Machine vision systems are being used more and more to perform a wide range of tasks from automated barcode reading to multi-dimensional gauging and robotic guidance. And while recent advances in machine vision technology have significantly reduced their complexity, developers still face a number of challenges when building a reliable machine vision system.

A typical machine vision system is made up of a number of discrete components, including both hardware and software elements. Many developers place a significant amount of time and energy into the selection of the hardware components – including cameras, lighting, frame grabbers, additional image processing engines, etc. – and rightfully so.

Poor choices here will have a major impact on system performance on the factory floor.

The danger arises when this fixation on hardware eclipses or leaves little time for the consideration of the software elements. Software ultimately controls the hardware interface and drives the image acquisition, processing and analysis functions. The truth is, they are equally important, and developers need to consider both domains carefully.

This white paper will review critical considerations when selecting a machine vision software development package. This will include different programming languages and structure; algorithm performance; software reliability and robustness; and the impact of upgrades both to computing platforms and image processing programs.

KEY MACHINE VISION SOFTWARE EVALUATION CRITERIA:

**Performance** Does the software fulfill all the application requirements in terms of correctness and speed?

**Hardware Independence** Does the software API abstract key hardware features?

**Learning** How much time will it take for engineers to be able to use effectively this software (documentation, course, formation, support)?

**Reliability/Robustness** Does the software behave well in all operational conditions?

**Maintainability and Evolution** Is the software able to evolve smoothly?

**Support** What kind of technical support will you receive during your development phase and after?

**START WITH THE APPLICATION**

After evaluating a potential machine vision application, including an analysis of the environmental factors around the system (ambient lighting, vibration, process variations etc.)
and a detailed understanding of the part to be inspected, vision system designers would then have the basis for making hardware and software decisions.

For instance, if the application is a gauging application on a fast moving line that requires high-contrast images with very fine resolutions, the system may require a megapixel or larger CCD camera that generates many megabytes of image data per second and call for high-performance frame grabbers, image processing engines, and software capable of advanced memory management. If the application is less challenging in its complexity, then an embedded vision system or “smart camera” (a machine vision system with the camera, image processor and software all in one box), may be the most cost-effective and quickest solution to implement.

If the application falls in the latter category, the developer can have some confidence that a menu or “point and click” driven high-level programming interface will provide an adequate software solution with the added benefit of a shortened learning curve. (A high-level programming interface refers to the front end GUI of a machine vision application. Such methods, also called “point and click” programming environments, do not require the system designer to understanding programming languages such as C, C++, Visual Basic, Active X, etc.) These point and click interfaces significantly decrease development time and are recommended when the user has little or no programming experience and when flexibility and performance are issues of secondary importance.

On the other hand if the application requires tight control of peripherals and image processor(s), advanced memory management for large images, optimized algorithms and data integrity monitoring functions for system reliability and robustness – then the application likely will require a highly configurable, more powerful API that provides all of these capabilities. These API’s typically require the system designer to have good familiarity with programming languages and is a good choice for medium to advanced programmers.

That being said it’s important to note that the leading machine vision suppliers have invested tremendously to create both interface levels, such as DALSA Coreco’s Sherlock for point and click, and Sapera Processing API. Both packages leverage the companies experience, resources and expertise to break down the complexity of programming into easy-to-understand parts.

One other cautionary note when evaluating software to fit your application, make sure the software was designed specifically for machine vision applications rather than ‘generic’ image processing software sold into medical imaging, microscopy or other non-industrial, general purpose image analysis applications. While powerful, these packages do not provide the real-time control typical of machine vision applications.

**MODULARITY AND FLEXIBILITY**

Any complex system becomes simpler when broken down into its constituent parts.

An important way to judge the character of a machine vision API is how it’s organized.

API’s that are broken down into modules for image acquisition, image processing, memory management, and display allow the developer to focus on a smaller set of functions rather than a single list of hundreds of commands. A modular approach significantly reduces programming time while shortening the learning curve.
THE COMMAND AND CONTROL API
The choice of hardware vendor will usually determine the command and control API that will be used. While third party control API's exist (in the context of a complete imaging API) they usually reside on top of the hardware vendor's API. The latter will also offer more functionality and flexibility. These command and control API's come in the form of high-level programming libraries to control and optimize the image capture, transfer and display process. These libraries also allow the programmers to set operational parameters within the software platform to interface the peripheral's specific I/O connections, setting clock speeds and trigger conditions among other specifications. A good command and control API will default to the most common operational parameters for a given application, but still allow the programmer to change important parameters to achieve specific performance levels.

An important consideration when evaluating the modularity of a software package are the utility tools. One extremely important utility to consider is the camera configuration tool. The Camera configuration utility usually comes bundled with the command and control software and facilitates the efficient set-up and performance of the camera interface. But not all camera configuration utilities are alike.

Here are a few key features to look for:

- An easy to use drag and drop GUI with camera-centric parameters
- Live grab and display windows for real-time parameter tweaking
- High performance waveform tools to optimize signal inter-relationships and precise setting
- Analog and digital camera control
- Memory management (images, vectors, lookup-tables, etc)
- Image file support

HARDWARE INDEPENDENCE
An important long term consideration in the choice of a machine vision software platform is hardware independence. Put simply, this capability permits a machine vision application to operate, without modification, with several different hardware platforms from the same vendor. The advantage to the machine vision OEM is the capability to offer a range of system platforms at different price and performance points with little or no software modifications. In order to leverage this capability look for a software API that abstracts key hardware capabilities and characteristics in the programming environment. Vendors should openly promote this capability in their product literature.

HELPFUL APPLICATION DEMOS
Another way to assess the sophistication and strength of image analysis software is by the quality, number and type of demos that come with the program. Demos can be invaluable to users or integrators facing new applications because they walk the programmer through the most common image analysis functions and highlight critical parameters for each object class. Dedicated machine vision software vendors bundle these demos with their offering as an extension of their software support programs.

Don't forget, the level of bundled support a company provides may be indicative of overall quality of after-market support you'll receive, which is often the difference between a successful
machine vision application and one that gets shelved because of project overruns and poor performance. Additionally demos should be provided as source code demos, and become a starting point for OEM’s in their application development.

**PROGRAMMING LANGUAGE**

For advanced image processing and analysis programs, programmers also need to consider the software’s native programming language (C, C++, Visual Basic, etc.), as well as its flexibility to accommodate code from other languages. Each programmer has his or her own favorite coding language, but the best image analysis programs provide methods for programmers to work in familiar surroundings, regardless of the platforms native language. For instance, a program may be based on the object/class structure common to C++, but if the system also includes Active X functionality, this mid-level language can act as a go between to allow the programmer to work in Visible Basic, Borland, Visual C++, and C++ Builder. A final consideration in this area is the question of Active X versus Microsoft’s new .Net command structure.

Many machine vision software providers are maintaining Active X support until legacy support of Active X is guaranteed in the .Net environment. Moving completely to .Net today could mean that your image analysis program cannot talk to programs that use Active X or .Com functions.

**THE IMAGE PROCESSING AND ANALYSIS LIBRARIES**

Now it’s time to discuss what might be considered the primary engine of the API _ the libraries. Machine vision software libraries can be broken down into two sections _ image processing and image analysis. Imaging processing libraries work to manipulate and optimize the image. Developers need to look for libraries that deliver a broad range of functionality that includes a comprehensive pallet of image filters including

- Point-to-point operators (averaging, subtraction)
- Neighborhood filters (with variable and flexible kernel sizes)
- Morphological tools (grey scale and binary)
- Basic and locally adaptive threshold techniques
- Geometry tools that allow the image to be cropped, flipped, rotated, or sheared
- Measurement tools to perform horizontal and vertical projections and calculate vector differences
- Segmentation to separate foreground objects from the background

The key here is to ensure that there is a wide range of image manipulation functions that are both fast and flexible. Look for image processing libraries that are highly optimized to take advantage of technological advances like SIMD (single instruction, multiple data) instruction sets. MMX-2 and Altivec are two examples of popular optimizations targeted for operation on the Intel x86 architecture and the Motorola PowerPC (G4) platforms respectively.

While the image processing libraries condition the image data to a useful state, it is the image analysis libraries that actually extract useful information from the image itself upon which accept/reject decisions can be made. Common image analysis routines include: gauging or measurement, pattern matching, blob analysis, optical character recognition (OCR) and barcode decoding. Both image processing and analysis functions are combined in sequences to form machine vision algorithms such as golden template matching.
Eighty percent of machine vision applications can be broken down into two primary functions: location analysis and inspection. Moreover, the latter, location analysis is commonly used as a first step to inspecting un-fixtured parts. Location analysis therefore is a very important tool to be found in an image analysis library. Location analysis relies on pattern matching, to locate unique patterns on objects that help to locate and determine the orientation of the inspected component. Therefore an API’s search algorithms are important and are worth some time to evaluate properly.

The most common pattern matching technique, normalized cross correlation (NCC), compares the normalized energy of a trained pattern to the normalized energy of similarly sized Region of Interest (ROI) in the image being inspected. This mathematically intense technique is robust against intensity variations in the image (compared to the trained pattern) but suffers when either the orientation (rotation) or scale of the trained and inspected patterns is different.

Therefore most machine vision component vendors have transitioned their pattern matching or search tools from the older NCC approach to shape or geometric searches that represent the image as vectors rather than a map of pixel values. Geometric search algorithms operate efficiently regardless of whether the part under inspection appears to be the same size (scale) or pointing in the same direction (orientation) as the trained pattern. Therefore look for vendors that offer both NCC and Geometric search tools in their libraries.

When analyzing the efficacy of any image analysis tool, the user should consider how the supplier has implemented the algorithm based on ease-of-use and performance.

Ease of use relates to the training of the algorithm. For example, training a Search tool should be intuitive, allowing the user to view and alter the stored pattern as it is displayed on the vision system’s monitor through an easy-to-understand interface.

The algorithm should automatically extract the edges and contours that define the pattern, and allow the programmer to change them on screen as necessary. This should all be accomplished through an intuitive GUI that guides the programmer through part/pattern training, algorithm optimization and execution.

Machine vision performance depends on the speed, robustness and accuracy of the underlying software tools and algorithms. These are always a compromise between what runs the fastest and what is most comprehensive. A desirable algorithm will not bottleneck the inspection process while minimizing false negative or false positive results. Robustness describes the algorithms ability to identify a part even in poor conditions, such as changing lighting conditions, blurry images, when parts are grouped together, partially occluded by another part or touching. One example of accuracy in a Search tool is the degree to which the tool determines the pose of the part under test versus the parts actual position. Illumination, image distortion from camera lenses, and jitter introduced by the frame grabber are all factors that adversely affect accuracy. A common method to check the accuracy of a Search tool is to take an image from the vision system, offset the image’s position by less than a pixel and rotate the image by a fraction of a degree. Then run the Search tool on using the original pattern on the transformed image and see if the results match the known transformation. Repeat several times using different images and statistically measure the standard deviation using the 3-sigma rule.
SOFTWARE RELIABILITY AND ROBUSTNESS
While programming language, applications demos, control APIs or utilities and image processing and analysis libraries are critical considerations in evaluating machine vision software programs, there is another important factor. Developers need to consider the overall reliability and robustness of the entire system – here too, software selection can play a critical role. Unfortunately, the question of reliability and robustness in machine vision software is a complex one. There is no statistical measure of robustness and reliability that will work across all vision software platforms, however, machine vision users should understand how reliability and robustness are achieved in machine vision software and select a vendor accordingly. Reliability and robustness refer to the software’s ability to track and verify all I/O and internal image processing functions to make sure they performed as expected.

Reliability and robustness can only be delivered through a comprehensive framework that delves into every aspect of the system’s operation from triggering and image acquisition through processing and output.

Consider an inspection application on a high-speed manufacturing line. A part approaches a sensor that triggers the machine vision system to acquire an image.

The vision system, in turn, sends a signal to the camera to begin reading out the image information, and then the frame grabber begins collecting image data in a memory buffer before passing it to the image processing engine. Hopefully, the part has not moved while the microprocessor was servicing other system resources or busy with some other function. Once the image is in the processor’s memory, the algorithms convolve the image and a result is issued for action and archival. But how does the operator know that any of this actually happened? What if two parts are placed too close together, or the trigger sensor “bounces” and sends two “initiate” signals rather than just one? What if there’s an unusually long delay between trigger and acquisition, or the camera only reads out half of the image? The results are false negatives and high reject rates that cost the end user productivity. To insure the reliability and robustness of the vision system, there must be the ability to track the beginning and end of each event very precisely, store the tracking data, and react when events transpire out of sequence or contrary to expectations. Many software providers interject some form of tracking measure, but only a system that tracks from trigger-to-image and uses an external temporal (clock) or spatial (encoder) sequencer can guarantee that the data coming into the system is the right data at the right time, otherwise, it’s “garbage in, garbage out.”

Unfortunately, software providers that do not integrate tightly with the hardware in the vision system may depend solely on the microprocessor’s internal clock for time stamping events, which is sometimes unavailable for tracking because of OS or other computational requirements. In these cases, reliability tracking systems should also be configurable so that the user can set how much processing power is used for tracking purposes and avoid creating additional error conditions. This solution is adequate for many applications, but the best solution requires the closest possible integration between hardware and software. For instance, not all cameras are programmable or confirm trigger signals, the beginning of the CCD readout or the end of the frame. Not all frame grabbers contain an external clock or some mechanism to speak directly with an encoder on the manufacturing line that is separate from the internal microprocessor.

And just as important as tracking internal operations are the software’s ability to react to problem conditions. For instance, if the frame grabber collects only half a frame, will it dump the
bad data? Store the bad data for later review? Or will it append that image data to the next frame, potentially creating two error conditions?

When selecting software for high-end applications, system designers should ask vendors about their reliability tracking mechanisms throughout the trigger-to-image chain as well as what the system will do when problems arise...because, even with the best automation system, machines DO occasionally make mistakes.

**OPERATING SYSTEMS - PLAN FOR THE FUTURE**

Although the machine vision industry has recently been transformed by the emergence of dedicated or smart systems for low-end applications, the overall trend – even among these systems – is to leverage advances in electronics to provide more powerful systems. In the digital world of machine vision, power means processing speed and bandwidth.

Cutting edge machine vision applications are already collecting images that exceed the standard 32-bit Windows 1 GB limitation on a single block of memory. At 1 GB,

Windows can still manage memory, but it significantly slows down the image processing functions. Adding frame grabbers with large memory buffers and software capable of mapping this additional memory without introducing significant latency in data transfers allows single memory chunks to grow up to 4 GB, however, today, web inspection and similar high-speed applications are generating image files up to 16 GB; unfortunately, at 4 GB, 32-bit operating systems fall apart.

There are 64-bit OS available today, however, these systems are not compatible with 32-bit code, posing a problem for machine vision software providers. If your application is likely to generate multiple gigabit data streams, then ask the software vendor what they provide in terms of additional memory allocation functions within the 32-bit environment and what their plans are for developing software in 64-bit environments.

Unlike previous OS updates, the move to 64-bit requires software providers to essentially start from scratch. So if a vendor has not already begun developing a 64-bit architecture, it is not likely to be done any time soon. Other operating system issues include the gathering momentum behind Linux, and Linux 64, along with Windows Server 64. Serious machine vision software providers already offer products for these OS or will share their development plans with you.

A good software selection will guarantee hardware and OS independence, while planning for future growth through a clear support structure, both today and tomorrow.

DALSA Coreco

www.dalsa-coreco.com