The Basics of Video Decoders in Supervision and Inspection

By Witold Kaczurba

Video inspection[1] systems are used in many commercial and industrial processes. Cameras—which range from those in inexpensive, low-definition black-and-white closed-circuit television (CCTV) systems to those in state-of-the-art high-definition digital-video systems—are used in diverse applications ranging from product inspection to traffic monitoring to real-time face recognition.

Video inherently carries a lot of data, which can complicate signal-processing and data-storage tasks. Video inspection can often be simplified by cropping useless information and passing only the essential parts of the picture, which saves both memory and computational cycles. Figure 1 shows the elements of a typical system.

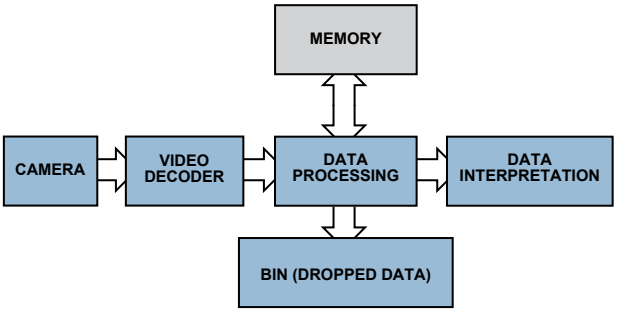

Figure 1. Simplified video-inspection data flow.

This article shows a few examples of how extracting useful data can minimize processing, memory size, and DSP usage—and illustrates how special features of Analog Devices video decoders[2] can simplify video algorithms and speed up development of video inspection systems.

Example 1. Counting and Inspecting Objects

Imagine a wide conveyor belt containing many rapidly moving products. The large number of products makes manual counting difficult. In addition to automating the counting task, a camera can be used to monitor product quality. This can be accomplished by modifying the simple count algorithm to focus on particular details and artifacts.

Storing all of the video data requires a huge amount of memory, and processing a large amount of data will cost a lot in terms of hardware and power. Instead of collecting whole pictures in memory, the system can find interesting details in the pile of data and drop as much useless data as possible when inspecting the products on the conveyor belt.

In most cases, gray-scale pictures can carry enough information. So chrominance information can be dropped by converting RGB signals to Y (luminance only). The resulting monochromatic picture can then be examined for content by using *edge detection* to find products on the belt and compare their shapes with a template to determine whether the product is misshapen.

Edge detection algorithms—which require only a few lines of active video and a small amount of memory—find discontinuities in the brightness of adjacent pixels by calculating the first and second derivatives of active pictures, as described in Digital Image Processing by Bernd Jähne.[3] Edge detection can be implemented in practice by extracting information using matrix calculations, such as the Sobel[4] matrix operator. In an FPGA (field-programmable gate-array) implementation, doing this on a pixel basis gives satisfying results. A simple FPGA implementation is shown in "A proposed FPGA Based Architecture for Sobel Edge Detection Operator" by Tanvir A. Abbasi and Mohm. Usaid Abbasi.[5] Noise can be removed by adding a Gaussian 2D filter, as described in "Hardware Acceleration of Edge Detection Algorithm on FPGAs" by Mathukumar Venkatesan and Daggu Venkateshwar Rao,[6] which describes a successful implementation of a detector similar to the Canny edge detector.[7]

Several other optimization algorithms can enhance the picture quality, but all occupy significant space on the FPGA design. However, some integrated-circuit (IC) video decoders are already equipped with useful preprocessing algorithms or filters; so choosing one of these would save space in the FPGA. For example, the ADV7802 video decoder[8] includes both *luma transient improvement* (LTI) and *chroma transient improvement* (CTI) blocks. These blocks, which enhance the resulting picture by improving the steepness of luma and chroma transitions, use adaptive peaking and nonlinear methods—without increasing noise or introducing artifacts—and can be very useful in the process of edge detection. In addition, luma-shaping and other built-in input filters can remove high-frequency noise from the source—focusing on the signal and ignoring incidental noise.

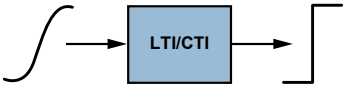

Figure 2. LTI/CTI operation diagram.

Edge detection provides information on an object's edge transitions instead of a full picture of the object. This reduction, from 3 × 8 bits per pixel (bpp) to 1 bpp, saves a lot of memory:

- 640 pixels × 480 pixels = 307,200 bits at 1 bpp
- 800 pixels × 600 pixels = 480,000 bits at 1 bpp
- 1024 pixels × 768 pixels = 786,432 bits at 1 bpp
- 1280 pixels × 720 pixels = 921,600 bits at 1 bpp

By converting RGB to Y, storing just a few lines of active video in memory, and using FPGA algorithms, we can detect objects and see their shapes. Once their locations on the moving belt are known, we can estimate their movement and collect color or other information from the next frames with the assurance that a minimum amount of memory is being used. The process involves

1. Edge detection
2. Storing information
3. Predicting the next position x_{n+1}
4. Extracting information in area where product is supposed to be.

Example 2. Detecting Motion and Quality

A robot is looking for items at a particular distance and within a limited range. Ultrasound can be used in some applications; but if the surface absorbs ultrasound or the items are behind glass, video can be used. The camera is set to focus on nearby objects. Items within a narrow range will have sharp edges, but background items—which are outside that range—have fuzzy edges (Figure 3).

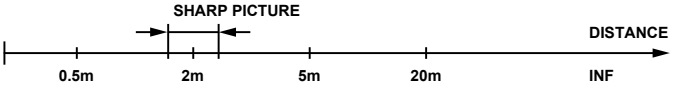

Figure 3. Focus—narrow depth of field.

Edge detection can be used to distinguish the items within the target range, as these are the only ones with sharp edges. Items in the background will be fuzzy enough to fail an edge detection test. Processing yields a binary bitmap where a 1 means that an edge was detected and a 0 means that no edge was detected. The position (x, y) of each detected edge pixel can be used to approximate the middle of an isolated object using Equation 1:

$$x_{average} = \frac{\sum_{n}^{N} x_n}{N} \; ; \qquad y_{average} = \frac{\sum_{n}^{N} y_n}{N} \tag{1}$$

Where x_n is the x-position of edge pixel, n; y_n is the y-position of edge pixel, n; and N is the number of edge pixels detected.

Once the position of the object and its edges are known, we can try to trace it. The key is to extract exactly one object from the picture, transforming its edges to an outline that can be used to determine if the item is moving toward the camera by checking the average distance of pixels from the middle of the object to see if the size of object is changing, as shown in Equation 2 (below).

N is the number of edge pixels in FRAME; M is the number of edge pixels in FRAME–1.

Focusing on the horizontal axis leads to Equation 3 (below).

The value of this equation will be positive when the object is moving toward the camera (pixels are spreading from the middle of object). A negative value means that the object is moving away from the camera, as shown in Figure 4.

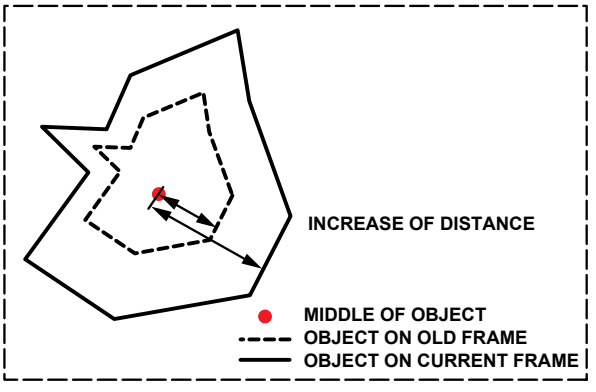

Figure 4. Frame change of moving object.

Note that the object has to be within the camera's range of focus. By modifying the algorithm, we can actively change the focus[9] to scan a wider area. Once the objects are detected, they can be segmented, processed, and tracked.

Tracking objects becomes more difficult as video complexity increases, especially with textured objects and objects that lose sharpness because they move quickly. Some tracking algorithms are shown in "Good Features to Track" by Jianbo Shi.[10] As objects lose sharpness, edge detection fails. Tracking can still be done by using complex correlation techniques such as block matching—used to estimate motion—or other methods detailed in "Video Processing and Communications" by Yao Wang, Jörn Ostermann, and Ya-Qin Zhang.[11]

Thanks to continuous data flow from the camera, an object can be tracked to determine its acceleration and other parameters. However, a high-quality video sequence must be used in order to obtain good video analysis results. When detecting edges by analyzing adjacent pixels, the resolution will be better if progressive-scan video is used instead of low-quality interlaced PAL or NTSC signals. The ADV7401 and ADV7403 video decoders[12] accept a variety of video standards, including progressive modes. Capable of digitizing video signals up to 140 MHz, they can handle SD, ED, and HD component signals, CVBS, and graphics. In addition, they support nonstandard video modes, allowing the use of less-popular standards, such as STANAG. The flexible pixel output bus allows data processing in 4:2:2, 4:4:4 YCbCr, or 4:4:4 RGB formats. Nonstandard video formats can be oversampled or undersampled to get a given horizontal width, as described in AN-0978, "Component Processor Nonstandard Video Formats".[13]

The built-in *color-space converter* (CSC), shown in Figure 5, transforms the color space to suit user requirements (Equation 4 below, where A1… A4, B1… B4, C1… C4 are adjustable CSC parameters). YPrPb or RGB input signals can be converted to other formats using configurable matrix conversion. For instance, converting RGB to YCbCr allows chroma information (Cb, Cr) to be dropped, simplifying edge detection with a monochrome picture.

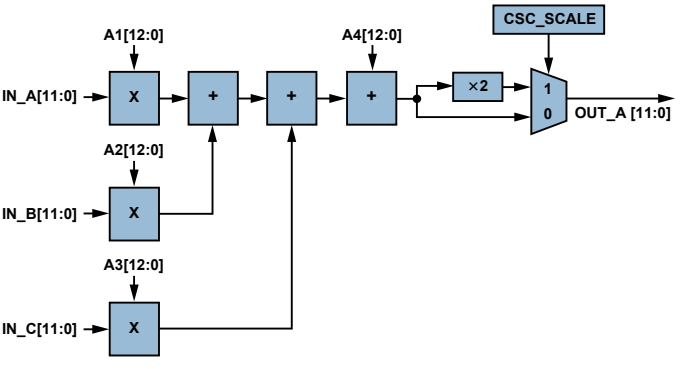

Figure 5. Single CSC Channel (ADV7403).

$$\frac{1}{N}\sum_{n}^{N}\left(\sqrt{\left(x_n^{FRAME} - x_{average}^{FRAME}\right)^2 + \left(y_n^{FRAME} - y_{average}^{FRAME}\right)^2}\right) - \\ + \frac{1}{M}\sum_{m}^{M}\left(\sqrt{\left(x_m^{FRAME-1} - x_{average}^{FRAME-1}\right)^2 + \left(y_m^{FRAME-1} - y_{average}^{FRAME-1}\right)^2}\right) \tag{2}$$

$$\frac{1}{N}\sum_{n}^{N}\left(x_i^{FRAME} - x_{average}^{FRAME}\right) - \frac{1}{M}\sum_{n}^{M}\left(x_i^{FRAME-1} - x_{average}^{FRAME-1}\right) \tag{3}$$

$$\begin{bmatrix} Channel\_A\_out \\ Channel\_B\_out \\ Channel\_C\_out \end{bmatrix} = \begin{bmatrix} A1 & A2 & A3 \\ B3 & B1 & B2 \\ C2 & C3 & C1 \end{bmatrix} \begin{bmatrix} Channel\_A\_in \\ Channel\_B\_in \\ Channel\_C\_in \end{bmatrix} + \begin{bmatrix} A4 \\ B4 \\ C4 \end{bmatrix} \tag{4}$$

The CSC is very useful. With an RGB or YCbCr input, color information can be simply transformed using a color-space matrix. Figure 6 shows a YUV color space that is similar to YCbCr.

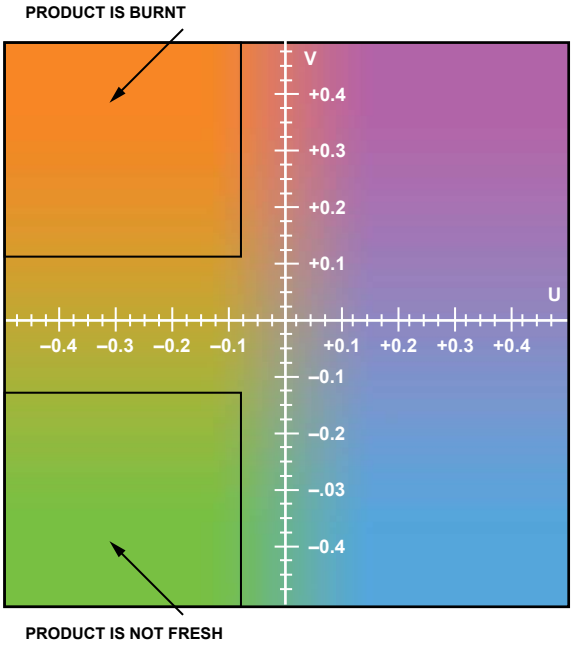

Figure 6. YUV color space in product-quality evaluation can be used to detect (for example) when a product is burned or moldy. Y (luma) is constant.

As Figure 6 shows, the color (or YPrPb value), can help to detect the quality of the product, for example, whether it is burned or moldy. Color-space conversion is necessary in video processing and for interfacing to ICs that use other standards. The ADV7401/ADV7403 include an input multiplexer that enables easy switching of video sources, a useful feature when switching from a stopped conveyor belt to a working one.

Example 3. Adjusting White and Color Balance for Video Inspection

Significant effort is required to develop a video system that extracts objects from a picture, as simple changes in light angle or intensity can affect the inspection results. Video engineers can use the ADV7401/ADV7403 gain and offset adjustments to adjust the brightness and contrast by adding two small reference stripes (one dark, one bright) to the conveyor belt. The offset and gain of the ADV7401/ADV7403 are adjusted to get comparable values, thus allowing the system to compensate for changes in light, color, angle, and intensity.

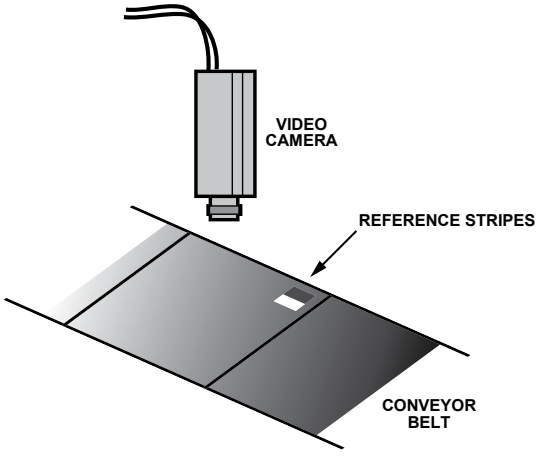

Figure 7. Small reference stripes are added to the visible area.

The algorithm for adjusting proper white balance[14] can be very simple. First, get reference RGB (or YCbCr) values for the stripes. Then, to compensate for light, simply change the offset and gain to get the same values as the reference. This algorithm can be used:

1. Get RGB (or YCbCr) values of the dark stripe.
2. Adjust offset to match desired RGB (or YCbCr) value of dark stripe.
3. Get RGB (or YCbCr) values of the light stripe
4. Adjust gain to match desired RGB (or YCbCr) value of light stripe.
5. To improve accuracy, repeat steps 2 and 4.

This procedure is especially useful during system development, as it provides the correct offset (brightness) and gain (contrast)—even when the light is too strong or too weak, as shown in Figure 8. The offset and gain registers are available via the I^2C bus, allowing quick adaptation.

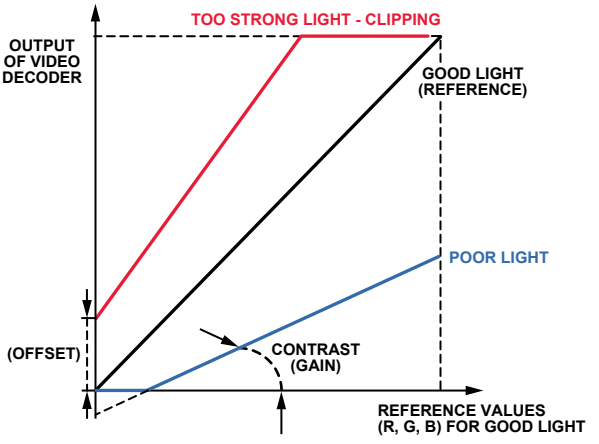

Figure 8. The offset and gain are adjusted to compensate for changes in ambient lighting.

Colors can also be used for the reference stripes. This compensation is similar to white balance, which is widely used, but while white balance matches a human's perception, the color correction is to compensate for changes due to different lighting. Although the algorithm is similar, an additional offset causes dark colors to look unnatural. The ADV7401/ADV7403 color-space conversion, flexible output pixel port, and offset and gain adjustment registers allow engineers to quickly develop algorithms using data that is already prepared for processing. As discussed earlier, it is important to reduce the amount of data required for video processing and to avoid advanced algorithms if they're not needed for simple video. An evaluation board for the ADV7401/ADV7403 with an easily accessible pixel port is available to speed up the start of new design. It's a matter of simply plugging a video-capture board into the pixel port of the evaluation board and capturing the video-data (Figure 9).

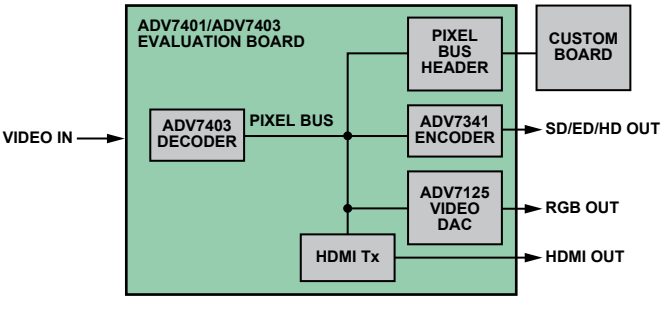

Figure 9. Pixel bus on ADV7401/ADV7403 evaluation board.

The video encoder, video DAC, and AD9889B HDMI transmitter[15] are connected to the same pixel bus, allowing viewing of the current picture on the second output. Analog Devices video decoders include the blocks required for processing video, giving robust performance and stable pictures.

Conclusion

Video cameras provide many benefits in industrial applications. This is particularly important when moving items must be sorted, tracked, or recorded. Video technology and real-time processing with highly integrated video decoders can be used to efficiently analyze items or sort mixed products on a moving conveyor belt.

References

[1] http://www.analog.com/en/industrial-solutions/machine-vision/applications/index.html

[2] http://www.analog.com/en/audiovideo-products/video-decoders/products/index.html

[3] http://www.amazon.com/Digital-Image-Processing-CD-ROM-Bernd/dp/3540677542

[4] http://en.wikipedia.org/wiki/Sobel_operator

[5] http://www.oldcitypublishing.com/FullText/JAPEDfulltext/JAPED2.4fulltext/Abbasi.pdf

[6] http://www.uweb.ucsb.edu/~shahnam/HAoEDAoF.pdf

[7] http://en.wikipedia.org/wiki/Canny_edge_detector

[8] http://www.analog.com/en/analog-to-digital-converters/video-decoders/adv7802/products/product.html

[9] http://en.wikipedia.org/wiki/Autofocus

[10] http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=323794

[11] http://www.amazon.com/Video-Processing-Communications-Prentice-Hall-Signal/dp/0130175471

[12] http://www.analog.com/en/analog-to-digital-converters/video-decoders/adv7403/products/product.html

[13] http://www.analog.com/static/imported-files/application_notes/AN-0978.pdf

[14] http://en.wikipedia.org/wiki/White_balance

[15] http://www.analog.com/en/audiovideo-products/analoghdmidvi-interfaces/ad9889b/products/product.html

The Author

Witold (Vito) Kaczurba [witold.kaczurba@analog.com] is an applications engineer in our Advanced TV group in Limerick, Ireland, where he supports decoders and HDMI products. He joined ADI in 2007, following graduation from Technical University of Wroclaw, Poland, with a master's degree in electronic engineering. As a student, he gained experience while working for small electronic and IT companies, then joined Analog Devices in Ireland as a co-op student—and subsequently as an applications engineer. In his spare time, he enjoys sightseeing Ireland, tasting Guinness beer with his friends, and reading.