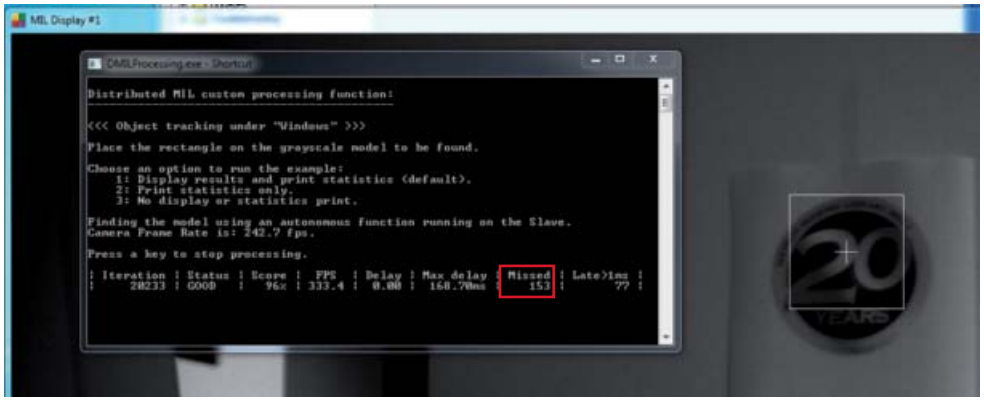

Bild: IntervalZero

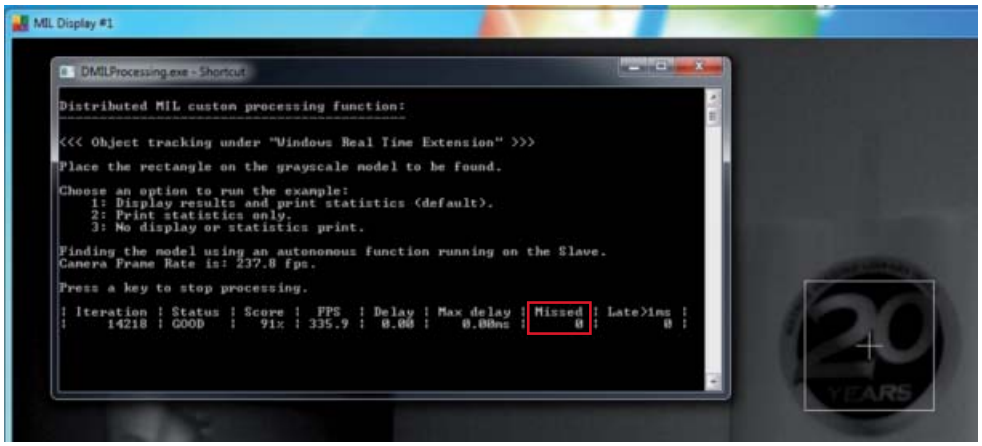

Bild: IntervalZero

Bild | Mit 335fps werden Bilder an einen Windows 7 PC gesendet (oben). Das Testprogramm ohne RTX64-Subsystem zeigt erhebliche Verzögerungen bei den Berechnungen, bis hin zu verlorenen Bildern (unten). Beim Einsatz des RTX64-Subsystems geht kein Bild verloren.

Bildverluste vermeiden

Echtzeit-Bildverarbeitung unter Windows

Die meisten der derzeit angebotenen Bildverarbeitungssoftwares wie Halcon, Matrox Imaging Library (MIL), VisionPro usw. laufen bevorzugt unter Windows. Allerdings wurde dieses Betriebssystem für Desktop-Anwendungen entwickelt. Dadurch ergeben sich aufgrund des Windows-Schedulers (der die aktuelle Reihenfolge der Abarbeitung der Aufgaben bestimmt) Latenzzeiten, die bis zu 500ms betragen können. Jeder Anwender kennt das Verhalten, wenn er die Maus bewegt, diese aber für eine kurze Zeit nicht reagiert und quasi stillsteht, um sich dann plötzlich irgendwo hinzubewegen.

Dieses Verhalten kann dazu führen, dass bei der Bildaufnahme (Transport der Bilddaten in den Speicher z.B. über GigE-Vision) Verzögerungen auftreten. Wenn die Maschinensteuerung bzw. der Prozess darauf angewiesen ist, dass das Bild zu einer genauen Zeit vorhanden sein muss, kann dies zu einer inakzeptablen Zeitdifferenz führen, die einen Maschinenstillstand zur Folge haben kann. Die Algorithmen (z.B. zur Form- oder Farberkennung) werden normalerweise auf jedes sich im Windows-Speicher befindliche, eingelesene Bild angewendet und können sehr komplex sein, d.h. benötigen ggf. auch unterschiedlich lange Berechnungszeiten. Sollten diese zu lange dauern, können Abbruchmechanismen definiert werden. Jeder Algorithmus, der unter Windows-Kontrolle läuft, kann in der Abarbeitung durch den Windows-Scheduler aufgerufene Services unterbrochen bzw. verzögert werden. Dadurch steht das Ergebnis der Berechnung eventuell nicht zum benötigten Zeitpunkt zur Verfügung und könnte zu einer Störung der Maschine führen. Jeder gute Ingenieur sucht daher nach einer Lösung für die beschriebenen Probleme, weshalb wir uns verschiedene Strategien genauer ansehen.

Auszug aus der inVISION, Machine Vision • Identification • Imaging, Ausgabe 1/2015.

FPGA-Framegrabber

Es gibt Framegrabber, auf denen nicht nur die Bildaufnahme, sondern auch die komplette Berechnung (nicht nur eine Bildvorverarbeitung) direkt vorgenommen werden kann. Die Algorithmen sind entsprechend vorab als VHDL-Programm zu erstellen, mit dem passenden VHDL-Compiler zu übersetzen und anschließend zu testen, was einen hohen Zusatzaufwand und zusätzliche Zeit in der Entwicklung bedeutet. Die Framegrabber kosten ohne Programmierung einige hundert Euros pro Stück, wobei dort noch nicht das Interface (GigE, CL oder CXP) bzw. die Kamera mit einbezogen ist. Falls das FPGA zu klein und damit die Maschine nicht so flexibel wie gewünscht ist, bedingt dies ein Re-Design der Karte mit einem größeren FPGA.

PCI-Karte auf DSP-/FPGA-Basis

Eine andere Strategie wäre die komplette Eigenentwicklung einer PCI-Einsteckkarte auf Basis eines DSPs bzw. FPGAs. Dies bedeutet, dass darauf die Bilderfassung (das Interface muss selbst programmiert werden) sowie die gesamte Echtzeitverarbeitung (die auch entwickelt und getestet werden muss) abläuft und somit getrennt von Windows läuft. Dafür benötigt man ein Betriebssystem für den DSP bzw. FPGA, das dann noch anzupassen ist. Auch die Algorithmen müssen an den verwendeten Chip angepasst werden. Ebenfalls kommt man um eine eigene Hardware-Entwicklung mit den zugehörigen Platinenentwicklungen und Fehler-Iterationen sowie den Testszenarios nicht herum. Nicht zu vergessen ist die nachgelagerte Lagerhaltung für Service und Ausfälle. Ganz zu schweigen von dem Problem möglicher Abkündigungen von Bauteilen. Zudem ist das Board nur für eine begrenzte Leistung entwickelt, sodass, wenn mehr Power benötigt wird, ein neues Board notwendig ist.

Industrie-PCs

Eine andere Lösung ist ein Industrie-PC für die Bildverarbeitung. An diesem darf niemand etwas bewegen (z.B. die Maus) und es gibt Systeme, bei denen verschiedene Windows-Services deaktiviert wurden, um möglichst keine Störung des BV-Systems zu haben, was aber u.U. zu einem instabilen Windows führt. Selbst mit einem Stand-Alone-PC nur für die Bildverarbeitung lassen sich die Probleme nur bedingt lösen. So kann es z.B. aus nicht nachvollziehbaren Gründen einige Male am Tag zu einem Produktionsstopp kommen. Zudem bedeutet ein weiterer PC zusätzliche Kosten sowie erhöhter Platzbedarf. Mangels kostengünstiger Alternativen ist diese Strategie aber ein weit verbreitetes Szenario.

Windows-Echtzeitverarbeitung

Um alle Windows-Latenzzeiten in den Griff zu bekommen, ist es nötig, dass die Bildaufnahme und -verarbeitung unabhängig von Windows läuft. Die Bildakquisition müsste, abhängig vom verwendeten Kamerainterface, dann als eigenständiger Prozess auf einem eigenen Kern laufen, auf den Windows keinen Zugriff hat und der somit unabhängig von Windows ist. Das von der Kamera übertragene Bild bzw. die zugehörigen Bilddaten wären dann nicht im Windows-Speicher, sondern in einem von Windows unabhängigen, getrennten Speicher zu legen. Wenn aber die Bildalgorithmen unabhängig von Windows auf die Bilddaten angewendet werden, dann gibt es die Möglichkeit, mit modernen Mehrkernprozessoren eine komplette Maschine inkl. der SPS und Feldbus-Anbindung auf einem einzigen IPC zu implementieren. Eine Lösung für diese Echtzeit-Bildverarbeitung wird seit Kurzem angeboten. Sie besteht aus einem PC, einem CL-Framegrabber von Matrox Imaging mit zugehörigem RTX64-Treiber sowie der Portierung der gesamten MIL-Algorithmen auf dem RTX64-Subsystem. Dieses kann nachträglich zu Windows installiert werden und ist nach kurzer Konfigurierung einsatzbereit. Es fungiert als SMP-Erweiterung (Symmetric Multi Processing) und transformiert Windows in ein RTOS (Real-Time Operating System). Gleichzeitig stellt es die unabhängige Nutzung vorhandener Kerne für Echtzeitanwendungen sicher, wobei ein Kern immer für Windows reserviert bleibt. Sämtliche Mechanismen (z.B. Mutex, Semaphore....), die ein Software-Entwickler von einem RTOS erwartet, sind vorhanden. Zudem erlaubt die Entwicklungsumgebung (Microsoft VisualStudio) die Implementierung eigener Algorithmen unter Standard C/C++. Je nach benötigtem Interface (bei GigE kann die vorhandene Ethernet-Schnittstelle genutzt werden) kommen nur Software-Komponenten zum Einsatz, was die Kosten zusätzlich senkt. Somit fallen nur die Lizenzkosten für das linear skalierende Subsystem RTX64 an. Allerdings wird ein PC mit Windows bzw. die MIL-Bibliothek benötigt.

Erfolgreicher Praxistest

Für die abgebildeten Screenshots wurde eine Demoapplikation, bestehend aus einer Kamera mit 2x1k Pixel mit 8Bit Tiefe, benutzt, die über ein doppeltes CL-Kabel und einem Radient-CL Framegrabber mit 335fps die Bilder an den PC sendet. Das gleiche Testprogramm zeigte speziell bei Bewegung der Maus oder beim Öffnen des Windows-Taskmanagers, erhebliche Verzögerungen bei den Berechnungen, bis hin zu verlorenen Bildern. Beim Einsatz des RTX64-Subsystemes wird dagegen kein Bild verloren und auch die geforderte Abweichung von kleiner 1ms wird erfüllt. Gleichzeitig besteht die Möglichkeit einen echtzeitfähigen-Master-Stack und eine selbstprogrammierte Ablaufsteuerung auf dem PC zu integrieren, sodass sich ein leicht zu wartendes Software-Gesamtsystem ergibt. ■

www.intervalzero.de

Autor | Bernhard Hartmann, Sales Manager Central Europe, IntervalZero