

# Four Keys to Successful Multicore Optimization for Machine Vision

White Paper

**COGNEX**

Optimizing a machine vision application for multicore PCs can be a complex process with unpredictable results. Developers need to pay close attention in order to achieve the best overall system performance. Field testing under real world operating conditions is the only way to fully measure system throughput.

*By John Petry, Cognex Corporation, Business Unit Marketing Manager, Vision Software*

## **INTRODUCTION**

---

For many years, increases in machine vision speed came almost automatically with increasing microprocessor speeds. However, this is no longer true with multicore PC architectures, which require major software design changes to take advantage of the parallel processing architecture.

A successful multicore strategy for machine vision can be implemented at multiple levels. Independent high-level tasks—especially those with hardware dependencies, such as acquisition and I/O—can be written to run asynchronously on separate cores. This leaves the processor free to concentrate on those tasks that are not blocked. Individual vision tools can also be parallelized so that they divide their processing task among several cores.

## **MULTICORE PC ARCHITECTURE**

---

In the past, vision applications have depended on advances in PC hardware performance to handle bigger and more complex applications. Improved performance resulted from faster CPUs and associated hardware improvements. But faster processors require greater and greater heat dissipation, to the point where cooling has become a limiting factor.

Manufacturers such as Intel and AMD have addressed this by moving to an approach that uses multiple processors to do the job previously done by a single processor. These processors are packaged on a single chip. Each processor is called a “core”, and the new chips are called multicore processors. Two, four and eight-core processors are now common, while much higher density models are also being designed.

A 2 GHz dual-core processor might appear to have the same computing power as a 4 GHz single-core processor, but this is rarely true. To take full advantage of each core, software applications must be written to distribute the computation between the cores. Otherwise one core will sit idle for at least part of the time.

## Optimized Software Is the Key

One cannot simply move an existing machine vision application from a single-core PC to a multicore PC and expect to see a significant performance improvement. In fact, some applications may not run any faster on a multicore machine due to operating system overhead and other inefficiencies. Application developers and vision software vendors must rewrite their programs if they want to take advantage of multicore architectures to speed up their applications. This can be a complex task, and many algorithms do not easily lend themselves to parallel processing.

This paper discusses four keys to successful multicore optimization for machine vision applications:

1. Application Optimization
2. Vision Tool Optimization
3. Tuning for Overall System Performance
4. Software Portability

## Processes and Threads

The PC operating system manages programs as separate processes. Each process has an associated *context* which makes it appear to the program that it owns all of the computer resources such as CPU, memory, I/O, etc. When a process is blocked—for example, when it is waiting for an I/O resource—or when its time slice ends, the operating system saves the current context and swaps in another process. The operating system juggles process priorities in order to be as responsive as possible to a wide range of demands, most of which are invisible to the user.

A multithreaded program can be written so that different sections run simultaneously and independently. This is similar to running multiple processes, but threads are much *lighter weight*; in particular, they share the same address space. This allows the operating system to quickly switch between them, and makes it easy for them to share data when running in parallel.

Multithreaded applications do not require a multicore architecture. They can be very efficient on a single-core machine, but rely on the operating system to manage hardware resources for them. Multithreading is especially well suited for multicore PCs. Those parts of a machine vision algorithm that previously ran sequentially can be partitioned into separate threads that now run in parallel on separate cores.

## Commercial Multithreaded Software

Writing multithreaded application code is not simple, and there are often timing dependencies that make it hard to debug in a real world environment. It may also require the underlying machine vision libraries to be written in a *re-entrant* manner that allows multiple instances of the program to execute in parallel. It takes a skilled programmer to write robust multithreaded applications. For this reason, writing custom software at the application layer to take advantage of a multicore PC is usually only justified in very demanding applications.

As a practical matter, it's usually much more effective for machine vision users to purchase commercial software that is already optimized for multicore PCs. Off-the-shelf solutions may not be as efficient as custom code, but they can provide significant benefits at very low cost.

### 1. APPLICATION OPTIMIZATION

---

Application-level software can be optimized for multicore PCs in three ways:

1. By creating separate threads for tasks with hardware dependencies, such as image acquisition, accept/reject results, and operator interaction,

These threads are often designed to minimize unpredictable hardware delays. For example, the system needs to be ready to respond to a trigger event, but should not delay image processing in order to poll the triggering hardware every several milliseconds.

2. By creating separate threads for each camera in a multi-camera application,

This allows each thread to run as soon as its camera is triggered.

3. By creating separate threads for different machine vision tasks within a vision application.

For example, one thread might handle part alignment while another measures critical dimensions. However, this only works if the tasks are not dependent on each other, and the benefit will be small if one task is much shorter than the other.

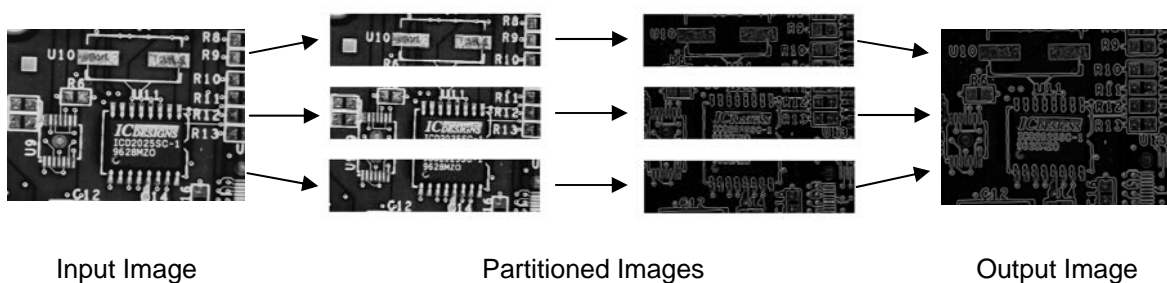
Some commercial machine vision products build in these features. For example, Cognex VisionPro™ software can automatically create separate threads for image acquisition and vision processing. The software is designed to automatically detect the number of cores in a PC and create threads based upon the number of cores available. This type of scalability is a great advantage in multicore PCs for applications with multiple image acquisition and vision processing tasks which need to perform simultaneously. It's even beneficial on single-core PCs, because image acquisition does not use much CPU time and can therefore run in parallel with image processing operations.

## 2. VISION TOOL OPTIMIZATION

---

In addition to application-level optimization, it's possible to optimize machine vision tools by parallelizing their algorithms so they use multiple cores simultaneously.

However, not all vision tools can be easily parallelized. In general, parallelization is most helpful for image processing filters or other vision tools that run local operations on small regions of the image. Commonly used filters include median, Gaussian and morphology operations. These can be optimized by dividing the image into different pieces and assigning each one to a separate thread. The results from each thread are then combined to produce the final result (see Figure 1). The final speedup depends on the algorithm and the number of cores. Because of overhead, there will always be some inefficiencies, so even a well-optimized vision tool may not run eight times faster on an eight-core PC.



*Figure 1. Example of partitioning an image across multiple threads*

Unfortunately, many vision applications spend most of their processing on tools that are much more complex than simple image processing filters. It's not always possible to parallelize complex vision tool algorithms such as alignment. In these cases, optimizing the tool might only benefit a small portion of the algorithm.

Cognex understands this and is working to optimize its most important alignment and inspection tools. For example, the PatInspect™ tool has been redesigned so that inspection steps are divided among the available cores. Even when the percentage improvement is lower than for simple image processing filters, the overall application may benefit more, since complex machine vision tools generally consume a larger portion of the overall application.

### 3. TUNING FOR OVERALL SYSTEM PERFORMANCE

---

It might seem that the fastest vision application would be one that had control over every processor core in the PC, and which created one thread to run on each core. Real world applications are not that simple, however. The PC must also support operating system, machine control and other background tasks. In practice, the optimum number of threads for the vision application may not necessarily be the same as the number of cores in the PC, and it may not make sense to assign each thread to a specific core.

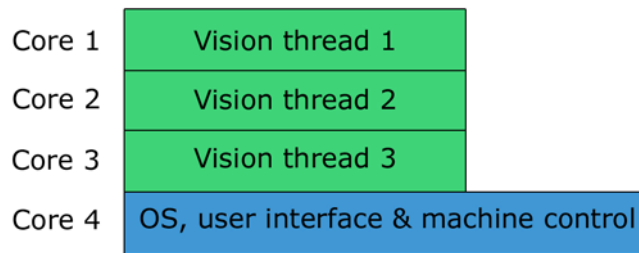


Figure 2. A vision application divided into three threads running on a four-core PC

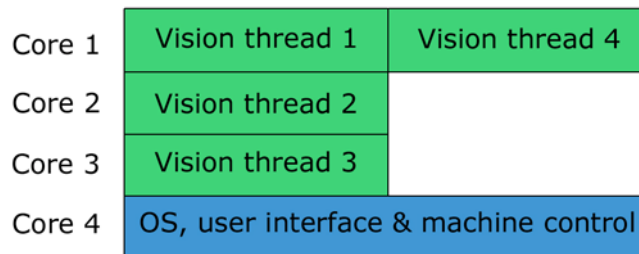


Figure 3. One vision thread per core may not be the optimum choice. This application finishes later than the one in Figure 2, even though this one has one thread per core. Depending on the application, more or fewer threads may be better.

The only way to determine the optimum number of machine vision threads is to test it under realistic conditions. For this reason, Cognex's CVL™ and VisionPro™ software libraries give users a simple method to set the number of threads for multicore-aware vision tools in an application. This top-level ability lets users easily tune the system for best overall performance.

## 4. SOFTWARE PORTABILITY

---

Another real world concern is software portability from one PC to another. PC hardware changes so quickly that many vision applications will be deployed on multiple PC models over their lifetime, either when new vision stations are deployed or when a PC needs to be replaced. The machine vision application is frequently developed on a different PC than the one on which it is deployed. Additionally, replacing PCs deployed in manufacturing lines is a constant maintenance issue.

Since the number of cores available may change over time, it's important to have a vision application that can account for any number of cores in the system. Otherwise, redeploying the existing system on a different PC may require recompilation, or worse, rewriting the application software. This can be cost prohibitive for many companies as development stations are modified and developers move on to other projects.

To avoid this, CVL and VisionPro libraries can automatically detect the number of cores on a PC and dynamically adjust the number of threads that they create. This allows applications written for a four-core PC to run efficiently on an eight-core PC without touching the source code or recompiling. That provides huge downstream maintenance savings, while offering the ability to upgrade performance simply by deploying the system on a PC with more cores.

### SUMMARY

---

Optimizing a machine vision application for multicore PCs can be a complex process with unpredictable results. Even in this simple overview, it is clear that developers need to pay close attention in order to achieve the best overall system performance. In particular, field testing under real world operating conditions is the only way to fully measure system throughput.

In order to maximize the benefits of multicore PC technology in machine vision applications, developers should consider several key questions when evaluating machine vision software products. These should include not only obvious points such as whether some image processing filters have been optimized for multicore, but also other factors that can significantly impact the performance of the overall application, including:

- Can the software product automatically create separate acquisition and processing threads to speed system throughput and responsiveness?
- Does the software allow users to write their own multithreaded application?
- Can users tune the number of threads for best overall system performance?

- Does the software have the ability to automatically detect and adjust the number of threads, based on the number of cores, without having to rewrite the application?

By keeping these points in mind, users can maximize their options (and minimize their work!) to take full advantage of multicore PC technology.

~

### **About the author**

John Petry has worked in machine vision for twenty years and is currently the marketing manager for Cognex Corporation's Vision Software Business Unit, which includes the VisionPro™ and CVL™ product lines, along with numerous automation products for industries as diverse as vision-guided robotics, pharmaceutical packaging and semiconductor manufacturing. He has been a software developer, an engineering manager, and a product manager for a wide range of products, including the VisionPro software library and Cognex's wafer identification business. He holds five patents in machine vision and a bachelor of science degree from the Massachusetts Institute of Technology.



Companies around the world rely on Cognex vision to optimize quality and drive down costs.

Corporate Headquarters One Vision Drive, Natick, MA 01760 USA Tel: +1 508.650.3000 Fax: +1 508.650.3344

**Americas**

United States, East +1 508-650-3000  
United States, South +1 615-844-6158  
United States, West +1 650-969-4812  
United States, Detroit +1 248-668-5100  
United States, Chicago +1 630-649-6300  
Canada +1 905-634-2726  
Mexico +52 81 5030-7258  
Central America +1 972-365-3463  
South America +1 972-365-3463

**Europe**

Austria +43 1 23060 3430  
France +33 1 4777 1550  
Germany +49 721 6639 0  
Hungary +36 1 501 0650  
Ireland +353 1 825 4420  
Italy +39 02 6747 1200  
Netherlands +31 402 668 565  
Spain +34 93 445 67 78  
Sweden +46 21 14 55 88  
Switzerland +41 71 313 06 05  
United Kingdom +44 1908 206 000

**Asia**

China +86 21 6320 3821  
India +91 80 4022 4118  
Japan +81 3 5977 5400  
Korea +82 2 539 9047  
Singapore +65 632 55 700  
Taiwan +886 3 578 0060

[www.cognex.com](http://www.cognex.com)