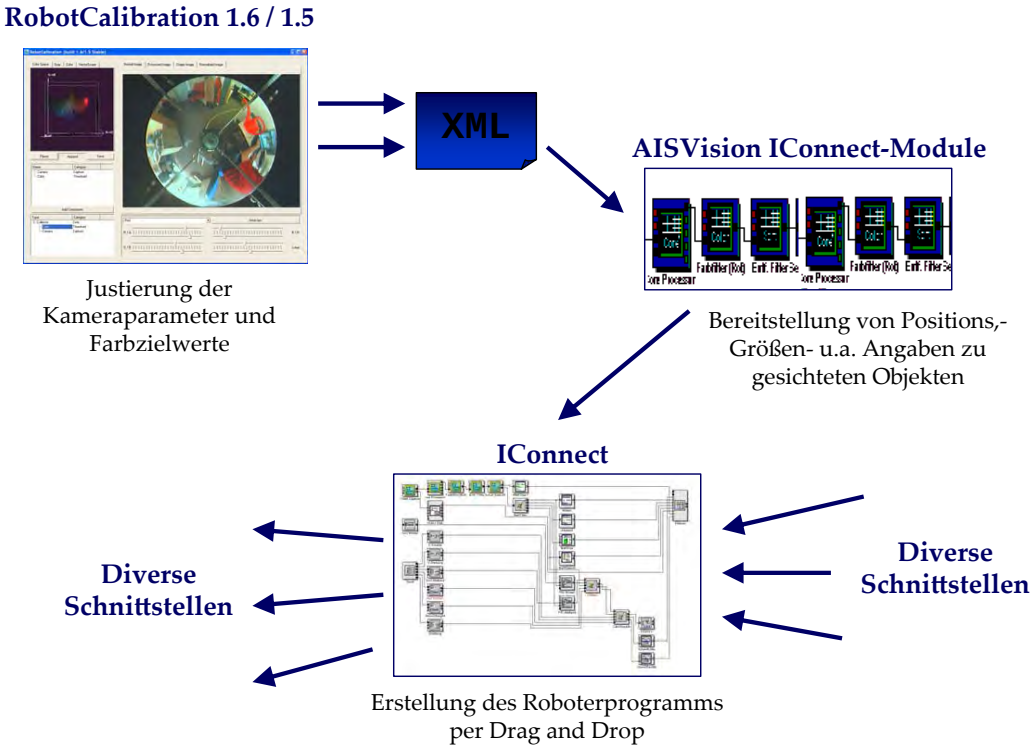

Abbildung 2.3: Überblick über die AISVision-Kernkomponenten zur Entwicklung eigener Roboterapplikationen

Zwar ist die Anordnung von Icons gegenüber der Programmierung in einer imperativen Programmiersprache zunächst nicht grundlegend einfacher, wie es auf den ersten Blick bei der Betrachtung eines zumeist recht intuitiv und verständlich anmutenden Flussgraphen scheinen mag (vgl. Abbildung 2.4). Insbesondere ist jedoch die Konvertierung zwischen den verschiedenen Datentypen in ICONNECT nicht trivial und erfordert anfangs viel Sucharbeit, um das jeweils passende Modul für die Ausgabe eines anderen Moduls zu finden. Auch das Triggerverhalten und andere Eigenheiten von den jeweiligen Bausteinen müssen beachtet werden. Hierbei hilft jedoch die ausgezeichnete Online-Dokumentation von ICONNECT gut weiter. Zu jedem Modul gibt es ein Beispielprogramm, dass dessen korrekte Verwendung demonstriert - zu finden direkt per Rechtsklick auf das jeweilige Modul. Der Einstieg in die generelle Arbeitsweise von ICONNECT wird durch das Tutorial „IConnect in 60 Minuten“ [ME04] zudem sehr vereinfacht und kann allen Erstanwendern empfohlen werden.

Probleme bereitete allerdings die Stabilität von ICONNECT: Komplexere Signalgraphen, die insbesondere durch den Einsatz von Visualisierungen sehr viel Rechenzeit benötigten, führten häufig dazu, dass der Signalgraph nicht mehr gestoppt werden konnte und ICONNECT neu gestartet werden musste. Zumindest führte das jedoch zu keinerlei Datenverlust. Weiterhin war es mit der vorliegenden ICONNECT-Version nicht möglich, ein wirklich eigenständig ausführbares Programm oder eine Bibliothek zu generieren. Vielmehr wird bei der Wahl des entsprechenden Menüpunktes eine Art Runtime-Version von ICONNECT erzeugt, in der dann der erstellte Graph ausgeführt werden kann. Auch können in der dem VolksBot beiliegenden Version von ICONNECT keine eigenen, hardwarenahen Erweiterungsmodule in C++ integriert

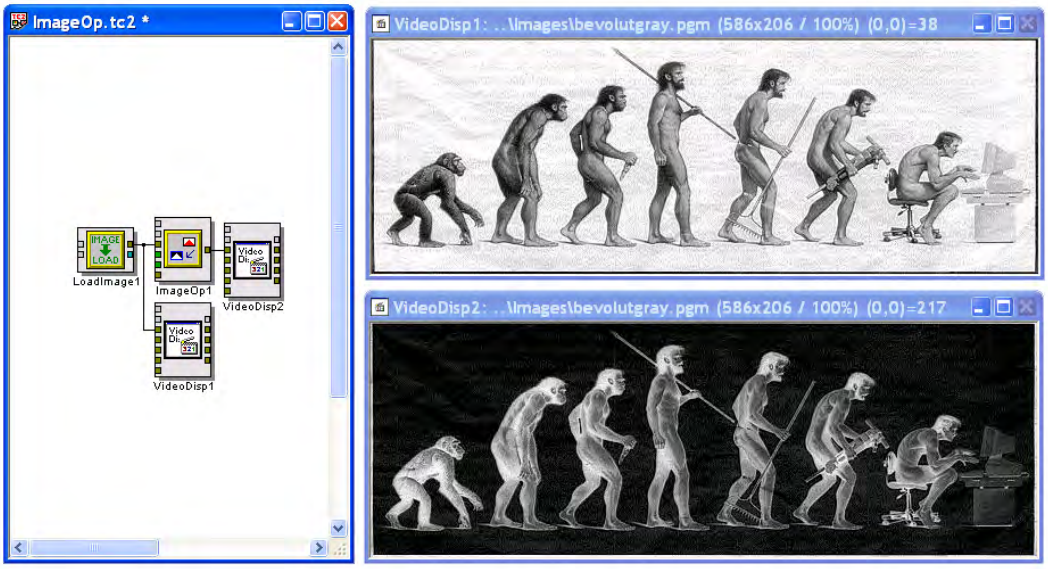

Abbildung 2.4: Einfacher Flussgraph (=Programm) in IConnect

werden – hierfür ist eine zusätzliche Lizenz erforderlich.

2.3 AISVision Signalverarbeitungsmodule für IConnect

Zusätzlich zu den in ICONNECT standardmäßig vorhandenen Modulen sind für den Volks-Bot spezielle AISVISION-Module verfügbar, mit denen die grundlegende Verarbeitung der Kameradaten geschieht. Die Module übernehmen dabei im einzelnen die im folgenden näher erläuterten Funktionen.

2.3.1 Capturing

Das Capturing-Modul „1394 Capture“ übernimmt die Konfiguration der Kamera anhand von XML-Dateien und die Kommunikation mit dem Firewire-Treiber. Das Modul stellt das Rohbild in einem nicht dokumentierten Datenformat an seinem Ausgang zur Verfügung („Vision Out“). Alternativ zur Verwendung des Live-Bildes des Kamerasensors lässt sich auch eine AVI-Datei als Eingangsstream auswählen (siehe Abb. 2.5), über die z. B. eine zuvor aufgezeichnete Roboterfahrt mehrfach reproduziert werden kann.

2.3.2 Segmentierung und Generierung von Blobs

Das „Core Processing“-Modul nimmt die Daten des Capturing-Moduls entgegen und segmentiert die Frames in Bereiche ähnlicher Farbe. Das hierzu verwendete Segmentierungsverfahren

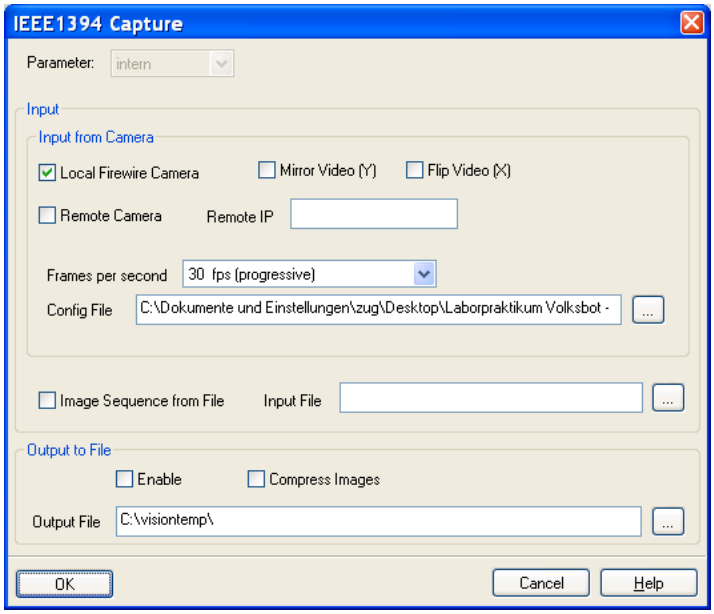

Abbildung 2.5: Screenshot der Optionen des Capturing-Moduls

ist sehr einfach und basiert auf vorher fest zu definierenden Schwellwerten für jede Farbkomponente. Diese Schwellwerte werden ebenso wie die Kameraparameter aus einer XML-Datei eingelesen. Bereiche, die innerhalb gleicher Schwellwertgrenzen liegen und aneinander angrenzen, werden zu „Blobs“ zusammengefasst. Ein Blob repräsentiert also eine Fläche mit annähernd gleicher Farbe. Das Modul berücksichtigt hierbei die Eigenheiten des Sensors und erkennt auch Bereiche als ein zusammenhängendes Blob, die durch eine Strebe der Spiegelhalterung geteilt sind (siehe z. B. Kamerabild in Abb. 2.9).

Zu jedem dieser Blobs werden Informationen zu dessen Größe und Position ermittelt (Breite in Grad, Abstand in cm, Fläche usw.), die ein weiteres Modul, das „Serial Output“-Modul, in eine Matrix umwandeln kann. Diese Matrix kann dann in beliebigen ICONNECT-eigenen Modulen weiterverarbeitet werden. Die Syntax dieser Matrix (das heißt Umfang und Reihenfolge der enthaltenen Daten) ist in der Onlinehilfe zu den Modulen dokumentiert, allerdings unterschiedet sich diese Dokumentation von der in einem dem VolksBot-Kit beigelegten Beispielprogramm verwendeten (und im Versuch als korrekt ermittelten) Syntax.

Eine wichtige Auswahlmöglichkeit, die im Konfigurationsdialog zum Core Processing-Modul besteht (vgl. Abbildung 2.6), betrifft die Auflösung, mit der die Segmentierung vorgenommen wird: Hier kann zwischen Varianten mit über das gesamte Sensorbild konstanter Auflösung und Varianten, die eine gleichbleibende Auflösung in der Realwelt bieten, gewählt werden. Die höchstmögliche Auflösung liegt bei 5 cm bzw. 8% der ursprünglichen Pixeldaten. Es können auch noch geringere Auflösungswerte verwendet werden.

Einen großen Nachteil stellt hierbei dar, dass diese starke Datenreduktion stets global auf das gesamte Kamerabild angewendet wird. Sie lässt sich nicht für bestimmte Bildbereiche abschalten (z. B. volle Auflösung im Bereich von 0-10 Grad; im restlichen Blickfeld reduzierte

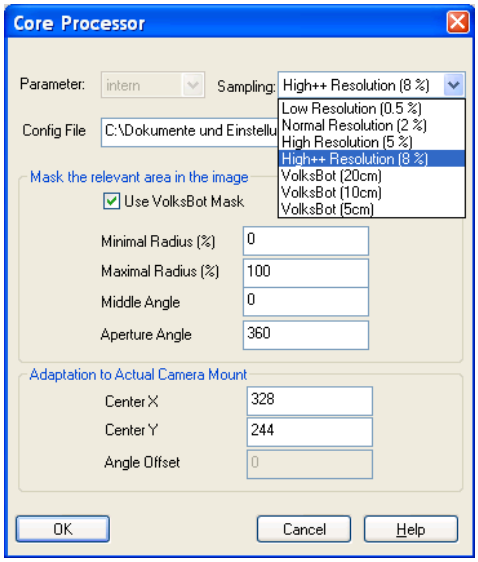

Abbildung 2.6: Screenshot AISVision-Modul: Core Processor zur Segmentierung und Blobbildung

Auflösung). Inwiefern der Kamerasensor selbst ein deutlich höheres Auflösungsvermögen ermöglichen würde als die maximal wählbaren 5 cm, wird in Abschnitt 3.2 noch ausführlicher dargestellt.

Neben dieser unflexiblen Art der Datenreduktion ist ist das Core Processing-Modul in einem weiteren, entschiedenen Punkt eingeschränkt: Das Segmentierungsverfahren des Bildes ist fest auf die erwähnte Schwellwertsegmentierung mit festen Schwellwerten festgelegt. Komplexere Segmentierungsverfahren – zum Beispiel auf Textur oder Konturbasis – sind in dem Modul nicht vorgesehen und somit nicht nutzbar.

2.3.3 Sortieren und Filtern von Blobs:

Ein weiteres Modul ermöglicht die Sortierung von Blobs nach Kriterien wie Größe und Entfernung (siehe Screenshot in Abbildung 2.7). Mit diesem Modul soll offenbar auch Objekt-Tracking möglich sein (Sortierung nach Objektalter) – dies funktionierte jedoch nicht und war auch nicht dokumentiert. Die Onlinehilfe (Abb. 2.8) und das tatsächliche Modul unterscheiden sich in diesem Punkt. Es fehlt weiterhin die Möglichkeit zur gezielten Selektion von nur n Blobs (z. B. „betrachte nur die nächsten 10 Objekte“) – dies muss manuell über ein „Interpret“-Modul geschehen, über das frei JavaScript-artiger Quellcode in ICONNECT eingefügt werden kann.

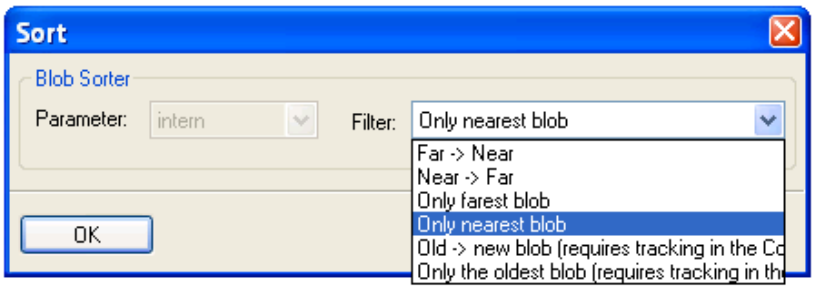

Abbildung 2.7: Screenshot eines AISVision-Moduls: Sortierung von Blobs

Abbildung 2.8: Zu Screenshot 2.7 zugehörige, unvollständige Onlinehilfe

2.3.4 Visualisierung von Blobinformationen:

Schließlich ist es auch möglich, mit Hilfe des Serialisierungs-Moduls die gefundenen Blobs zu visualisieren. Das Modul kann hierzu eine Vektorgrafik generieren, in der wahlweise einzelne Punkte eingezeichnet werden – repräsentativ für Abtastpunkte, die zwischen den Schwellwerten liegen – oder auch Richtungspfeile und grobe Konturen (siehe Abb. 4.1).

Leider sinkt bei sehr vielen visualisierten Objekten hierbei die ansonsten auf aktuellen Notebooks akzeptable Performance der Signalgraphen immens ab. Um eine hohe Framerate (und damit eine hohe Reaktionsgeschwindigkeit) zu erzielen, sollte möglichst nur ein Blob oder einige wenige von Bedeutung über dieses Modul visualisiert werden.

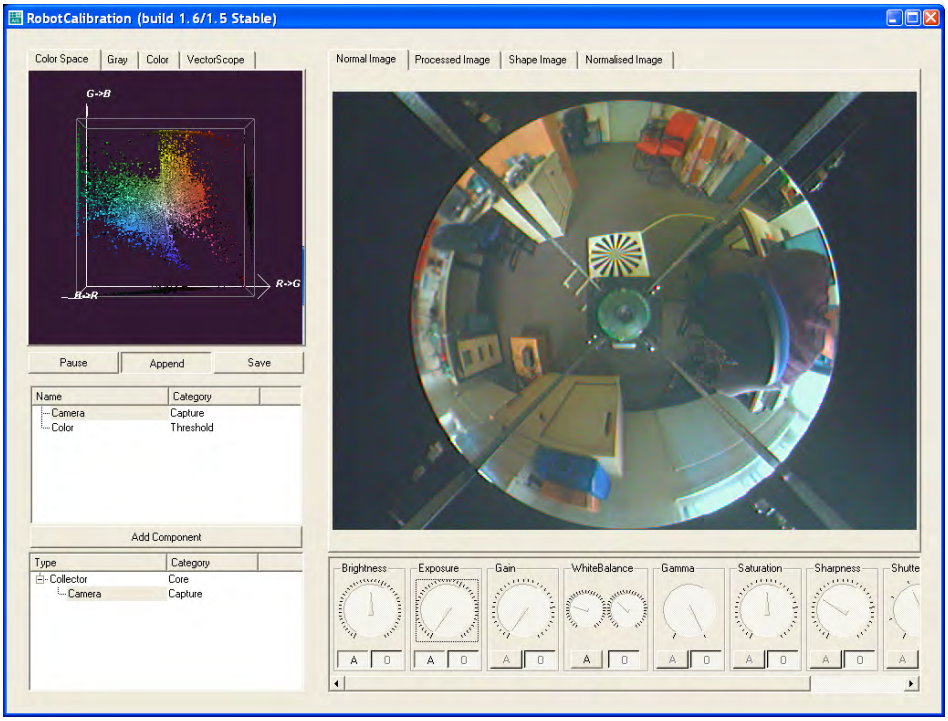

Abbildung 2.9: Screenshot der Anwendung „Robot Calibration“: Justierung der Kameraparameter über Drehregler; Visualisierung des dreidimensionalen Farbraums als drehbare Punktwolke

2.4 Kalibrierung des Kamerasensorsystems

Zur Kalibrierung des Kamerasensorsystems steht die Applikation „RobotCalibration“ [AIS04a] mit einem grafischem Nutzerinterface zur Verfügung. Mit Hilfe dieser Anwendung lassen sich sowohl Kameraeinstellungen (Belichtung, Weißabgleich, ...) als auch Farbbereiche festlegen, welche anschließend als XML-Dateien mit einem proprietären Schema exportiert werden können. Über diese XML-Dateien erfolgt schließlich die Steuerung der entsprechenden AISVision-Module in ICONNECT.

Die Kalibrierung erfolgt dabei in zwei Schritten. Zunächst werden mit Hilfe von stilisierten Drehreglern die Kameraparameter so justiert, dass die gewünschten Farbbereiche visuell gut zu differenzieren sind. Anhand mehrerer Echtzeit-Visualisierungen (z. B. 3D-Farbwürfel, Histogramme zur Belichtung, Vektordiagramm für den Weißabgleich) lassen sich die Einstellungen einfach regeln. Dabei kann bei verschiedenen Parametern je nach Beleuchtungssituation zwischen automatischer und manueller Einstellung gewählt werden. Als nächstes können die Farbschwellwerte durch Klicks in das Live-Bild festgelegt werden – alternativ über Schieberegler, die etwas exakter zu bedienen sind als die virtuellen Drehknöpfe (siehe Abbildung 2.10). Einen absoluten Wert kann man jedoch auch für die Farbschwellwerte nicht direkt eingeben – hierfür bleibt nur die manuelle Bearbeitung der generierten XML-Dateien. Die Syntax dieser ist nicht ausführlich dokumentiert, allerdings recht intuitiv und selbsterklärend.

Auffällig bei der Durchführung der Kalibrierung ist die Fehleranfälligkeit der RobotCalibrati-

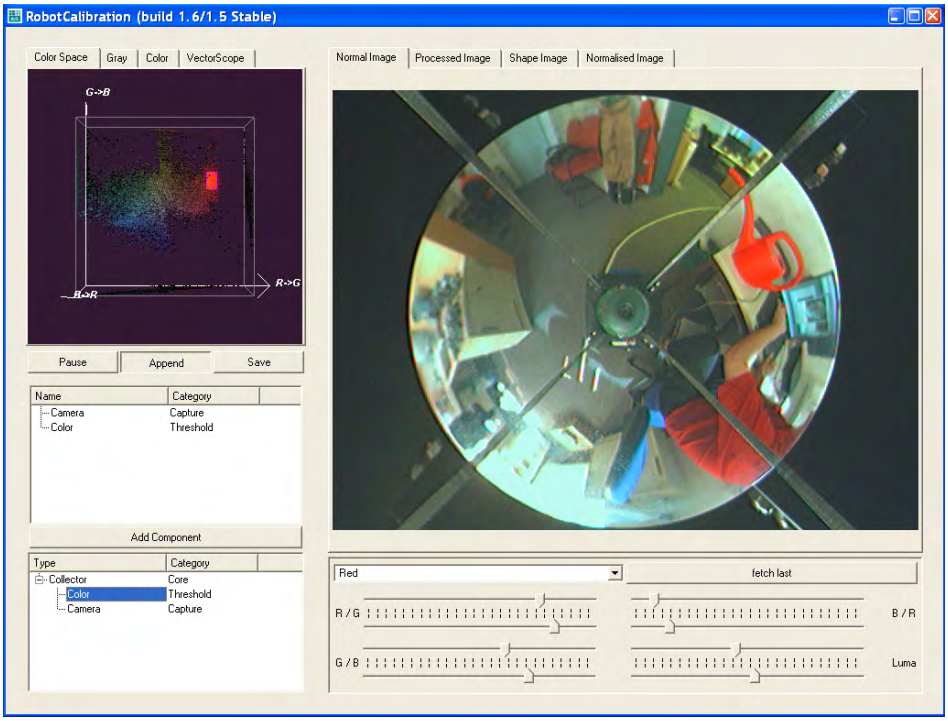

Abbildung 2.10: Setzen der Farbschwellwerte in der Anwendung RobotCalibration

on-Anwendung: Ein Klick in das Vorschaubild der Kamera direkt nach Programmstart führt unmittelbar und reproduzierbar zum Absturz der Applikation.

3 Eigenschaften und Leistungsfähigkeit des Kamerasensorsystem

Im nachfolgenden Abschnitt werden Untersuchungen geschildert, die an dem Kamerasensor als solchem, das heißt ohne Nutzung der vorhandenen AISVision-Softwaremodule, durchgeführt wurden. Die Messergebnisse charakterisieren somit die Leistungsfähigkeit der Sensorhardware bestehend aus Spiegel und CCD-Modul in Verbindung mit dem zugehörigen Kameratreiber. Alle Messungen wurden mit Hilfe des 1394-Camera-Demo-Programms [Bak06, Version 6.3] der Treibersoftware durchgeführt.

Parameter	Wert
Auto-Gain	an
Shutter	aus
Saturation	128 von 255
Sharpness	66 von 255
Auflösung	640 * 480
Framerate	30 fps
Farbraum	YUV 4:1:1 (YCbCR)

Abbildung 3.1: Empfehlenswerte Treibereinstellungen für den Betrieb des Kamerasensors

Alle Messungen wurden dabei, insofern nicht anders angegeben, bei weitgehend homogenem Zimmerlicht (Neonlicht, ca. 6000K Farbtemparatur) und den in Tabelle 3.1 aufgelisteten Treibereinstellungen durchgeführt.

Die Parameter für Auflösung, Framerate und Farbtiefe entsprechen damit den Werten, mit denen die AISVision-Softwarekomponenten standardmäßig arbeiten. Die Werte für Schärfe, Verstärkung und Farbsättigung stellten sich als in vielen Beleuchtungssituationen empfehlenswert heraus. Höhere Werte für Schärfe und Sättigung führten teils zu deutlichen Geisterbildern und Farbsäumen, die bei den genannten Werten nicht auftraten.

3.1 Automatische Belichtungssteuerung

Obwohl im Konfigurationsdialog die automatische Belichtungssteuerung (Auto Exposure) und -Helligkeit (Brightness) ausgewählt wurde, müssen die Belichtungszeit (Shutter) bzw. die Blende (Iris) dennoch manuell justiert werden. Die Kamera selbst regelt entgegen den Angaben der Dokumentation des Fraunhofer Instituts [AIS04a] offenbar lediglich den Gain, also

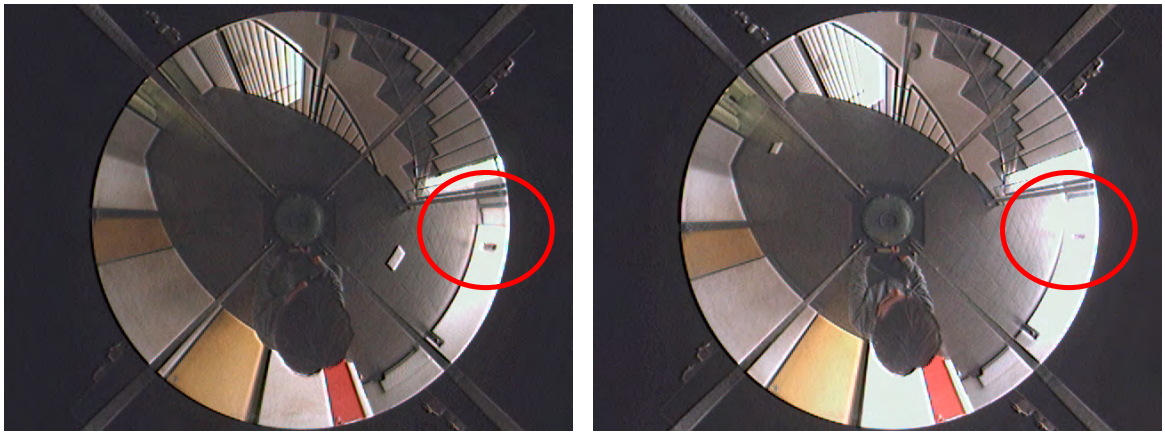

Abbildung 3.2: Dynamikumfang der Kamera: Vergleich zwischen dunkler und heller Blendenstufe

die kamerainterne Signalverstärkung, und nicht die Belichtungszeit oder die Blende. Dies ist daran erkennbar, dass der Regler für die Verstärkung keinen Effekt mehr hat, wenn „Auto Exposure“ gewählt wurde; der Shutter- und der Blendenregler jedoch schon.

Eine automatische Belichtungsanpassung der Kamera ist somit nicht möglich (bzw. nur eingeschränkt über die Verstärkung) und müsste folglich selbst implementiert werden. Ohne Belichtungssteuerung ist davon auszugehen, dass die Kamera in inhomogen beleuchteten Umgebungen (Fahrt von innen nach außen, Fahrten zwischen Sonne und Schatten) ohne jeweils vorher durchgeführte Neujustierung keine verwendbaren Resultate liefert und es zu überbelichteten Bereichen im Bild kommt (siehe Abbildung 3.2).

Offensichtlich auf einen Treiberfehler ist zurückzuführen, dass die Regler für Verschlusszeit und Blendenöffnung vertauscht sind. Die Verwendung eines neueren Treibers für die Kamera könnte hier ggf. Abhilfe schaffen, führt aber zu Problemen bei der Verwendung in Zusammenhang mit ICONNECT (bestimmte Treiberversion erforderlich).

Für die weiteren Tests wurde der Shutter deaktiviert (d. h. die Verschlusszeit entsprach der Dauer eines Frames, also 1/30 s) und die Blende so geregelt, dass die Aufnahmen eher unter- als überbelichtet erschienen. Dabei ist noch erwähnenswert, dass die Kamera über nur 3 verschiedene Blendenstufen verfügt (hell / mittel / dunkel), so dass hier bei der Konfiguration des Roboters wenig Entscheidungsfreiheit besteht. Die Regelung der Verstärkung wurde auf „automatisch“ gestellt – eine Einstellung, die auch bei Verwendung der Kamera in ICONNECT-Roboterapplikationen empfehlenswert ist, da so das Bild zumindest in gewissen Grenzen normalisiert wird.

Messung Nr.	R	G	B	Ø (Grauwert)	**dR**	**dG**	**dB**
1	215	184	145	181	+18,6%	+1,5%	-20,0%
2	232	238	235	235	-1,3%	+1,3%	±0,0%
3	149	137	128	138	+8,0%	-0,7%	-7,2%
4	172	171	182	175	-1,7%	-2,3%	+4,0%
5	126	112	102	113	+11,2%	-1,2%	-10,0%
...	...	...	...	...	...	...	...
				Mittelwert:	**+9,2%**	**-1,3%**	**-7,9%**
				Varianz:	**4,4%**	**1,2%**	**4,0%**

Tabelle 3.1: Messwerte zur Bestimmung der Genauigkeit des Weißabgleichs (gekürzt)

3.2 Automatischer Weißabgleich

Der automatische Weißabgleich funktioniert bei der oben genannten Beleuchtungssituation (d. h. ausschließlich Beleuchtungsquellen gleicher Farbtemparatur) ausreichend exakt. Es wurden hierzu mehr als 20 Aufnahmen in verschiedenen Innenräumen angefertigt. Mit Hilfe einer Bildverarbeitungssoftware wurden dann die RGB-Farbwerte von hellgrauen und weißen Motiven bestimmt und die jeweiligen Abweichungen zum zugehörigen Grauwert (Mittelwert aus der R-, G- und B-Komponente) ermittelt.

Die Messungen (siehe Tabelle 3.1) ergaben einen leichten Rotstich der Aufnahmen. Dieser hat jedoch aufgrund seiner Konstanz keinerlei negativen Einfluss auf Messungen mit dem Kamerasensor – vorausgesetzt die jeweilige Applikation arbeitet nicht auf absoluten Farbwerten, sondern basiert auf einer Kalibrierung, bei der der Farbstich miterfasst wird.

3.3 Örtliche Auflösung

	Größe des Sterns auf Sensor		Größe des unscharfen Innenteils des Sterns		**Errechnete Auflösung**	
Dist *[cm]*	b_{proj} *[px]*	h_{proj} *[px]*	b_{unsch} *[px]*	h_{unsch} *[px]*	$\mathbf{A_{vert}}$ ***[cm]***	$\mathbf{A_{view}}$ ***[cm]***
50	56	43	17	13	**1,6**	**1,6**
100	48	29	20	9	**2,2**	**1,6**
200	34	15	27	6	**4,1**	**2,1**
400	23	7	> 23	5	**> 5,2**	**3,7**

Tabelle 3.2: Messwerte zur Bestimmung der Auflösung des Kamerasensors mittels auf dem Boden liegenden Siemensstern

Zur Bestimmung der örtlichen Auflösung der Kamera wurde ein Siemensstern mit $n = 32$

	Größe des Sterns auf Sensor		Größe des unscharfen Innenteils des Sterns		**Errechnete Auflösung**	
Dist *[cm]*	b_{proj} *[px]*	h_{proj} *[px]*	b_{unsch} *[px]*	h_{unsch} *[px]*	$\mathbf{A_{senk}}$ ***[cm]***	$\mathbf{A_{waag}}$ ***[cm]***
50	81	63	18	11	**1,2**	**0,9**
100	57	53	16	13	**1,5**	**1,3**
200	39	40	15	14	**2,0**	**1,8**
400	24	24	15	15	**3,3**	**3,3**

Tabelle 3.3: Messwerte zur Bestimmung der Auflösung des Kamerasensors mittels senkrecht stehenden Siemensstern

Abbildung 3.3: Bestimmung des Auflösungsvermögens mittels Siemensstern von waagerechten (links) und senkrechten Objekten (rechts) im Abstand von 100 cm zum Sensor

schwarz-weißen Strahlen und einem Durchmesser von $d = 53\,\text{cm}$ unter der genannten Beleuchtungssituation in verschiedenen Entfernungen (zwischen 50 cm und 4 m) aufgenommen. Dabei wurden zwei Messreihen – eine, bei der der Siemensstern flach auf dem Boden lag und eine, bei der er senkrecht auf dem Boden stand – durchgeführt, da der Aufbau des Sensorssystems für diese zwei Orientierungen im Vorfeld ein unterschiedliches Auflösungsvermögen vermuten ließ (siehe Abbildung 3.3). Die daraus resultierenden Aufnahmen wurden manuell mit Hilfe eines Bildverarbeitungsprogramms vermessen und die Größe des gesamten Siemenssterns (b_{proj}, h_{proj}) sowie der des verschwommenen Innenkreises (b_{unsch}, h_{unsch}) pixelgenau bestimmt. Je größer der Quotient aus der Breite des unscharfen Kreises und der Gesamtbreite des Siemenssterns, desto geringer die Auflösung für Konturen, die tangential zur Blickachse verlaufen (in Richtung der roten bzw. blauen Linie in Abbildung 3.3). Bildet man entsprechend den Quotient aus der Höhe des verschwommenen Kreises und des Siemensterns, erhält man die Auflösung für in Blickrichtung verlaufende Konturen (gelbe bzw. grüne Linie in Abbildung 3.3). Es gelten somit:

$$A_{vert} = \frac{\pi \cdot d \cdot}{n} \cdot \frac{b_{unsch}}{b_{proj}} \qquad A_{waag/view} = \frac{\pi \cdot d \cdot}{n} \cdot \frac{h_{unsch}}{h_{proj}}$$

Der so errechnete Auflösungswert gibt an, wie groß ein Objekt, welches einen hohen Kontrast zu seiner Umgebung aufweist, mindestens sein muss, um von der Kamera gerade noch erfasst werden zu können – z. B. wie breit die dünnsten Linien in einem schwarz/weißen Strichcode sein müssten, ohne dass der Strichcode als graue Fläche erscheint. Für Objekte mit geringerem Kontrast (z. B. graue Linien auf schwarzem Grund) oder bei schlechterer Beleuchtung ist davon auszugehen, dass sich dieser Wert erheblich vergrößert. Wird die Kamera im 4:1:1 YUV-Modus betrieben, sinkt das Auflösungsvermögen für nichtmonochrome, farbige Strukturen ebenfalls (siehe Abschnitt 3.4).

Die Ergebnisse der Messungen finden sich in den Tabellen 3.2 und 3.3. Aus den Messwerten lässt sich eine Abhängigkeit der Auflösung sowohl von der Entfernung des betrachteten Objekts als auch seiner Orientierung ableiten. Nahe Objekte werden mit einer maximalen Auflösung von knapp unter einem Zentimeter erfasst, während in einem Abstand von 4 m die Auflösungsgrenze bei ca. 5 cm liegt. Plane, senkrecht auf dem Boden stehende Objekte (z. B. an Wänden angebrachte Muster) werden wegen der geringeren Verzerrung generell besser aufgelöst als flache, waagerecht auf dem Boden liegende Objekte und Konturen.

Die Auflösung wird des weiteren relativ stark von der Richtung der Objektkonturen beeinflusst: Konturen, die tangential zur Blickrichtung des Kamerasensensors verlaufen (blaue bzw. rote Linie in Abbildung 3.3, repräsentiert durch Messwert A_{vert}) werden schlechter aufgelöst als Konturen, die in Blickrichtung des Sensors liegen – also z. B. Linien auf dem Boden, die strahlenförmig vom Kamerasensor weg führen (gelbe / grüne Linie in der Abbildung). Je weiter das Objekt vom Sensor entfernt ist, desto stärker der Effekt. Insbesondere bei der Suche nach feinen Linien im Bild (Begrenzungslinien etc.) könnte diese Richtungsabhängigkeit der Auflösung problematisch sein. Ein Richtwert für die minimale Größe beliebiger Muster stellt daher jeweils der schlechtere Messwert dar.

3.4 Datenvolumen: Brutto vs. Nettodaten

Der Kamerasensor arbeitet in allen AISVision-Applikationen mit einer progressiven[1] Auflösung von $640 \cdot 480 = 307.200$ Pixel bei einer YCbCr 4:1:1 Farbabtastung mit einer Farbtiefe von 8 Bit pro Kanal. Diese Art der Abtastung bedeutet, dass Farbinformationen nur für jeden zweiten Pixel in jeder zweiten Zeile ermittelt werden – es werden also nur für insgesamt 76 800 Pixel Farbinformationen per Firewire an den Host-PC, an den die Kamera angeschlossen ist, übertragen. Durch diese Art der Datenreduktion ist eine hohe Framerate von 30 Bildern pro Sekunde (fps) über den maximal 480 Mbps schnellen IEEE 1394-Bus möglich. Die Kamera selbst unterstützt im Gegensatz zu den AISVision-Modulen auch noch eine höhere Farbauflösung bei geringerer Framerate (z.B. 7,5 fps bei voller Farbauflösung).

Es ergibt sich bei 30 fps ein Rohdatenstrom von $(307\,\text{kB} + 2 \cdot 76{,}8\,\text{kB}) \cdot 30 = 13{,}5\,\text{MB/s}$, der vom Host-PC entgegengenommen werden muss (ohne 4:1:1-Abtastung wären es 27 MB/s). Dieser enthält jedoch noch einen großen Anteil an nicht verwertbaren Informationen. Der Spiegel deckt nicht die komplette Sensorfläche ab, sondern nur eine Kreisfläche von ca. 472

[1] d. h. vollbild-basiert; ohne Zeilensprung

Helligkeitspixeln Durchmesser, was rund 17 500 Pixeln entspricht. Die Streben, mit denen der Spiegel über der Kamera befestigt ist (vgl. Abbildung 2.1) sowie die im Spiegel reflektierte Kamera reduzieren die nutzbare Sensorfläche weiter auf insgesamt rund 164 300 Helligkeitspixel (bzw. 41 000 Farb-Pixeln) – das entspricht nur rund 55% der ursprünglichen Pixelzahl. Andersherum ausgedrückt ist fast die Hälfte der Informationen, die der Host-PC vom Sensor erhält, für eine verarbeitende Applikation wertlos und muss verworfen werden.

Die nutzbare Netto-Datenrate liegt damit im YUV 4:1:1-Modus bei etwa 7,2 MB/s.

3.5 Leistungsaufnahme des Kameramoduls

Der Stromverbrauch des Sensormoduls beträgt etwa ein Watt bei 12 V (ca. 82-85 mA). Dieser Wert ist unabhängig vom gewählten Betriebsmodus (Framerate, Auflösung) und ändert sich auch nach längerem Nichtbetrieb nicht, d. h. die Kamera besitzt keinen Energiesparmodus. Vor der ersten Initialisierung der Kamera fließt allerdings nur ein Strom ca. 60 mA.

3.6 Bestimmung der Abbildungsfunktion des Spiegels

Die AISVision-Module geben für jedes identifizierte Objekt (d. h. jeden Farbbereich) in der Umgebung des Sensors neben seinem Winkel einen Entfernungswert in cm an, der recht genau mit der realen Entfernung des jeweiligen Objektes übereinstimmt. Diesem Wert liegt die simple Annahme zugrunde, dass sich das Objekt auf dem Boden befindet und die Höhe des Kamerasensors stets konstant ist. Dann ist die reale Entfernung s_{real} des Objektes anhand seines Abstandes in Pixeln vom Sensormittelpunkt s_{sensor} bestimmbar.

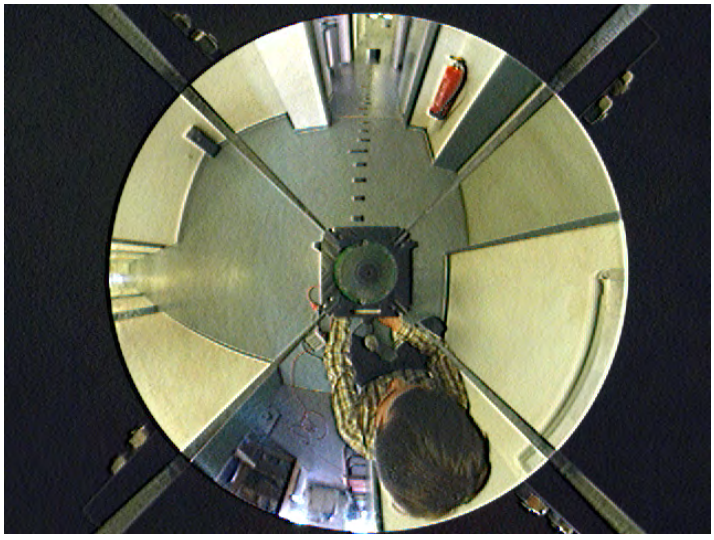

Abbildung 3.4: Ermittlung der Abbildungsfunktion des Kameraspiegels über Messpunkte auf dem Boden

Messung Nr.	s_{real} *[cm]*	s_{sensor} *[px]*
1	20	45
2	40	64
3	60	80
4	80	95
5	100	107
6	120	118
7	140	126
8	160	134
9	180	141
10	200	147

Messung Nr.	s_{real} *[cm]*	s_{sensor} *[px]*
11	300	169
12	400	182
13	500	188
14	600	194
15	700	197
16	800	201
17	1000	205
18	1200	209
19	1400	212

Tabelle 3.4: Messergebnisse zur Bestimmung der Abbildungsfunktion des Kameraspiegels

Zur Umrechnung des Pixel-Abstandes in den Realwert-Abstand muss jedoch die Abbildungsfunktion $s_{real} = f_{proj}(s_{sensor})$ des Kamerasystems bekannt sein. Diese Abbildungsfunktion ist vor allem von der Form des Spiegels abhängig. Sie ist in der Dokumentation des VolksBots nicht enthalten und wurde daher experimentell bestimmt. Um sie zu ermitteln, wurden Aufnahmen des Kamerasensors von Messpunkten mit bekanntem Abstand zum Kamerasensor ausgewertet (vgl. Tabelle 3.4). Aufgrund der Zentralsymmetrie des Spiegels genügte es, die Messpunkte auf einer Geraden anzubringen, wie sie in Abbildung 3.4 zu sehen ist.

Eine Applikation kann die ermittelte Messwertreihe als Lookup-Tabelle nutzen. Alternativ wurde aus den Messwerten mittels Polynomapproximation die Abbildungsformel interpoliert. Es gilt im betrachteten Messbereich (bis 14 m Abstand):

$$s_{real} = 2 \cdot 10^{-9} \cdot s_{sensor}^6 - 1 \cdot 10^{-6} \cdot s_{sensor}^5 + 0,0003 \cdot s_{sensor}^4$$
$$-0,0446 \cdot s_{sensor}^3 + 3,3775 \cdot s_{sensor}^2 - 128,72 \cdot s_{sensor} + 1937,3$$

Wobei $[s_{real}] =$ cm und $[s_{sensor}] =$ Pixel. Im Nahbereich bis 2m Entfernung kann auch eine kürzere Formel verwendet werden:

$$s_{real} \approx 9,4 \cdot 10^{-5} \cdot s_{sensor}^3 - 0,016 \cdot s_{sensor}^2 + 2,0077 \cdot s_{sensor} - 47$$

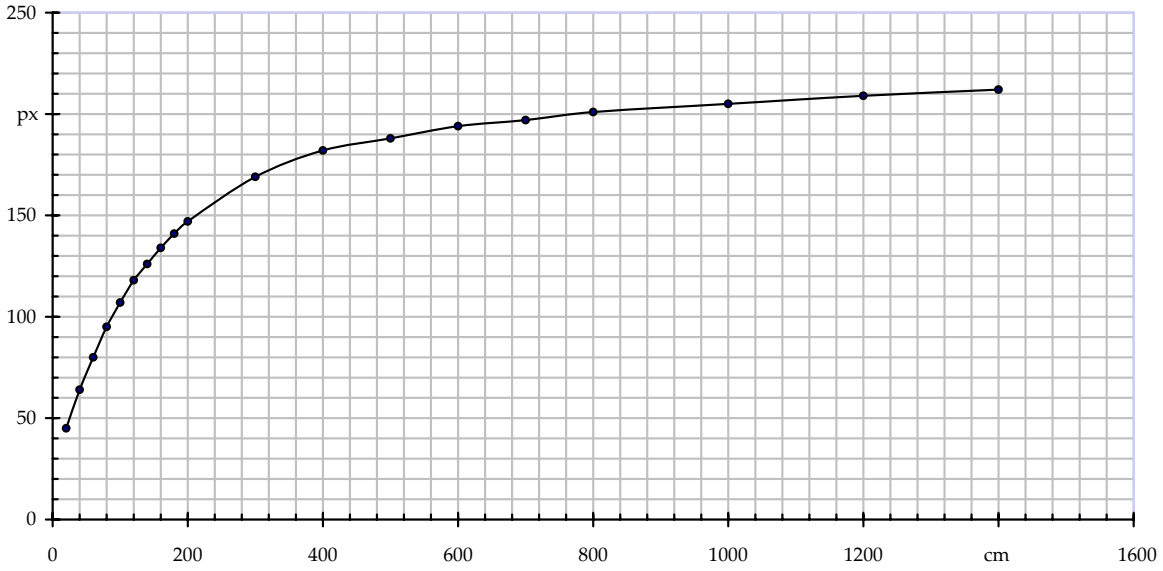

Abbildung 3.5: Darstellung der Messwerte zur Bestimmung der Abbildungsfunktion des Kameraspiegels

4 Entwicklung einer Roboterapplikation zur Linienverfolgung

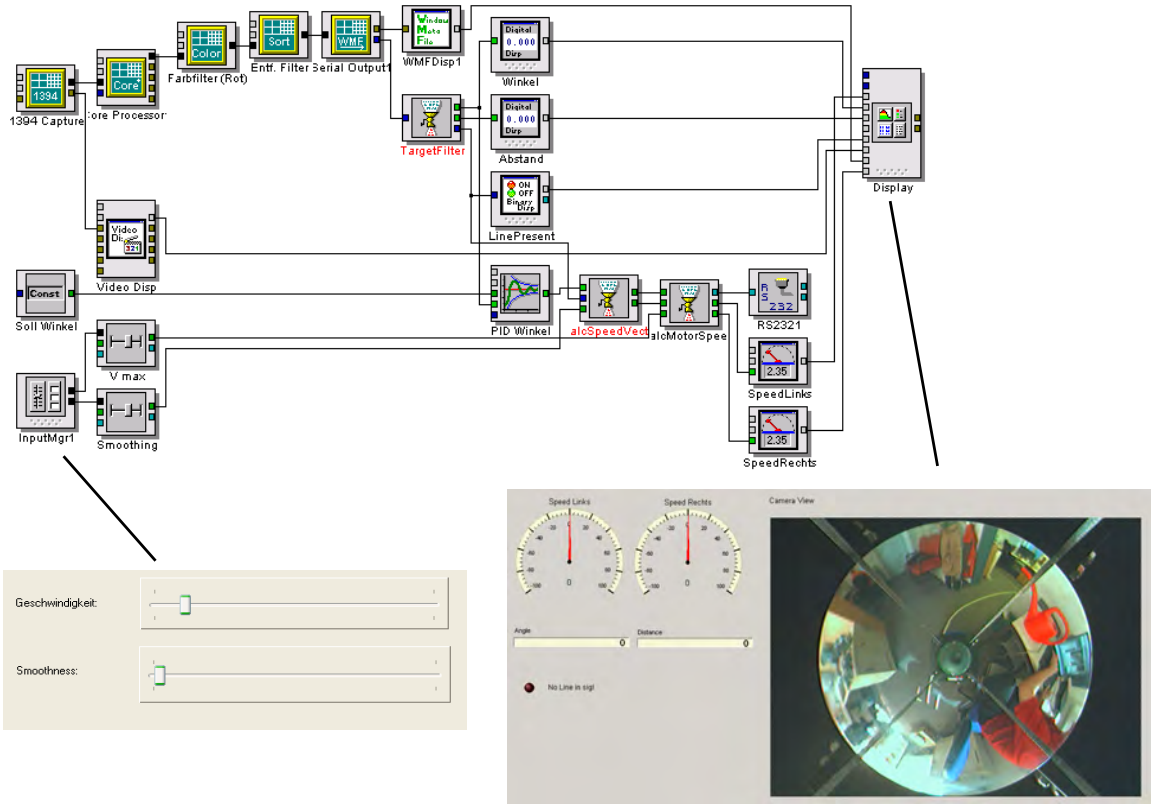

Abbildung 4.1: Signalgraph mit Ein- und Ausgabe Form der entwickelten Roboterapplikation zur Linienverfolgung

Um das Entwicklungskit praktisch zu erproben, wurde mit ihm auf Basis eines mit dem VolksBot mitgelieferten Beispielprogramms – einer einfachen Objektverfolgung – eine neue Anwendung erstellt. Die allgemeinen Erkenntnisse bezüglich Qualität und Eigenheiten von ICONNECT als Entwicklungsumgebung, die dabei gewonnen wurden, wurden bereits in Abschnitt 2.2 dargestellt. Nachfolgend soll dagegen primär die Struktur und das Funktionsprinzip der erstellten Anwendung näher erläutert werden.

Nachdem das vorhandene, aber weitestgehend undokumentierte Beispielprogramm zur Objektverfolgung – bei dem sich der Roboter einem beliebigen, auch beweglichen Ziel mit einer möglichst grellen Farbe und einer Mindestgröße von ca. 20 cm nähert – nach leichten Modifi-

kationen recht zuverlässig funktionierte, war es das Ziel, den Roboter entlang einer auf einem dunkelgrauen Boden aufgebrachten weißen Linie (Breite ca. 4 cm, siehe z.B. Abb. 4.1) fahren zu lassen. Hierzu wurde die Beispielanwendung stark modifiziert und erweitert.

4.1 Grundalgorithmus der Applikation

Bei dem erstellten Signalgraphen handelt es sich prinzipiell um einen einfachen Regelkreis. Das Kamerabild wird Frame für Frame auf das Vorhandensein einer Linie hin untersucht. Je nachdem, ob und wo eine Linie gefunden wird, wird der Bewegungsvektor des Roboters, bestehend aus Geschwindigkeits- und Richtungskomponente, berechnet: Wird keine Linie gefunden, wird die Geschwindigkeitskomponente der Bewegung allmählich auf 0 gesetzt, bis der Roboter stoppt. Andernfalls wird sie bis auf einen konstanten Wert erhöht.

Die Richtungskomponente der Bewegung wird durch ein PID-Regelmodul ermittelt, welches den Winkel des nächstgelegenen Punktes der gefundenen Linie auf 0 regelt – mit anderen Worten, den Roboter immer möglichst in Richtung des Nahpunktes der gefundenen Linie schickt, unabhängig von der Orientierung der Linie. So ist gewährleistet, dass der Roboter, auch wenn er sich nicht direkt auf der Linie befindet, auf direktem Wege auf sie zufährt – und ihr ansonsten folgt.

4.2 Besonderheiten der Implementation

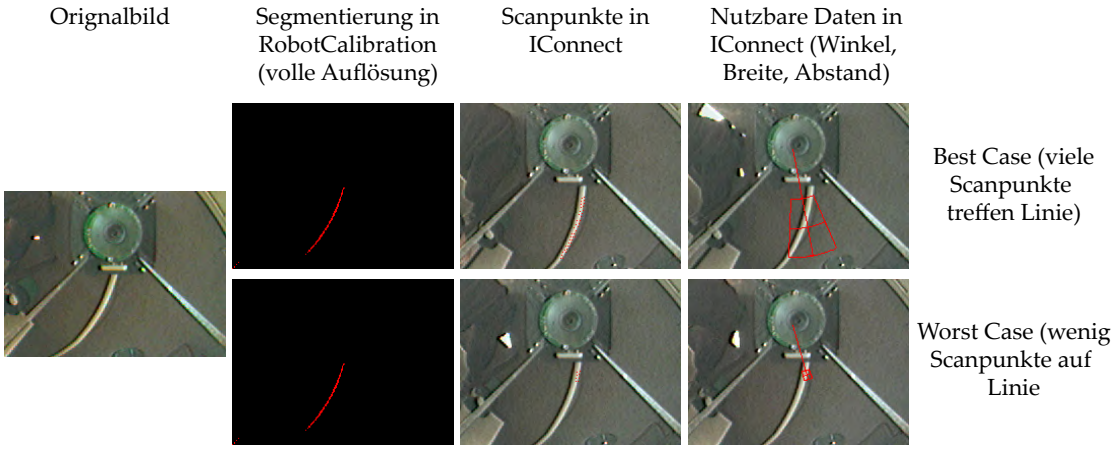

Abbildung 4.2: Gegenüberstellung von nutzbaren und durch das Sensorsystem gelieferten Daten am Beispiel einer 4 cm breiten Linie

Da das Sichtfeld des Sensors volle 360 Grad umfasst, musste eine Vorkehrung getroffen werden, dass bevorzugt Linien, deren Nahpunkt in Fahrtrichtung des Roboters liegen, verfolgt werden. Andernfalls würde der Roboter, insbesondere wenn er sich korrekt auf der Linie positioniert hat und sich vor und hinter ihm ein Liniensegment in etwa gleichem Abstand befindet, ständig

wenden. In der erstellten Applikation wurde dazu einfach der Sichtbereich des Sensors von 360 auf 225 Grad hart begrenzt, also Objekte hinter dem Roboter softwareseitig ausgefiltert. Alternativ wären hierfür auch weitergehende Verfahrensweisen denkbar gewesen (z. B. Objekte in Abhängigkeit von ihrem Winkel in einer Prioritätsliste anordnen und so Objekte hinter dem Roboter nicht komplett ignorieren).

Ein weiteres Problem stellt die Unterscheidung zwischen Linien und anderen Gegenständen dar. Die zu verfolgende Linie war weiß – und damit von anderen Objekten im Raum (Tapete, Tische, überbelichtete Bereiche auf dem Sensor) anhand der Farbe allein nicht sicher zu unterscheiden. Hierzu ist die Betrachtung der Form des Gegenstandes unumgänglich und im Fall von Linien über Skelettierung mittels morphologischer Operatoren auch einfach zu implementieren [Tön07]. Eine solche formbasierte Unterscheidung ist jedoch aufgrund der ausreichend hohen Auflösung des Kamerasensors nicht möglich gewesen. Grund dafür ist der begrenzte Datenumfang, die die AISVision-Sensorkomponenten zur Verfügung stellen. Während in der Anwendung zur Robotorkalibrierung, in der eine Vorschaumöglichkeit für das segmentierte Sensorbild besteht, die Linie visuell stets sehr sauber segmentiert wurde, lassen die Daten, die die AISVision-Module in IConnect liefern, keine Rückschlüsse auf die Form der Linie zu. Hier stehen nur abstrahierte Angaben über die Entfernung, Fläche, Breite und Höhe eines Objektes zur Verfügung (vgl. Abb. 4.2 bzw. Abschnitt 2.3). Selbst wenn auf die einzelnen Scanpunkte der AISVision-Module zugegriffen werden könnte, ist ihre Dichte auch in der höchstmöglichen Einstellung zu gering, um die 4 cm breite Linie sicher z. B. von einem Quadrat unterscheiden zu können. Je nachdem, welche Orientierung die Linie aufwies, bedeckte sie teilwiese deutlich mehr bzw. weniger zusammenhängende Scanpunkte, wie ebenfalls in Abbildung 4.2 zu erkennen ist.

Als Workaround wurde auf eine Überprüfung der Form des gefundenen Objektes verzichtet. Es wird von jedem weißen Objekt angenommen, dass es sich um eine Linie handelt. Um dennoch zumindest grob ausschließen zu können, dass z. B. entfernte weiße Wände versehentlich verfolgt werden, werden die Objekte nach ihrer Entfernung zum Roboter sortiert und nur die nächstgelegenen n Objekte daraufhin untersucht, ob sie sich vor dem Roboter, also im erwähnten Sichtbereich von 225 Grad befinden, und somit ein Ziel darstellen.

Weitaus besser wäre die Möglichkeit, den über die Farbsegmentierung ermittelten, ungefähren Aufenthaltsbereich eines Objektes in höherer Auflösung untersuchen zu können, also eine Art „Region of Interest“ aus der Roboterapplikation heraus spezifizieren zu können. Dies würde den Vorteil des generell stark reduzierten Datenvolumens der AISVision-Module mit weitaus besseren Möglichkeiten der Bildanalyse kombinieren.

4.3 Details zur Implementation

Abbildung 4.1 zeigt den Signalflussgraphen und die erstellten GUI-Elemente. Die zwei Slider sind dabei die Eingabeelemente: Mit einem kann die Geschwindigkeitskomponente des Bewegungsvektors des Roboters justiert werden, mit dem anderen die Glattheit der Bewegung, also wie stark der Roboter beschleunigt, wenn er eine Linie entdeckt. Im Flussgraphen sind

die Slider durch die Module `V max` und `Smoothing` repräsentiert. Für die Ausgabe sind die Module mit den Bezeichnungen `WMFDisp1` und `VideoDisp` (Ausgabe des Videobildes mit Markierung der verfolgten Linie) sowie die Module `Winkel`, `Abstand`, `LinePresent`, `SpeedLinks` und `SpeedRechts` (Anzeige des jeweiligen skalaren bzw. binären Wertes) zuständig.

Die AISVision-Module (grüne Module oben links im Graphen) implementieren die in Abschnitt 2.3 beschrieben Funktionen. Das letzte von ihnen, das `Serial Output`-Modul, wandelt schließlich die aus dem Kamerabild aufbereiteten Objekt-Informationen in eine Matrix um, welche mit Standard-IConnect-Modulen verarbeitet werden kann. Die Funktion dieser weiteren Module im Graphen wird nachfolgend kurz erläutert:

TargetFilter: Interpret-Modul, d. h. Modul, in das eigener Quelltext in JavaScript-ähnlicher Syntax eingefügt werden kann. Das Modul ist für die Eingrenzung der gefundenen Blobs mit weißer Farbe auf ein einzelnes zuständig. Sein Quelltext ist in Listing 4.1 zu finden. Ausgabe des Moduls ist die Entfernung und der Winkel des gefundenen (Linien-) Objektes.

PID Winkel: PID-Regelmodul. Gleicht den Winkel, in dem die Linie gefunden wurde, mit dem Sollwinkel (0 Grad, also „Objekt soll sich direkt vor dem Roboter befinden“) ab.

Soll Winkel: Konstante. Liefert die erwähnten 0 Grad als Eingabe für den PID-Regler.

CalcSpeedVector: Interpret-Modul. Errechnet aus der Ausgabe des Regelmoduls, den Geschwindigkeits-Slidern und der Angabe, ob eine Linie überhaupt in Sicht ist den aktuellen Bewegungsvektor.

CalcMotorSpeeds: Interpret-Modul. Berechnet aus dem Bewegungsvektor die zwei normierten Radgeschwindigkeiten, wie sie vom Motorcontroller benötigt werden.

RS2321: Schnittstellen-Modul. Steuert den Motorcontroller an.

```
execute {
  foundBlob = 0;
  anzZeilen = Matrix[0]; anzSpalten = Matrix[1];

  // Nur die nahesten MAX_BLOBS Elemente betrachten
  // (ein Blob pro Matrixzeile)
  if (anzZeilenMatrix > MAX_BLOBS) {zeilenMatrix = MAX_BLOBS;}

  // Blobs durchlaufen
  for (i=0; i < anzZeilen; i++)
    {
     //Position in der Matrix aus deren Breite berechnen
     matrixIndex = (i * anzSpalten) + MATRIXOFFSET;

     // Winkel des Objektes (0 == direkt vor Roboter)
     // steht in 4. Spalte:
     TAngle = Matrix[matrixIndex + 4];

     // Entfernung des Objektes aus 5. Spalte:
     TDistance = Matrix[matrixIndex + 5];

     if ( (TAngle    > MIN_ANGLE && TAngle < MAX_ANGLE) &&
          (TDistance < MAX_DISTANCE) )
        {foundBlob = 1; break;}
    }
  // Rückgabe
  TargetAngle << itof(TAngle);
  TargetDistance << itof(TDistance);
  isPresent << foundBlob;
}
```

Listing 4.1: JavaScript-ähnlicher Quelltext des „TargetFilter"-Interpret-Moduls

5 Zusammenfassung und Ausblick

Das AISVision-Kamerasystem ermöglicht einen relativ unkomplizierten Einstieg in die Roboterentwicklung. Ist der Bausatz montiert und eine kompatible Firewire-Karte gefunden, gelingt die Erstellung einfacher Applikationen mit Hilfe des mitgelieferten Beispielprogramms auch ohne Vorkenntnisse in der Entwicklungsumgebung IConnect relativ schnell. IConnect bietet zahlreiche Schnittstellen und ist trotz seinem zur klassischen imperativen Programmierung grundlegend anderen Konzept einfach zu bedienen. Negativ hervorzuheben sind in diesem Zusammenhang lediglich Mängel in der Dokumentation der AISVision-Komponenten für IConnect und die fehlende Stabilität der Entwicklungsumgebung bei der Ausführung rechenintensiverer Signalgraphen, die währedn der durchgeführten Versuche einen häufigen Neustart erforderlich machten.

Weiterhin negativ aufgefallen ist die fehlende Möglichkeit, auf die Art und Weise der Bildsegmentierung Einfluss zu nehmen, so dass mit Hilfe der AISVision-Module für IConnect grundsätzlich nur Applikationen entwickelt werden können, die auf einer Farbsegmentierung der Bilder mit statischen Schwellwerten basieren. Auch arbeitet die AISVision-Software nur auf einem Bruchteil der Kamera-Pixeldaten, wodurch die sowieso schon recht geringe örtliche Auflösung des Kamerasensors weiter reduziert wird. Wünschenswert wäre hier die dynamische Spezifierbarkeit einer ROI (Region of Interest), in der der Robotorapplikation alle verfügbaren Pixeldaten des Sensor zur Verfügung gestellt werden. Dadurch ließen sich auch Applikationen wie eine zuverlässige Linienverfolgung umsetzen – dies gelang im praktischen Versuch mit einer ca. 5 cm breiten, kontrastreichen Linie nicht, da diese trotz sorgfältiger Kalibrierung nicht als zusammenhängendes Objekt erkannt wurde.

Somit ist für die weitere Verwendung des AISVision-Sensorsystems zu Ausbildungszwecken die Entwicklung einer universelleren Abstraktion des Sensorelementes als erster Schritt für darauf aufbauende Applikationen empfehlenswert.

Literaturverzeichnis

[AIS04a] AIS, Fraunhofer: *Robot Calibration.* Begleit-CD zum VolksBot, 2004. – Softwareprogramm / Dokumentation zur Software, Version build 1.6/1.5 Stable

[AIS04b] Autonome Intelligente Systeme, Fraunhofer I.: *Kurzanleitung zum VolksBot.* Begleit-CD zum VolksBot, 2004

[Bak06] Baker, Christopher: *CMU 1394 Digital Camera Driver (Open-Source).* http://www.cs.cmu.edu/~iwan/1394/, 2006. – Softwareprogramm/Treiber, Version 6.13 / 6.44

[Dou07] Douxchamps, Damien: *Coriander: The Linux GUI for IEEE1394 / Firewire cameras.* http://damien.douxchamps.net/ieee1394/coriander/, 2007. – Softwareprogramm, Version 2.0.0 RC 4

[ME04] Micro-Epsilon: *IConnect in 60 Minuten.* Dokumentation im Lieferumfang von IConnect, 2004

[Tön07] Tönnies, Klaus: *Grundlagen der Bildverarbeitung.* http://wwwisg.cs.uni-magdeburg.de/bv/skript/gbv/BV13.pdf, 2007. – Folien zur Vorlesung 13

[Wef00] Wefels, Hans-Gerd: *Der IEEE-1394-Bus (FireWire).* http://www.cs.cmu.edu/~iwan/1394/, 2000. – Begleitunterlagen Technische Informatik I der Fernuniversität Hagen

Abbildungsverzeichnis