

Hardware und Programmierung einer **inte**

Die Bedeutung der automatisierten Bildverarbeitung hat in den letzten Jahren in vielen Bereichen stark zugenommen: BV-Lösungen findet man mittlerweile in der Robotik, Automatisierungstechnik, Medizintechnik, im Verkehrswesen, in Fahrerassistenzsystemen und auch im Home-Entertainment-Bereich (z. B. Sony Playstation2, Eye Toy). Parallel zu dieser Entwicklung hat auch die Verbreitung eingebetteter Systeme zugenommen, und so liegt es für viele Applikationen nahe, diese beiden Technologien zu verbinden. Hierdurch entstehen intelligente Kameras, welche sich in der Bauform kaum von herkömmlichen Digitalkameras unterscheiden, zusätzlich aber mit DSP- und Ethernet-Funktionalität sowie mit Prozessschnittstellen beispielsweise zur Anlagen-SPS aufwarten. Da die bisher notwendige PC-Anbindung somit entfällt, werden die Systeme preiswerter, baukleiner und robuster.



von Andreas Böttlinger
und Tilo Gockel

Mit diesem Artikel möchten wir solch eine Kamera vorstellen und den Software-Entwicklungsprozess anhand der Aufgabe der automatischen Pfandflaschenerkennung und -sortierung genauer beleuchten. Wir haben dieses Beispiel ausgewählt, weil wohl allen Lesern die Pfandautomaten im Supermarkt bekannt sind und der Nutzen dieser Systeme auf der Hand liegt. Zu anderen möglichen Applikationen wie Barcodeerkennung, Werkstückvermessung, Objekterkennung mit Lageausgleich, 3D-Laserscanning und 3D-Tracking siehe beispielsweise [Azad 07].

Zum Einsatz kommt eine OEM-Platinenkamera der Firma Vision Com-

ponents in Ettlingen bei Karlsruhe. Diese wird für die Leserschaft der Zeitschrift Elektor zusammen mit der notwendigen Software-Bibliothek und dem Kabelsatz als Bundle günstig angeboten, vgl. hierzu [VC 07].

Die verwendete Hardware

Im vorliegenden Projekt wird eine OEM-Platinenkamera mit folgenden Eigenschaften verwendet (Bild 1 und Bild 2 sowie Datenblatt unter [VC 07: Dokument 3]):

- Produktbezeichnung: Intelligente OEM-Platinenkamera ohne Gehäuse. VCSBC4018 inkl. montiertem Lensholder für 12 mm-LowCost-Objektive und Kabelset. Objektiv: M12 x 0,5-Gewinde, Brennweite $f = 6$ mm,

Intelligenten Kamera

Fallbeispiel: Flaschensortierung

Bezugsquelle: [VC 07; Edmund 07]; Anmerkung: verwendet wird die Graustufenversion, die Farbversion hat die Bezeichnung VCSBC4018C

- Maße: Platine: 60 x 80 mm², Bauhöhe: ca. 40 mm, je nach Objektivstellung
- Bildaufnehmer: 1/3"-CCD Sony ICX424AL, Auflösung 640 x 480, 32 frames-per-second, full-frame, Progressive Scan, Shutter: 36,2 Mikrosekunden...2 Sekunden
- Prozessor: TI TMS320C64xx DSP, 400 MHz, 3200 MIPS („Million Instructions Per Second“: der Wert 3200 entspricht etwa der Leistung eines 2,4 GHz Pentium-PCs (!))
- Speicher: 32 MByte DRAM, 4 MByte Flash EPROM
- Digitale Ein- und Ausgänge: zwei Eingänge, vier Ausgänge, 12 V/24 V-SPS-kompatibel (4 x 400 mA pro Ausgang), acht zusätzliche TTL-DIOs
- RS232-Schnittstelle
- Trigger: 1x Bildtrigger-Eingang, 1x Blitztrigger-Ausgang, 1x Beleuchtungsanschluss („Illumination Controller“)

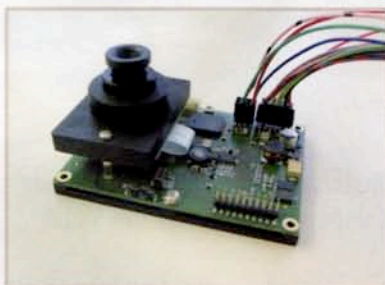


Bild 1. OEM-Platinenkamera mit 400 MHz-DSP.

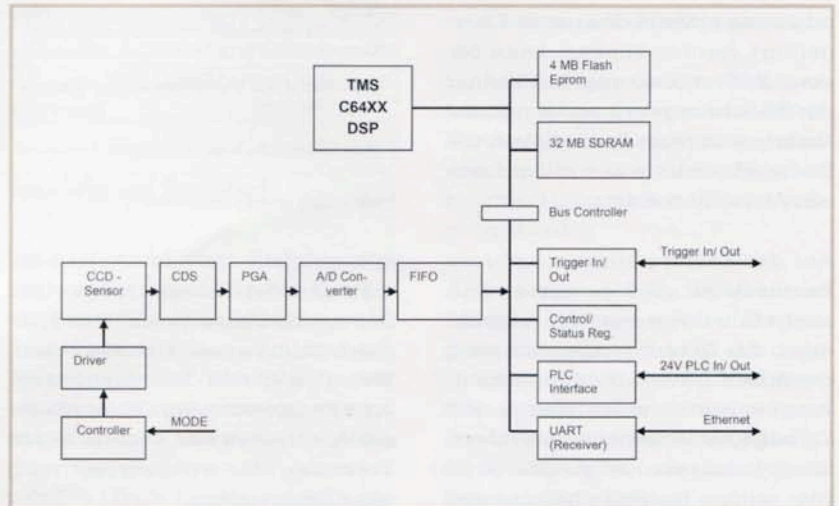


Bild 2. Blockschaltbild der VCSB4018-Kamera (Quelle: [VC 07: Dokument 3]).

- Netzwerk: 100 MBit-Ethernet-Anschluss
- Videoausgang und Programmierung: Livebild über Ethernet, Programmierung über Ethernet
- Stromversorgung: 12...24 VDC unstabilisiert (die Stabilisierung geschieht on-board), Maximalwerte: 9 VDC...30 VDC, die Leistungsaufnahme beträgt 2,4 Watt im Betrieb ohne Last an den DIOs
- Notwendiges Zubehör: Kabelsatz für Stromversorgung und Netzwerkanschluss, Netzwerk-Cross-over-Adapter, Mini-Objektiv (12 mm-Anschluss, Brennweite $f = 6$ mm, fixierbar im Sockel durch eine kleine Inbusschraube), unstabilisiertes 800 mA-Steckernetzteil

Anmerkung: Kleine dreieckförmige Markierungen an den Printbuchsen geben die Orientierung der Steckverbinder vor.

Für die Verwendung der Kamera mit den VC-Programmibliotheken ist der Einsatz des Betriebssystems VCRT auf dem DSP Voraussetzung (Vision Components Real-Time OS). Dieses Betriebssystem erlaubt Multitasking und somit beispielsweise auch die Übertragung von Live-Bildern über TCP/IP im Hintergrund. Die zugehörige VCRT-Kamera-Shell ermöglicht den Transfer und die Ausführung neu erstellter Programme sowie einfache Datei- und Verzeichnisoperationen (vgl. auch [VC 07: Dokumente 1 und 4]). Das VCRT-Betriebssystem wird im Bundle mit der zugehörigen Software-Programm-Bibliothek VCLib ausgeliefert. Eine Evaluationsversion ist erhältlich unter [VC 07].

Kamera-Betriebssystem und Shell

Im nachfolgenden Text wird das Betriebssystem der Kamera und die zugehörige Shell mit den wichtigsten Kommandos vorgestellt.

Das Betriebssystem VCRT

Das VC-Real-Time-Betriebssystem wurde speziell für den Einsatz auf intelligenten Kameras entwickelt. Es ist echtzeit- und multitasking-fähig und schöpft die Rechenleistung der neuesten DSPs von Texas Instruments voll aus. Weiterhin verwaltet es die Datenverarbeitung und den Zugriff auf den Bildaufnehmer. Da Aufnahme und Verarbeitung eines Bildes parallel ausgeführt werden können, kann bei einer Bildverarbeitungszeit kleiner der Belichtungszeit als maximale Verarbeitungsgeschwindigkeit die Bildwiederholrate des Bildaufnehmers erreicht werden.

Auf der Kamera können mehrere Prozesse parallel gestartet und ausgeführt werden, die sowohl durch das Betriebssystem als auch durch den Anwender unterschiedlich priorisiert werden können. Ein Zeitscheiben-Scheduling-Verfahren weist Prozessen mit gleicher Priorität reihum für einen bestimmten Zeitraum den Prozessor zu und ermöglicht somit die Echtzeitverarbeitung.

Häufig ist es notwendig, parallel arbeitende Prozesse zu synchronisieren. Hierfür wird für die Interprozess-Kommunikation das VCRT-Event-System angeboten. Da ein Prozess in den Ruhezustand versetzt werden kann, bis ein Event eintritt, wird verhindert, dass der wartende Prozess CPU-Zeit beispielsweise durch Polling beansprucht. Sobald das entsprechende Ereignis eingetreten ist, wird der ruhende Prozess wieder aufgeweckt und bekommt entsprechend seiner Priorität den Prozessor zugeteilt. Das Betriebssystem fängt eingehende Hardware-Interrupts wie zum Beispiel detektierte Änderungen an den SPS-Ein- und Ausgängen oder die Meldung eines abgeschlossenen Bildeinzuges ab und benachrichtigt Anwenderprogramme mithilfe des Event-Systems.

Die Speichermanagementfunktionen für Flash, DRAM und MMC/SD-Card ermöglichen die dynamische Speicherreservierung zur Laufzeit. Es stehen Befehle zur Verfügung,

Funktion	Tastaturbefehl
Upload von Programmen vom PC in den Flash-Speicher der Kamera	\$ lo , dann Aktivierung der Uploadfunktion des Terminalprogramms
Programmausführung	\$myprogram
Ausgabe des Verzeichnissesinhaltes auf fd	\$dir fd:/
Dateien löschen	\$del myfile
Packen des EPROMS (der Speicher wird von gelöschten Dateien gesäubert)	\$pk
Umschaltung der Videoausgabe in den Live- bzw. Still-Modus	\$vd -l (bzw. -d)
Ausgabe der Betriebssystemsversion der Kamera	\$ver
Einstellung der Verschlusszeit der Kamera (engl.: Shutter)	\$sh <value> (Angabe in Mikrosekunden)
Hilfe (Übersicht über verfügbare Kommandos)	\$he

Tabelle 1. Häufig benötigte Shell-Kommandos.

mit welchen sich der Anwender während der Entwicklungs- und Debugging-Phase die aktuelle Speichernutzung anzeigen lassen kann. Weiterhin erlaubt die industrietaugliche Realisierung des Dateisystems ein Abschalten der Kamera durch Trennung der Stromversorgung ohne Datenverlust.

Die Shell und die wichtigsten Kommandos

Mithilfe der Kamera-Shell können die wichtigsten Kamerafunktionen vom PC aus kontrolliert werden. Die Shell ist der für den Anwender sichtbare Teil des Betriebssystems und bietet Interaktionsmöglichkeiten über Ethernet bzw. über ein serielles Interface. Gewöhnlich wird hierfür ein Terminalprogramm auf einem PC verwendet. Wie bei den meisten Betriebssystemen üblich, können Kommandos eingegeben sowie Programme ausgeführt werden. Die VCRT-Kommandos dienen zum Einen der direkten Steuerung der grundlegenden Kamerafunktionen (Einstellung der Belichtungszeit, Umstellung der Videomodi), zum Anderen können aber auch Dateioperationen durchgeführt und Informationen, wie zum Beispiel TCP/IP-Statistiken, abgerufen werden. Die eingebaute Hilfe (Kommando **he**) bietet einen schnellen Überblick über die vorhandenen Funktionen.

Eine ausführbare Datei wird durch die Eingabe des Dateinamens aufgerufen: das Programm wird geladen, Befehlszeilenparameter werden in den Speicher geschrieben und das Programm wird gestartet. Die Shell wird wieder in den Hauptspeicher geladen, nachdem das Programm terminiert. Um den Umgang mit der Shell zu vereinfachen, können Kommandos in Stapelverarbeitungsdateien zusammengefasst werden. Eine spezielle Stapelverarbeitungsdatei ist die **autoexec**, die beim Systemstart automatisch ausgeführt wird.

Die Shell und die verfügbaren Kommandos können auch aus einem Anwenderprogramm heraus aufgerufen werden. Eine Übersicht über die wichtigsten Shellkommandos zeigt Tabelle 1.

Entwicklungsumgebung, Toolchain und Probelauf

Im nachfolgenden Text erfolgen einige Erklärungen zum verwendeten C-Compiler, zu der Entwicklungsumgebung Code Composer Studio (CCS) von der Fa. Texas Instruments und zum Entwicklungsablauf.

Implementierungen mit dem Code Composer-Studio und der VCLib

Die Software-Entwicklung für die Kamera geschieht mittels der inte-

grierten Entwicklungsumgebung Code Composer Studio der Fa. Texas Instruments. Das CCS erlaubt die Programmierung des DSPs in Standard-C und C++. Compilieren, Linken und Kopieren des lauffähigen Codes in das send-Verzeichnis für den Upload in die Kamera ist mit einem einfachen Mausklick möglich. Weiterhin ermöglicht ein optional erhältlicher JTAG-Emulator das In-System-Debugging.

Das CCS ist als kostenfreie 120-Tage-Testversion erhältlich unter [TI 07]. Die Installation und die kostenfreien Tools zur FTP-Übertragung und zum Aufbau einer Telnet-Verbindung werden ausführlich beschrieben im Schnellstart-Handbuch unter [VC 07: Dokument 1].

Vision Components bietet für die Programmierung der Kamera die VCRT- und die VCLib-Bibliothek an. Erstere ermöglicht den Zugriff auf Funktionen des Betriebssystems, zweitere bietet grundlegende Bildverarbeitungsfunktionen. Eine erweiterte Funktionalität wird geliefert in Form der optional erhältlichen ExtensionLib-Bibliothek. Beide BV-Bibliotheken basieren auf den folgenden Bilddatenstrukturen:

- Grauwertbilder
- Binärbilder in Run Length Code-Darstellung (RLC, vgl. untenstehenden Text)

- Binärbilder in Labelled Run Length Code-Darstellung (SLC)
- Image-Variablen
- Addresslisten (Pixel Lists)
- Contour Code (CC)
- JPEG Data (JPG)

Ferner existieren Bibliotheken für die Farbbildverarbeitung (ColorLib) und spezielle Anwendungen wie z. B. Barcode- und DataMatrix-Code-Auswertung.

Toolchain und Probelauf

Vor der eigentlichen Programmerstellung für die intelligente Kamera wird in vielen Fällen die Programmierung eines Demonstrators bzw. Testsystems auf dem PC erfolgen. Mit dieser Vorgehensweise ist eine schnelle und bequeme Evaluierung des geplanten BV-Ansatzes möglich (Tools hierfür vgl. z. B. die IVT-Bibliothek aus [Azad 07]). Nach Abschluss dieser Phase erfolgt die Programmerstellung für die Zielplattform:

1. Erstellung (Veränderung) des C-Quellcodes im CCS (zu Workspaces vgl. die Doku unter [VC 07: 2])
2. Compilieren, Linken und Ablage der entstandenen .out-Datei im Verzeichnis c:\ti\myprojects\

send\ mittels Button-Klick, etwaige Debug-Infos sind in Form von print()-Kommandos eingebaut (*)

3. Übertragung der .out-Datei mithilfe eines FTP-Programms, z. B. mithilfe des freien Programms „Total Commander“ (vgl. [VC 07: 2])
4. Start des Programms auf der Kamera, u. U. Rücksprung zu 1.

(*) In der vorgestellten Toolchain hat der optional einsetzbare JTAG-Emulator keine Erwähnung gefunden, da dieser uns für die Programmentwicklung noch nicht zur Verfügung stand.

Anhand des nachfolgenden kurzen Beispiel-Listings werden die Programmierung und auch die Leistungsfähigkeit der Bibliotheken demonstriert. Zu den einzelnen Funktionen und Variablen vgl. [VC 07: 2], hier finden sich auch weitere Beispiele zum Einstieg.

In Bild 3 ist die Ausgabe des Beispielprogramms gezeigt: eine Kantenerkennung des aufgenommenen Bildes. Da das hier verwendete preiswerte Kameramodul keinen VGA-Ausgang besitzt, erfolgt die Bildausgabe über den PC. Zu diesem Zweck wird vom Hersteller das Programm **img3r** zur Verfügung gestellt, welches den aktuellen Inhalt des Bildspeichers via TCP/IP auf den PC überträgt. Das Übertragungsprogramm läuft parallel zum jeweiligen Anwenderprogramm und überträgt den aktuellen Inhalt des Bildspeichers immer dann, wenn ihm das Anwenderprogramm den Prozessor überlässt (vgl. den Befehl `time_delay()` im Quelltext).

Für die hierfür notwendige parallele Ausführung zweier Programme wird eines der beiden an der Shell mit einem nachfolgenden Ampersand-Zeichen gestartet. Im vorliegenden Falle lautet entsprechend der Aufruf: **img3r &**

Anschließend wird auf dem PC das Client-Programm **AtxClient** gestartet, dessen ActiveX-Komponente vorher registriert wurde mittels



Bild 3. Ausgabe der Beispiel-Applikation, Visualisierung mittels AtxClient (vgl. Text).

Startmenü / Ausführen.../ regsvr32.exe „c:\ti\util\image transfer\atxcontrol.ocx“ (der Pfad ist entsprechend anzupassen). Beide Programme sind unter [VC 07] im Download-Bereich erhältlich.

Der Flaschenerkennung: Systemaufbau und Prozesskette

Systemaufbau

Im vorliegenden Beispiel werden die zu klassifizierenden Flaschen im sogenannten Durchlichtverfahren aufgenommen. Entsprechend befindet sich die Flasche zwischen einer matten, großflächigen Beleuchtung und der Kamera.

Im Testaufbau dient ein Tageslichtprojektor, auf dessen Auflagefläche einige Lagen weißes Papier angebracht wurden, als Lichtquelle (Bild 4). Mit dieser Art der Beleuchtung werden Schatten oder störende Reflexionen am Glas weitgehend vermieden, und das zu vermessende Objekt erscheint nahezu als Schebenschnitt. Falls kein Tageslichtprojektor zur Verfügung steht, so kann auch eine diffus angestrahlte weiße Wand als Lichtquelle verwendet werden.



Bild 4. Testaufbau

```
#define NEW_IMAGE_VAR
#include <vcrt.h>
#include <vclib.h>
#include <macros.h>
#include <sysvar.h>

void main(){
    U32 origin; // Startadresse des Bildes
    U32 sobel; // Startadresse des Ergebnisbildes
    image originImage; // Bildvariablen fuer das Originalbild
    image sobelImage; // und das Ergebnisbild
    char cKeyPressed = 0;

    origin = (int) ScrGetPhysPage;
    sobel = origin + ScrGetColumns /2;

    ScrSetLogPage(origin);

    // Zuweisung der Bilddimension an die Image-Variablen:
    // (systemabh. Bildgroessen aus der macros.h, zu pitch usw. vgl [VC 07: 2])

    ImageAssign(&originImage, origin, ScrGetColumns/2, ScrGetRows, ScrGetPitch);
    ImageAssign(&sobelImage, sobel, ScrGetColumns/2, ScrGetRows, ScrGetPitch);

    vmode(vmStill); // Einzelbildeinzug

    while (cKeyPressed!= 'x'){
        if (kbhit()){
            cKeyPressed = getchar();
        }

        tpict(); // take picture: Bildeinzug
        sobel(&originImage, &sobelImage); // Bild kann von img3r uebertragen werden
        time_delay(100);
    }
}
```

Durch die sogenannte Shading-Korrektur (vgl. den ersten Prozessschritt in Bild 5) ist eine ungleichmäßige Beleuchtung nicht allzu kritisch. Zu beachten ist allerdings, dass die Flasche keinen harten Schatten werfen darf, da dieser als zum Objekt zugehörig interpretiert werden würde.

Prozesskette

Das Beispiel der Flaschenerkennung bzw. -sortierung ist in Form einer Prozesskette realisiert, wie sie häufig in der industriellen Bildverarbeitung zu finden ist.

Zuerst erfolgen einige Operationen zur Bildverbesserung, dann eine Segmentierung des relevanten Bereiches, eine Merkmalsbestimmung und abschließend eine Klas-

sifizierung mittels Vergleiches mit den abgelegten Merkmalsvektoren (siehe Bild 5).

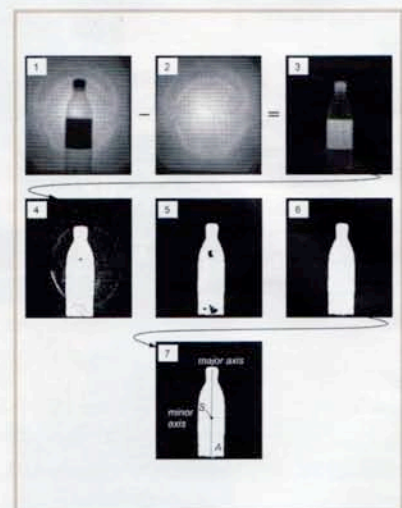


Bild 5. Prozesskette zur Flaschenerkennung (zu den einzelnen Schritten siehe. Text).

Funktion in der ExtensionLib	Bedeutung
<code>mom_calc_cgx</code> , <code>mom_calc_cgy</code>	Schwerpunkt: x-, bzw. y-Koordinate
<code>mom_calc_angle</code>	Hauptträgheitsachsen: Winkel in Grad
<code>mom_calc_rad</code>	Hauptträgheitsachsen: Winkel in rad
<code>mom_calc_ellipse_a</code> , <code>mom_calc_ellipse_b</code>	Hauptträgheitsachsen: Halbdurchmesser der zugehörigen Ellipse
<code>mom_calc_ecc</code>	Exzentrizität
<code>mom_calc_phi1</code> , <code>mom_calc_phi2</code>	Die ersten und zweiten Hu Momente (rotations-, translations- und skalierungsinvariante Merkmale eines Objektes)

Tabelle 2. Momentenbasierte Regionenmerkmale.

Differenzbild [1] – [2] = [3]

Um in der gegebenen Aufnahmesituation [1] den störenden Einfluss des inhomogenen Hintergrundes [2] zu minimieren und das Objekt (die Flasche) im nachfolgenden Schritt der Binarisierung optimal herauszulösen, wird eine sogenannte Shading-Korrektur vorgenommen. Hierzu wird pixelweise die Differenz zwischen dem Kamerabild mit Flasche und einem zuvor aufgenommenen Hintergrundbild errechnet. Auf dem Ergebnisbild [3] ist die Flasche leicht vom Hintergrund zu unterscheiden.

Binarisierung und Codierung als Run-Length-Code [4]

Im nächsten Prozessschritt wird das Graustufenbild binarisiert. Hierzu wird ein globaler Schwellwert experimentell derart bestimmt, dass eine optimale Unterscheidung zwischen schwarzem Hintergrund und weißer Flasche gegeben ist. Als Bildformat dient ab jetzt der sogenannte Run-Length-Code (RLC). Bei Verwendung der VCLib kann dieser Schritt mit der Operation `rlcmk` durchgeführt werden. Zur Theorie des RLC und zur Anwendung in der Bildverarbeitung vgl. [Wikipedia 07: Lauflängencodierung] sowie [Ercal 02], [Messom 02].

Entfernung kleiner Störungen [5]

Bei der mittlerweile vorliegenden Objektdarstellung treten aufgrund von Rauschen, Beleuchtungsänderung oder anderen Faktoren Störungen auf, die die Binarisierung besonders deutlich hervortreten

lässt [4]. Abhilfe lässt sich mit den formverändernden Operationen der Erosion und Dilation schaffen (zur Theorie vgl. [Azad 07] oder [Burger 06]). Die ExtensionLib bietet mit den Makros `erode` und `dilate` morphologische Operatoren, die auf RLC-kodierten Binärbildern arbeiten.

Schließen der Löcher im Flascheninneren [6]

Mittlerweile ist die Kontur der Flasche gut herausgearbeitet [5], nun werden die verbleibenden Lücken mithilfe eines einfachen Algorithmus entfernt. Hierzu wurde der sogenannte Region-Growing-Algorithmus angepasst (zu Details vgl. den Quelltext unter [Download 07] bzw. die genauere Erklärung in [Azad 07: Kap. 8]).

Merkmalsextraktion [7]

Nachdem die beschriebenen Operationen zur Bildverbesserung ausgeführt wurden, können nun im letzten Prozessschritt die Merkmale der Objektregion (der Flasche) aufgenommen werden, um eine Klassifizierung zu ermöglichen. Im vorliegenden Beispiel wurden folgende Merkmale der Region als besonders aussagekräftig bestimmt: Flächeninhalt, Schwerpunkt, Länge der Hauptträgheitsachsen und Exzentrizität (Abweichung von der Ellipsenform). Zu Details hierzu vgl. [Azad 07: Kap. 6] und [Burger 06].

Zu beachten ist hierbei, dass die Bildinformation mittlerweile als RLC vorliegt und entsprechend die Berechnung der Momenteninformationen hierauf angepasst werden muss

[Ercal 02, Messom 02]. Die ExtensionLib stellt hierfür folgende Funktionen zur Verfügung:

`rlc_moments` speichert die ersten beiden Zentralmomente in einem struct ab, welches als Eingabe für die Funktionen `mom_calc_<Merkmal>` dient. Die damit zur Verfügung stehenden Merkmale sind in Tabelle 2 aufgelistet.

Alle für die vorliegende Applikation gewählten Merkmale sind translations- und rotationsinvariant. Entsprechend werden keine besonderen Genauigkeitsanforderungen an die Positionierung des Objektes (der Flasche) im Bild gestellt, auch bei einer leichten Verschiebung oder Verkippung wird die Flasche erkannt. Der erhaltene Merkmalsvektor wird für jede Flasche zusammen mit der Flaschenbezeichnung abgelegt.

Bedienung

Nach Compilieren, Linken und Übertragung des Auswerteprogrammes auf die Kamera wird zunächst das Bildübertragungsprogramm `img3r` gestartet (vgl. vorstehenden Text). Anhand des dieserart erhältlichen Live-Bildes wird dann die Verschlusszeit der Kamera auf eine optimale Bildqualität eingestellt (Befehl `sh`, vgl. Tabelle 1). Beim ersten Durchlauf werden nun einige Referenzflaschen aufgenommen und die zugehörigen Merkmalsvektoren nebst Flaschenbezeichnung abgelegt. Hierzu startet man das Einlernprogramm auf der Kamera.

Um im eigentlichen Prozess neue Flaschen zu klassifizieren, werden diese an die vorgesehene Position im Bild eingebracht, der Merkmalsvektor wird bestimmt und der euklidische Abstand zu allen abgespeicherten Merkmalsvektoren berechnet. Ist der Abstand zu einem der abgelegten Merkmalsvektoren kleiner als der gegebene Maximalabstand, so gilt die Flasche als erkannt.

Ausblick

Moderne intelligente Kameras besitzen mittlerweile eine Verarbeitungsgeschwindigkeit, die an die Leistungsklasse schneller PCs heranreicht. Mit den Vorteilen des geringeren Preises, der Robustheit, der geringen Leistungsaufnahme und der kleinen Bauform eröffnen sich völlig neue Anwendungsfelder, und entsprechend wird in den nächsten Jahren ein enormer Zuwachs dieser Produkte in den Bereichen Automatisierungstechnik, Robotik, Medizintechnik, Überwachungstechnik und im Consumer-Markt erwartet.

Mit der vorgestellten überschaubaren Applikation der Flaschenerkennung konnten wir nur einen kleinen Einblick in die Technik der intelligenten Kameras bieten, haben aber hoffentlich das Thema anschaulich aufbereitet und das Interesse an eigenen Experimenten und Applikationen geweckt. Einen guten Eindruck dazu, was mit einer solchen Kamera in Verbindung mit einer leistungsfähigen Software tatsächlich möglich ist, bietet [Eyespector 07]. Die Quelltexte zum vorgestellten Projekt „Flaschensortierung“ sind erhältlich unter [Download 07].

Kontakt:

Andreas Böttinger
andreas.boettinger@web.de

Tilo Gockel
gockel@ira.uka.de

Fa. Vision Components

Infos und Bestellung zum Elektor-Bundle (Hardware und Libs) via E-Mail an:

sales@vision-components.com

„Eintrag in der Betreffzeile:

„Stichwort ELEKTOR“

Quellen

[Azad 07] Pedram Azad, Tilo Gockel, Rüdiger Dillmann, „Computer Vision – Das Praxisbuch“, Elektor-Verlag, Aachen, 2007.

[Burger 06] Wilhelm Burger, Mark James Burge, „Digitale Bildverarbeitung – Eine Einführung mit Java und ImageJ“ (2. Auflage), Springer-Verlag, Heidelberg, 2006.

[Download 07] Download-Bereich für die Programmdateien zum Projekt „Intelligente Kamera“:

<http://www.i3k.de>

[Edmund 07] Fa. Edmund Optics GmbH in 76131 Karlsruhe. Online-Produktspektrum. Bezugsquelle für 12mm-Objektive:

<http://www.edmundoptics.com>

[Ercal 02] F. Ercal, F. Bunyak u.a., „A fast modular RLE-based inspection scheme for PCBs“, in: Proc. of SPIE Conf. On Architectures, Networks and Intelligent Systems, SPIE Vol. 3203-06, Pittsburgh, Oct. 1997. Online erhältlich über

<http://citeseer.ist.psu.edu>

[Eyespector 07] Homepage und Produktvorstellung zum Produkt Eyespector (intelligente Kamera mit leicht anzuwendender Programmierumgebung, für industrielle Anwendungen):

<http://www.eyespector.de>

[Messom 02] C.H. Messom u.a., „Size / Position Identification in Real-Time Image Processing using Run Length Encoding“, Proc. of IEEE Int. Conf. on Instrumentation and Measurement, Anchorage, Alaska, USA, May 2002. Online erhältlich über

<http://citeseer.ist.psu.edu>

[TI 07] Firma Texas Instruments: Download-Möglichkeit für den verwendeten Cross-Compiler (Code Composer CCStudio 3.1 EVAL), 120-Tage-Testversion. Anmerkung: Der Weg zum Download der richtigen CCS-Version ist im Schnellstart-Manual unter [VC 07: Dokument 1] beschrieben.

<http://www.ti.com>

[VC 07] Firma Vision Components GmbH in 76275 Ettlingen. Produktspektrum und Handbücher. <http://www.vision-components.de>, Für den Zugang zum Download-Bereich ist eine Registrierung via E-Mail an sales@vision-components.com erforderlich (Eintrag in der Betreffzeile: „Stichwort ELEKTOR“). Registered User Area:

1. Schnellstart-Handbuch: Datei Getting_Startet_VC.pdf, Rev. 2.0 bzw. Schnellstart_VC.pdf, Rev. 2.1 (deutsch)
2. Programming Tutorial Basics: Datei Prog_Tut.pdf, Rev. 1.6
3. Handbuch zum Kameramodul (Hardware): Datei VCSBC40XX.pdf, Rev. 3.1
4. Handbuch zum RT-Betriebssystem VCRT: Datei VCRT5.pdf, Rev. 5.08
5. Handbuch zu den TCP-IP-Funktionen von VCRT: Datei VCRT5_TCP_IP.pdf, Rev. 5.01

[Wikipedia 07] Online-Enzyklopädie Wikipedia, deutsche Version, <http://de.wikipedia.org>