Use of Video Technology To Improve Automotive Safety Becomes More Feasible with Blackfin™ Processors

By David Katz [david.katz@analog.com]
Tomasz Lukasiak [tomasz.lukasiak@analog.com]
Rick Gentile [richard.gentile@analog.com]

Dozens of processors control every performance aspect of today's automobiles, and not a single feature of the "vehicle experience" remains untouched by technology. Whether it's climate control, engine control, or entertainment, there has been constant evolution of capabilities in manufacturer offerings over the last decade. One of the forces behind this evolution, the rapidly increasing performance-to-cost ratio of signal processors, is about to have a profound impact on another critical automotive component—the safety subsystem.

While most currently available safety features utilize a wide array of sensors—principally involving microwaves, infrared light, lasers, accelerometers, or position detection—only recently have processors been introduced that can meet the real-time computation requirements that allow *video image processing* to contribute substantially to safety technology. The Analog Devices Blackfin *media-processor* family offers attractive solutions for this growing market, with its high processing speeds, versatile data-movement features, and video-specific interfaces. This article will discuss the roles that Blackfin processors can play in the emerging field of video-based automotive safety.

VIDEO IN AUTOMOTIVE SAFETY SYSTEMS

In many ways, car safety can be greatly enhanced by video-based systems that use high-performance media processors. Because short response times are critical to saving lives, however, image processing and video filtering must be done deterministically in real time. There is a natural tendency to use the highest video frame rates and resolution that a processor can handle for a given application, since this provides the best data for decision making. In addition, the processor needs to compare vehicle speeds and relative vehicle-object distances against desired conditions—again in real time. Furthermore, the processor must interact with many vehicle subsystems (such as the engine, braking, steering, and airbag controllers), process sensor information from all these systems, and provide appropriate audiovisual output to the driver. Finally, the processor should be able to interface to navigation and telecommunication systems to react to and log malfunctions, accidents, and other problems.

Figure 1 shows the basic video operational elements of an automotive safety system, indicating where image sensors might be placed throughout a vehicle, and how a lane departure system might be integrated into the chassis. There are a few things worth noting. First, multiple sensors can be shared by different automotive safety functions. For example, the rear-facing sensors can be used when the vehicle is backing up, as well as to track lanes as the vehicle moves forward. In addition, the lane-departure system might accept feeds from any of a number of camera sources, choosing the appropriate inputs for a given situation. In a basic system, a video stream feeds its data to the embedded processor. In more advanced systems, the processor receives other sensor information, such as position data from GPS receivers.

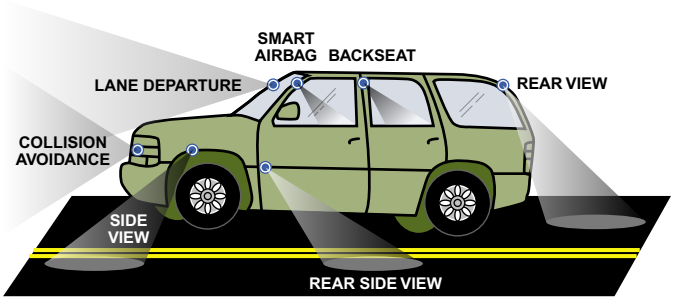

Figure 1. Basic camera-placement regions for automotive safety applications.

Smart Airbags

An emerging use of media processors in automotive safety is for "intelligent airbag systems," which base deployment decisions on who is sitting in the seat affected by the airbag. At present, weight-based systems are in widest use, but video sensing will become popular within five years. Either thermal or regular cameras may be used, at rates up to 200 frames per second, and more than one might be employed—to provide a stereo image of each occupant. The goal is to characterize the position and posture of the occupants—not just their size. In the event of a collision, the system must choose whether to restrict deployment entirely, deploy with a lower force, or deploy fully. In helping to determine body position, image-processing algorithms must be able to differentiate between a person's head and other body parts.

In this system, the media processor must acquire multiple image streams at high frame rates, process the images to profile the size and position of each occupant under all types of lighting conditions, and constantly monitor all the crash sensors, located throughout the car, in order to make the best deployment decision possible in a matter of milliseconds.

Collision Avoidance and Adaptive Cruise Control

Another high-profile safety application is *adaptive cruise control* (ACC), a subset of *collision avoidance systems*. ACC is a convenience feature that controls engine and braking systems to regulate the speed of the car and its distance from the vehicle ahead. The sensors employed involve a combination of microwave, radar, infrared, and video technology. A media processor might process between 17 and 30 frames per second in real time from a camera—focused on the roadway—mounted near the car's rear-view mirror. The image-processing algorithms may include frame-to-frame image comparisons, object recognition, and contrast equalization for varying lighting scenarios. Goals of the video sensor input are to provide information about lane boundaries and road curvature, and to categorize obstacles, including vehicles ahead of the car.

ACC systems are promoted as a convenience feature, while true collision avoidance systems actively aim to avoid accidents by coordinating the braking, steering, and engine controllers of the car. As such, they have been slower to evolve because of the complexity of the task, the critical reliability considerations, and legal and social consequences. It is estimated that deployment of these systems may be well on its way by 2010. In view of the typical 5-year automotive design cycle, such system designs are already underway.

Collision *warning* systems, like ACC, are a subset of the collision-avoidance category. These provide a warning of a possibly impending accident, but they don't actively avoid it. There are two main subcategories within this niche:

Blind spot monitors—Cameras are mounted strategically around the periphery of the vehicle to provide a visual display of the driver's blind spots—and to sound a warning if the processor senses the presence of another vehicle in a blind-spot zone. In reverse gear, these systems also serve as back-up warnings, cautioning the driver about obstructions in the rear of the car. A display could be integrated with the rear-view mirror, providing a full, unobstructed view of the car's surroundings. Moreover, the system might include a video of "blind spots" *within* the car cabin, allowing the driver to monitor a rear-facing infant, for example.

Lane-departure monitors—These systems can notify drivers if it is unsafe to change lanes or if they are straying out of a lane or off the road—thus aiding in detecting driver fatigue. Forward-facing cameras monitor the car's position relative to the roadway's centerline and side markers, up to 50 to 75 feet in front of the car. The system sounds an alarm if the car starts to leave the lane unintentionally.

LANE DEPARTURE—A SYSTEM EXAMPLE

In addition to the role that a media processor can play in video-based automotive safety applications, it is instructive to analyze typical components of just such an application. To that end, let's probe further into a lane-departure monitoring system that could employ the Blackfin media processor.

The overall system diagram of Figure 2 is fairly straightforward, considering the complexity of the signal processing functions being performed. Interestingly, in a video-based lane departure system, the bulk of the processing is image-based, and is carried out within a signal processor rather than by an analog signal chain. This represents a big savings on the system bill-of-materials. The output to the driver consists of a warning to correct the car's projected path before the vehicle leaves the lane unintentionally. It may be an audible "rumble-strip" sound, a programmed chime, or a voice message.

The video input system to the embedded processor must perform reliably in a harsh environment, including wide and drastic temperature shifts and changing road conditions. As the data stream enters the processor, it is transformed—in real time—into a form that can be processed to output a decision. At the simplest level, the lane departure system looks for the vehicle's position with respect to the lane markings in the road. To the processor, this means the incoming stream of road imagery must be transformed into a series of lines that delineate the road surface.

The processor can find lines within a field of data by looking for edges. These edges form the boundaries within which the driver should keep the vehicle while it is moving forward. The processor must track these line markers and determine whether to notify the driver of irregularities.

Keep in mind that several other automobile systems also influence the lane-departure system. For example, use of the braking system and the turn signals typically will block lane departure warnings during intentional lane changes and slow turns.

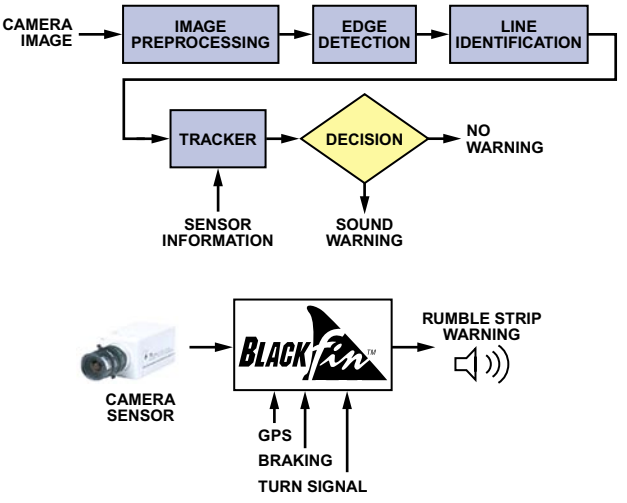

Figure 2. Basic steps in a lane-departure algorithm and how the processor might connect to the outside world.

Let's now drill deeper into the basic components of the lane-departure system example. Figure 3 follows the same basic operational flow as Figure 2 but with more insight into the algorithms being performed. The video stream coming into the system needs to be filtered and smoothed to reduce noise caused by temperature, motion, and electromagnetic interference. Without this step, it would be difficult to find clean lane markings.

The next processing step involves edge detection; if the system is set up properly, the edges found will represent the lane markings. These lines must then be matched to the direction and position of the vehicle. The *Hough transform* will be used for this step. Its

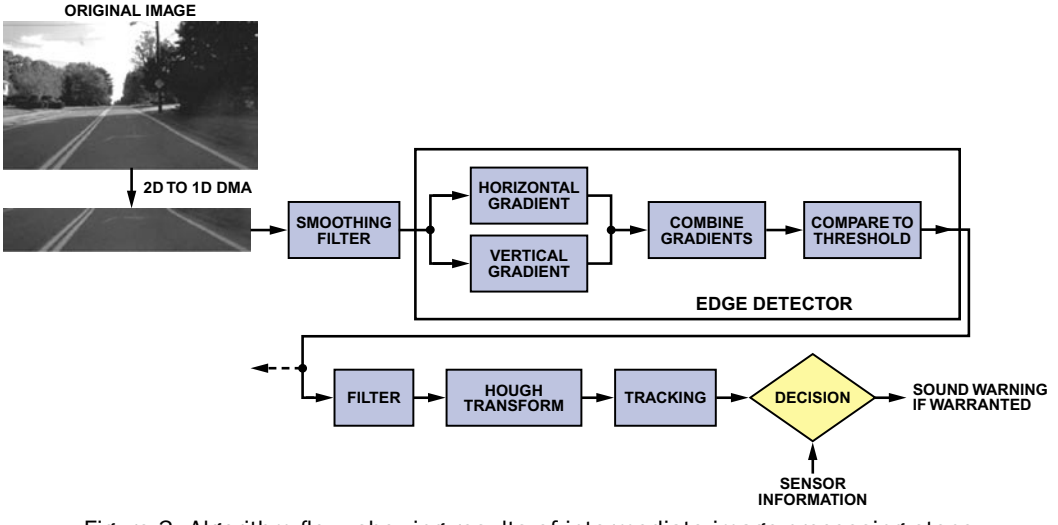

Figure 3. Algorithm flow, showing results of intermediate image-processing steps.

output will be tracked across frames of images, and a decision will be made based on all the compiled information. The final challenge is to send a warning in a timely manner without sounding false alarms.

Image Acquisition

An important feature of the Blackfin Processor is its *parallel peripheral interface* (PPI), which is designed to handle incoming and outgoing video streams. The PPI connects without external logic to a wide variety of video converters. In addition to ITU-R 656-compliant video encoders and decoders, the PPI can connect to CMOS camera chips and LCD displays, which find common use in the automotive industry. Because it can capture video in real time, the PPI is instrumental for the kinds of auto safety applications discussed in this article.

In devices supporting ITU-R 656, each boundary between blanking data and active video data is set using a 4-byte data sequence that is embedded within the data stream. The PPI automatically decodes this sequence, without processor intervention, to collect the incoming active video frames. With this embedded control scheme, the physical connection is simply eight data lines and a clock.

The PPI also connects to a wide range of image sensors and data converters that do not have an embedded control scheme. In these cases, the PPI provides up to three frame syncs to manage incoming or outgoing data. For a video stream, these frame syncs function as physical horizontal sync, vertical sync and field lines (HSYNC, VSYNC, and FIELD).

For automotive safety applications, image resolutions typically range from VGA (640 × 480 pixels/image) down to QVGA (320 × 240 pixels/image). Regardless of the actual image size, the format of the data transferred remains the same—but lower clock speeds can be used when less data is transferred. Moreover, in the most basic *lane-departure warning systems*, only gray-scale images are required. The data bandwidth is therefore halved (from 16 bits/pixel to 8 bits/pixel) because chroma information can be ignored.

Memory and Data Movement

Efficient memory usage is an important consideration for system designers because external memories are expensive, and their access times can have high latencies. While Blackfin processors have an on-chip SDRAM controller to support the cost-effective addition of larger, off-chip memories, it is still important to be judicious in transferring *only the video data needed* for the application. By intelligently decoding ITU-R 656 preamble codes, the PPI can aid this "data-filtering" operation. For example, in some applications, only the active video fields are required. In other words, horizontal and vertical blanking data can be ignored and not transferred into memory, resulting in up to a 25% reduction in the amount of data brought into the system. What's more, this lower data rate helps conserve bandwidth on the internal and external data buses.

Because video data rates are very demanding, frame buffers must be set up in external memory, as shown in Figure 4. In this scenario, while the processor operates on one buffer, a second buffer is being filled by the PPI via a DMA transfer. A simple semaphore can be set up to maintain synchronization between the frames. With Blackfin's flexible DMA controller, an interrupt can be generated at virtually any point in the memory fill process, but it is typically configured to occur at the end of each video line or frame.

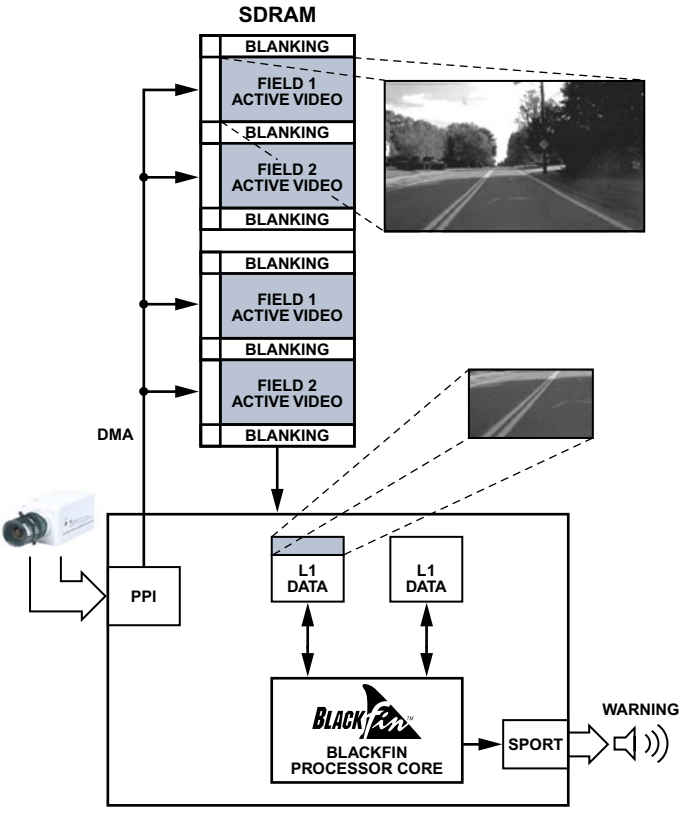

Figure 4. Use of external memory for a frame buffer.

Once a complete frame is in SDRAM, the data is normally transferred into internal L1 data memory so that the core can access it with single-cycle latency. To do this, the DMA controller can use two-dimensional transfers to bring in pixel blocks. Figure 5 shows an example of how a 16 × 16 "macroblock," a construct used in many compression algorithms, can be stored linearly in L1 memory via a 2D DMA engine.

To efficiently navigate through a source image, four parameters need to be controlled: X Count, Y Count, X Modify, and Y Modify. X and Y Counts describe the number of elements to read in/out in the "horizontal" and "vertical" directions, respectively. *Horizontal* and *vertical* are abstract terms in this application because the image data is actually stored linearly in external memory. X and Y Modify values achieve this abstraction by specifying an amount to "stride" through the data after the requisite X Count or Y Count has been transferred.

From a performance standpoint, up to four unique SDRAM internal banks can be active at any time. This means that in the video framework, no additional bank-activation latencies are observed when the 2D-to-1D DMA is pulling data from one bank while the PPI is feeding another.

Projection Correction

The camera used for the lane departure system can be located in the center-top location of the front windshield, facing forward, in the rear windshield, facing the road already traveled, or as a "bird's-eye" camera, which gives the broadest perspective of the upcoming roadway and can thus be used instead of multiple line-of-sight cameras. In this latter case, the view is warped because of the wide-angle lens, so the output image must be remapped into a linear view before parsing the picture content.

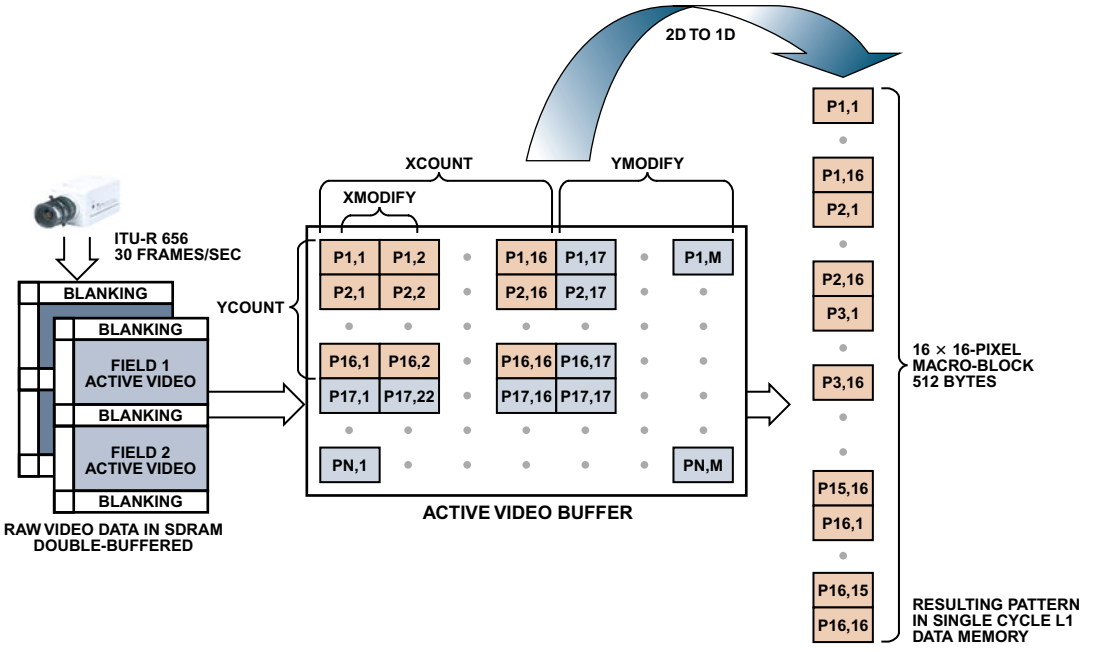

Figure 5. A 2D to 1D DMA transfer from SDRAM into L1 memory.

Image Filtering

Before doing any type of edge detection, it is important to filter the image to smooth out any noise picked up during image capture. This is essential because noise introduced into an edge detector can result in false edges output from the detector.

Obviously, an image filter needs to operate fast enough to keep up with the succession of input images. Thus, it is imperative that image filter kernels be optimized for execution in the fewest possible number of processor cycles. One effective means of filtering is accomplished with a basic two-dimensional convolution operation. Let's look at how this computation can be performed efficiently on a Blackfin Processor.

Convolution is one of the fundamental operations in image processing. In two-dimensional convolution, the calculation performed for a given pixel is a weighted sum of intensity values from pixels in the neighborhood of that pixel. Since the neighborhood of a mask is centered on a given pixel, the mask area usually has odd dimensions. The mask size is typically small relative to the image; a 3 × 3 mask is a common choice because it is computationally reasonable on a per-pixel basis but large enough to detect edges in an image.

The basic structure of the 3 × 3 kernel is shown in Figure 6. As an example, the output of the convolution process for a pixel at row 20, column 10 in an image would be:

$$Out(20,10) = A \times (19,9) + B \times (19,10) + C \times (19,11) + D \times (20,9) + E \times (20,10) + F \times (20,11) + G \times (21,9) + H \times (21,10) + I \times (21,11)$$

A	B	C
D	E	F
G	H	I

Figure 6. Basic structure of the 3 × 3 convolution kernel.

The high-level algorithm can be described with the following steps:

1. Place the center of the mask over an element of the input matrix.
2. Multiply each pixel in the mask neighborhood by the corresponding filter mask element.
3. Sum each of the multiplies into a single result.
4. Place each sum in a location corresponding to the center of the mask in the output matrix

Figure 7 shows an input matrix, F, a 3 × 3 mask matrix, H, and an output matrix, G.

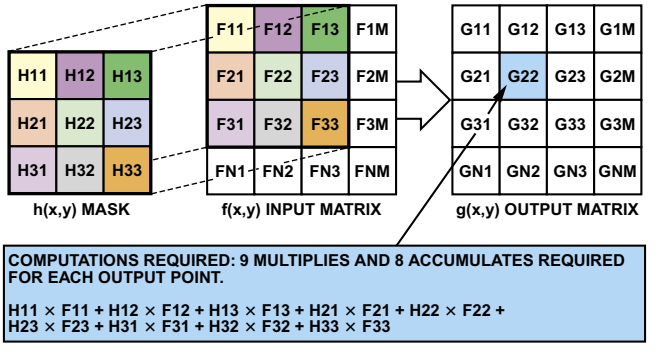

Figure 7. Input matrix, F; 3 × 3 mask matrix, H; and output matrix, G.

After each output point is computed, the mask is moved to the right. On the image edges, the algorithm wraps around to the first element in the next row. For example, when the mask is centered on element F2M, the H23 element of the mask matrix is multiplied by element F31 of the input matrix. As a result, the usable section of the output matrix is reduced by one element along each edge of the image.

By aligning the input data properly, both of Blackfin's *multiply-accumulate* (MAC) units can be used in a single processor cycle to process two output points at a time. During this same cycle, multiple data fetches occur in parallel with the MAC operation. This method allows efficient computation of 2 output points for each loop iteration, or 4.5 cycles per pixel instead of the 9 cycles per pixel of Figure 7.

Edge Detection

A wide variety of edge detection techniques are in common use. Before considering how an edge can be detected, the algorithm must first settle on a suitable definition for what an edge actually is, then find ways to enhance the edge features to improve the chances of detection. Because image sensors are non-ideal, two issues must be dealt with—*noise* and the effects of *quantization errors.*

Noise in the image will almost guarantee that pixels having equal gray scale levels in the original image will not have equal levels in the noisy image. Noise will be introduced based on many factors that can't be easily controlled, such as ambient temperature, vehicular motion, and outside weather conditions. Quantization errors in the image will result in edge boundaries extending across a number of pixels. These factors work together to complicate edge detection. Because of this, any image-processing algorithm selected must keep noise immunity as a prime goal.

One popular detection method uses a set of common derivative-based operators to help locate edges within the image. Each of the derivative operators is designed to find places where there are changes in intensity. In this scheme, the edges can be modeled by a smaller image that contains the properties of an ideal edge.

We'll discuss the Sobel Edge Detector because it is easy to understand and illustrates principles that extend into more complex schemes. The Sobel Detector uses two convolution kernels to compute gradients for both horizontal and vertical edges. The first is designed to detect changes in vertical contrast (S_x). The second detects changes in horizontal contrast (S_y).

$$S_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} S_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

The output matrix holds an "edge likelihood" magnitude (based on horizontal and vertical convolutions) for each pixel in the image. This matrix is then thresholded in order to take advantage of the fact that large responses in magnitude correspond to edges within the image. Therefore, at the input of the Hough Transform stage, the image consists only of either "pure white" or "pure black" pixels, with no intermediate gradations.

If the true magnitude is not required for an application, this can save a costly square root operation. Other common techniques in building a threshold matrix include summing the gradients from each pixel or simply taking the largest of the two gradients.

Straight Line Detection—*Hough Transform*

The Hough transform is a widely used method for finding global patterns such as lines, circles, and ellipses in an image by localizing them in a parameterized space. It is especially useful in lane detection because *lines* can be easily detected as *points* in Hough transform space, based on the polar representation of Equation 1:

$$\rho = x\cos\theta + y\sin\theta \qquad (1)$$

The meaning of this equation can be visualized by extending a perpendicular from the given line to the origin, such that θ is the angle that the perpendicular makes with the abscissa and ρ is the length of the perpendicular. Thus, one pair of coordinates (ρ, θ) can fully describe the line. Lines L1 and L2 in Figure 8a demonstrate this concept. Figure 8b shows that L1 is defined by θ_1 and the length of the red perpendicular, while L2 is defined by θ_2 and the length of the blue perpendicular line.

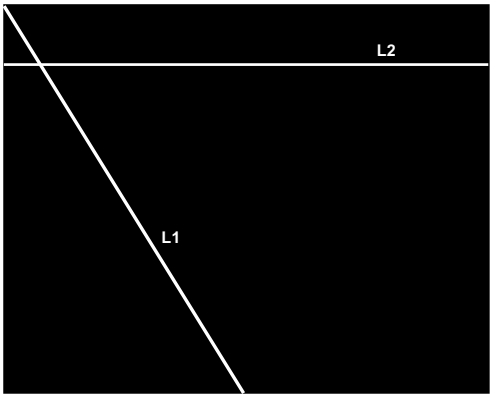

Figure 8a. The output of an edge detector is a binary image like this one, which can be visually inspected by a human observer to show lines. A Hough Transform allows localization of these two lines.

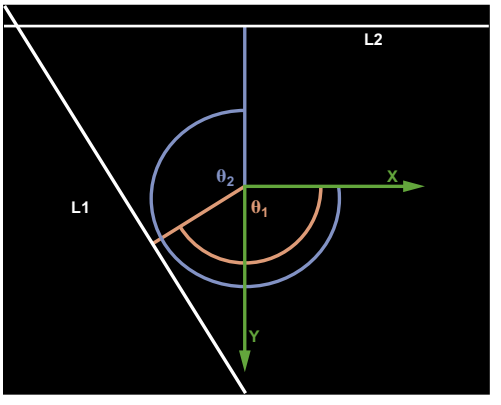

Figure 8b. The two white lines in the image above can be described by the lengths and angles of the red and blue perpendicular line segments extending from the origin.

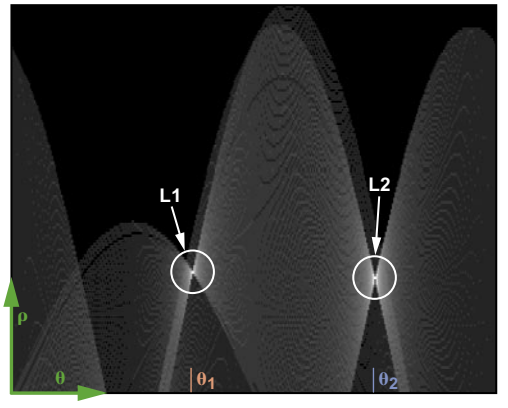

Figure 8c. The Hough transform of the image in Figure 8a. The range for θ is $[0, 2\pi]$, and the range for ρ is one-half the diagonal of the input image in Figure 8a. The two bright regions correspond to local maxima, which can be used to reconstruct the two lines in Figure 8a.

Another way to look at the Hough Transform is to consider a way that the algorithm could be implemented intuitively:

1. Visit only white pixels in the binary image.
2. For each pixel and every θ value being considered, draw a line through the pixel at angle θ from the origin. Then calculate ρ, which is the length of the perpendicular between the origin and the line under consideration.
3. Record this (ρ, θ) pair in an accumulation table.
4. Repeat steps 1–3 for every white pixel in the image.
5. Search the accumulation table for the (ρ, θ) pairs encountered most often. These pairs describe the most probable "lines" in the input image, because in order to register a high accumulation value, there had to be many white pixels that existed along the line described by the (ρ, θ) pair.

The Hough transform is computationally intensive because a sinusoidal curve is calculated for each pixel in the input image. However, certain techniques can speed up the computation considerably.

First, some of the computation terms can be computed ahead of time, so that they can be referenced quickly through a lookup table. In Blackfin's fixed-point architecture it is very useful to store the lookup table only for the *cosine* function. Since the sine values are 90 degrees out of phase with the cosines, the same table can be used, with an offset. With the lookup tables in use, the computation of Equation (1) can be represented as two fixed-point multiplications and one addition.

Another factor that can improve performance is a set of assumptions about the nature and location of lane markings within the input image. By considering only those input points that could potentially be lane markings, a large number of unnecessary calculations can be avoided, since only a narrow range of θ values need be considered for each white pixel.

The output of a Hough Transform is a set of straight lines that could potentially be lane markings. Certain parameters of these lines can be calculated by simple geometric equations. Among the parameters useful for further analysis are the *offset* from the camera's center axis, the *widths* of the detected lines, and the *angles* with respect to the position of the camera. Since lane markings in many highway systems are standardized, a set of rules can eliminate some lines from the list of lane-marking candidates. The set of possible lane-marking variables can then be used to derive the position of the car.

Lane Tracking

Lane information can be determined from a variety of possible sources within an automobile. This information can be combined with measurements of vehicle-related parameters (e.g., velocity, acceleration, etc.) to assist in lane tracking. Based on the results of these measurements, the lane-departure system can make an intelligent decision as to whether an unintentional departure is in progress. In advanced systems, other factors could be modeled, such as the time of day, road conditions, and driver alertness.

The problem of estimating lane geometry is a challenge that often calls for using a *Kalman filter* to estimate the road curvature. Specifically, the Kalman filter can predict future road information—which can then be used in the next frame to reduce the computational load presented by the Hough transform.

As described earlier, the Hough transform is used to find lines in each image. But these lines also need to be tracked over a series of images. In general, a Kalman filter can be described as a *recursive filter that estimates the future state of an object*. In this case, the object is a line. The state of the line is based on its location and its motion path across several frames.

Along with the road state itself, the Kalman filter provides a variance for each state. The predicted state and the variance can be used in conjunction to narrow the search space of the Hough transform in future frames, which saves processing cycles.

Decision Making—*Current Car Position or Time to Lane-Crossing*

From our experience, we know that false positives are always undesirable. There is no quicker way to get a consumer to disable an optional safety feature than to have it indicate a problem that does not exist.

With a processing framework in place, system designers can add their own intellectual property (IP) to the decision phase of each of the processing threads. The simplest approach might be to take into account other vehicle attributes when making a decision. For example, a lane-change warning could be suppressed when a lane change is perceived to be intentional—as when a blinker is used or when the brake is applied. More complex systems may factor in GPS coordinate data, occupant driving profile, time of day, weather, and other parameters.

CONCLUSION

In the foregoing discussion, we have only described an example *framework* for how an image-based lane departure system might be structured. The point we have sought to establish is that when a flexible media processor is available for the design, there is plenty of room to consider feature additions and algorithm optimizations.