

High Speed, Real-Time Machine Vision

Perry West – Automated Vision Systems, Inc.



Table of Contents

- Introduction** 1
- Latency** 3
- Imaging** 5
 - Triggering – The Stimulus 5
 - External Sensor 5
 - Part Detection Through Imaging 6
 - Image Acquisition 6
 - The Role of the Camera 6
 - The Role of the Frame Grabber 14
- Image Processing** 20
 - Processing Architecture 20
 - Processor Speed 20
 - Instruction Set 21
 - Software Development 21
 - Single Processor 21
 - Multi-Processor 22
 - Operating System 23
 - Application Software 23
 - Resynchronization 24
- Latency Calculation** 26
 - Part Detector 26
 - Capture Command 26
 - Strobe/Shutter Trigger 26
 - Exposure Time 26
 - Video Transfer 27
 - Transfer to CPU 27
 - Image Processing 27
 - Resynchronization 27
 - Time Base 28
 - Output Activation 28
- Example 1** 29
 - The Requirements 29
 - Design 29
 - Latency Evaluation 30
- Example 2** 31
 - The Requirements 31
 - Initial Design 31
 - Solving the Uncertainty in Part Position 33
 - Solving the Image Blur and Motion Tear Problems 33
 - Number of Cameras 34
 - Frame Grabber 36
 - Computer 36
 - Operating System 36
 - Application Software 36
 - Ejector Operation 36
 - Summary 37

List of Figures

Figure 1 – Value Functions for Soft and Hard Real-Time	1
Figure 2 – Event A	4
Figure 3 – Event B	4
Figure 4 – Events A & B Combined	4
Figure 5 – Event C	4
Figure 6 – Events A, B, & C Combined.....	4
Figure 7 – Simplistic CCTV Timing	7
Figure 8 – Basic CCTV Image Acquisition Latency.....	7
Figure 9 – Interlace Scanning.....	8
Figure 10 – Interlace Timing.....	8
Figure 11 – Motion Tear	9
Figure 12 – Progressive Scan Camera Timing	9
Figure 13 – Asynchronous Camera Timing	11
Figure 14 – Asynchronously Resettable Camera Timing.....	11
Figure 15 – Camera Timing with Strobe Illumination	13
Figure 16 – Camera Timing with Electronic Shutter	14
Figure 17 – Basic Frame Grabber	15
Figure 18 – Simple Frame Grabber System Timing.....	15
Figure 19 – Frame Grabber with Ping-Pong Memory.....	16
Figure 20 – Ping-Pong Frame Grabber System Timing.....	16
Figure 21 – PCI Bus Transfer Latency.....	17
Figure 22 – Single Processor.....	21
Figure 23 – Common Multi-Processor Approach	22
Figure 24 – Inspecting Bottles.....	31
Figure 25 – Dual Camera Configuration	35

List of Tables

Table 1 – Adding Up Latencies.....	3
Table 2 – Latency Compilation Template.....	26
Table 3 – Latencies for Example 1.....	30
Table 4 – Initial Compilation of Latencies, Example 2.....	32
Table 5 – Final Latencies for Example 2.....	37

Introduction

A widget moves along a continuously moving conveyor belt, trips a sensor, the vision system performs its inspection, and operates an ejector to remove a defective widget – all at high speed and in real-time. So, exactly what is high-speed, what is real-time` and what is the difference?

High-speed has no formal definition. To most people it means a very short time elapses between the command to take an image and the vision system's output. A better meaning for high-speed is that the vision system can take in image data at a high rate – continuously. That is, it can acquire and process images at a high rate (frames per second). Another meaning of high-speed sometimes used in industrial machine vision is that the machine vision system is always much faster than any other process step that limits the manufacturing line speed. A final use for the term high-speed relates to its ability to image products that are moving at rates that can cause significant image blurring. What these definitions don't tell us is what rate sets the threshold for a high-speed vision system. There is no technical definition of high-speed such as 10 images/second. The implication, though, is that a high-speed system is one that can operate faster or have a higher data rate than ordinary systems.

Real-time, on the other hand, doesn't necessarily mean really fast. It does mean the system will be timely – the results will be provided when they are needed. A vision system's timeliness can be viewed as a value function. In a typical machine vision application where the part stops for viewing and is not advanced until the vision system has made its decision, there is some value to the vision system making a decision by a certain time so that it is not the slowest element of the production process. If the vision system is earlier than this time, its value is not increased because the production line cannot go any faster. If the vision system is almost always on time, but very occasionally a little slower, its value might be diminished only slightly. However, as the vision system becomes slower on average or more frequently, it becomes a bottleneck, and its value declines eventually becoming negative. Such a system would be called soft real-time.

In the example in the first paragraph above of a widget on a continuously moving conveyor, the value function is high only when the reject decision is output while the widget is passing next to the ejector. If the vision system is either too early or too late, its value is negative. This system would be called hard real-time.

Figure 1 below shows value functions for soft and hard real-time.

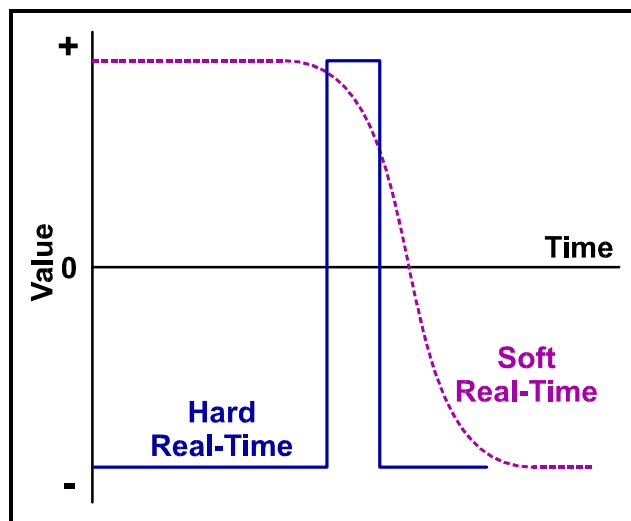


Figure 1 – Value Functions for Soft and Hard Real-Time

In the example at the beginning the widget tripping the sensor is the event and the vision system's decision to operate or not operate an ejector is the response. In technical terms the time between an event and a response to that event is the latency. The important characteristic in a real-time system is that latency is deterministic. That is, the response will always happen at a predictable time after the event.

The requirements for a system's speed and the determinism of its latency greatly affect the engineering challenges in developing the system. The key to achieving high-speed is understanding how to combine system components to get the best throughput. The key to achieving real-time performance is understanding and managing latency.

Although high-speed and real-time are technically different criteria, they are most often found together. This paper focuses more on real-time issues with discussions about how to achieve high-speed performance. The paper starts with a description of latency and then focuses on high-speed and real-time issues in two areas: imaging and image processing. Finally, this paper looks at two examples and discusses how the systems can be engineered to meet requirements. These examples are composites of real-world machine vision systems. In that regard they represent real engineering challenges that have been faced and met. The examples do not show the complete development of a vision system. A comprehensive design might affect the choices illustrated in the examples.

Special thanks go to the following people who provided input and discussion on the topics covered in this white paper: Duke Bettis, Coleman Gormly, Allan Jackson, Brian Smithgall, Keith Vasilakes, and Jim West.

Latency

Latency is the time between an activity's initiation and its result. While a system has an overall activity, from trigger event to output signal, the system itself is comprised of a number of internal events usually occurring serially. Each latency in the system has some degree of uncertainty ranging, in our technology, from a few nanoseconds to several seconds. A system's latency is comprised of the latencies of the individual processes within the system. In the example shown graphically below in Figure 2 through Figure 6, there are three serial events A, B, and C, each with its own latency and uncertainty. The latency is shown as a probability function for the time the event will complete. If we know the probability functions for the latencies, we can combine them by convolving their probability functions together as shown.

Fortunately, in real-time systems, the designer is usually not faced with convolving the latencies together. Determining the total latency and its minimum and maximum from the individual latencies is sufficient. For the example below:

Event	Latency	
	Minimum	Maximum
A	3	12
B	8	57
C	17	35
Total	28	104

Table 1 – Adding Up Latencies

Note that in designing a real-time system, typical latency is not meaningful unless the variation in latency is negligible in comparison to the timeliness required. Following sections will discuss how to minimize latency and its variation for different parts of the machine vision process and how to reduce the variation in latency by resynchronizing the data flow. Resynchronization is a form of pipelining which removes the variation in latency by using a time reference.

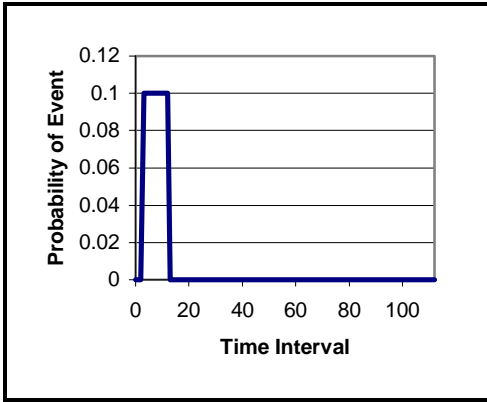


Figure 2 – Event A

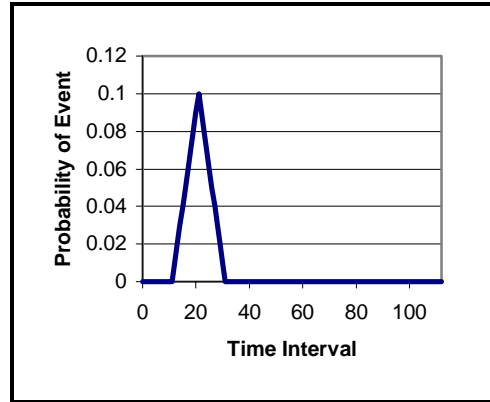


Figure 5 – Event C

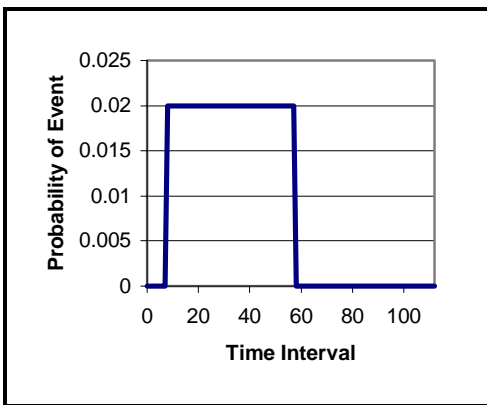


Figure 3 – Event B

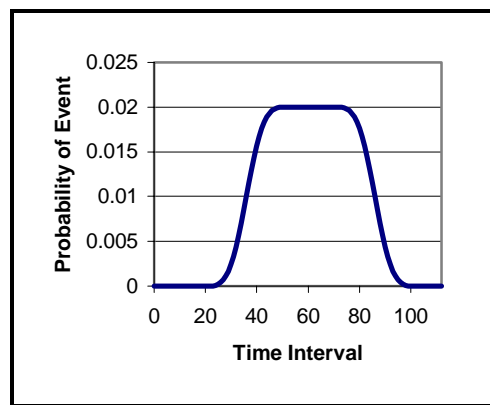


Figure 6 – Events A, B, & C Combined

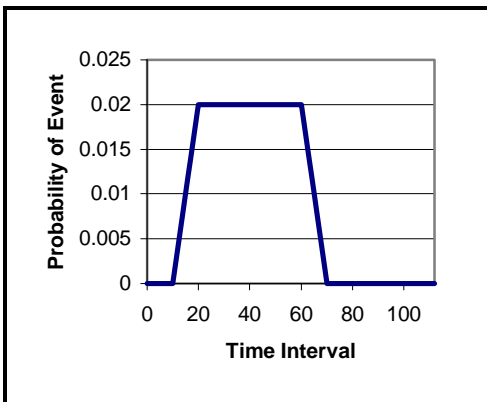


Figure 4 – Events A & B Combined

Imaging

There are three major elements to real-time imaging: the trigger event that starts the process, acquisition of the image by the camera, and the transfer of image data from the camera to frame grabber and from the frame grabber to the processor.

Triggering – The Stimulus

From the perspective of designing a vision system, the trigger is a signal received by the vision system that tells the vision system to acquire an image for processing. However, from a system design perspective, the critical event is the arrival of a part in position to be imaged. There is always a difference in time between the arrival of a part and the receipt of a trigger by the vision system that can range from nanoseconds to milliseconds.

There are two principal methods of triggering a vision system. The first is to use an external device such as a photosensor to signal the presence of a part in the viewing area. The second is to have the vision system continuously image the viewing area and through image processing determine when a part is present. We will discuss both methods from the perspective of real-time operation.

External Sensor

Perhaps the most common way to trigger a vision system is with a photosensor or other device that detects the presence, or more accurately the arrival, of a part in the viewing area. There are certainly other methods of externally triggering. For example, in some system a programmable logic controller (PLC) monitors the motion of parts on an index table, and sends a trigger signal to a vision system after a part is indexed into the viewing area.

The important characteristic with external triggering is to understand there is some delay and some variability in timing. With the photosensor, the delay (latency) may be between a few microseconds to several milliseconds, and the uncertainty may be less than a millisecond or even as short as a few microseconds. In the case of the PLC and the index table, the uncertainty is the uncertainty of the signal out of the PLC. In general, programmable controllers have a delay determined by their scan time: the time for them to interrogate all inputs, determine a new state, and update all outputs. The scan time depends on the speed of the PLC, the number of inputs, the number of outputs, and the complexity of the logic. A typical PLC scan time is around 3 msec., but the uncertainty associated with the PLC is $\pm 2/3$ ds of a scan time.

The external trigger signal must be received by the vision system, and it must in turn initiate a response. The response usually starts with the initiation of the camera's exposure including controlling a shutter and triggering a strobe light.

If the signal is received into a general-purpose input port on the vision system, it will be up to software to detect the presence of the signal, and initiate events. Some frame grabbers accept an external signal directly, and initiate the exposure without software intervention.

If software is used to detect the trigger signal, it will do this either by polling or by an interrupt. When polling, the software periodically tests the input to see if the signal is present. The minimum latency is the time it takes the processor to read the input port, to determine the trigger signal is present, and to issue a capture command to the frame grabber that starts the image acquisition. The maximum latency is the minimum latency plus the maximum interval between the processor checking the input port for the trigger

signal. This maximum time may not be just the software loop time to reread the input port, but it also includes whatever overhead time may be imposed by the operating system and whatever delays are possible through other concurrent tasks the processor is running.

When the trigger signal generates an interrupt, the minimum latency is the time it takes the processor to recognize the interrupt, transfer control to the interrupt service routine, and issue a capture command to the frame grabber to start the camera's exposure. The maximum latency is the minimum latency plus whatever delays the computer and operating system might impose in recognizing and servicing an interrupt. Requiring the processor to service a higher priority interrupt might cause such a delay.

Frame grabbers that receive and process the trigger signal directly initiate image acquisition with negligible latency; the delay is usually less than a microsecond. Depending on the frame grabber, it may also have the option to send an interrupt to the CPU when acquisition is initiated as well as when acquisition is completed.

Part Detection Through Imaging

There are some situations well suited to having the vision system examine the image data to detect the presence of a part. In these cases, image acquisition must be commanded by software; the software triggers the event. After image acquisition, some image processing is performed to determine if a part is present for viewing. In some implementations of this technique, the same image that is used for detection (triggering) is retained and analyzed as the primary image. In other cases, a new image is acquired by software command and its contents analyzed.

The latency for part detection through imaging is the latencies for exposure, transfer of image data through the system, and initial processing. If the same image can be used for analysis, there is no further latency for image acquisition.

For discrete parts, part detection through imaging is generally more complex, less reliable, and has longer latency than using an external sensor to trigger the process.

Image Acquisition

This section on image acquisition covers area cameras and their various operating modes together with the frame grabber. It covers the challenges involved in high-speed and real-time image acquisition and how components may be used to manage the system's latencies. Image acquisition finishes with the image data being transferred to the processor.

The Role of the Camera

The camera is the vision system's sensing element. For economic reasons, television cameras came to dominate machine vision applications, but when applied to machine vision they brought with them compromises that limited the speed and real-time performance of systems. New camera technology has evolved which addresses the needs in real-time and high-speed imaging. The material that follows discusses the characteristics of television cameras, and shows how each limitation of camera performance affecting real-time and high-speed performance has been addressed with other camera technology. It should be noted that consideration of a camera with its unique characteristics must necessarily include the camera's interface, the frame grabber, which needs to support those characteristics.

CCTV Camera

To begin an examination of how camera exposure time affects both high-speed and real-time performance, we will look at the classic case of where a CCTV camera is used in a machine vision application. The simplified timing diagram below, Figure 7, shows how this might work. The basic scenario is that the vision system receives a trigger and acquires the image from the camera. However, the CCTV camera is continuously running. That is, while it may be synchronized to a frame grabber in a vision system, it is always exposing and reading out images. So with a trigger arriving asynchronously with the camera's scanning, the vision system has to wait between 0 and 33 milliseconds¹ before the camera's image sensor transfers the photogenerated charge to its output registers and begins sending a video signal to the frame grabber. The video transfer time from camera to frame grabber adds an additional 33 msec. to the total time to acquire an image.

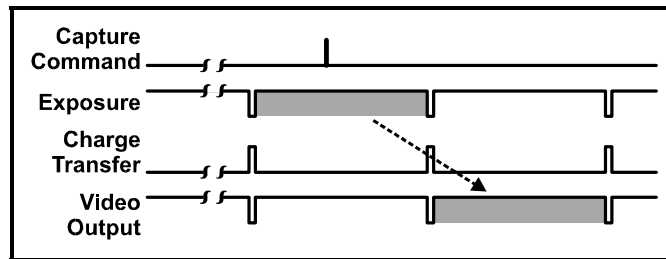


Figure 7 – Simplistic CCTV Timing

Neglecting the trigger signal's latency, we can look at this vision system's latency in acquiring an image combining both exposure and video transfer. This is shown in Figure 8.

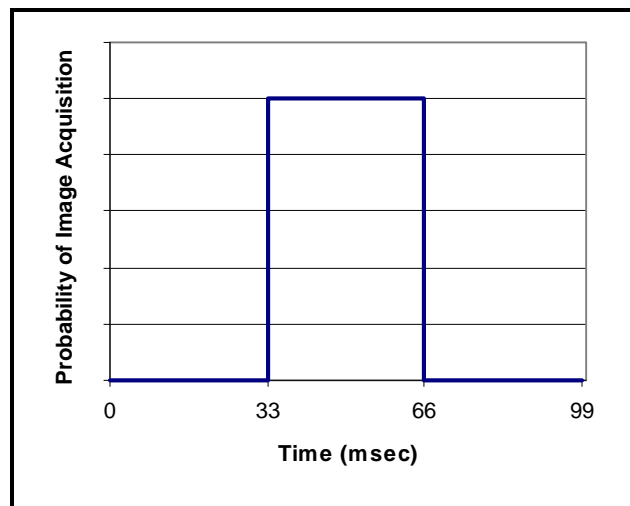


Figure 8 – Basic CCTV Image Acquisition Latency

It is apparent that a basic CCTV based vision system will take from 33 to 66 msec. to acquire an image into the frame grabber, not including the trigger signal latency or image transfer from the frame grabber to the processor. For relatively slow speed applications (a few parts per second or slower) where the part is stationary during imaging, this vision system is adequate. However, for systems needing higher speed or imaging moving parts, this configuration is usually not satisfactory.

¹ In this paper, a frame rate of 30 frames per second or 33 msec per frame will be used. For much of the world, the frame rate is 25 frames per second or 40 msec per frame.

The latency variation of a CCTV based machine vision system can be cut in half by understanding how the CCTV image is transferred as video, and having a more flexible frame grabber. The CCTV camera exposes and transfers an image (frame) as video data in two interlaced fields. The first field (odd field) consists of scan lines 1, 3, 5, 7, The second, or even, field consists of scan lines 2, 4, 6, 8, This is shown below in Figure 9. Also shown below in Figure 10 is the timing diagram showing the exposure time and video transfer for interlaced scanning.

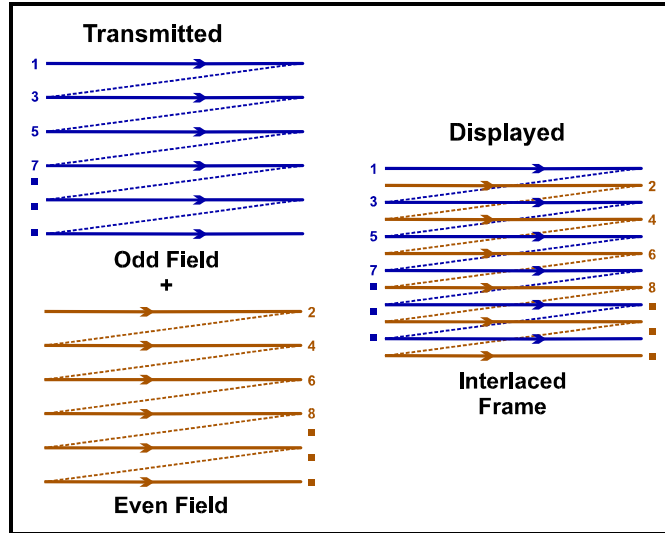


Figure 9 – Interlace Scanning

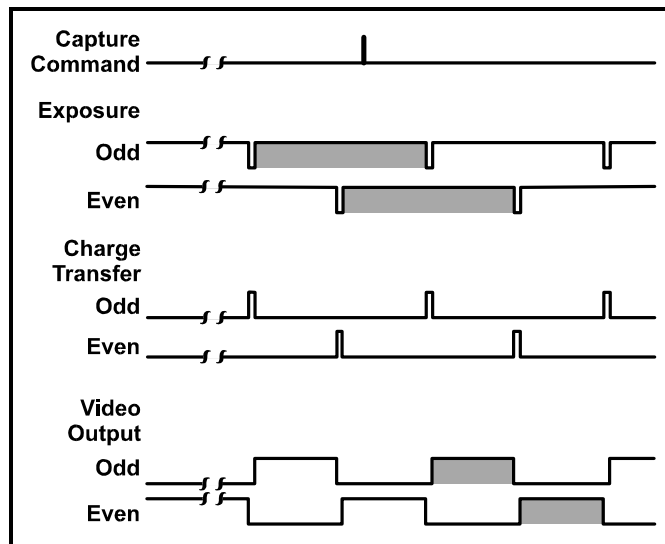


Figure 10 – Interlace Timing

Note that there are two options for video transfer. Most basic frame grabbers only start acquisition when the odd field is being transferred; for these, the uncertainty in latency is 33 msec. However, there are frame grabbers that can begin acquisition with either the odd or the even field. These devices reduce latency uncertainty by half to 17 msec.

Motion Tear

Many real-time and high-speed vision systems need to image moving parts. One of the first issues with CCTV cameras imaging moving parts is that their exposure as well as their video output is interlaced. The effect of interlaced exposure on a moving part's image is that the vertical edges are offset giving the appearance of a picket fence. This effect is called motion tear and is shown in Figure 11.

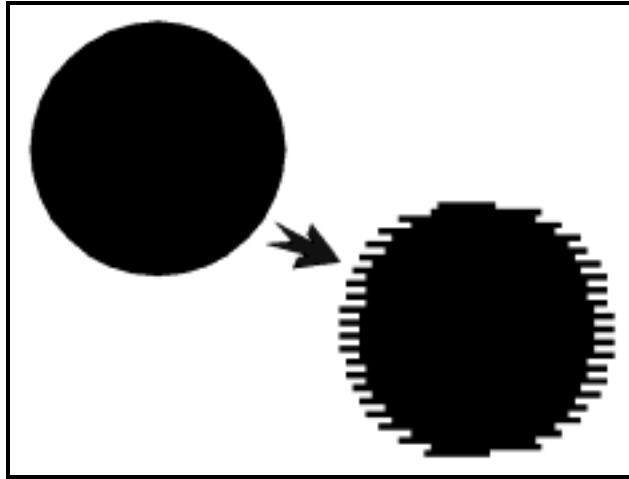


Figure 11 – Motion Tear

If the camera were turned 90 degrees to the direction of motion so that the normal horizontal image direction was vertical, the interlace offset would not be apparent in the image. However, this would introduce a one scan line (pixel) uncertainty in the location of vertical edges due to interlaced scanning. It would also distort the edge waveform and make sub-pixel measurement impractical.

The magnitude of motion tear can be computed from the equation:

$$MT = V_p * T_f * N_{ph} / FOV_h$$

Where:

- MT is the magnitude of motion tear in pixels
- V_p is the part velocity
- T_f is the time for each field
- N_{ph} is the number of pixels in a scan line
- FOV_h is the field-of-view size in the horizontal direction

One common solution to eliminate interlace offset is to use a progressive scan camera. A progressive scan camera can have the same resolution as a CCTV camera, but does not use interlaced scanning. The waveform for a progressive scan camera is shown in Figure 12.

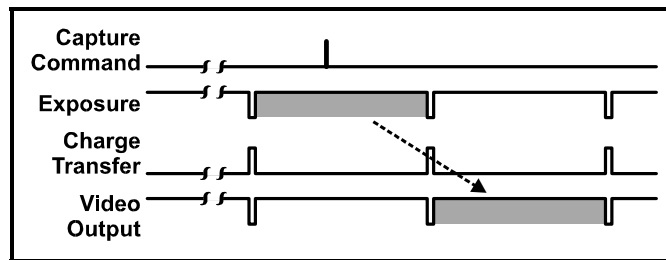


Figure 12 – Progressive Scan Camera Timing

Progressive scan cameras do not usually conform to television standards. So the selection of a progressive scan camera will also change the choice of a frame grabber. However, there are a few progressive scan cameras which expose the entire frame at one time, transfer it into storage registers, and then read out the storage register in an interlaced fashion. These cameras can be used with frame grabbers which do not accommodate full-frame non-interlaced video data output.

Part Position Variation

In acquiring an image of a moving part, the variation in latency causes some part-to-part variation in the position of parts in the field-of-view. The vision system must have a field-of-view large enough to tolerate this positional variation. The positional variation is given by:

$$P_V = L_U * V_P$$

Where:

P_V is the variation in part location in the direction of part motion

L_U is the uncertainty in latency in the acquisition of the image

V_P is the speed of the part

For example, a part traveling at 1cm/sec imaged by a vision system with an uncertainty in image acquisition latency of 33 msec. will have a position variation of:

$$P_V = .033 \text{ sec} * 1 \text{ cm/sec} = .033\text{cm}$$

That is, a part traveling at only 1 cm/sec will have a position uncertainty of 1/3mm when imaged with a standard CCTV camera. It should be emphasized that this is the uncertainty in position of the object in the image, and not the blur in the image caused by the part's motion. Concerns about blur are addressed below.

Motion uncertainty can only be assessed in terms of its impact on the vision system's performance. One consideration must be any additional complexity of image processing to handle the variation in part position. Another consideration is that allowing for positional change forces the field-of-view to be larger than if there was no uncertainty in position, and this larger field-of-view reduces the vision system's resolution and accuracy. There are two approaches to working with part location uncertainty. The first approach calculates the field-of-view taking location uncertainty into consideration, and from the field-of-view determines the required image resolution of the camera. The second approach is to determine from some given equipment, including the camera's image resolution, the maximum field-of-view that meets resolution and accuracy requirements, and from that field-of-view determine how much allowance for part position variation is available. In this second approach where the maximum part location uncertainty is known, the maximum latency uncertainty is:

$$L_U = P_U / V_P$$

High Speed Cameras

Using a high-speed camera improves system speed, it also improves real-time performance, such as the uncertainty in part location discussed above, by reducing both the latency and its uncertainty. There are cameras that operate at 60 frames per second. These reduce the latency to between 17 and 33 msec. Obviously these cameras do not conform to television standards; they do not have to implement interlaced scanning. They are progressive scan cameras, and are discussed above.

There are also specialized cameras that provide even higher speeds that can be used in machine vision. Typically, these cameras use two or more image data outputs in parallel to achieve the higher bandwidth.

They require special frame grabbers that can accept multiple simultaneous image data sources, and reconstruct the image in memory.

Partial Scanning Cameras

Still another way to speed up image acquisition is to use a camera that offers the option of partial scanning. In partial scanning, only a group of scan lines are transferred (e.g., scan lines 151 through 280). Data in the remaining scan lines is discarded, and not sent to the frame grabber. This trades off reduced vertical information for increased speed.

Asynchronous Cameras

An even more effective way to reduce latency is to use an asynchronous camera or an asynchronously resettable camera. These cameras start their exposure and video transfer sequence on command. An asynchronous camera works by holding the image sensor reset until a start signal is received. An asynchronously resettable camera is constantly scanning and providing an output; even if the image sensor is being held reset. Upon receiving a signal to start a new image, it resets the image sensor at the start of the next video line. After receipt of a start signal, the exposure is begun, and at the conclusion of the exposure time, the video is transferred to the frame grabber. The timing diagram for an asynchronous camera is shown in Figure 13 below. For an asynchronously resettable camera, the timing is shown in Figure 14 below.

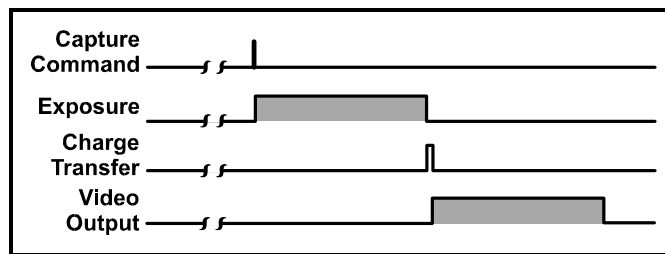


Figure 13 – Asynchronous Camera Timing

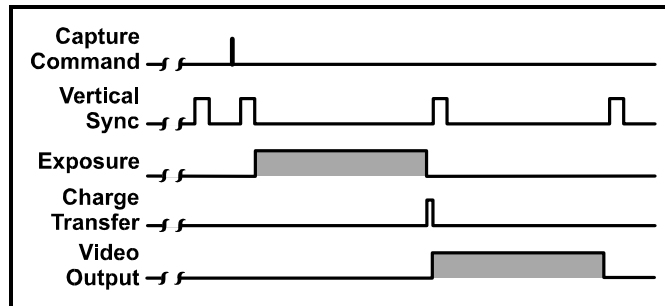


Figure 14 – Asynchronously Resettable Camera Timing

For the asynchronous camera, the latency is the sum of the exposure time, the charge transfer time (which is negligible), and the time to transfer the video data to the frame grabber. Because these cameras do not conform to any standards like those for CCTV, the actual latency is dependent on the particular camera and how it is operated. One important factor is that the uncertainty in the camera's latency is on the order of one pixel clock – less than 100 nsec.

The asynchronously resettable camera adds a small additional latency uncertainty on the order of one horizontal scan line time typically up to 63.49 μ sec.

Image Blur

If part motion is possible during exposure, there will be image blur. The magnitude of the image blur depends on the rate of part motion, the size of the field-of-view, and the exposure time. Suppose a camera is viewing a 10cm field-of-view, its exposure time is 33 msec., the image is digitized into 640 pixels horizontally, and the part is moving at a speed of 1 cm/sec. The magnitude of blur in pixels is:

$$B = V_p * T_E * N_p / FOV$$

Where:

- B is the blur in pixels
- V_p is the part velocity
- FOV is the field-of-view size in the direction of motion
- T_E is the exposure time in seconds
- N_p is the number of pixels spanning the field-of-view

In the example above, V_p is 1 cm/sec, T_E is 33 msec., N_p is 640 pixels, and FOV is 10cm. Then:

$$B = 1 \text{ cm/sec} * .033 \text{ sec} * 640 \text{ pixels} / 10\text{cm} = 2.1 \text{ pixels}$$

In most cases, blurring becomes an issue when it exceeds one pixel. However, in precision measurement applications where sub-pixel precision is needed, even one pixel of blur may be too much.

A useful equation gives the exposure time (T_E) for a Blur of one pixel:

$$T_E = FOV / (V_p * N_p)$$

The only practical methods of reducing image blur are either to slow or stop the part motion or reduce the exposure time. In a real-time system, and especially in a high-speed system, slowing the part is usually impractical. There are very good methods for reducing the exposure time either by using an electronically shuttered camera or by using strobed illumination or both.

Strobe Illumination

Using a strobe light, whether a xenon strobe or a pulsed LED light source, can freeze part motion in the image. The light pulse out of a xenon strobe source is typically around 10 μ sec, but can range from as little as 250 nsec to as much as 8 msec. Pulsed LED light sources have been built which operate with a light pulse shorter than 1 μ sec, but normal light pulse widths range 5 μ sec and longer. The timing diagram below in Figure 15 shows typical strobe timing.

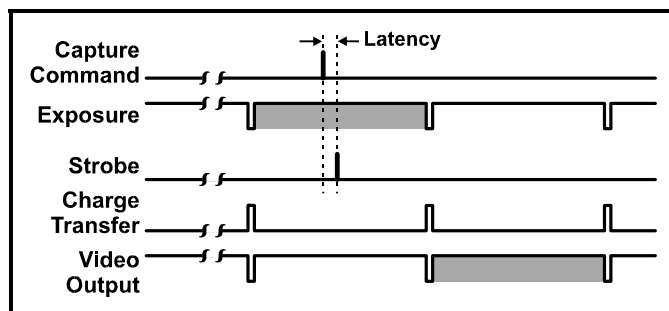


Figure 15 – Camera Timing with Strobe Illumination

Notice there is a delay or latency between the capture command signal and the firing of the strobe light. This latency is the sum of the delay between the capture command and the command to the strobe plus the delay between when the strobe source receives its command and the light pulse is generated. The strobe lamp and its controller also contribute an uncertainty to the latency called jitter. Typical jitter time for xenon strobes is on the same order as the duration of the light pulse.

It is critical that the strobe be fired during the camera's exposure time and not during a time when the camera's image sensor is transferring the photogenerated charge to the output registers. For CCD image sensors, both interline transfer and frame transfer, it is important that the strobe not fire during the charge transfer time. For sequential readout image sensors such as CID and many CMOS sensors that do not use an output register, the strobe should only fire during the period between image readouts. It is very difficult to control strobe triggering through software because of the great uncertainty in software latency. Systems that successfully use strobe illumination have the trigger signal received by and the strobe light triggered by hardware circuits; typically these circuits are part of the frame grabber. Frame grabbers that can automatically trigger the strobe normally have the circuitry to recognize when the strobe is allowed to fire during the camera's scanning cycle, and will hold off triggering the strobe until it is allowable. Using hardware circuits such as on a frame grabber to control the strobe insures the latency and its uncertainty are minimized. It also minimizes both the part's blurring and its uncertainty in location in the image.

Strobed light sources are very intense for a very brief duration, but their average light output is not necessarily as great as ambient illumination. Therefore, it may be necessary to shield the viewing area from ambient light to insure good signal-to-noise (the ratio of engineered illumination to ambient illumination). Also, the allowable intensity for a strobed light source is a function of the pulse repetition rate.

Electronic Shutter

An alternative to a strobed light source is an electronic shutter. The sensing elements in a camera's image sensor are reset at the end of each exposure period as the charge from the sensing elements is transferred to the output register. This reset on the sensing elements can be maintained, and released for a short time just before transferring the charge from the sensing elements into the output register. The timing for a progressive scan camera with electronic shuttering is shown in Figure 16. The maximum speed (shortest duration) of electronic shutters varies from camera to camera; it is typically around 20 μsec , but can be as low as 1.25 μsec .

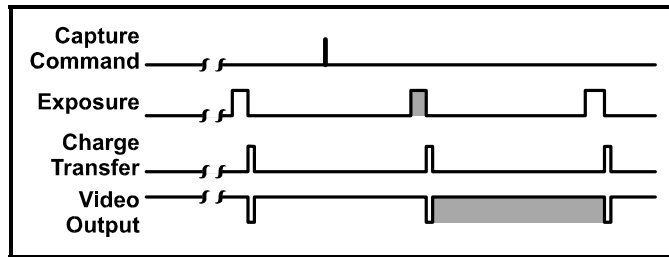


Figure 16 – Camera Timing with Electronic Shutter

Depending on the design of the camera, an electronic shutter is controlled either by circuitry in the camera or by an external timing signal provided from the vision system. In most cameras, the electronic shutter causes exposure during the interval just prior to transfer to the image sensor’s output registers. Sony has implemented a proprietary method, which they call Donpisha, to allow the electronic shutter to operate during almost any period of the camera’s timing.

If the shutter duration is fixed in the camera (e.g., by jumpers), there are no special interface considerations. If the shutter time is remotely selected or controlled, then the camera interface (frame grabber) must provide that control. In the case of externally controlled shutter duration, the shutter time must be closely coordinated with the rest of the camera timing. This can only be accomplished with a frame grabber with the appropriate ability.

One use of an electronic shutter other than to stop motion, is to control the camera’s total exposure, the product of light energy and exposure time. Some cameras implement this control in internal circuitry. However, the camera’s means of internally sensing correct exposure may work well for surveillance, but may not be well suited to a particular industrial application. For machine vision and scientific applications, a better approach is to have the vision system determine the exposure requirements, and adjust the exposure time through the frame grabber.

The Role of the Frame Grabber

In its most basic form, the frame grabber is the interface between a camera and a computer. However, even in this simple view, there are architectural trade-offs in how the image data is stored and transferred that may be more attractive for high-speed or real-time application than others.

Under the discussion of cameras above, there were a number of special camera features like progressive scan, asynchronous operation, and partial scanning that can have real value in real-time or high-speed systems. To realize the benefit of these camera features when needed, the frame grabber must support them.

Real-time and high-speed systems often demand special I/O control managed by hardware and synchronized with the camera, and this circuitry must be on the frame grabber. Some of these requirements were discussed under cameras above, including external triggering, remotely controlled exposure time, and strobe light triggering.

Image Transfer

In general, the industry is moving towards frame grabbers with shared memory and PCI bus master designs. However, it is valuable to understanding the benefits of these designs to real-time and high-speed systems to first review simple frame grabbers and ping-pong memory options as the basics of image transfer.

Simple Frame Grabber

The principal issue with high-speed or real-time performance of frame grabbers is the speed at which they make image data available to the processor. To understand this, look at the most basic frame grabber for a CCTV camera shown in Figure 17 below.

The frame grabber receives the composite video data, and separates the video and synchronization information. The video data is digitized and stored in the frame grabber's memory. When the frame is complete, the memory is switched onto the computer's bus, and image data is now available to the computer. Usually, the frame grabber also asserts an interrupt to the processor to notify it that data is available. In addition to the latencies identified above that have to do with image acquisition, the transfer of the image data from the frame grabber to the computer presents additional time challenges.

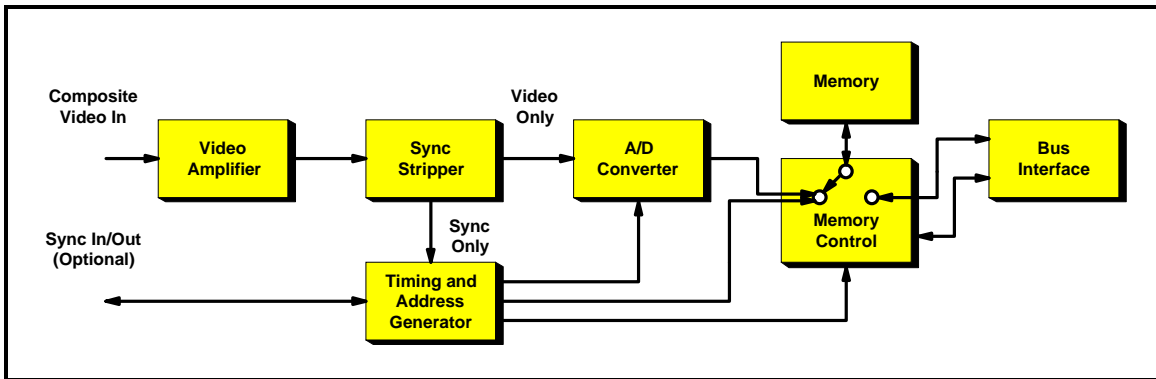


Figure 17 – Basic Frame Grabber

A critical latency is the time required to allow the CPU access to the image; only one image can be acquired or processed at a time. This means that, for a CCTV camera, one or more images will be missed depending on the length of time it takes the processor to access and perhaps process an image. The very fastest this frame grabber can support acquiring and processing images is one every 67 msec. That is, the maximum rate a system with this frame grabber can run is 15 images per second if the parts arrive at exactly that rate. With asynchronous part arrival, considering the variability in acquisition latency, the maximum rate for a system using this frame grabber is 7.5 images per second.

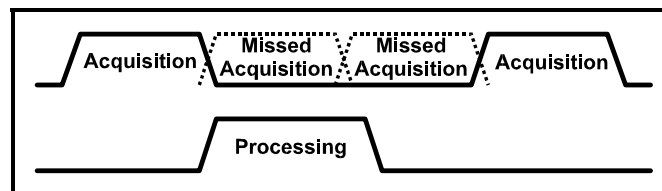


Figure 18 – Simple Frame Grabber System Timing

Ping-Pong Memory

A common modification to improve the speed of a vision system is to employ ping-pong memories on the frame grabber. In this configuration, the frame grabber has two memories as shown in Figure 19. One of the memories is available to receive incoming image data, and the other is connected to the computer's bus to make image data available. When the computer signals it is finished with the memory connected to the bus and the frame grabber finishes loading new image data into the other memory, the two memories are swapped. In this way, acquisition and image processing can always overlap (see Figure 20). A frame

grabber with ping-pong memories can theoretically support acquisition and processing of 30 images per second; a practical maximum rate is 15 images per second.

Note that the use of a single memory or a ping-pong memory affects the system's speed – the number of images per second it can process. It does not affect the system's real-time performance – whether or not its output is timely with respect to the input. The use of a ping-pong memory does not increase the transfer speed of image data into the processor; it does allow the activities of image acquisition and processing to occur in parallel.

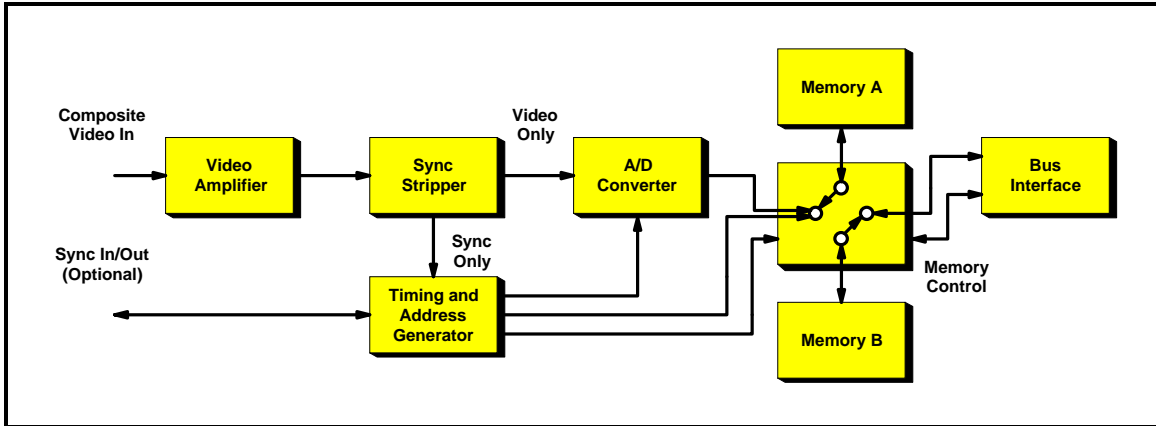


Figure 19 – Frame Grabber with Ping-Pong Memory

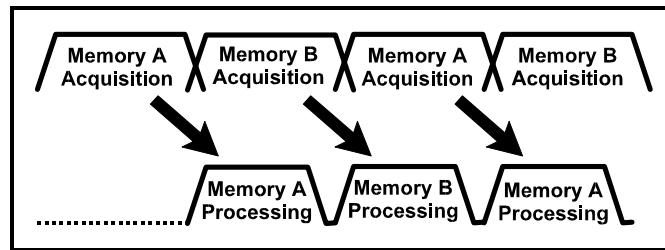


Figure 20 – Ping-Pong Frame Grabber System Timing

Improved Data Transfer

The transfer speed of image data from the frame grabber to the processor over the bus is one area where frame grabbers have improved greatly over the years. Most current frame grabbers reside on the PCI or compactPCI bus. The theoretical maximum transfer rate is 132 Mbytes/second², and devices on the bus can achieve this rate for short bursts of data; the best practical sustained average transfer rate is around 100 Mbytes per second when transferring four bytes in parallel. The actual transfer rate depends on the chip set accompanying the CPU that manages the PCI bus and other data transfers that may be taking place on the bus such as disk accesses. Needless to say, real-time, high-speed vision system designers check into the chip set used with their CPU, closely manage how the computer is configured, and minimize other traffic on the bus.

While the PCI bus is definitely fast by comparison to other, older computer buses, it does not compare with the memory access speed a processor realizes when accessing data out of its local RAM. Individual byte

² This is for the most common implementation of the PCI bus: 32 bits wide with a 33 MHz clock. The PCI bus specification allows data widths of 32 or 64 bits and bus clock speeds of 33 or 66 MHz. The higher bus capacity, while desirable is largely unsupported.

(pixel) accesses by the processor from the frame grabber over the PCI bus will be far slower than the potential bus speed, and very much slower than similar accesses to the CPU's local RAM. To minimize memory access latency, high-speed systems usually transfer the image data out of the frame grabber into main memory. While the CPU could handle this transfer, it is much faster, simpler, and more efficient to have the frame grabber operate as a PCI bus master and manage the transfer itself. That way, the processor is free to use its resources for other tasks while the image data is being transferred.

Frame grabbers that are bus masters do not usually need ping-pong memories. Many of them eliminate the traditional frame grabber memory, and use a dual port memory or a FIFO to buffer image data onto the bus. The frame grabber's memory or FIFO must be large enough to hold image data that may be delayed while waiting for bus access when other transfers are taking place.

The PCI bus transfer proceeds in blocks during the period the image is being acquired from the camera. While it is possible for the CPU to begin processing the image data as it becomes available and before the end-of-frame, this approach carries a reasonable risk that the processor will overrun the image data being transferred from the frame grabber. Typically, the frame grabber issues an interrupt after a frame of data has been transferred to inform the processor that it can start image processing. The transfer latency is the delay between the end of transfer of the data from the camera and the end of servicing the end-of-frame interrupt from the frame grabber.

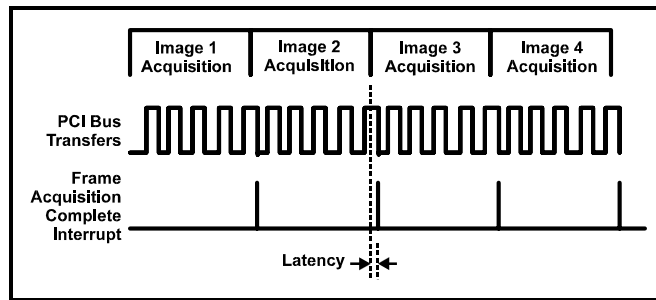


Figure 21 – PCI Bus Transfer Latency

An important feature to look for on PCI bus master devices is that they handle scatter-gather. When an application requests a block of memory, say to hold image data, the operating system allocates the required amount of memory, but does not guarantee that the memory is physically contiguous. In a Pentium based PC, the memory is a collection of 4 Kbytes pages. Between the processor and the memory, is a table that translates the logical address issued by the processor into a physical address in memory. A device on the PCI bus addresses physical, not logical memory. If the bus master does not have scatter-gather capability, either the software must insure that the memory receiving the image data is contiguous, a difficult task, or the software driver must control each block of data transferred, providing the logical to physical address translation. However, with scatter-gather capability, the software driver for the board loads a table on the board with the translation from logical to physical address space. The board can then manage the complete image transfer without software intervention.

A final very high-speed memory option for frame grabbers is emerging. PCs are becoming available with an AGP (accelerated graphics processing) slot. A board in this slot has direct access to system RAM and very high bandwidth, and no on-board memory or even a FIFO is needed. The latency in image acquisition added by an AGP board is just the latency to service the end-of-frame interrupt from the frame grabber. Since AGP is intended for servicing a display interface board, frame grabbers using the AGP slot also incorporate the display driver circuitry.

Region of Interest

Some frame grabbers have a programmable region of interest. With this feature, only image data within the region is transferred over the bus to the CPU's RAM. This can be a significant savings especially in applications where there is significant data flow over the bus.

For example, if the full image resolution is 640 x 480 pixels, the time to transfer the entire image over the bus would be about 4 msec. best case. With an ROI of 320 x 240 pixels, the bus transfer time is reduced to one-quarter of that value.

Input/Output Control

In addition to its basic function of facilitating image transfer, a high-speed and a real-time system can place additional responsibilities on the frame grabber to accept other input signals and coordinate other functions through outputs. To minimize latencies and their variation, there are functions that must often be coordinated by hardware circuits rather than by software. While it is possible to build special circuits to monitor the camera timing and manage these functions, this would duplicate functionality already on the frame grabber. Some frame grabbers include resident circuits to manage most critical I/O functions.

These functions commonly needed by real-time and high-speed vision systems were identified and discussed earlier under triggering and cameras, and will be reviewed here. Finally, a new control signal, the pixel clock will be discussed and shown how it can improve performance.

Most vision applications use an external trigger to start the process of image acquisition and analysis. When time is critical, the software latency and its variation are unacceptable. Some frame grabbers have the ability to accept the trigger and initiate the process with negligible latency.

When a vision system uses an asynchronous camera or an asynchronously resettable camera, a signal is required to be sent to the camera to begin image acquisition. The most practical source for this signal is the frame grabber, and since some cameras do not provide a signal when a commanded image is being acquired, the frame grabber must have this control to work with the camera.

When a vision system uses a strobe lamp or pulsed LED light source for illumination. It is critical that the light pulse occur at a specific time in the camera's exposure time. Otherwise, the image will be seriously degraded. Only the frame grabber that is synchronized to the camera can reliably provide this signal at the correct time.

There are some cameras that require control of exposure time. In some cases, this is just the static assertion of a particular signal line. In other cases, one or two pulses with the proper timing are required. Especially in the latter case, the frame grabber is the logical, and perhaps only, choice to provide this control.

Pixel Clock

As a final subject on frame grabbers, the pixel clock can be a factor in high-speed systems. There are two options for a pixel clock: a common clock shared by the camera and frame grabber or a clock replicated by the frame grabber from synchronization signals. For a common pixel clock either the camera or the frame grabber generates a pixel clock and sends it to the other device. In a replicated clock, the frame grabber uses the camera's synchronization signals to control circuitry that reproduces the pixel clock.

For a common clock, the favored technique is to have the camera send the clock to the frame grabber. The advantage is that, if the same cable carries the image and clock, there is no phase error between the clock and image data introduced by cable delay. In some instances, the frame grabber will generate the clock, send it to the camera, and the camera will return the clock to the frame grabber. The reason for preferring the clock be sent from the camera to the frame grabber is to compensate for cable delays. Propagation

delay for a signal is around 1.5 nsec/foot of cable length. For a round trip, the delay is 3 nsec/foot, and this does not include any circuit delays contributed by line drivers and receivers or camera circuits. For a clock period of 70 nsec, a 46 foot camera cable introduces a one-way error of one clock period; a 20 foot cable introduces almost one-half clock period of error.

With a replicated clock, the concern is the stability of the replicated clock in the frame grabber with respect to the original pixel clock in the camera. This is called clock jitter and is typically in the range of ± 3 to ± 10 nsec. Clock jitter is a source of measurement error. There is no clock jitter with a shared clock.

Image Processing

The aspects of image processing in a high-speed, real-time environment will be discussed as impacts of the processor architecture, operating system, application software, and output timing. This last topic is especially important in real-time systems, because it is where vision systems typically compensate for the uncertainties in latencies.

Processing Architecture

The majority of vision systems, including real-time vision systems, use a single processor; usually a PC, but many high-speed applications are addressed with multiple processors; operating in either a SIMD (Single Instruction Multiple Data) or a MIMD (Multiple Instruction Multiple Data) mode. While an understanding of multi-processing or parallel processing is valuable to the designer of high-speed vision systems, the scope of the topic is too broad for this paper. This paper will touch on some of the issues in processor architecture and common solutions.

In choosing a processor for a real-time, high-speed application, there are a number of important considerations. Three will be briefly discussed: processor speed, instruction set, and software development. There are also decisions about using a single processor or a multi-processor architecture.

Processor Speed

Obviously, processor speed is very important. While the CPU's clock speed is a major determinant of speed, the architecture of the circuits surrounding the CPU can have just as significant a bearing on the actual processing speed as does the clock speed. The chips surrounding the CPU often determine the speed of access to memory and other peripherals, and, if not efficiently implemented, can cause unnecessary wait states to be inserted. The amount and speed of memory will have a bearing on the processor's speed of execution as will the amount and speed of cache memory if used by the processor.

Benchmark programs published on the Intel web site (www.intel.com) for current processors show that processor speed increases at about 65% of the increase in clock speed. Depending on processor and chip set, processor speed increased between 37% and 88% of clock speed increase. In one case, with a Pentium III processor and an 815E chip set, the processor speed with a 750 MHz clock was slower than with a 733 MHz clock.

The speed of processors is often judged by the time it takes to complete one or a series of benchmark algorithms. For this to be meaningful, the system developer needs to use benchmarks that have similar processing characteristics to those the application requires. For example, most DSPs optimize their speed for multiply-accumulate operations common to signal processing, but sacrifice all but rudimentary input/output capability. An application environment that uses extensive correlation or convolution algorithms, and only basic I/O will be well served by a DSP. Other applications that use, say, binary image processing and require more sophisticated I/O will be served better by a more general purpose processor than by a DSP.

To realize the processor's speed potential, one requirement of a high-speed or a real-time vision system using a computer is that concurrent tasks performed by the computer not add to the latency of the vision task. In most implementations, this is accomplished simply by having the processor not perform any other task than the primary vision task. In more sophisticated implementations using real-time multi-tasking operating systems, there can be concurrent tasks running at a priority lower than the vision task. In multi-tasking environments with priorities set appropriately, the activity of task switching uses up some available processor time.

Instruction Set

In real-time applications, a processor with a more deterministic instruction set may be more desirable. A deterministic instruction is one that executes in the same time regardless of the data. For example, logical instructions (AND, OR, etc.) usually execute in a fixed time regardless of the data values. However, a divide instruction often takes a variable amount of time depending on the values of the operands. The most frequently used instructions in a real-time system need to be deterministic to insure a minimum variation in latency.

Achieving a deterministic instruction set is only practical in embedded processors designed specifically to provide this characteristic. Current general purpose computers make extensive use of pipeline scheduling, data and instruction caches that make prediction of execution time for any instruction extremely difficult. In addition, operating systems such as Windows 2000 will insert delays in the execution of the program to handle task scheduling, interrupt processing, and virtual memory management.

An additional consideration in instruction sets is any provision for parallel calculations. The prime example of this is the MMX instruction set in the Intel microprocessors. These instructions, operating on 64 bits of data representing up to 8 pixels of 8 bits each, are an implementation of SIMD parallel processing. People using these instructions to code image processing routines report they accelerate image processing by up to a factor of 4 over code using the traditional single operand instructions.

Software Development

As mentioned above, the ability to develop and support the application must be part of the selection decision. Considerations include the development tools available to speed development and insure software quality, and the availability of skilled programmers who can develop and support code written for a specific processor. Some of the higher speed and more esoteric processors may be attractive until it is realized the software support tools are minimal and only a few people have the skills necessary to program them to achieve operation near their potential speed.

Single Processor

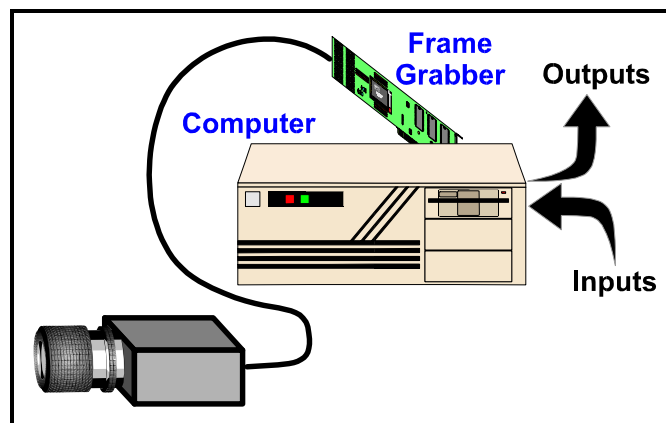


Figure 22 – Single Processor

Undoubtedly, the most prevalent processor used for machine vision and scientific imaging is the PC. This is due to its very high performance to cost factor, availability of peripherals, availability of programming

and engineering talent that is familiar with the hardware and software environment, and its being available in low-cost office or industrial packaging. However, even with its power, the PC does not have the processing power to handle many high-speed applications. Other processors such as the Power PC from Motorola and IBM have been used for image processing, usually in as embedded processors in a dedicated application.

Multi-Processor

A multi-processor design may be used either to gain more raw processing power or to use different processors, each for their individual strengths.

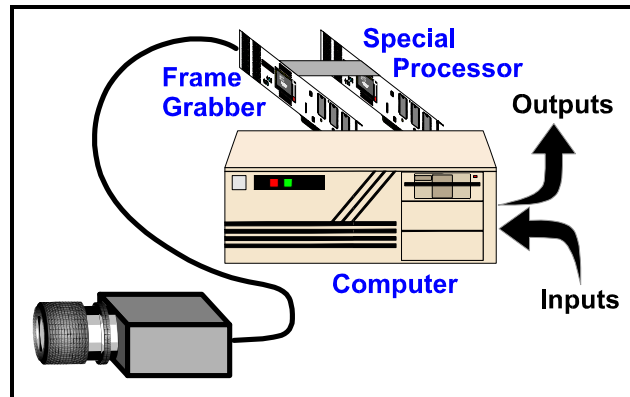


Figure 23 – Common Multi-Processor Approach

Perhaps the most common approach to multi-processor image processing is to use an embedded processor; a DSP, Intel Pentium, or Power PC being the most common, within a PC. The embedded processor, which does not have the system overhead of the main CPU, performs the data intensive image processing like image transforms, correlation, and convolution. The CPU may be used to make a final decision and to control the outputs. Sometimes the embedded processor also makes the decision and controls the real-time outputs leaving the CPU to handle supervisory tasks. The embedded processor may be on a separate board inside the PC and rely on image data being transferred over the computer's bus, it may be on a separate board with a private data interconnect to the frame grabber, or it may be integrated onto the frame grabber itself. The embedded processor makes more processing power available, may have an architecture that is faster at certain image processing functions, or may just be more efficient by virtue of not having to support a complex operating system.

Another common approach to gaining processing power for image processing is to use ASICs. These integrated circuits process image data in hardware, and are very fast. They can be designed to provide any number of image processing functions, but they are usually not programmable by the user; the available functions are determined by the design of the ASIC or its programming at the factory. The most common functions available in ASIC processing are look-up tables (LUT) and certain convolutions. Commonly, the ASIC chips are part of a frame grabber or a mezzanine board that plugs on to a frame grabber.

Still another common approach to multi-processor image processing is to use a PC with multiple CPUs; typically either two or four. This multiplies the available processing power almost by the number of CPUs available. The operating system still requires resources from one processor, but a portion of that processor and the resources of the others are available for image processing. The successful use of such a multi-processor environment requires an appropriate operating system that supports multi-processors (e.g., Windows 2000), application software written to run in a multi-processor environment, and careful attention to competition for common resources (e.g., data transfers over the bus).

Other processing architectures used for high-speed applications have included dedicated array processors, pipeline processors, and even arrays of PCs.

Operating System

Every processor needs an operating system or kernel³. The operating system or kernel manages system resources. Most computers use a powerful, general-purpose operating system (e.g., Windows, Linux) which provide a very rich array of capabilities to simplify the development of applications, but are not particularly high-speed, and are definitely not real-time.

Developers have managed to configure some of these general-purpose operating systems to work adequately for high-speed and real-time applications. Embedded Windows NT is common in this regard. By installing only the absolute minimum part of the operating system and carefully avoiding any task, like disk accesses, which can take up significant portions of the processor's resources, the operating system can perform adequately for some high-speed or real-time applications. However, the application usually must be developed and debugged under a fully functional non-real-time operating environment, and then transferred to the special operating environment for testing.

Some applications use an operating system or kernel developed specifically to support real-time applications (e.g., VRTX, RTXC, QNX, VxWorks.). Real-time versions or extensions are becoming available for the general-purpose operating systems. The latest version of Windows CE has real time capabilities. Extensions are available for Windows NT and Windows 2000 that give them many, but not necessarily all, the features of a real-time operating system. Real-time versions of Linux are available from several suppliers, but Linux drivers for devices such as frame grabbers are still rare.

What distinguishes a real-time operating system or kernel is its facilities to support timeliness in applications running under them. These features include preemptive interrupt processing with guaranteed latencies, and the ability to prioritize software tasks or threads so that time critical tasks get executed before more mundane tasks.

Device drivers for special devices like the frame grabber come either from the frame grabber supplier or from a supplier of an application program or library. It is necessary to evaluate these device drivers for efficiency. For example, device drivers for one frame grabber opened five threads. While most of these threads are rarely used during system operation, each resident thread requires a certain amount of overhead from the operating system to give them an opportunity to operate even if there is nothing for them to do.

Application Software

It is a given that the efficiency of the application code, especially data intensive image processing, will be a prime determinant of system speed. Developers have adopted different courses to achieving application software that executes at high speed. One common approach is to shift the burden off software development and onto hardware, by using a higher speed processor, special ASIC chips, or multi-processing architecture as discussed above. A second common approach is optimization of algorithms.

An example of where programmable hardware can enhance real-time, high-speed performance and reduce the effort in developing application software was touched upon in image acquisition. If the frame grabber has circuits controlled by software that can manage and respond to real-time events, the application software can manage these functions in non-real-time and still achieve real-time performance.

³ A kernel is a subset of an operating system. It generally remains resident in memory, and manages critical resources like memory. It does not provide the rich set of utilities (API) normally associated with an operating system.

Application software may be written from scratch specifically for the application, or it may be developed using a more general-purpose software package as its foundation. In writing a specific software package from scratch, it is possible to optimize the code more than is possible when using routines out of a general-purpose package. However, this requires programmers who are familiar with image processing techniques, the specific hardware features of the processing being used, and the techniques for writing real-time programs.

Because most common operating systems are not real-time nor particularly high-speed, many commercially available software packages are not oriented toward supporting real-time operation and do not have all the high-speed optimization possible. When selecting a commercial image processing package, it is necessary to investigate them carefully. Using a software package can reduce development time significantly, and should be considered as more desirable than writing the program from scratch unless the drawbacks of the package are too great.

Here are some things the application software developer can do to speed up operations:

- Write specific code for specific image processing functions. For example, most convolutions are symmetrical. A symmetrical convolution over an area can be decomposed into a row convolution and a column convolution. Assuming the convolving kernel coefficients are stored in cache, a 3x3 convolution requires 9 memory reads, 9 multiplies, 8 adds, and one memory write. A 3x1 plus a 1x3 convolution require a total of 6 memory reads, 6 multiplies, 4 adds, and 2 memory writes. For larger convolving kernels, like a 7x7, the savings are even more dramatic. For specific convolving kernels, a programmer might find an even more cleverly efficient coding method.
- Design the application around image processing functions that allow efficient coding.
- Write code that uses the processor's special features like MMX instructions in the Pentium processors.
- Structure the code so that data is retained in cache to the extent possible. This minimizes wait states for memory accesses.
- Avoid complex loops with embedded decisions. Not doing this increases the probability the processor will need to go outside cache to fetch instructions.
- Write the application using threads, but only the number necessary. That allows the developer to assign priority to different parts of the application program.
- Use compiler optimizations for all time-critical routines. Even more efficient is assembly language programming. However, as processors migrate to more complex instruction sets, the practicality of assembly language coding is reduced, and assemblers are not even made available. The risk of assembly language coding is loss of portability to future generations of processors.
- Avoid creating or destroying objects (allocating and releasing memory) as part of the image processing routine. These activities cause a variable amount of operating system overhead.

In special cases, developers have coded special purpose processors (e.g., DSP) so that the execution time exactly matched the rate at which image data was being acquired and the latency is essentially zero with no uncertainty. That is, the loop for processing each pixel took exactly the same time in clock cycles as the period between each pixel being received by the vision system regardless of the image content. This requires that the pixel clock and the processor clock be derived from the same source. It also requires very careful attention to algorithm development and coding. Such code may be very fast, but it is very difficult to maintain or update.

Resynchronization

While the techniques above can be used to advantage in high-speed systems, they do not offer any particular advantage for real-time performance. A common characteristic of image processing is that, except for low-level image processing, processing time is dependent on the image content. While it is conceivable for the developer to write code that executes in constant time, such an effort is extremely difficult and makes development and support virtually impractical.

A more practical technique is to resynchronize the results. To achieve resynchronization, the incoming image is tagged with a time stamp. The time stamp need not be an actual time, but must be created from a regular periodic event that is relevant to the application. One such time base for a continuously operating system is the occurrence of a new image frame from a continuously operating camera whether or not the frame is acquired and processed. Another time base might be a signal from an encoder tracking a conveyor belt. The events that create the time base are input to the processor where the processor maintains a running count; the count is the time stamp used by the application. One of the challenges in maintaining a count is to anticipate and deal with the overflow of the counter.

The time stamp follows the image data as it is processed, and is also attached to the results. When the result of processing an image is available, it is placed into an output queue according to its time stamp. The output queue resembles a shift register in which any element of the shift register can be loaded (with output results) at any time. However, rather than use processor resources to actually shift the contents, it is much more common to use an address pointer to the next output data. The vision computes the location in the output queue from the time stamp on the data, the current time count and the known distance (time) to the output. It inserts the output data into the queue the calculated number of time intervals ahead of the current queue output pointer. Placing the output into a queue based on time resynchronizes the result.

The requirements are:

- The vision system must always finish processing an image at least one time interval before the result is needed.
- The vision system must know the time required between the trigger event and the time the output must be activated in terms of the intervals of the time stamp.
- The output queue must be at least as long as the number of time (distance) intervals between the imaging and output positions.

Latency Calculation

The table below provides an aid for calculating latencies and their effects. Some applications may not need all the entries, while other applications will need some additional entries. For vision systems, latency times are typically expressed in milliseconds.

	Latency			
	Minimum	Maximum		
Part detector				
Capture command				
Strobe/shutter trigger				
Subtotal			Position Variation ⇒	
Exposure time			Blur ⇔	
Video transfer				
Transfer to CPU				
Image processing				
Subtotal				
Resynchronization		0		
Time Base	0			
Output activation				
Total				

Table 2 – Latency Compilation Template

Part Detector

The latency for the part detector would be the time between when the part physically arrives in position (e.g., breaks a beam in a photodetector) and when the trigger signal is received by the vision system.

Capture Command

The capture command latency is the time between when the trigger signal is received by the vision system and the actual initiation of image acquisition starts.

Strobe/Shutter Trigger

This latency is the time between when the image acquisition starts and the pulse from the strobe light begins or the electronic shutter opens. Of course, systems may not use these functions, in which case the latency is zero.

Exposure Time

This is the time during which light is creating an exposure. It is controlled by the camera's exposure time and the duration of the light. If a steady light is used, it is just the camera's exposure time. If a pulsed light

source is used, like a strobe light, then it is the duration of the strobe light pulse. In rare occasions when both an electronic shutter and a pulsed light source are used, it is that time during which both the shutter is open and the light pulse is present.

When an electronic shutter of fixed duration is used, the maximum and minimum latencies are both equal to the duration of the exposure time. If an electronic shutter is used in a programmable mode to control exposure through controlling time, then the latencies are the minimum and maximum times of the shutter. A pulsed light source usually has a non-linear output, with a rise time and a fall time. Typically, the pulse width is measured at the 50% points of the light output. There is always some variation in the width of the light pulse.

Video Transfer

Video transfer time is the duration of time required to transmit the image data from the camera to the frame grabber. Usually, if the image size is fixed, the video transfer time is also fixed, and the maximum and minimum values for the latency will be the same.

Transfer to CPU

This is the time between the end of video transfer from the camera to the frame grabber to the end of the transfer of image data from the frame grabber to the CPU. Some frame grabbers buffer the entire image, and start transferring image data to the CPU only after the entire image has been received from the camera. Other frame grabbers have a limited size buffer for image data, and start the transfer to the CPU shortly after the video transfer begins. Other frame grabbers have the option to work in either mode.

The latency time will depend on the way the frame grabber transfers data to the CPU and on other bus traffic competing with the frame grabber for bandwidth.

Image Processing

The image processing time is the time between when image transfer to the CPU ends and the result of image processing is available. Although it is customary for the CPU to wait until the frame grabber finishes transferring image data before starting the image processing, it is at least conceivable that it could start after only a portion of the image data is received. The latency time will vary depending on the complexity of the individual image and on competition for processor resources (e.g., cache memory, CPU cycles) by the operating system and other processes that may be running. Generally, this time is either estimated from experience, or measured using actual or simulated conditions.

Resynchronization

Resynchronization is the correction for latency variations that is accomplished by having a real-time time base. It relies on the image being time stamped with the real-time time base, and computing a minimum additional delay based on the value of the real-time time base when image processing is finished. Usually the image is time stamped either when the capture command is issued or when the image transfer to the CPU is complete. Frame grabbers often provide interrupts to the CPU for both of these events, and one or the other of those interrupts can be used to create the time stamp.

Resynchronization is unique in that the maximum value is normally zero (no additional time is required), and the minimum value is a positive number (a delay is introduced to attempt to make the total minimum latency closer to the maximum latency).

To compute the resynchronization time, total the latencies for the steps from the time the image is time stamped through image processing. Get a total for the maximum values and for the minimum values. The difference in these values is the correction resynchronization will attempt to provide. However, since resynchronization is based on discrete time periods of the time base, the difference must be divided by the time base period, rounded down, and then multiplied by the time base period. Enter this value for resynchronization in the minimum column. If resynchronization is not used, just leave the minimum value zero.

Time Base

Because of the granularity of the time base used for resynchronization, the system may introduce an additional latency of from zero to one time base interval into the process. Enter the time period for one time base interval in the maximum column for the time base.

Output Activation

The output activation latency is the time between the end of image processing or resynchronization and the final event in the process (e.g., operation of the reject mechanism). It includes processing overhead, any delays in propagating the signal, and mechanical delays of any mechanism operated.

Example 1

The Requirements

A system is to inspect small circuit modules after assembly for correct component placement. The inspection system must handle a variety of different modules. The assembly system sequentially indexes the modules into position for each step in the assembly operation to be completed. Because the vision system is being retrofit to the assembly system, it will have to view the modules between stationary positions; that is, it will have to view the modules while they are moving. The inspection system is not to reduce the throughput of the assembly system. The results of the inspection are needed when the module reaches its next position, or the assembly system will be slowed. Control of the assembly system is through a PLC. In addition to rejecting non-conforming modules, the inspection system is expected to create reports on failures and quality levels and to make these reports available over a network.

- Module size 20mm x 30mm
- Accuracy of position measurement 0.03mm
- Production rate (time/module) 0.8 seconds
- Transfer time/module 0.5 seconds
- Distance between assembly stations 250mm

Design

A photosensor will be used to detect when a module is in position. The photocell has a latency of 600 ± 150 μ sec. The photosensor's signal will be received by the frame grabber, which will initiate image acquisition.

Assuming sub-pixel resolution can be used and that an accuracy of 1 pixel can be achieved, the needed image resolution is:

$$20\text{mm} / .03\text{mm} = 667 \text{ pixels by } 30\text{mm} / .03\text{mm} = 1000 \text{ pixels}$$

A high-resolution camera having an image resolution of 1024 x 1024 pixels should work. At this resolution, the camera will be a progressive scan camera. Sub-pixel analysis is necessary to meet the accuracy requirements. To insure image quality free from timing artifacts that would degrade the sub-pixel algorithms, the camera will supply its pixel clock to the frame grabber.

The field-of-view will be 30.72mm x 30.72mm.

The module must travel 250mm in 0.5 seconds. In most practical systems, the profile of velocity versus time will be a trapezoid or similar shape to allow for acceleration. For simplicity in this example, a uniform velocity is assumed. The velocity is 500mm/second.

To hold image blur to one pixel, the exposure time must be:

$$T_E = \text{FOV} / (V_p * N_p) = 30.72 / (500 * 1024) = .00006 \text{ seconds (60 } \mu\text{sec)}$$

This exposure time can be achieved with either an electronic shutter or strobed illumination. For many cameras, the electronic shutter can only occur at a particular time in the video timing. The strobe, on the other hand, can be triggered almost any time during the video timing except for a brief period, typically less

than one horizontal line time at the beginning of a frame. The video transfer takes place in 33 msec.; each scan line can take no more than 32 μ sec. Therefore, the maximum uncertainty in strobe triggering is 32 μ sec.

The camera is free running. The video transfer to the frame grabber can begin any time within a 33 msec. time after exposure (strobe lamp firing), and takes an additional 33 msec. to complete.

The inspection system views the module during motion. Assuming the inspection system is positioned half way between two assembly positions, and that it must have its result ready by the time the module reaches the next assembly position, the time available to the vision system is one-half the transfer time of a module, or 0.25 seconds. The vision system will have 0.8 seconds - 0.25 seconds = 0.55 seconds of idle time between inspecting parts. This time can be used to compile and save reports.

The image processing time depends on the complexity of the module and the competition for computer resources. It is estimated to take between 10 and 80 msec. The output is a pass/fail signal sent to the PLC, and is dependent only on the availability of the CPU's resources. Typical output time could be a few microseconds to perhaps 200 μ sec.

Latency Evaluation

	Latency			
	Minimum	Maximum		
Part detector	.45	.75		
Capture command	nil	nil		
Strobe/shutter trigger ⁴	.02	.072		
Subtotal	.47	.822	Position Variation \Rightarrow	.176mm
Exposure time	.02	.02	Blur \Leftrightarrow	.91 pixels
Video transfer	33	66		
Transfer to CPU	13	26		
Image processing	10	80		
Subtotal	56.49	172.84		
Resynchronization	not used	not used		
Time Base	not used	not used		
Output activation	.005	0.2		
Total	56.5	173.04		

Table 3 – Latencies for Example 1

⁴ This is the total of the latencies for waiting for the camera to become available for the strobe event (0 to 32 μ sec) and the strobe response to its trigger (20 to 40 μ sec).

Example 2

The Requirements

A system is to image new bottles on a conveyor line to insure the bottle's neck is centered on the bottle and not leaning. Some of the important specifications are:

- Bottle diameter 3 to 6 inches in diameter
- Throughput 240 bottles/minute (0.25 seconds/bottle)
- Conveyor speed nominally 160 feet/minute (32 inches/second)
- Variation in conveyor speed ± 16 feet per minute
- Ejector location 24 inches past the point of imaging

Initial Design

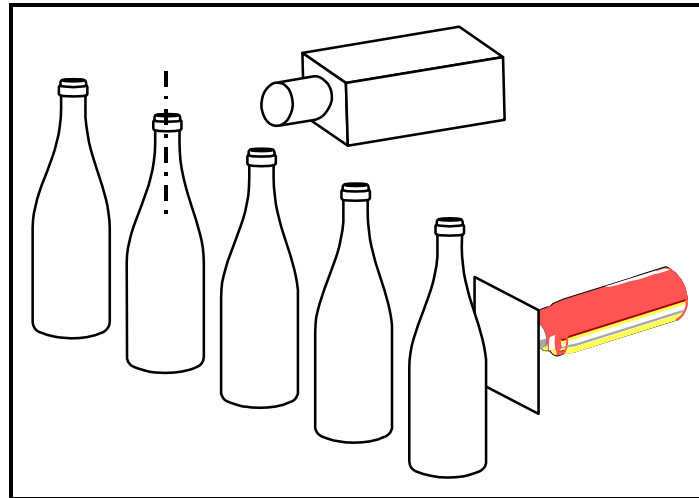


Figure 24 – Inspecting Bottles

We start the design by assuming the simplest system possible. One CCTV camera images the bottle. The CCTV camera is continuously operating, and acquires one image every $1/30^{\text{th}}$ of a second. The assumed system will use a PC running Windows NT with an ordinary, simple frame grabber. Image resolution will be 640×480 pixels. The design anticipates having sufficient processing power so that the processor can acquire and analyze each image coming in. It will detect the presence of a bottle in the image, and find its neck position.

We examine the latencies of the assumed simple system (see Table 4 below). There is no part detector, the camera is free running, and a software trigger causes the frame grabber to acquire a new image and provide it to the CPU. Since images are constantly being sent from the camera to the frame grabber, the time between the software trigger and the start of image data transfer to the CPU ranges from 0 to 33 msec. This represents the latency of the capture command. The exposure time and the video transfer time are both fixed at exactly 33 msec. The transfer of image data from the frame grabber under CPU control over the bus could take place in as little as 9 msec., but could be preempted by the operating system and other bus activities; we can estimate that it might occasionally take as long as 100 msec. Again, it is not unreasonable in a fast computer with good code to perform the image processing for the measurements

needed in as little as 10 msec. But again, the operating system may preempt the program, and the time could stretch to 100 msec. on occasions.

These latencies total to a range of 85 to 299 msec. With bottles coming every 250 msec., it is apparent the vision system might possibly be too slow on occasion, and some adjustment to the design will be needed. However, it is good practice to work through the rest of the assumed design to identify any other possible problems. The total uncertainty of these latencies along with the nominal conveyor speed gives us the variation of the position of the bottle in the image:

$$P_V = L_U * V_P = 0.214 \text{ sec} * 32 \text{ in/sec} = 6.85 \text{ inches}$$

For a 6 inch diameter bottle, the horizontal field-of-view could probably as narrow as 7 inches if there were no uncertainty in the bottle's location. With the variation factored in, the field-of-view will need to be 13.85 inches horizontally. The uncertainty in latency almost doubles the required size of the field-of-view, which halves the measurement precision.

	Latency			
	Minimum	Maximum		
Capture command	0	33		
Exposure time	33	33	Blur ↔	49 pixels
Video transfer	33	33		
Transfer to CPU	9	100		
Image processing	10	100		
Subtotal	85	299	Position Variation ⇒	6.85 inches
Resynchronization	33	0		
Time Base	0	55		
Output activation	0	10		
Total	118	364		

Table 4 – Initial Compilation of Latencies, Example 2

Another needed calculation is the blur due to the bottle's motion.

$$B = V_P * T_E * N_P / FOV = 32 \text{ in/sec} * .033 \text{ sec} * 640 \text{ pixels} / 13.85 \text{ in.} = 48.8 \text{ pixels}$$

It is intuitively obvious that a reasonable measurement would be extremely difficult with a blur stretching over 49 pixels. While there are some advanced image processing techniques used for image restoration of reconnaissance images that remove much of the effects of motion, they are extremely computationally intensive and impractical for on-line machine vision.

Another factor that must be considered when using interlaced cameras is motion tear caused by the camera's interlaced scanning. This is calculated as:

$$MT = V_P * T_F * N_{Ph} / FOV_h = 32 \text{ in/sec} * .0167 \text{ sec} * 640 \text{ pixels} / 13.85 \text{ in} = 24.7 \text{ pixels}$$

The motion tear is half the size of the image blur, and it also is generally unacceptable in this simple system.

Assuming the vision system could somehow make sufficiently precise measurements, it could also tell the lateral position of the bottle in the image. The vision system could compute the time from inspection to ejector assuming certain nominal latencies to get its answer, and apply the correction based on the bottle's position in the image to the ejector time. However, the variation in position relates to when the image was acquired (including the capture command above), but it does not include any uncertainty introduced in the

subsequent process steps. Correcting for the bottle's position in the image is necessary, but only eliminates the variation in what has been labeled the capture command. Thus, the resynchronization in the compilation only corrects for this one factor that varies between 0 and 33 msec.

If we assume the design will use the PC's internal clock that generates an interrupt 18.2 times/second or about every 55 msec. Also, we can assume that the output (ejector) will be interrupt driven from the PC's clock, and the interrupt service time will vary between 70 μ sec (virtually 0 msec.) and 10 msec. depending on the operating system's state.

In summary, we find this simple system will have a latency of between 118 and 364 msec. The uncertainty in latency of 246 msec. represents about 7.9 inches of conveyor travel, or about 1 bottle on the conveyor. Clearly the simple system is very inadequate. The remainder of this example will look at these and other issues and at ways to improve the real-time performance of the system.

Solving the Uncertainty in Part Position

The original plan is to have the system analyze each image to see if a bottle is present for measurement. However, the analysis of latencies uncovered several problems with this. The first was the uncertainty in latencies for image processing meant the process might take too long to keep up with line rates. The other is that the uncertainty in latency required a much larger field-of-view than desirable.

One way to mitigate some of these affects is to use an independent sensor like a photodetector to detect the presence of a bottle. A typical photodetector that can work in this application responds in 600 ± 150 μ sec.

The application's requirements present a challenge in using a photodetector. The photodetector must be mounted so as not to interfere with the camera's view. The most reasonable position is near the bottom of the bottle. The vision system must handle change over to products of differing diameters. Experience shows that repositioning an element of the system like the photodetector extends the time required for changeover and presents risks of error in setup. Best practice in the design of a vision system is not to move any of the system components. The result of a fixed photodetector position is that the position of the neck in the camera's image will be in different locations depending on the diameter of the bottle. While it is more desirable to always have the bottle's neck centered in the image, it would require positioning the photodetector up stream from the inspection and timing the bottle's movement into the camera's view. At this point in the system's evolution, this additional complexity is not needed.

It might seem that using the photodetector alone, will reduce the latency uncertainty to capture an image from 33 msec. to 300 μ sec. However, since the camera is continuously operating, even when the photosensor is triggered, the vision system will still have to wait between 0 and 33 msec. before reading out the image data. However, by using an asynchronously resettable camera triggered by the photosensor rather than a free-running camera, the camera's latency uncertainty can be reduced to one scan line or 63.49 μ sec maximum which is less than the 300 μ sec of the photodetector. In order to avoid undesirable latency uncertainty, the frame grabber should receive the trigger signal from the photodetector and handle the timing of the asynchronous camera.

Solving the Image Blur and Motion Tear Problems

In the initial design with a CCTV camera, blur was 49 pixels and motion tear was 25 pixels. This is clearly unacceptable. The solution for motion blur is to use either a strobed light source or a camera with an electronic shutter. The solution to motion tear is to eliminate the effects of separate exposure time for the two interlaced fields.

When selecting either strobe illumination or an electronic shutter time, we need to know the maximum practical exposure time for tolerable blur. The calculation for exposure time requires knowing the size of the field-of-view. In the initial design, the field-of-view was 13.85 inches; clearly too large. Until latency uncertainties are reduced, the exact size of the field-of-view cannot be known. However, it must be at least 6 inches to accommodate the largest bottle. Using this lower bound, we calculate the maximum exposure time to give only one pixel of blur:

$$T_E = \text{FOV} / (V_p * N_p) = 6 \text{ inches} / (32 \text{ in/sec} * 640 \text{ pixels}) = 293 \text{ } \mu\text{sec}$$

Many system developers will choose to use the strobed illumination. Either a xenon strobe lamp or a pulsed LED illuminator will easily give the short exposure time required. Other system designers prefer to use cameras with electronic shutters. The electronic shutter on most cameras offering this capability is more than fast enough for this application. We will assume a shutter speed of 1/4,000 second (250 μsec).

One consideration in using a strobed light source or an electronic shutter is the required intensity of the light source. Assuming the intensity of the light source and lens aperture are known to get good image on the CCTV camera and that the progressive scan camera has the same sensitivity as the CCTV camera, then the light intensity reaching the sensor must increase in inverse proportion to the exposure time. Assuming an exposure time of 200 μsec , the light intensity reaching the sensor needs to be:

$$.033 \text{ sec} / .0002 \text{ sec} = 165 \text{ times more intense.}$$

This increase can be achieved by a combination of opening the lens aperture and increasing the intensity of the light source. With today's sensitive cameras, wide aperture lenses, and efficient light sources, this increase is quite attainable.

If an electronic shutter is used to reduce blur, then a progressive scan camera must be used so that there is no image tear. If a strobe light is used, then each field of an interlaced camera will need to be exposed for an entire frame time (see Figure 10). This also eliminates image tear. The strobe light can be triggered any time during the scan except during the charge transfer time in the camera's image sensor. The latency uncertainty for firing the strobe is one horizontal scan line, or 63.49 μsec .

The acceptable choices in cameras and lighting are between an asynchronously resettable, progressive scan camera with electronic shutter and a steady light source or an asynchronously resettable, interlaced camera with strobe light illumination. However, some interlaced asynchronously resettable cameras require a one field delay before starting image data transfer. The system could be designed to accommodate this fixed delay, but choosing a progressive scan camera and relying on the electronic shutter with a steady light source avoids the delay.

Number of Cameras

The anticipated design has a significant flaw: a single camera cannot measure the position of the bottle's neck independently of the bottle's rotation. To do this takes two cameras. The least complex arrangement for image processing is to have the two cameras view at right angles to each other.

Another requirement for measurement precision is that both cameras view the bottle at exactly the same time. That means that the cameras must be synchronized to each other and their images must be acquired simultaneously. This can be done using two frame grabbers, each accommodating one of the cameras, or it can be accomplished with a single frame grabber that has the ability to accept simultaneous camera image data streams. Using a two frame grabbers requires that the frame grabbers be synchronized with each other. For the evolving architecture using asynchronous cameras, the only requirement to achieve sufficient synchronization is that the trigger signal from the photodetector goes to both frame grabbers simultaneously.

There are frame grabbers that can handle two or more camera inputs simultaneously. However, the frame grabber must assert a signal resetting the cameras to start the image acquisition process. Generally, the outputs of a frame grabber are designed to drive only one load. This is especially true if the signal is a differential signal such as RS-422 or LVDS. Normally, the camera's input is terminated. It would be possible to go into one of the cameras and modify it by removing the load. This would allow both cameras to be driven from the one signal. However, now the two cameras are different, and servicing the system is more complex. By using two frame grabbers, one for each camera, each frame grabber drives the reset input of its corresponding camera. Because the cameras are not synchronized, the timing of the two images can differ by as much as one scan line (63.49 μ sec). The maximum discrepancy between the two images can be calculated using the formula for blur and assuming the 6 inch field-of-view. It is:

$$V_P * T_E * N_P / FOV = 32 \text{ in/sec} * .00006349 \text{ sec} * 640 \text{ pixels} / 6 \text{ in.} = 0.2 \text{ pixels}$$

This value is essentially negligible for this application, and can be ignored.

For this system, we will select analog, progressive scan, asynchronously resettable cameras with an electronic shutter, continuous illumination, and two compatible frame grabbers.

The system as configured presently, is shown below in Figure 25.



Figure 25 – Dual Camera Configuration

This arrangement is necessary, but it carries with it a certain risk. In the initial concept, the camera was viewing perpendicular to the direction of motion. Any variation in timing for when the image was acquired simply translated into a change in location for the bottle in the image. However, in the modified design, both cameras are viewing at 45 degrees to the direction of motion. Now any uncertainty in timing of the image results not only in a positional change but also in a magnification change as the object distance is changed. Thus, the latency variation in the start of image exposure adds new processing burdens on the image processing to compensate for magnification change (this can be done by knowing the position of the bottle in the two images). It also can demand a larger depth-of-field from the optics to keep the bottle in focus.

The uncertainty in acquisition latency due to the photosensor and the operation of the asynchronously resettable cameras is 0.306 msec. This translates into about .001 inch of location uncertainty. With the cameras viewing at 45 degrees, the variation in object distance is only .0007 inches; negligible in this application.

Frame Grabber

So far, the design has evolved to require two frame grabbers that accept an external trigger from a photosensor, can operate an asynchronously resettable camera, and accept analog video data.

Our system only has to process one bottle every 250 msec.; a rather modest speed by current PC standards. It might work with even a very simple frame grabber with only one image memory and requiring the processor to manage image data transfer over the bus. However, most frame grabbers that work with progressive scan, asynchronously resettable cameras, are designed for speed, and do not employ such a slow architecture. Our frame grabber of choice will be a PCI bus master card with scatter-gather capability.

Computer

Next we turn our attention to the computer – both the hardware and the software. For inspecting the bottles at the specified speed, a high-end PC has more than enough speed.

Operating System

This application can almost certainly work well with embedded Windows NT without any real-time extensions. Its installation would have to be very careful to include only the minimum components needed, to structure the hardware so interrupts are assigned to give the frame grabber the highest priority possible, and to insure that no other application or operating system service was running. In addition, the application program must be written using threads, and it must assign priorities to the threads to insure the most time-critical threads execute when necessary.

Application Software

For this application we should choose a commercial software package as the foundation of the application software. Its selection should come after evaluating several candidate packages for functions needed by this application as well as for their speed and real-time compatibility.

Ejector Operation

Our final consideration in this example system is operating the ejector. The ejector is located downstream of the inspection location, a common configuration for parts being inspected on a continuously moving conveyor. In the initial design, it was planned to use the computer's clock to time the ejector's operation. However, the interval between clock interrupts of 55 msec., means there is an uncertainty in the bottle's position of 1.76 inches due to just the clock interval alone.

Also, the speed of the conveyor can vary by ± 16 feet per minute or, related to the conveyor speed of 240 feet per minute, $\pm 6.7\%$. For a distance of 24 inches, the speed variation will add an uncertainty in the bottle's location of ± 1.6 inches. The total uncertainty in the bottle's position at the ejector is $1.76 + 2 * 1.6 = 4.96$ inches.

A second scheme is to create an "inventory" of inspected bottles in a software simulation of a FIFO memory, and use a second photodetector at the eject position to call out the results of an inspection as each bottle arrives. The problem with this scheme is that any conveying problem, such as a jam, that might cause the ejector's photodetector to miscount bottles, either missing one or double counting one, will result

in an error that will continue until the vision system is reset. For this reason, most vision systems do not use this scheme.

A better scheme is to use an encoder attached to the conveyor as a time base and resynchronize the output as discussed earlier. The encoder can give a finer time (distance) resolution and inherently compensates for conveyor speed variation. A reasonable implementation would have the encoder producing a pulse every 1/8th of an inch of conveyor movement or about every 3.9 msec. At this short a time interval, the signal needs to be serviced by an interrupt. Some frame grabbers facilitate external signal lines in, and may allow those signal lines to initiate an interrupt. If this is not the case, a separate digital I/O board is needed that can use the encoder signal to generate an interrupt.

The encoder provides one more advantage. At the beginning, it was decided to use a photodetector to detect the presence of a bottle. Because the equipment handles different diameter bottles, the position of the bottle's neck in the image can change with changes in bottle diameter. However, with the encoder, the photodetector can be mounted a short distance upstream of the imaging area, and the known distance from the photodetector to the imaging area along with the diameter of the bottle, which is known in advance, the system can determine the elapsed time between the detection of the bottle and when it will be positioned in the center of the field-of-view for imaging. Similar to the output queue for resynchronizing, an input queue is established to time the bottles coming into view. Repeatability of results will be better if the bottles are always positioned in the center of the field-of-view. If the frame grabber cannot accept the photodetector signal and produce an interrupt without starting image acquisition, its output will need to be serviced with a general purpose digital I/O board that has interrupt capability.

Summary

The latencies can be recompiled as shown below in Table 5.

	Latency			
	Minimum	Maximum		
Part detector	0.45	0.75		
Capture command	nil	nil		
Strobe/shutter trigger	0	.064		
Subtotal	0.45	0.814	Position Variation ⇒	.01 inch
Exposure time	0.25	0.25	Blur ⁵ ⇔	0.8 pixel
Video transfer	33	33		
Transfer to CPU ⁶	6.2	16		
Image processing	10	20		
Subtotal	49.9	70.1		
Resynchronization	19.5	0		
Time Base	0	3.9		
Output activation	0	10		
Total	69.4	84.0		

Table 5 – Final Latencies for Example 2

The maximum latency of 84 msec. is well within the 250 msec. rate at which bottles appear. The final uncertainty in latency of 15 msec. means the uncertainty in bottle location at the ejector is no more than 0.48 inch total (± 0.24 inch); well within workable parameters for a mechanical ejector.

⁵ Assumes a 7 inch wide field-of-view spanned by 640 pixels.

⁶ Allows time for the transfer of two images, one per frame grabber.

To summarize the changes from the original conception to a design meeting real-time and high-speed requirements, the following changes were made:

- One CCTV camera was replaced with two progressive scan, asynchronously resettable cameras with electronic shutters set to 250 μ sec.
- A photodetector was added to detect bottles coming into view. The photodetector is positioned upstream from the imaging area, and its output is placed in an input queue to let the application software time the start of image acquisition.
- An encoder is added to the conveyor to be the system's clock. It provides timing pulses that enable the system to time the arrival and rejection of bottles.
- An elementary analog frame grabber is replaced two frame grabbers that can accept an external trigger and control the camera's asynchronous reset. The frame grabber will be a bus master that has scatter-gather capability.
- Either the frame grabber or a separate digital I/O board will also be able to accept the signals from the encoder and the photodetector and generate an interrupt without starting image acquisition.
- The processor is a high-end PC running the very minimum version of embedded Windows NT.
- The application software will be built around a commercially available package that has good performance.

Written by:
Perry C. West
President
Automated Vision Systems, Inc.
127 Belcrest Dr.
Los Gatos, CA 95032 U.S.A.
(408) 356-2171
www.autovis.com

Commissioned by:
CyberOptics - Imagenation
10220 SW Nimbus, Suite K5
Portland, OR 97223 U.S.A.
(503) 495-2200
(800) 366-9131
www.imagenation.com

