RTX64: A Powerful, Flexible Platform for Industrial Vision Control Systems

IntervalZero

Windows combined with multicore real-time performance, along with huge shared memory, creates a software environment for machine vision systems that uses non-specialized hardware yet is powerful, easy to program and maintain, while remaining flexible and configurable.

From sorting pieces of broken glass out of a manufacturing process to picking the face of a felon out of a crowded street, digital vision systems are permeating the industry and indeed, society. This is coming about partly because such systems mimic the way humans interact with objects, and also because the power and the cost of the needed technology are becoming ever more accessible.

The range of applications is truly wide—from inspection to robotic assembly to biometrics such as fingerprint, DNA and facial analysis, to medical devices to automobile systems and robotic warehousing to name but a few. The demands made on machine vision systems are as varied as the applications in terms of speed, resolution, depth and detail of image processing and the speed, accuracy and complexity of the control systems they are driving.

In the past, vision systems often consisted of two parts—the vision capture and processing system and the control system. Today, there is a growing need to bring these into a unified and integrated environment to bring the costs down as well as to make the system more easily configurable—mostly through software—for a changing range of applications. The RTX64 system from IntervalZero brings together real-time performance in a multicore SMP environment with shared memory that enables parallelism to speed image capture and processing along with real-time control. This is combined with a Windows user environment that can easily accommodate the latest camera technology and link to modern equipment through standard interfaces. Such a platform enables the rapid implementation and configuration of application-specific machine vision (ASMV) solutions that can be retargeted for different applications using what is essentially the same hardware platform.

This provides a powerful and flexible environment for implementing vision systems with a wide range of capabilities, one of which will be vision directed motion, also known as visual servoing. Some of this type of functionality has been offset using what are termed “smart cameras.” These cameras essentially have significant processing power on the camera itself

to either fully process or partially process images at speeds that allow for fast capture and high bandwidth that traditional control systems could not deal with. This process offloading allowed only the vision data that was needed to be passed to the system, thereby reducing the amount of information the system had to process directly. However, such on-camera processing comes at a very steep price in terms of the cost of the camera, which served to limit the types of systems that use vision. With RTX64 and GigE Vision, all of this capture and processing can be handled by the system itself, reducing the need for smart cameras and their steep cost. The result is that the number of applications that have access to integrate vision can grow at an increased rate and stoke the fires of innovation.

When we talk about machine vision, one thing that we can all agree on is that it begins with the capture of an image of some scene or objects by means of a camera. In the case of embedded vision systems, this is almost always a digital camera that will acquire an image that can be processed and understood in some way. But beyond that, more definition is needed. Just because an image falls on the retina does not mean we are "seeing." The brain has to process and understand the image, and the body has to respond to its information.

By the same token, a machine vision system has to do something. Across the capabilities of such systems, that "something" can be quite simple, such as recognizing and classifying parts for quality, size or some other properties. Or it can be very complex, such as robot arms manipulating objects into an assembly in 3D space. So to be considered machine vision, a system must bring together three basic components: image capture to acquire the visual data upon which to base the system's function; image processing to extract and/or apply the relevant data and information needed for the system's goals; and a control loop to carry out the actual function.

That control loop can be simple and even non-physical such as simply logging data about objects derived from the image capture and image processing. It could add a very simple physical control mechanism such as a puff of air or some actuator to push substandard parts off a belt for quality control. And, of course, it could include a complex control loop that would manipulate arms and instruments to accomplish some task directed by the processing of the visual data in cooperation with the control program and other predefined data serving as instructions to the mechanical subsystem.

The range from simple to complex also applies to the cameras used to capture the data in the first place. There is a wider range of cameras with different resolutions, color capabilities and speeds along with a corresponding variety of hardware interface standards that have evolved over time. In addition, these hardware interface standards come with different data rates, cables and connectors. One important designation among these is that some of them, such as CameraLink, CameraLinkHS and CoaXPress, must be connected to frame grabber hardware, while others, including the legacy IEEE1394 and USB 2.0 and newer, more powerful standards like GigE Vision and USB 3.0 Vision, connect directly to a PC.

While the above camera, cable and interface selection does provide a wide and seemingly complex choice of capabilities, the good news is that there is a much smaller number of software standards that are able to ensure interoperability among all these system components used to define drivers and that are supplied from various vendors as part of a software development kit. GenICam and IIDC2 take care of mapping internal camera functions, the basic transport layer and libraries into a generic API. This also makes it easy to change cameras connected to a system to meet different application requirements without a huge amount of low-level reprogramming.

As noted above, machine vision has huge potential to move into increasing numbers of automation and control applications if cost and complexity can be reduced, and if such systems can be made more flexible and more easily adaptable to changing application needs. Traditionally, a vision control system consisted of a separate vision and image processing unit that communicated with a separate and specialized control unit such as a welder, drill or robot arm. Bringing those components together in a single hardware/software system where the actual control device (drill, welder, etc.) is treated like a standard peripheral with a USB, Ethernet or PCIe interface, could greatly reduce the barriers to wider application and lower cost.

Consider the example of a vision-directed welding or drilling operation. The vision system would capture the image of two parts to be joined, or pieces of sheet metal to be welded, or the like. It would use image processing algorithms to recognize some key features of the objects, either from their inherent shape or by markings placed on them. Then it would apply a known pattern to the image using such things as edge recognition or other pattern matching algorithms to correspond exact-

ly to the desired placement on the work piece. The control of the tool would be guided by a course run on the applied pattern, and since it is exactly in place on the work piece, the weld, drill holes or other operations would be exactly placed as well.

Such a scenario has a number of immediate advantages, one of which is coming to be known as "cooperative robotics." Here, vision-directed motion is teamed with a human operator who places the work piece in position but does not operate the tool, which is under the control of the vision system. There can be slight imperfections in the placement of the piece because they are compensated for by the placement of the pattern. This saves significant amounts of cost and complexity in terms of touch and pressure sensors and actuators for positioning and holding the piece as well as the preparation of specialized jigs for that same purpose. If this strategy is applied to a number of different applications, the potential savings in sensors and positioning equipment can be tremendous. It also lends itself to greater flexibility, since an entirely different operation can be performed with different software and by plugging a different tool into what is essentially the same piece of equipment.

If all this functionality can also be put on a single general-purpose, multicore PC-class system and mostly implemented in software, the savings and flexibility will be significant. Such a system, based on RTX64, could be implemented on a multicore PC with multiple parallel real-time cores with Microsoft Windows as a human interface, and link to external networking such as the Internet, and also be programmed using Microsoft Visual Studio. Examples of other PC-based vision and image processing tools include MathWorks' MatLab, Wolfram Research's Mathematica, and libraries from Matrox and others along with real-time control programs that can run routines in parallel on several cores. Parallelism is further enhanced by the ability of threads on different cores to access shared memory, lessening the possibility of jitter or duplicating data.

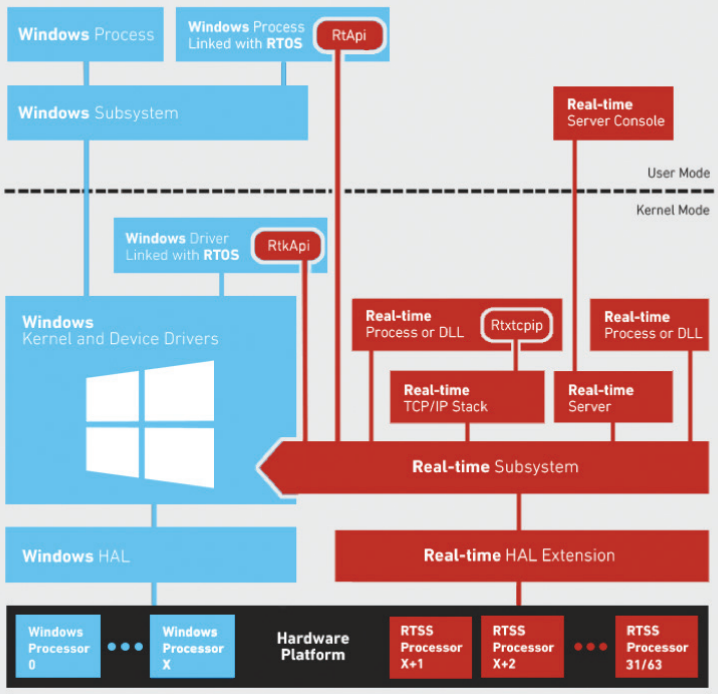
RTX64 Brings Together Windows and Specialized Real-Time OS in One Hardware Platform

IntervalZero's RTX64 transforms Windows into a fully functional real-time operating system that runs entirely on x64 multicore hardware. It also provides access to 128 Gbytes of non-paged memory, depending on actual mapped physical RAM size. Overall, Windows' 512 Gbytes

of physical memory dwarfs the 4 Gbyte physical memory limitation of 32-bit Windows. This vast amount of available memory opens the door to the implementation of truly powerful machine vision systems, which can accommodate large amounts of image data that is equally available to all cores running on the system.

Above all, RTX64 provides a single commodity hardware environment in the form of multicore x64 devices. This enables a single software environment that can accommodate Windows with its rich user interface, available applications and development environment. And Windows is seamlessly connected to the full-function, real-time symmetric multiprocessing (SMP) RTX64 environment that can scale from 1 to 63 cores. Applications compile to a single code base with no need for FPGAs or DSPs to execute logic based on different code that must be separately compiled and linked with the main application. One set of hardware, one operating system environment, one set of tools and one base of code. That translates to one team that can communicate and work together and produce high-performance, scalable applications while dramatically shortening time-to-market.

RTX64 provides an architecture that takes advantage of the advancing technologies—specifically, high-speed, multicore x64—that can outperform and outscale the traditional embedded environment that relies on DSPs, FPGAs and microcontrollers.



RTX64 has a hardware abstraction layer (HAL) that is distinct from the Windows HAL, but it operates alongside it, and no modification of Windows is needed. The two systems operate side by side and communicate via existing mechanisms. The RTX64 HAL can deliver deterministic real-time performance with timing down to 1 µs (dependent on hardware support). The scheduler, which resides in the RTX64 real-time subsystem (RTSS), can assign threads to cores to achieve symmetrical multiprocessing (SMP) without relying on virtualization or complex inter-process communications.

The vast memory space is available to all cores without memory partitioning. This is a huge advantage for machine vision applications and advanced industrial control systems that must not only present visual data to the user, but also process it in real time to drive motion control of tools and for the inspection of parts produced by the process.

Having this large a memory space available to such a high-performance general-purpose hardware platform allows OEMs to develop specialized software that can perform extremely specialized functions that would have otherwise required specialized hardware components. Scaling such a system only brings increased complexity with each disparate piece of additional hardware with its own interfaces and unique software needs.

IntervalZero's RTX64 has opened the world to Windows-based real-time systems with high-end vision, visualization and rich user interfaces. It has done this by giving the developer the ability to transform the functions of hardware components into software components by harnessing the power of the underlying multicore processing hardware. For the OEM, there is nothing to inventory and the parts can be replicated infinitely. For the software team, there is no need for specialized knowledge of hardware such as DSPs and FPGAs. The code exists in a unified code base and can be managed as such.

RTX64 Hosts Powerful Vision Systems

The high-performance and general-purpose nature of an RTX64 system makes it possible to implement a powerful and versatile visual servoing system defined mostly in software. The main hardware choices would then be the selection of a camera and the particular device to be con-

trolled such as a robot arm, drill, welding apparatus, milling device or something as simple as an air stream to remove rejected parts from a conveyor belt.

There is a very wide selection of cameras available with internal functionality ranging from basic VGA to high resolution in multiple megapixels, choices of frame rates and, of course, different degrees of internal processing capabilities if these are considered necessary to the application. As noted earlier, such a wide range of choices also comes with a wide range of associated costs. Given the very high processing capabilities of a multicore RTX64 system, careful consideration of camera features versus costs is an important step. Tasks that can be accommodated by the available processing power of the RTX64 system can greatly reduce the costs for the needed camera.

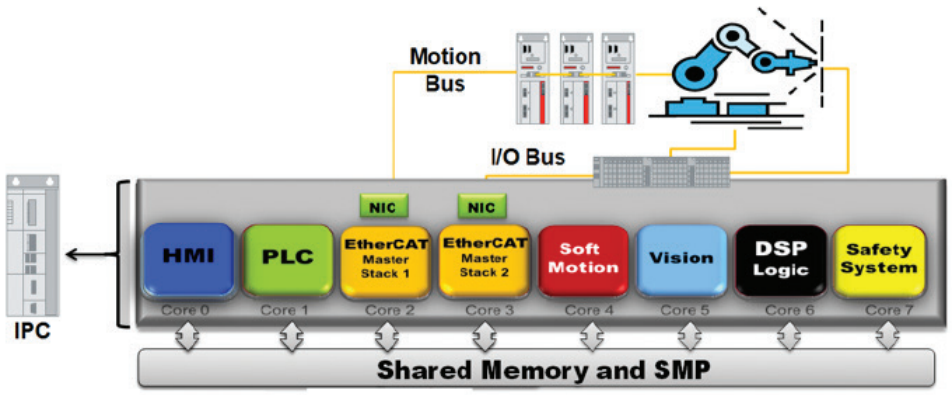
For this reason, IntervalZero recommends the GigE Vision global camera interface standard, which allows for fast image transfer using low-cost standard Ethernet cabling and connectors with the ability to span lengths up to 100 meters or even 5,000 meters using fiber optic cable. Since GigE Vision directly uses an Ethernet connection, it does not rely on any specialized hardware such as a frame grabber card to transfer its image data to the application. In addition, it can support multiple cameras on the same network with each camera having its own IP address.

Additionally, in the spirit of a generic interface, GigE Vision relies on the Generic Interface for Cameras, or GenICam. GenICam is able to work with any hardware interface technology, and therefore can accommodate any camera that is connected to the RTX64 system by way of the GigE Vision Ethernet or EtherCAT cable connection. It offers several modules including a generic control protocol, naming convention, API and transport layer that can easily be adapted to accommodate different camera/application combinations should the need arise due to the demands of a different or upgraded application.

As on the camera end, the control loop end also takes advantage of the versatility of EtherCAT, which can also integrate such legacy fieldbus components as Profibus or CANopen. EtherCAT runs under RTX64 in software without the need for any specialized EtherCAT card plugged into the system bus. Running on one or more processor cores, EtherCAT communicates directly with whatever network interface chip (NIC)

is used in the system. The individual device can be selected during EtherCAT configuration. Of course, other standard interfaces, such as USB or PCI Express, can also be easily and inexpensively integrated if needed. The advantage is that machines or devices connected to these interfaces are run by the control loop software running under RTX64, which is itself controlled by the vision processing running on the same system.

The ability to assign different functions to different cores that communicate via shared memory makes it easy to change out and control such things as different tools that are to be controlled by the vision system.



Thus, motion is not treated here as some stand-alone function that is to be driven by an independent vision system. Rather vision and motion are integral parts of the single system. At the same time, however, thanks to the multicore nature of RTX64, the control/motion functionality is modular and can be placed on a dedicated core and communicate with the vision system via the shared memory provided by the RTX64 environment. So if the same drill or welder is to be used to do some different operations on a different work piece, the application can be altered by simply upgrading the part of the software—for example, the visual processing—that will drive the tool according to different images, patterns and instructions. Or, if need be, the entire application can be changed out and a different tool selected, but it would still use the same RTX64 Windows-based hardware/operating system environment.

Again, the multicore character of the RTX64 system allows for parallelism among different functions so that they do not interfere with each other. Therefore, an interrupt that is meant for the control loop will not affect the image processing that is running on a different core. Data collection and the human interface running under Windows can function smoothly. Depending on how many 64-bit cores are included

in the system, even some of the visual processing that in other environments might have required the use and specialized programming of FPGAs, can be partitioned among a selected number of cores using familiar software approaches and tools. With Windows as the human interface, it is also possible to access the Internet, other applications, and even to bring up the images that the vision system is working with.

Based on the Microsoft Visual Studio environment, software development can take place under Windows using powerful tools like C and C++ with a powerful suite of editors, debuggers and analysis tools. In addition, there are a number of software libraries available that are aimed specifically at image processing with functions such as edge detection, object recognition, feature detection, extraction and so on. IntervalZero is currently working with a number of these suppliers to port their libraries to RTX64.

The flexibility and savings of such a versatile system extend beyond the configurability of the platform alone. The nature of a visual servo system is that it need not rely on sensors and positioning mechanisms whose parameters have to be precisely defined independently of the component running the tool. For example, a stage to hold a part would need a jig that was specifically defined to load and hold that part in an exact position. A drill would have to be separately programmed to meet exact locations based on the position of the part. Then, for a different part and/or operation, positioning and control would have to be defined all over again, perhaps with different mechanical components a well.

With vision, the piece does not need tobe placed so precisely since the vision system will match the coordinates in its learned pattern with the captured image to control the tool with the needed accuracy. If a human is involved, this may require only placing and clamping a piece in place and changing to the next piece. Different operations might be able to use the same positioning device and certainly the same human operator. And with an RTX64-based visual system, the different tools, operations and part positioning can be switched out quickly, easily and reliably. For both humans and machines, the ability to see what you are doing makes all the difference.▪