dsp@stettbacher.ch I +41 43 299 57 23 I CH-8600 Dübendorf

# Embedded Bildverarbeitung

08. August 2017

*Zunehmend wird moderne Bildverarbeitung nicht mehr nur auf sperrigen PCs implementiert, sondern direkt auf schlanken und preiswerten Embedded Systemen. Maschinen und Geräte werden sehend! Aber: Einige Gewohnheiten und Reflexe, die sonst Gültigkeit haben, sind bei der Anwendung im Embedded Bereich zu überdenken.*

Mikroprozessoren und kleine PC-Module sind so leistungsfähig geworden, dass inzwischen selbst die digitale Bildverarbeitung in den Embedded-Bereich vorgedrungen ist. Das ist nicht selbstverständlich, denn die numerische Verarbeitung von Kamerabildern in Echtzeit erfordert in der Regel sehr viel Rechenleistung auf Grund der grossen Datenmengen und der Komplexität der Vision-Algorithmen. Zum Glück sind Leistungsaufnahme und Preis bei den neuen Computer-Modulen nicht proportional zur Rechenleistung gestiegen. Im Unterschied zu Desktop-PCs kommen moderne dual- oder quad-core PCs im Kreditkartenformat mit wenigen Watt aus und kosten nur 100 bis 200 Franken.

Eine ähnliche Entwicklung hat bei den Kameras statt gefunden: Günstige und kompakte CMOS-Sensoren haben die aufwändige CCD-Technologie weitgehend abgelöst. Allerdings werden nach wie vor nicht die hochauflösendsten Kameras eingesetzt, sondern vorzugsweise jene, die das betreffende Objekt zweckmässig abbilden. Denn so spart man viel unnötige Rechenleistung.

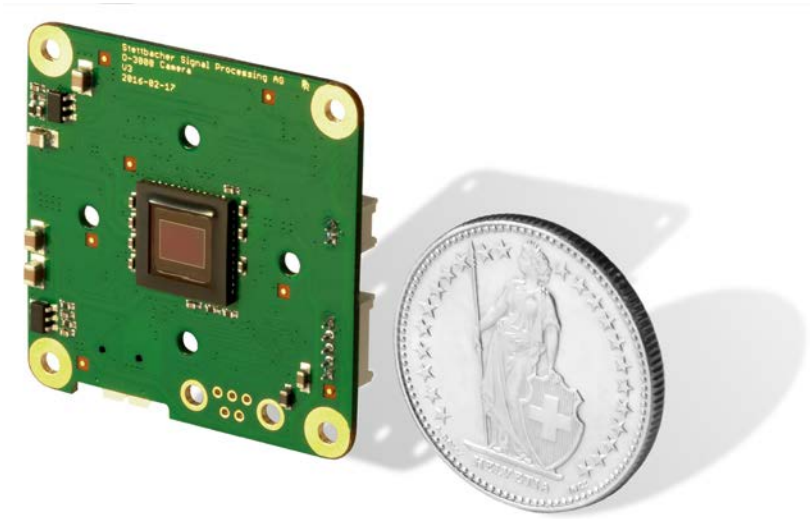

Bild 1: O-3000 Embedded Kamera.

## Herausforderung für Entwickler

Es gilt nicht grundsätzlich mehr ist besser, grösser ist schöner. Statt dessen ist der Entwickler von Embedded Image Processing Systemen mit einer Reihe von ganz elementaren Fragen konfrontiert:

1. Wie wird die Bildverarbeitung realisiert (Standard-Software)?
2. Welcher Rechner wird eingesetzt (Standard-Formfaktoren)?
3. Wie wird die Kamera an den eigenen Rechner angeschlossen (Standard-Schnittstelle)?
4. Wie gelangen die Bilder in die eigene Software (Standard-Treiber)?
5. Und so weiter ...

Um die Antwort gleich verweg zu nehmen: Es ist schlecht bestellt um Standards, jedenfalls im Embedded Bereich. Und – das möchten wir hier zeigen – das ist ganz gut und zweckmässig so. Denn Embedded Systeme sind so vielfältig wie ihre Anwendungen heterogen sind. Wichtig sind am Ende der Stückpreis, die Baugrösse, die Anschlussleistung, etc. Jeder Standard würde nur einer kleinen Gruppe von entsprechenden Lösungen gerecht und hätte in der Folge nur wenig Nutzen. Zudem verursacht jeder Standard im Vergleich zu einer spezifischen Lösung Mehrkosten für die Einarbeitung, die Hardware, den grösseren Platzbedarf und allenfalls die lizenzpflichtige Software. All dies widerspricht den ursprünglichen Zielen – voraus gesetzt natürlich, man verfügt über geeignetes Personal, das in der Lage ist, die spezifische Lösung zu entwickeln.

Bild 2: Rasberry Pi 3 mit O-3000 Kamera.

## Alternativen zu gängigen Standards

Betrachten wir zuerst die Prozessoren: In der Regel sind weder die klassischen embedded Mikroprozessoren noch Rechnermodule x86-kompatibel, sondern sehr oft ARM-basiert. Damit entfallen typischerweise die diversen ressourcenhungrigen Windows-Derivate. Ebenfalls aus dem Rennen ist in der Folge so manches schwergewichtige und teure Standard-Bildverarbeitungsprogramm. Statt dessen haben schlanke, auf die Anwendung zugeschnittene Linux-Derivate das Feld erobert. Diese Lösungen lassen sich sehr leicht für den betreffenden Anwendungsfall skalieren. Freie Tools wie OpenEmbedded, Bit-Bake, Buildroot, etc. unterstützen diesen Prozess. Statt einem käuflichen Standardprogramm für die Bildverarbeitung wird oft eine der grossen und sehr weit entwickelten open-source Grafikbibliotheken für C/C++ verwendet. Ein professionelles Beispiel dafür ist OpenCV, das mit Blick auf Anwendungen in Echtzeit entwickelt wurde und dank der Unterstützung von OpenCL und CUDA die spezifischen Fähigkeiten der betreffenden Hardware nutzt, wie Grafikbeschleunigung und Multiprocessing. Auf diese Weise erhält man für we-

nig Geld ein leistungsstarkes und überaus vielseitiges Bildverarbeitungssystem.
LVDS und MIPI CSI sind verbreitete Standards für die Anbindung von Kameras an Rechner. Aber es fehlen genormte Kabel und Stecker, daher verfügen die meisten Embedded-PCs über keine entsprechenden Anschlüsse. Dagegen ist praktisch jeder Rechner, ob gross oder klein, mit USB und Ethernet ausgestattet. Selbst kleinere Mikroprozessoren sind üblicherweise mit diesen Schnittstellen versehen. Sie sind folglich primäre Kandidaten für den Anschluss von Kameras an Embedded Systeme. Dabei ist USB preiswerter und kompakter als Ethernet. Ausserdem ist die Übertragungsrate von USB 2.0 höher als jene von Ethernet 100 Mbit/s. (Falls USB 3.x oder Gigabit-Ethernet auf Embedded Plattformen verfügbar sind, so vermögen sie in der Regel nicht die volle Übertragungsrate zu entfalten.)

Schliesslich bleibt die Frage nach dem Kamera-Treiber für die Software-Anbindung. UVC ist eine populäre USB Device-Klasse in der Consumer-Welt, jedoch ist der Standard wegen seiner ausufernden Komplexität wenig geeignet für Embedded Systeme. Er verursacht lediglich einen grossen Overhead. Zahlreiche Anbieter umschiffen die Probleme, indem sie zu ihren Kameraprodukten eigene vorkompilierte Treiber abgeben. Aber auch dies hat seine Tücken, denn welche Kombinationen von Hardware und Betriebsystem werden unterstützt? In der Regel sind es gerade nicht jene, die man gerne im Embedded Bereich einsetzen würde. Eine Alternative bietet Stettbacher Signal Processing AG mit seinen O-3000 Kameras (siehe Bild 1): Statt sich an halbherzig passende Standards anzubiedern, wird einfach das Kamera-Interface offen gelegt. Der Anwender hat über ein einfaches XML-Protokoll direkten Zugriff auf alle Funktionen der Kamera und auf die Bilddaten. Ein im Quellcode offen gelegter Treiber und Anwendungsbeispiele erleichtern dem Entwickler den Einstieg und das Einbinden der Kameras in sein System. Dabei ist es egal, ob es sich um einen Mikrocontroller, ein FPGA oder einen Embedded-PC handelt. Es bestehen keine Einschränkungen bezüglich Hardware, Betriebsystem, Programmiersprache, usw.

```
1  >> sudo apt-get update
1  >> sudo apt-get install libusb-1.0-0-dev cmake libtiff5-dev
   >>
2  >> git clone https://github.com/StettbacherSignalProcessing/O-3000.git O-3000
   >>
3  >> mkdir -p O-3000_build/drv
3  >> cd O-3000_build/drv
3  >> cmake ../../O-3000/driver/src/
3  >> make
3  >> sudo make install
   >>
4  >> mkdir -p O-3000_build/demo
4  >> cd O-3000_build/demo
4  >> cmake ../../O-3000/demo/framedump
4  >> make
   >>
5  >> sudo bin/framedump
```

Bild 3: Befehle für Rasberry Pi 3.

## Anwendungsbeispiele

Zuerst wird gezeigt, wie ein Raspberry Pi 3 mit wenigen Handgriffen dazu gebracht wird, Bilder von einer O-3000 Kamera aufzunehmen und im TIFF-Format abzuspeichern. Dabei sei die O-3000 Kamera über USB mit dem Raspberry verbunden. Die Kamera wird gleichzeitig über

USB mit Strom versorgt, so dass nur ein Kabel notwendig ist. In Abbildung 2 sind zwei Varianten der Kamera gezeigt, für die Anwendung reicht aber eine davon. Es wird nun auf dem Raspberry eine Shell geöffnet und die Befehlsfolge von Bild 3 eingegeben. Beachten Sie, dass dabei eine Internetverbindung bestehen muss. Die mit ‹1› markierten Zeilen sorgen dafür, dass das System aktuell ist und dass die für die App notwenigen Software-Pakete sicher installiert sind. Bei ‹2› wird die O-3000 Software von Github herunter geladen. Die Befehle unter ‹3› bilden und installieren den O-3000 Treiber. Unter ‹4› wird die Demo-App kompiliert. Mit dem Befehl auf Zeile ‹5› wird das Demo-Programm ausgeführt. Der Raspberry liest nun Bilder von der Kamera ein und speichert sie je in eine TIFF-Datei. Der Vorgang kann mit CTRL-C beendet werden. Die aufgezeichneten Fotos lassen sich mit jedem geeigneten Programm ansehen.

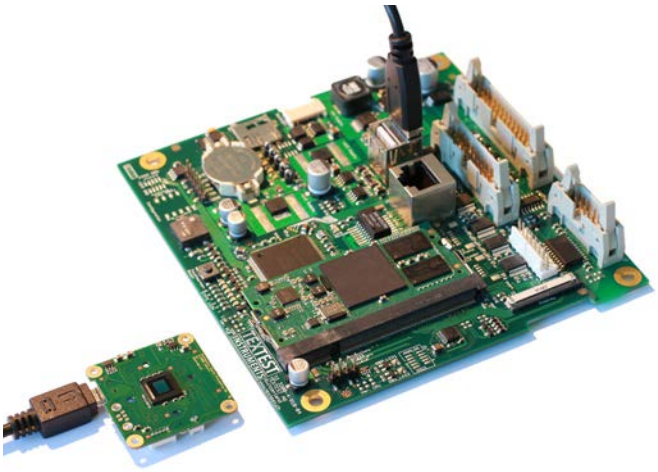

Bild 4: O-3000 Kamera in einer Maschinensteuerung.

Das zweite Anwendungsbeispiel ist in Bild 4 zu sehen. Es handelt sich um die Steuerung eines Textil-Prüfgeräts an der eine O-3000 Kamera (links vorne im Bild) via USB angeschlossen ist. Die Elektronik verfügt über diverse Ein- und Ausgänge für Sensoren und Aktoren im Gerät (Stecker auf der rechten Seite im Bild). Die Algorithmen der Maschinensteuerung laufen auf einem ARM Cortex A9-basierten Rechnermodul, das auf dem Hauptprint aufgesteckt ist (vorne im Bild). Als Betriebssystem wir ein massgeschneidertes Embedded Linux eingesetzt. Harte Echtzeitfunktionen des Geräts sind in einem FPGA realisiert. Das Rechnermodul empfängt auch die Bilder der Kamera und verarbeitet sie. Ziel ist es, während der Prüfung von Textilien Veränderungen am Gewebe zu detektieren. In diesem Fall ist die Aufgabe knifflig, weil erstens die Textilien beliebig bedruckt sein können und zweitens weil sich das Gewebe während der Prüfung verzieht.



Stettbacher Signal Processing AG
dsp@stettbacher.ch
www.stettbacher.ch
+41 43 299 57 23

Neugutstrasse 54
CH-8600 Dübendorf