IN THE UNITED STATES DISTRICT COURT
FOR THE EASTERN DISTRICT OF TEXAS

Image Processing Technologies, LLC Plaintiff, v. Samsung Electronics Co., Ltd and Samsung Electronics America, Inc. Defendant.	Civil Action No. 2:16-CV-505-JRG

DECLARATION OF GERARD P. GRENIER

I, Gerard P. Grenier, am over twenty-one (21) years of age. I have never been convicted of a felony, and I am fully competent to make this declaration.

1. I am Senior Director of Publishing Technologies of the Institute of Electrical and Electronics Engineers, Inc. ("IEEE"). My office address is 445 Hoes Lane Piscataway, NJ 08854.

2. Among my responsibilities as Senior Director of Publishing Technologies, I act as a custodian of certain records for IEEE.

3. IEEE received a Subpoena, dated November 10, 2016 in the above-captioned case, requesting the production of documents sufficient to show the date certain articles, which are identified below, were first published, sold, or otherwise made available to the public.

4. In response to that Subpoena, I make this declaration on behalf of the IEEE based on my personal knowledge and information contained in the business records of IEEE. It is the regular practice of IEEE to publish articles and other writings, including article abstracts, and make them available to the public through IEEE Xplore, an on-line digital library. IEEE maintains copies of publications in the ordinary course of its regularly conducted activities.

5. The following articles, accompanied by their abstracts, have been attached as Exhibits to this declaration:

A	Alton L. Gilbert, et al., "A Real-Time Video Tracking System"

B	O.D. Altan, H.K. Patnaik, and R.P. Roesser, "Computer Architecture and Implementation of Vision-Based Real-Time Lane Sensing"

6. I obtained copies of each of the attached Exhibits through IEEE Xplore, where they are maintained in the ordinary course of the IEEE's business. Each Exhibit is a true and correct copy of the Exhibit as it existed on or about November 16, 2016.

7. The attached Exhibits include a date of publication and the manner in which the publication was published. IEEE Xplore populates this information using the metadata associate with the publication. Based on IEEE's regular business practices, the date of publication identified in the abstract is the date when IEEE first made the article available to the public, first offered the article for sale in the United States, and/or first published the article.

Pursuant to Section 1746 of Title 28 of the United States Code, I declare under penalty of perjury that the foregoing is true and correct to the best of my personal knowledge or information

Executed on the 2nd day of November, 2016.

By: 

Gerard P. Grenier

Exhibit A

A Real-Time Video Tracking System

ALTON L. GILBERT, MEMBER, IEEE, MICHAEL K. GILES, GERALD M. FLACHS, MEMBER, IEEE, ROBERT B. ROGERS, MEMBER, IEEE, AND YEE HSUN U, MEMBER, IEEE

Abstract—**Object identification and tracking applications of pattern recognition at video rates is a problem of wide interest, with previous attempts limited to very simple threshold or correlation (restricted window) methods. New high-speed algorithms together with fast digital hardware have produced a system for missile and aircraft identification and tracking that possesses a degree of "intelligence" not previously implemented in a real-time tracking system. Adaptive statistical clustering and projection-based classification algorithms are applied in real time to identify and track objects that change in appearance through complex and nonstationary background/foreground situations. Fast estimation and prediction algorithms combine linear and quadratic estimators to provide speed and sensitivity. Weights are determined to provide a measure of confidence in the data and resulting decisions. Strategies based on maximizing the probability of maintaining track are developed. This paper emphasizes the theoretical aspects of the system and discusses the techniques used to achieve real-time implementation.**

Index Terms—**Image processing, intensity histograms, object identification, optical tracking, projections, tracking system, video data compression, video processing, video tracking.**

INTRODUCTION

IMAGE PROCESSING methods constrained to operate on sequential images at a high repetition rate are few. Pattern recognition techniques are generally quite complex, requiring a great deal of computation to yield an acceptable classification. Many problems exist, however, where such a time-consuming technique is unacceptable. Reasonably complex operations can be performed on wide-band data in real time, yielding solutions to difficult problems in object identification and tracking.

The requirement to replace film as a recording medium to obtain a real-time location of an object in the field-of-view (FOV) of a long focal length theodolite gave rise to the development of the real-time videotheodolite (RTV). U.S. Army White Sands Missile Range began the development of the RTV in 1974, and the system is being deployed at this time. Design philosophy called for a system capable of discriminatory judgment in identifying the object to be tracked with 60 independent observations/s, capable of locating the center of mass of the object projection on the image plane within about 2 percent of the FOV in rapidly changing background/foreground situations (therefore adaptive), able to generate a predicted observation angle for the next observation, and required to output the angular displacements of the object within the FOV within 20 ms after the observation was made. The system would be required to acquire objects entering the FOV that had been prespecified by shape description. In the RTV these requirements have been met, resulting in a real-time application of pattern recognition/image processing technology.

The RTV is made up of many subsystems, some of which are generally not of interest to the intended audience of this paper. These subsystems (see Fig. 1) are as follows:

1) main optics;
2) optical mount;
3) interface optics and imaging subsystem;
4) control processor;
5) tracker processor;
6) projection processor;
7) video processor;
8) input/output (I/O) processor;
9) test subsystem;
10) archival storage subsystem;
11) communications interface.

The *main optics* is a high quality cinetheodolite used for obtaining extremely accurate (rms error $\simeq 3$ arc-seconds) angular data on the position of an object in the FOV. It is positioned by the *optical mount* which responds to azimuthal and elevation drive commands, either manually or from an external source. The *interface optics and imaging subsystem* provides a capability to increase or decrease the imaged object size on the face of the silicon target vidicon through a 10:1 range, provides electronic rotation to establish a desired object orientation, performs an autofocus function, and uses a gated image intensifier to amplify the image and "freeze" the motion in the FOV. The camera output is statistically decomposed into background, foreground, target, and plume regions by the *video processor*, with this operation carried on at video rates for up to the full frame. The *projection processor* then analyzes the structure of the target regions to verify that the object selected as "target" meets the stored (adaptive) description of the object being tracked. The *tracker processor* determines a position in the FOV and a measured orientation of the target, and decides what level of confidence it has in the data and decision. The *control processor* then generates commands to orient the mount, control the interface optics, and provide real-time data output. An *I/O pro-*

Manuscript received September 14, 1978; revised November 19, 1978. This work was supported by the U.S. Army ILIR Program and the U.S. Army Research Office.

A. L. Gilbert and M. K. Giles are with the U.S. Army White Sands Missile Range, White Sands, NM 88002.

G. M. Flachs and R. B. Rogers are with the Department of Electrical Engineering, New Mexico State University, Las Cruces, NM 88003.

Y. H. U was with the Department of Electrical Engineering, New Mexico State University, Las Cruces, NM 88003. He is now with Texas Instruments Incorporated, Dallas, TX 75222.

0162-8828/80/0100-0047$00.75 © 1980 IEEE

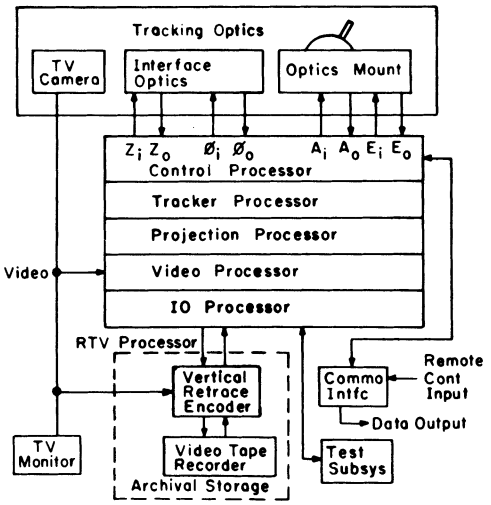


Fig. 1. RTV tracking system.

cessor allows the algorithms in the system to be changed, interfaces with a human operator for tests and operation, and provides data to and accepts data from the *archival storage subsystem* where the live video is combined with status and position data on a video tape. The *test subsystem* performs standard maintenance checks on the system. The *communications interface* provides the necessary interaction with the external world for outputing or receiving data.

The *video processor*, *projection processor*, *tracker processor*, and *control processor* are four microprogrammable bit-slice microprocessors [1], which utilize Texas Instruments' (TIs') new 74S481 Schottky processor, and are used to perform the real-time tracking function.

The four tracking processors, in turn, separate the target image from the background, locate and describe the target image shape, establish an intelligent tracking strategy, and generate the camera pointing signals to form a fully automatic tracking system.

Various reports and papers discuss several of the developmental steps and historical aspects of this project [2]-[7]. In this paper the video, projection, tracker, and control processors are discussed at some length.

Video Processor

The video processor receives the digitized video, statistically analyzes the target and background intensity distributions, and decides whether a given pixel is background or target [8]. A real-time adaptive statistical clustering algorithm is used to separate the target image from the background scene at standard video rates. The scene in the FOV of the TV camera is digitized to form an $n \times m$ matrix representation

$$P = (p_{ij})\, n, m$$

of the pixel intensities p_{ij}. As the TV camera scans the scene, the video signal is digitized at m equally spaced points across each horizontal scan. During each video field, there are n horizontal scans which generate an $n \times m$ discrete matrix representation at 60 fields/s. A resolution of $m = 512$ pixels per standard TV line results in a pixel rate of 96 ns per pixel. Every 96 ns, a pixel intensity is digitized and quantized into eight bits (256 gray levels), counted into one of six 256-level histogram memories, and then converted by a decision memory to a 2-bit code indicating its classification (target, plume, or background). There are many features that can be functionally derived from relationships between pixels, e.g., texture, edge, and linearity measures. Throughout the following discussion of the clustering algorithm, pixel intensity is used to describe the pixel features chosen.

The basic assumption of the clustering algorithm is that the target image has some video intensities not contained in the immediate background. A tracking window is placed about the target image, as shown in Fig. 2, to sample the background intensities immediately adjacent to the target image. The background sample should be taken relatively close to the target image, and it must be of sufficient size to accurately characterize the background intensity distribution in the vicinity of the target. The tracking window also serves as a spatial bandpass filter by restricting the target search region to the immediate vicinity of the target. Although one tracking window is satisfactory for tracking missile targets with plumes, two windows are used to provide additional reliability and flexibility for independently tracking a target and plume, or two targets. Having two independent windows allows each to be optimally configured and provides reliable tracking when either window can track.

The tracking window frame is partitioned into a background region (BR) and a plume region (PR). The region inside the frame is called the target region (TR) as shown in Fig. 2. During each field, the feature histograms are accumulated for the three regions of each tracking window.

The feature histogram of a region R is an integer-value, integer argument function $h^R(x)$. The domain of $h^R(x)$ is $[0, d]$, where d corresponds to the dynamic range of the analog-to-digital converter, and the range of $h^R(x)$ is $[0, r]$, where r is the number of pixels contained in the region R; thus, there are $r + 1$ possible values of $h^R(x)$. Since the domain $h^R(x)$ is a subset of the integers, it is convenient to define $h^R(x)$ as a one-dimensional array of integers

$$h(0), h(1), h(2), \cdots, h(d).$$

Letting x_i denote the ith element in the domain of x (e.g., $x_{25} = 24$), and $x(j)$ denote the jth sample in the region R (taken in any order), $h^R(x)$ may be generated by the sum

$$h^R(x_i) = \sum_{j=1}^{r} \delta x_i, x(j)$$

where δ is the Kronecker delta function

$$\delta_{i,j} = \begin{cases} 0 & i \neq j \\ 1 & i = j. \end{cases}$$

A more straightforward definition which corresponds to the actual method used to obtain $h^R(x)$ uses Iverson's notation [21] to express $h^R(x)$ as a one-dimensional vector of $d + 1$ integers which are set to zero prior to processing the region R as

$$h \leftarrow (d + 1)\, \rho 0.$$

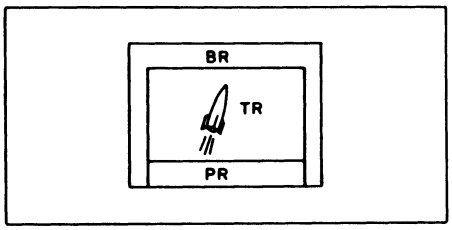


Fig. 2. Tracking window.

As each pixel in the region is processed, one (and only one) element of H is incremented as

$$h[x(j)] \leftarrow h[x(j)] + 1.$$

When the entire region has been scanned, h contains the distributions of pixels over intensity and is referred to as the feature histogram of the region R.

It follows from the above definition that h satisfies the identity

$$r = \sum_{i=0}^{d} h^R(x_i) \quad \text{or} \quad r = +/h.$$

Since h is also nonnegative and finite, it can be made to satisfy the requirements of a probability assignment function by the normalization

$$h \leftarrow h \div +/h.$$

Hereafter, all feature histograms are assumed to be normalized and are used as relative-frequency estimates of the probability of occurrence of the pixel values x in the region over which the histogram is defined.

For the ith field, these feature histograms are accumulated for the background, plume, and target regions and written

$$h_i^{BR}(x): \sum_x h_i^{BR}(x) = 1$$

$$h_i^{PR}(x): \sum_x h_i^{PR}(x) = 1$$

$$h_i^{TR}(x): \sum_x h_i^{TR}(x) = 1$$

after they are normalized to the probability interval $[0, 1]$. These normalized histograms provide an estimate of the probability of feature x occurring in the background, plume, and target regions on a field-by-field basis. The histograms are accumulated at video rates using high-speed LSI memories to realize a multiplexed array of counters, one for each feature x.

The next problem in the formulation of a real-time clustering algorithm is to utilize the sampled histograms on a field-by-field basis to obtain learned estimates of the probability density functions for background, plume, and target points. Knowing the relative sizes of the background in PR, the background in TR, and the plume in TR, allows the computation of estimates for the probability density function for background, plume, and target features. This gives rise to a type of nonparametric classification similar to mode estimation as discussed by Andrews [9], but with an implementation method that allows for real-time realization.

Letting

$$\alpha = \frac{\text{number of background points in PR}}{\text{total number of points in PR}}$$

$$\beta = \frac{\text{number of background points in TR}}{\text{total number of points in TR}}$$

$$\gamma = \frac{\text{number of plume points in TR}}{\text{total number of points in TR}}$$

and assuming that 1) the BR contains only background points, 2) the PR contains background and plume points, and 3) the TR contains background, plume, and target points, one has

$$h_i^{BR}(x) = h_i^B(x)$$

$$h_i^{PR}(x) = \alpha h_i^B(x) + (1 - \alpha) h_i^P(x)$$

$$h_i^{TR}(x) = \beta h_i^B(x) + \gamma h_i^P(x) + (1 - \beta - \gamma) h_i^T(x).$$

By assuming there are one or more features x where $h_i^B(x)$ is much larger than $h_i^P(x)$, one has

$$\alpha = \frac{h_i^{PR}(x)}{h_i^{BR}(x)} - \frac{\epsilon_p}{h_i^{BR}(x)}$$

where $\epsilon_p = (1 - \alpha) h_i^P(x) \ll h_i^B(x)$. Now for all features x where $h_i^P(x) = 0$, one has the solution $\alpha = h_i^{PR}(x)/h_i^{BR}(x)$. For all features x where $h_i^P(x) > 0$, the inequality $h_i^{PR}(x)/h_i^{BR}(x) > \alpha$ is valid. Consequently, a good estimate for α is given by

$$\alpha = \min_x \{h_i^{PR}(x)/h_i^{BR}(x)\}$$

and this estimate will be exact if there exists one or more features where $h_i^{BR}(x) \neq 0$ and $h_i^P(x) = 0$. Having an estimate of α and $h_i^B(x)$ allows the calculation of $h_i^P(x)$.

In a similar manner, estimates of β and γ are obtained,

$$\beta = \min_x \frac{h_i^{TR}(x)}{h_i^{BR}(x)}$$

$$\gamma = \min_x \frac{h_i^{TR}(x)}{h_i^P(x)}.$$

Having field-by-field estimates of the background, plume, and target density functions ($h_i^B(x)$, $h_i^P(x)$, $h_i^T(x)$), a linear recursive estimator and predictor [10] is utilized to establish learned estimates of the density functions. Letting $H(i|j)$ represent the learned estimate of a density function for the ith field using the sampled density functions $h_i(x)$ up to the jth field, we have the linear estimator

$$H(i|i) = w \cdot H(i|i-1) + (1 - w) h_i(x)$$

and linear predictor

$$H(i+1|i) = 2H(i|i) - H(i-1|i-1).$$

The above equations provide a linear recursive method for compiling learned density functions. The weighting factor can be used to vary the learning rate. When $w = 0$, the learning effect is disabled and the measured histograms are used by the predictor. As w increases toward one, the learning effect increases and the measured density functions have a

reduced effect. A small w should be used when the background is rapidly changing; however, when the background is relatively stationary, w can be increased to obtain a more stable estimate of the density functions.

The predictor provides several important features for the tracking problem. First, the predictor provides a better estimate of the density functions in a rapidly changing scene which may be caused by background change or sunglare problems. Secondly, the predictor allows the camera to have an automatic gain control to improve the target separation from the background.

With the learned density functions for the background, plume, and target features $(H_i^B(x), H_i^P(x), H_i^T(x))$, a Bayesian classifier [11] can be used to decide whether a given feature x is a background, plume, or target point. Assuming equal *a priori* probabilities and equal misclassification costs, the classification rule decides that a given pixel feature x is a background pixel if

$$H_i^B(x) > H_i^T(x) \quad \text{and} \quad H_i^B(x) > H_i^P(x),$$

a target pixel if

$$H_i^T(x) > H_i^B(x) \quad \text{and} \quad H_i^T(x) > H_i^P(x),$$

or a plume pixel if

$$H_i^P(x) > H_i^B(x) \quad \text{and} \quad H_i^P(x) > H_i^T(x).$$

The results of this decision rule are stored in a high-speed classification memory during the vertical retrace period. With the pixel classification stored in the classification memory, the real-time pixel classification is performed by simply letting the pixel intensity address the classification memory location containing the desired classification. This process can be performed at a very rapid rate with high-speed bipolar memories.

Projection Processor

The video processor described above separates the target image from the background and generates a binary picture, where target presence is represented by a "1" and target absence by a "0." The target location, orientation, and structure are characterized by the pattern of 1 entries in the binary picture matrix, and the target activity is characterized by a sequence of picture matrices. In the projection processor, these matrices are analyzed field-by-field at 60 fields/s using projection-based classification algorithms to extract the structural and activity parameters needed to identify and track the target.

The targets are structurally described and located by using the theory of projections. A projection in the x-y plane of a picture function $f(x, y)$ along a certain direction w onto a straight line z perpendicular to w is defined by

$$P_w(z) = \int f(x, y)\, dw$$

as shown in Fig. 3. In general, a projection integrates the intensity levels of a picture along parallel lines through the pattern, generating a function called the projection. For binary

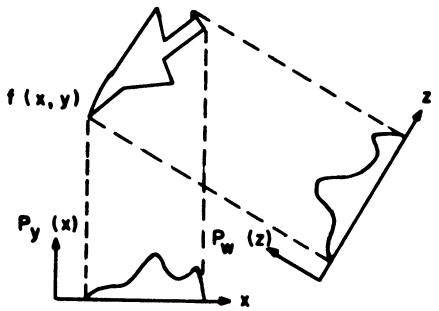


Fig. 3. Projections.

digitized patterns, the projection gives the number of object points along parallel lines; hence, it is a distribution of the target points for a given view angle.

It has been shown that for sufficiently large numbers of projections a multigray level digitized pattern can be uniquely reconstructed [12]. This means that structural features of a pattern are contained in the projections. The binary input simplifies the construction of projections and eliminates interference of structural information by intensity variation within the target pattern; consequently, fewer projections are required to extract the structural information. In fact, any convex, symmetric binary pattern can be reconstructed by only two orthogonal projections, proving that the projections do contain structural information.

Much research in the projection area has been devoted to the reconstruction of binary and multigray level pictures from a set of projections, each with a different view angle. In the real-time tracking problem, the horizontal and vertical projections can be rapidly generated with specialized hardware circuits that can be operated at high frame rates. Although the vertical and horizontal projections characterize the target structure and locate the centroid of the target image, they do not provide sufficient information to precisely determine the orientation of the target. Consequently, the target is dissected into two equal areas and two orthogonal projections are generated for each area.

To precisely determine the target position and orientation, the target center-of-area points are computed for the top section (X_c^T, Y_c^T) and bottom section (X_c^B, Y_c^B) of the tracking parallelogram using the projections. Having these points, the target center-of-area (X_c, Y_c) and its orientation can be easily computed (Fig. 4):

$$X_c = \frac{X_c^T + X_c^B}{2}$$

$$Y_c = \frac{Y_c^T + Y_c^B}{2}$$

$$\phi = \tan^{-1} \frac{Y_c^T - Y_c^B}{X_c^T - X_c^B}.$$

The top and bottom target center-of-area points are used, rather than the target nose and tail points, since they are much easier to locate, and more importantly, they are less sensitive to noise perturbations.

It is necessary to transform the projection functions into

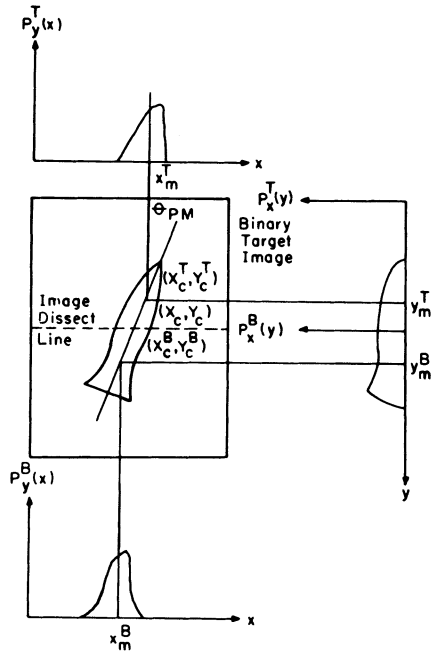


Fig. 4. Projection location technique.

a parametric model for structural analysis. Area quantization offers the advantage of easy implementation and high immunity to noise. This process transforms a projection function $P_w(z)$ into k rectangles of equal area (Fig. 5), such that

$$\int_{Z_i}^{Z_{i+1}} P_w(z)\,dz = \frac{1}{k}\int_{Z_1}^{Z_{k+1}} P_w(z)\,dz$$

$$\text{for } i = 1, 2, \cdots, k.$$

Another important feature of the area quantization model for a projection function of an object is that the ratio of line segments $l_i = Z_{i+1} - Z_i$ and $L = Z_k - Z_2$,

$$S_i = \frac{l_i}{L} \quad \text{for } i = 2, 3, \cdots, k-1$$

are object size invariant. Consequently, these parameters provide a measure of structure of the object which is independent of size and location [13]. In general, these parameters change continuously since the projections are one-dimensional representations of a moving object. Some of the related problems of these geometrical operations are discussed by Johnston and Rosenfeld [14].

The structural parameter model has been implemented and successfully used to recognize a class of basic patterns in a noisy environment. The pattern class includes triangles, crosses, circles, and rectangles with different rotation angles. These patterns are chosen because a large class of more complex target shapes can be approximated with them.

The architecture of the projection processor consists of a projection accumulation module (PAM) for accumulating the projections and a microprogrammable processor for computing the structural parameters. The binary target picture enters the PAM as a serial stream in synchronization

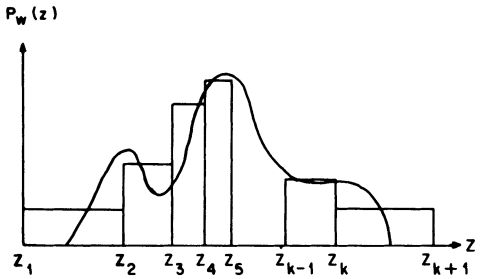


Fig. 5. Projection parameters.

with the pixel classifier of the video processor. The projections are formed by the PAM as the data are received in real time. In the vertical retrace interval, the projection processor assumes addressing control of the PAM and computes the structural parameters before the first active line of the next field. This allows the projections to be accumulated in real time, while the structural parameters are computed during the vertical retrace interval.

Tracker Processor

In the tracking problem, the input environment is restricted to the image in the FOV of the tracking optics. From this information, the tracking processor extracts the important inputs, classifies the current tracking situation, and establishes an appropriate tracking strategy to control the tracking optics for achieving the goals of the tracking system.

The state concept can be used to classify the tracking situations in terms of state variables as in control theory, or it can be interpreted as a state in a finite state automaton [15], [16].

Some of the advantages of the finite state automaton approach are as follows.

1) A finite state automaton can be easily implemented with a look-up table in a fast LSI memory.

2) A finite state automaton significantly reduces the amount of information to be processed.

3) The tracking algorithm can be easily adjusted to different tracking problems by changing the parameters in the look-up table.

4) The finite state automaton can be given many characteristics displayed by human operators.

The purpose of the tracker processor is to establish an intelligent tracking strategy for adverse tracking conditions. These conditions often result in losing the target image within or out of the FOV. When the target image is lost within the FOV, the cause can normally be traced back to rapid changes in the background scene, rapid changes in the target image due to sun glare problems, or cloud formations that obstruct the target image. When the target image is lost by moving out of the camera's FOV, the cause is normally the inability of the tracking optics dynamics to follow a rapid motion of the target image. It is important to recognize these situations and to formulate an intelligent tracking strategy to continue tracking while the target image is lost so that the target image can be reacquired after the disturbance has passed.

To establish an intelligent tracking strategy, the tracker processor evaluates the truthfulness and trackability of the track-

ing data. The truthfulness of the tracking data relates to the confidence that the measured tracking data truly define the location of the target under track. The trackability of the target image relates to the question of whether the target image has desirable tracking properties.

The inputs to the tracker processor are derived from the projection representation of the target image by the projection processor. Area quantization is used to transform each projection function $P(z)$ into $K = 8$ equal area intervals as shown in Fig. 5.

These inputs are:

1) target size (TSZ);
2) target location (TX, TY);
3) target orientation ($T\phi$);
4) target density (TDN);
5) target shape = $\{(SX_i, SY_i) | i = 1, 2, \cdots, 6\}$.

Target size is simply the total number of target points. Target location is given by the center-of-area points of the projections. Where X_i and Y_i are the parameters of Fig. 5 when projected on the x and y axes, respectively,

$$TX = X_5 \quad \text{and} \quad TY = Y_5.$$

The target orientation defines the orientation of the target image with respect to vertical boresight. Target density is derived from the target length (TL), width (TW), and size (TSZ) by

$$TDN = \frac{TL \times TW}{TSZ}.$$

The target shape is described by the ratio of the lengths of the equal area rectangles and the total lengths

$$SX_i = (X_{i+2} - X_{i+1})/(X_8 - X_2)$$

and

$$SY_i = (Y_{i+2} - Y_{i+1})/(Y_8 - Y_2)$$

for $i = 1, 2, \cdots, 6$. Observe that the first and last equal area subintervals are not used in the shape description, since they are quite sensitive to noise.

The tracker processor establishes a confidence weight for its inputs, computes boresight and zoom correction signals, and controls the position and shape of the target tracking window to implement an intelligent tracking strategy. The outputs of the tracker processor are as follows.

Outputs to Control Processor: 1) Target X displacement from boresight (DX), 2) target Y displacement from boresight (DY), 3) desired change in zoom (DZ), 4) desired change in image rotation ($D\phi$), and 5) confidence weight (W).

Outputs to Video Processor: 1) tracking window size, 2) tracking window shape, and 3) tracking window position.

The outputs to the control processor are used to control the target location and size for the next frame. The boresight correction signals are used to control the azimuth and elevation pointing angles of the telescope. The desired zoom is used to control the zoom lens, keeping the target visible within the FOV. The desired image rotation controls the image rotation element to keep the target image vertical. The confidence weight is used by the control processor much like a Kalman weight to combine the measured and predicted values. When the confidence weight is low, the control processor relies more heavily on the recent trajectory to predict the location of the target on the next frame.

The outputs to the video processor define the size, shape, and position of the tracking window. These are computed on the basis of the size and shape of the target image and the amount of jitter in the target image location. There is no loss in resolution when the tracking window is made larger; however, the tracking window acts like a bandpass filter and rejects unwanted noise outside the tracking window.

A confidence weight is computed from the structural features of the target image to measure the truthfulness of the input data. The basic objective of the confidence weight is to recognize false data caused by rapid changes in the background scene or cloud formations. When these situations are detected, the confidence weight is reduced and the control processor relies more heavily on the previous tracking data to orientate the tracking optics toward the target image. This allows the control processor to continue tracking the target so that the target image can be reacquired after the perturbation passes.

The confidence weight measures how well the structural features of the located object fit the target image being tracked. A linear recursive filter is used to continually update the structural features to allow the algorithm to track the desired target through different spatial perspectives. Experimental studies have indicated that the structural parameters $S = \{(SX_i, SY_i) | i = 1, 2, \cdots, 6\}$ and the target density are important features in detecting erratic data. Let $TDN(k)$ and $(SX_i(k), SY_i(k))$ for $i = 1, 2, \cdots, 6$ represent the measured target density and shape parameters, respectively, for the kth field, and let $(\overline{SX}_i(k), \overline{SY}_i(k))$ represent the filtered values for the target shape parameters. The linear filter is defined by

$$\overline{SX}_i(k+1) = \frac{(k-1)\,\overline{SX}_i(k) + SX_i(k)}{K}$$

$$\overline{SY}_i(k+1) = \frac{(k-1)\,\overline{SY}_i(k) + SY_i(k)}{K}$$

for $i = 1, 2, \cdots, 6$ and a positive integer K. The confidence weight for the kth field is given by

$$W(k) = \alpha_1 \max\{1 - C(k), 0\} + \alpha_2 \min\{1, TDN(k)\}$$

where

$$C(k) = \sum_{i=1}^{6} SX_i(k) - \overline{SX}_i(k) + \sum_{i=1}^{6} SY_i(k) - \overline{SY}_i(k)$$

and

$$0 \leqslant \alpha_1, \quad \alpha_2 \leqslant 1, \quad \alpha_1 + \alpha_2 = 1.$$

This formulation for the confidence weight has been experimentally tested, and it has demonstrated the ability to measure the truthfulness of the tracking data in the tracking environment. The filtered shape parameters are not updated

when the confidence weight falls below a given lower threshold. This prevents the shape parameters from being updated incorrectly during periods when the target image is lost or badly perturbed by the background noise.

To formulate an intelligent tracking strategy, the tracking algorithm needs the ability to respond to a current input based upon the sequence of inputs that lead to the current state (or situation). A finite state sequential machine possesses such a property because each state represents the collection of all input sequences that take the machine from the initial state to the present state (Nerode's tape equivalence [17]). By defining an equivalence relation R on the tape set as

$$XRy \quad \text{if} \quad \delta(s_0, X) = \delta(s_0, y) \quad \forall x,\ y \in \sum *$$

the tape set $\Sigma *$ can be partitioned into equivalent classes

$$[x] = s_i = \{y \mid xRy \ \forall y \in \sum *\}.$$

Consequently, a state represents all input sequences that produce a given tracking situation. This interpretation of input sequences transforms the development of the tracking algorithm into a problem of defining a finite state sequential machine

$$TA = (S, I, Z, \delta, W).$$

The states of the machine $S = \{s_i, s_2, s_3, \cdots, s_n\}$ define the different tracking situations that must be handled by the tracking algorithm. The inputs to the finite state machine are derived from the image parameters that characterize the size, shape, and location of the present target image. The output set Z defines a finite set of responses that the tracking algorithm employs for maintaining track and retaining high resolution data. The next state mapping δ: $S \times I \rightarrow S$ defines the next state $\delta(s_i, i_j) = s_k$ when an input i_j is applied to state s_i. The output mapping W: $S \rightarrow Z$ is a Moore output that defines the proper tracking strategy (response) for each state.

The inputs to the sequential machine are chosen to give the sequential machine a discrete measure of the trackability of the target image. These inputs are:

1) target size (TSZ);
2) confidence weight (W);
3) image orientation ($T\phi$);
4) image displacement from boresight (TX, TY);
5) image movement (ΔTX, ΔTY);
6) rate of change in target size (ΔTSZ);
7) rate of change in confidence weight (ΔW).

The image size is clearly an important measure of the trackability since the image is difficult to track when the image is either too small or too large. The confidence weight provides a measure of whether the image is the target under track. Image displacement from boresight gives a measure of whether the image is leaving the FOV. Image movement in the FOV measures the amount of jitter in the target image. The rate of change in the target size and confidence weight allows a prediction of what is likely to happen on subsequent fields. These inputs are quantized into an 8-bit binary input vector for the sequential machine.

The states of the sequential machine are chosen to define the major tracking situations. These states are:

1) target acquisition = S_1;
2) normal tracking = S_2;
3) abrupt change in FOV = S_3;
4) leaving or out of FOV = S_4.

S_1 corresponds to the situation where the tracker is trying to acquire the target. This situation occurs during initial acquisition and during periods of time when the target image is lost. S_2 corresponds to the situation where the target image is under track with the normal tracking algorithm and no special tracking strategy is required. S_3 corresponds to the situation where the target image undergoes erratic and abrupt changes in its shape or size within the FOV and S_4 corresponds to the situation where the target image is leaving or has left the FOV of the camera. These states only categorize the major tracking situations. State parameters are used to further refine the tracking situation by providing a parametric description of the tracking situation.

The tracking algorithm views the target structure and activity within the tracking window on a field-by-field basis. Based upon these observations, the tracking algorithm must classify the present tracking situation and enter the appropriate state whose outputs best handle the situation. The motivation of the next state mapping, δ: $I \times S \rightarrow S$, is to provide a tracking strategy to keep the tracker in the normal tracking state. Much experimental effort has been devoted to establish a next state mapping with desirable tracking characteristics. A simplified state diagram is given in Fig. 6 which defines the essential properties of the state transition behavior.

The output set Z defines a finite set of responses that the tracking algorithm can make to maintain track while retaining high resolution data. Several of these responses control the tracking window, and they are performed at electronic speed. However, the zoom lens and tracking optics operations are mechanical in nature and require more time to be performed.

The output set Z contains the following responses:

1) location of tracking window;
2) shape and size of tracking window;
3) target location (boresight corrections);
4) confidence weight;
5) desired zoom;
6) control shape parameter update procedure.

These output responses give the tracking algorithm the ability to respond to the tracking situations defined by tracking states.

The motivation for the output mapping, W: $I \times S \rightarrow Z$, is to associate with each state an intelligent strategy to maintain track with high image resolution. The outputs to the image decomposition algorithm control the size and position of the tracking window. The outputs to the control processor control the zoom and pointing angle of the tracking optics. The tracking algorithm must make many compromises between

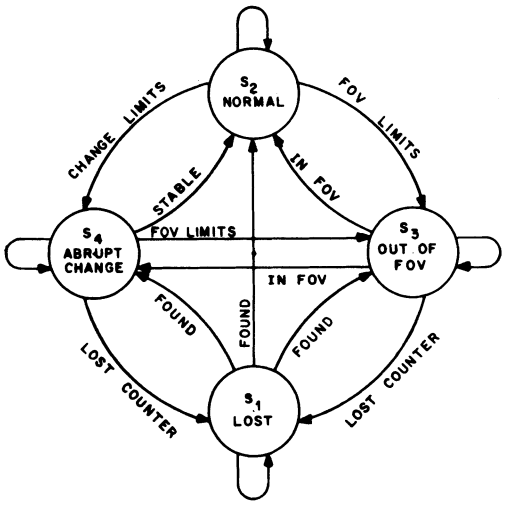


Fig. 6. Next-state mapping.

tracking ability and image resolution. For example, when the target image is about to leave the FOV, the zoom can be reduced to keep the target in the FOV. Increasing the FOV, however, decreases the target size and image resolution. These compromises have been formulated and solved with standard optimization techniques. The solutions are precomputed and stored in a look-up table for the tracking algorithm.

By implementing the tracking algorithm as a finite state machine where the outputs are obtained from the look-up tables, it is possible to realize the tracking algorithm with a standard microprogrammable processor.

Control Processor

The purpose of the control processor is to generate the control signals that drive the tracking optics. The processor receives inputs from the tracker processor on the target location (TX, TY), orientation $(T\phi)$, and length (TL), measured relative to boresight and on a confidence weight (W) that measures the reliability of the measured data. Using these inputs, the control processor predicts the desired tracker azimuth and elevation pointing angles $(\Theta_A(k+1), \Theta_E(k+1))$, the zoom lens setting $(Z(k+1))$ and the image rotation setting $(\phi(k+1))$ for the next field denoted by $(k+1)$.

The control processor provides the estimated and predicted values of the variables necessary to orientate the tracking optics toward the target. Using the confidence weight to combine the predicted variables with current measurements, the processor provides estimates of current target position variables. These estimates are combined with previous estimates to predict the target position variables for the next control interval.

Since the form of the estimator equations is similar for the azimuth, elevation, zoom, and image rotation variables, the following notation is used in the development:

$\Theta(k)$ variable to be estimated or predicted (azimuth, elevation, rotation, or zoom);
$\Theta_m(k)$ measured values of $\Theta(k)$;
$\hat{\Theta}(k)$ estimated value of $\Theta(k)$ using measurements through kth frame;
$\hat{\Theta}_j(k+1|k)$ predicted value of $\Theta(k+1)$ using measurements through kth frame and the j index type predictor;
$\hat{\Theta}(k+1|k)$ predicted value of $\Theta(k+1)$ using a combination of the set of predictors;
$w(k)$ confidence weight.

Using the confidence weight in a manner similar to that of a Kalman gain, the estimated value is obtained from the measured and predicted values by

$$\hat{\Theta}(k) = (1 - W(k))\,\hat{\Theta}(k|k-1) + W(k)\,\Theta_m(k)$$

where the confidence weight is normalized such that $0 \leqslant W(k) \leqslant 1$. Consequently, the predictors having the functional form

$$\hat{\Theta}(k+1|k) = F(\hat{\Theta}(k), \hat{\Theta}(k-1), \cdots, \hat{\Theta}(k-r))$$

rely more heavily on the estimated values when the weight is low. This is very important for continuing track when the target passes through clouds or noisy backgrounds such as the mountain/sky interface.

A simple filter-predictor that has demonstrated the ability to track highly maneuverable targets uses a linear convex combination of a linear-two-point and a quadratic-five-point polynomial filter-predictor. This scheme is based on the reasoning that the linear-two-point polynomial can better anticipate a rapid maneuver from the missile, while the quadratic-five-point polynomial is used to reduce the variance of error. The coefficients of the linear convex combination are chosen to obtain an unbiased minimum variance of error estimate.

In formulating the linear convex combination estimate of the linear $(\hat{\Theta}_l)$ and quadratic $(\hat{\Theta}_q)$ predictors

$$\hat{\Theta} = \alpha_1 \hat{\Theta}_l + \alpha_2 \hat{\Theta}_q,$$

it is assumed that the predictors $\hat{\Theta}_l$ and $\hat{\Theta}_q$ are independent; hence, the minimum variance error estimate [18] is given by the coefficients

$$\alpha_1 = \frac{\sigma^2(\hat{\Theta}_q)}{[\sigma^2(\hat{\Theta}_l) + \sigma^2(\hat{\Theta}_q)]}$$

$$\alpha_2 = \frac{\sigma^2(\hat{\Theta}_l)}{[\sigma^2(\hat{\Theta}_l) + \sigma^2(\hat{\Theta}_q)]}$$

and

$$\hat{\Theta}(t_{k+1}|t_k) = \frac{\sigma^2(\hat{\Theta}_q)\,\hat{\Theta}_l(t_{k+1}|t_k) + \sigma^2(\hat{\Theta}_l)\,\hat{\Theta}_q(t_{k+1}|t_k)}{\sigma^2(\hat{\Theta}_l) + \sigma^2(\hat{\Theta}_q)}$$

is the desired prediction of the chosen parameter.

Simulation studies [19] were conducted to compare the performance of this simple and somewhat primitive filter-predictor with the more robust extended Kalman filter [20]. The results of these studies indicated the total performance of the tracking system with the simple filter-predictor compared favorably with the extended Kalman filter. In the light of the real-time computational constraints, the simple filter-predictor is suitable for more real-time tracking problems.

Simulation and Hardware

A computer simulation of the RTV tracking system, incorporating the algorithms used by the four processors described in this paper, was developed and used to evaluate the system under realistic tracking conditions. The simulation model includes dynamic models for the target trajectory and the tracking optics and mount in addition to the RTV processor algorithms in order to verify the complete tracking system.

The simulation displays the target and plume points as they are identified by the video processor. These points are superimposed on a cross hair which locates the boresight of the tracking optics. In addition to the digitized images, the simulation displays the target and plume tracking windows and the projections accumulated for each video field. These outputs provide a direct measure of the performance of the four processors as well as an overall measure of the tracking accuracy of the system.

Two types of input data are available to the simulation—simulated digitized video fields and actual digitized video fields from video tape recordings of typical tracking sequences. Fig. 7 contains two representative simulation outputs selected from the first 100 fields of simulated digitized video. The RTV tracker performs well during this simulated tracking sequence. All processors function properly, and track is maintained throughout the tracking sequence.

Fig. 8(a) is a halftone graphics display of a field of actual digitized video which was injected into the RTV simulation. The simulator repeatedly processed the same video field superimposed on the simulated target trajectory in order to test the static and dynamic responses of the RTV processors. The result after six processing frames, shown in Fig. 8(b), verifies the effectiveness of the processing algorithms used in the RTV tracker. Since no plume is present, both windows are tracking the target. Several important conclusions may be deduced from this result: the video processor successfully identifies most of the target pixels; the projection data is accumulated and used effectively by the projection and tracker processors to obtain the target orientation and the displacement of the target from boresight; and both windows are tracking the target and closing down on it to reduce noise.

Subsequent to simulation, a hardware model was developed and is being deployed at the present time. Bench tests of all subsystems indicate that the projected performance of each processor has been achieved.

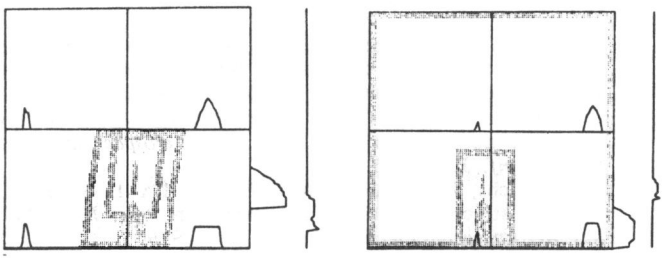
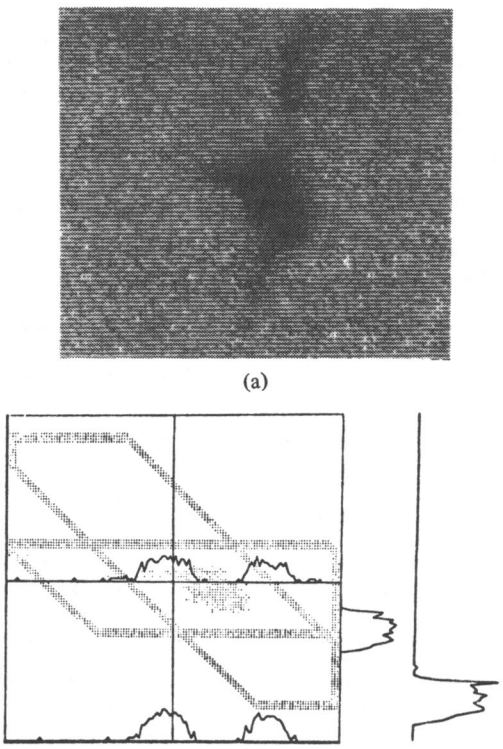


Fig. 7. Simulation outputs.



(a)

(b)

Fig. 8. (a) Digitized video. (b) Simulation output.

Conclusions

These four subsystems form only a part of the RTV; many of the other subsystems are of equal complexity. Changes currently taking place in digital technology coupled with aggressive research in real-time pattern recognition/image processing algorithms are now making possible highly sophisticated hardware for recognition and tracking purposes. While RTV is an example of this type of a system, work at WSMR and NMSU (and elsewhere) is continuing with the purpose of finding faster and better ways of processing video intelligently.

A lot of work remains before a truly versatile video analysis system exists. RTV is but a step in that direction. Research directed toward the solution of this class of problems needs to have two fundamental premises as a starting point. The first is that algorithms should be general, not specifically created for a particular scene. Too often pattern recognition methods have depended upon extensive "tweaking" of parameters to yield good results. This approach is not allowable for real-time processing. Secondly, the algorithms should not require extensive data storage, since manipulation of the data consumes resources that exceed those available for most applications.

As is evidenced by RTV, progress can be made; the future of video processing is bright. The applications in the nonmilitary community possibly exceed the military applications in importance. RTV is a forerunner of a large class of intelligent video processing machines that will perform a wide variety of tasks of importance to the human family.

Acknowledgment

Important contributions to this project were made by a number of persons not named herein, including faculty and students at universities and staff members at WSMR. The project has enjoyed many participants and cosponsors whose contributions are acknowledged.

REFERENCES

[1] P. I. Perez, "A multiprocessor architecture with a master shared control store," Ph.D. dissertation, New Mexico State Univ., 1978 (available from University Microfilms, Ann Arbor, MI).

[2] G. M. Flachs and A. L. Gilbert, "Automatic boresight correction in optical tracking," in *Proc. 1975 Nat. Aerospace and Electronics Conf.*, June 1975.

[3] G. M. Flachs, W. E. Thompson, Y. H. U, and A. L. Gilbert, "A real-time structural tracking algorithm," in *Proc. IEEE 1976 Nat. Aerospace and Electronics Conf.*, May 1976.

[4] G. M. Flachs and J. A. Vilela, "A structural model for polygon patterns," in *Proc. IEEE 1976 Nat. Aerospace and Electronics Conf.*, May 1976.

[5] G. M. Flachs, W. E. Thompson, J. M. Taylor, W. Cannon, R. B. Rogers, and Y. H. U, "An automatic video tracking system," in *Proc. 1977 Nat. Aerospace and Electronics Conf.*, May 1977.

[6] R. B. Rogers and G. M. Flachs, "Mathematical modeling and simulation in a programmed design methodology," in *Proc. 1st Int. Conf. on Mathematical Modeling*, Aug. 1977.

[7] A. L. Gilbert and M. K. Giles, "Novel concepts in real-time optical tracking," in *Proc. Army Sci. Conf.*, June 1978.

[8] N. J. Nilsson, *Learning Machines.* New York: McGraw-Hill, 1965, ch. 4.

[9] H. L. Andrews, *Introduction to Mathematical Techniques in Pattern Recognition.* New York: Wiley-Interscience, 1972, pp. 148–150.

[10] B. P. Lathi, *Random Signals and Communication Theory.* Scranton, PA: International Textbook, 1968, pp. 275–279.

[11] I. Selin, *Detection Theory.* Princeton, NJ: Princeton Univ. Press, 1965, pp. 11–13.

[12] S. K. Chang, "The reconstruction of binary patterns from their projections," *Commun. ACM*, vol. 14, Jan. 1971.

[13] Y. H. U, "Projection theory for real-time vision tracking," Ph.D. dissertation, New Mexico State Univ., Aug. 1978 (available from University Microfilms, Ann Arbor, MI).

[14] E. G. Johnston and A. Rosenfeld, "Geometrical operations on digitized pictures," *Picture Processing and Psychopictorics*, B. S. Lipkin and R. Rosenfeld, Eds. New York: Academic, 1970.

[15] R. Romovic and R. B. McGhee, "A finite static approach to the synthesis of bioengineering control systems," *IEEE Trans. Hum. Factors Electron.*, vol. HFE-7, pp. 65–69, June 1966.

[16] E. S. Angel and G. A. Bekey, "Adaptive finite state models of manual control systems," *IEEE Trans. Man-Mach. Syst.*, pp. 15–20, Mar. 1968.

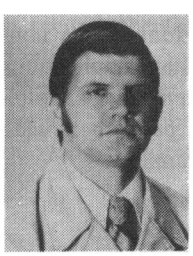
[17] T. L. Booth, *Sequential Machines and Automata Theory.* New York: Wiley, 1968.

[18] A. Papoulis, *Probability Random Variables and Stochastic Processes.* New York: McGraw-Hill, 1965, pp. 385–426.

[19] W. E. Thompson, G. M. Flachs, and T. R. Kiang, "Evaluation of filtering and prediction techniques for real-time video tracking of high performance targets," *Trans. 1978 Nat. Aerospace and Electronics Conf.*, June 1978.

[20] C. B. Chang, R. H. Whitney, and M. Atham, "Application of adaptive filtering methods to maneuvering trajectory estimation," Lincoln Lab. Tech. Note 1975-59, Nov. 1975.

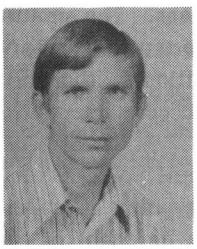
[21] K. E. Iverson, *A Programming Language.* New York: Wiley, 1962.



Alton L. Gilbert (S'68–M'73) was born in Elmira, NY, on April 13, 1942. He received the B.S.E.E., M.S.E.E., and Sc.D. degrees, all from the New Mexico State University, Las Cruces, in 1970, 1971, and 1973, respectively.

From 1961 to 1968 he served with the U.S. Navy in the Polaris missile program. From 1973 to the present he has been a Researcher and Research Manager with the Advanced Technology Office of the Instrumentation Directorate at U.S. Army White Sands Missile Range, NM. As Principle Investigator and Manager of the Real-Time Videotheodolite program, he has become widely known for his interests in video processing methods.

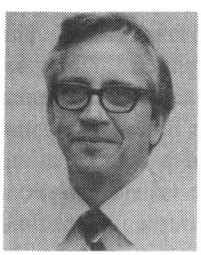
Dr. Gilbert is a member of the Technical Review Committee of the Joint Services Electronics Program, of the Electronics Coordinating Group for the U.S. Army, the Information Theory Group of the IEEE, Tau Beta Pi, Eta Kappa Nu, Phi Kappa Phi, and Sigma Xi.



Michael K. Giles was born in Logan, UT, on October 24, 1945. He received the B.E.S. and M.S.E.E. degrees from Brigham Young University, Provo, UT, in 1971, and the Ph.D. degree from the University of Arizona, Tucson, in 1976.

He was an Electronics Engineer with the U.S. Naval Weapons Center, China Lake, CA, from 1971 to 1977, where he specialized in the design and evaluation of electrooptical and optical systems. At China Lake he developed low-noise photodiode preamplifiers, photoparametric amplifiers, and rapidly tunable dye lasers for Navy applications. Since 1977 he has been with the Instrumentation Directorate, U.S. Army White Sands Missile Range, NM, where he is applying image-processing and pattern-recognition techniques to the development of real-time optical tracking systems. His research interests also include optical and electrooptical image and signal processing.

Dr. Giles is a member of the Optical Society of America, the Society of Photo-Optical Instrumentation Engineers, Tau Beta Pi, and Eta Kappa Nu.



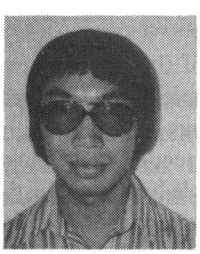
Gerald M. Flachs (S'68–M'68) received the B.S., M.S., and Ph.D. degrees, all from Michigan State University, East Lansing.

He is a Professor of Electrical and Computer Engineering at New Mexico State University, Las Cruces. His current research interests include image-processing techniques, distributive computer systems, and digital system design with microprogrammable processors.

Robert B. Rogers (M'77) was born in China Lake, CA on April 13, 1952. He received the B.S. degree in physics from the New Mexico Institute of Mining and Technology (NMIMT), Socorro, in 1973, and the M.S. and Ph.D. degrees in electrical engineering from New Mexico State University (NMSU), Las Cruces, in 1976 and 1978, respectively.

While at NMIMT, he performed research on cross-spectral analysis of thunder and lightning channel reconstruction. He is now at NMSU, where his primary research area is real-time video filtering with bit-slice microprogrammable processor systems. He is currently with the New Mexico Solar Energy Institute's Modeling and Analysis Section at NMSU.

Dr. Rogers is a member of the American Geophysical Union and the IEEE Computer Society.



Yee Hsun U (S'72–M'79) was born in Hong Kong, China, on May 30, 1949. He received the B.S.E.E. degree from Purdue University, West Lafayette, IN, in 1972, and the M.S.E.E. and the Ph.D. degrees in electrical engineering from New Mexico State University, Las Cruces, in 1974 and 1978, respectively.

From 1976 to 1978, he was a staff engineer with the Real-Time Video Tracker project at New Mexico State University. He joined Texas Instruments Incorporated, Dallas, in September 1978 as a member of technical staff of the Corporate Research, Development & Engineering Division. His research interests are in real-time tracking, vision systems, machine intelligence, and the application of pattern recognition to industrial automation.

A Real-Time Video Tracking System

5 Author(s)

Alton L. Gilbert ; Michael K. Giles ; Gerald M. Flachs ; Robert B. Rogers ; U Yee Hsun

View All Authors

Abstract:
Object identification and tracking applications of pattern recognition at video rates is a problem of wide interest, with previous attempts limited to very simple threshold or correlation (restricted window) methods. New high-speed algorithms together with fast digital hardware have produced a system for missile and aircraft identification and tracking that possesses a degree of ``intelligence'' not previously implemented in a real-time tracking system. Adaptive statistical clustering and projection-based classification algorithms are applied in real time to identify and track objects that change in appearance through complex and nonstationary background/foreground situations. Fast estimation and prediction algorithms combine linear and quadratic estimators to provide speed and sensitivity. Weights are determined to provide a measure of confidence in the data and resulting decisions. Strategies based on maximizing the probability of maintaining track are developed. This paper emphasizes the theoretical aspects of the system and discusses the techniques used to achieve real-time implementation.

Published in: IEEE Transactions on Pattern Analysis and Machine Intelligence (Volume: PAMI-2, Issue: 1, Jan. 1980)

Page(s): 47 - 56

DOI: 10.1109/TPAMI.1980.4766969

Date of Publication: 27 January 2009

Publisher: IEEE

ISSN Information:

Sponsored by: IEEE Computer Society

Keywords

IEEE Keywords
Real time systems, Pattern recognition, Clustering algorithms, Missiles, Video compression, Optical imaging, Image storage, Hardware, Aircraft, Adaptive systems

Author Keywords
video tracking, Image processing, intensity histograms, object identification, optical tracking, projections, tracking system, video data compression, video processing

Authors

Alton L. Gilbert
MEMBER, IEEE, U.S. Army White Sands Missile Range, White Sands, NM 88002.

Michael K. Giles
U.S. Army White Sands Missile Range, White Sands, NM 88002.

Gerald M. Flachs
MEMBER, IEEE, Department of Electrical Engineering, New Mexico State University, Las Cruces, NM 88003.

Robert B. Rogers
MEMBER, IEEE, Department of Electrical Engineering, New Mexico State University, Las Cruces, NM 88003.

U Yee Hsun
MEMBER, IEEE, Department of Electrical Engineering, New Mexico State University, Las Cruces, NM 88003; Texas Instruments Incorporated, Dallas, TX 75222.

Related Articles

Learning and optimization using the clonal selection principle
L.N. de Castro; F.J. Von Zuben

30 years of adaptive neural networks: perceptron, Madaline, and backpropagation
B. Widrow; M.A. Lehr

Real-time American sign language recognition using desk and wearable computer based video
T. Starner; J. Weaver; A. Pentland

Misconceptions about real-time computing: a serious problem for next-generation systems
J.A. Stankovic

The possibilistic C-means algorithm: insights and recommendations
R. Krishnapuram; J.M. Keller

The synchronous approach to reactive and real-time systems
A. Benveniste; G. Berry

Design of embedded systems: formal models, validation, and synthesis
S. Edwards; L. Lavagno; E.A. Lee; A. Sangiovanni-Vincentelli

A heuristic fuzzy logic approach to EMG pattern recognition for multifunctional prosthesis control
A.B. Ajiboye; R.Fff. Weir

A survey of fuzzy clustering algorithms for pattern recognition. I
A. Baraldi; P. Blonda

Real-time range acquisition by adaptive structured light
T.P. Koninckx; L. Van Gool

Exhibit B

COMPUTER ARCHITECTURE AND IMPLEMENTATION OF VISION-BASED REAL-TIME LANE SENSING

O. D. Altan, H. K. Patnaik, R. P. Roesser
General Motors Research Laboratories
Vehicle Systems Research Department
30500 Mound Road
Warren, MI 48090-9055
Phone: (313) 986-9100 Fax: (313) 986-3003

INTRODUCTION

Lane sensing is the detection of lane boundaries from a video image of the roadway and the determination of the vehicle's position and orientation with respect to the lane. The detection of lane boundaries involves the digital processing of a roadway image to bring out points comprising the lane edges and a search of the processed images to find those points. The determination of the vehicle-lane relationship consists of estimating vehicle offset, heading and lane curvature from the detected lanes and vehicle measurements. Lane sensing supports several other functions, including collision warning, collision avoidance, driving performance monitoring and lane control.

An algorithm that was developed at G.M. Research Laboratories [1] was specifically tailored for the purpose of demonstration on a vehicle in real-time. It involves generating two processed images that are separately searched for lane boundary points. The first processed image is generated by computing a local 5-by-5 pixel average of the intensity-normalized input image. The second processed image is generated by applying a Sobel edge operation [2] to the intensity-normalized image and then computing a local 5-by-5 intensity average. The first processed image is intended to bring out points from well marked lane boundaries, while the second image is intended to bring out transitions at the lane boundaries when they are not well marked.

This lane-detection algorithm was developed on a mainframe computer operating in non-real-time with images taken from a video tape of highway scenes. Numerous heuristic techniques have been incorporated to handle particular situations such as merging lanes, shadows, missing markers, and false targets. Limited attention was given to real-time implementation considerations.

A second algorithm was developed to estimate lane curvature, vehicle offset with respect to the center of the lane and vehicle heading. The algorithm uses information obtained from lane detection and from measurements of vehicle speed and steering angle. The estimation algorithm is in the form of a Kalman filter using linear models of both the vehicle dynamics and the road, plus camera and vehicle geometry. This estimation algorithm was also developed on a mainframe computer.

This paper describes the implementation of these two algorithms to achieve lane sensing in real-time on board a vehicle. This required developing the computer architecture, electronics and software to perform the processing steps in the algorithms. Attention was given to multi-tasking, multi-processing and interprocess communication. In particular, use of the VME bus [3] provides an open architecture and the disc-based real-time operating system [4] provides multitasking and a flexible environment.

The lane sensing system was installed in a vehicle and the real-time implementation was demonstrated on a test track and limited access highways.

COMPUTATIONAL REQUIREMENTS

The computational requirements of the real-time system are determined by the required processing time and latency. According to initial studies, the processing time or the time elapsed between the starting points of two successive process cycles should be no more than 0.2 seconds. Also, the latency or the time from the point an input image is sampled to the point that the processed results become available should be no more than 0.5 seconds.

The camera produces an image frame in approximately 33 milliseconds. Each image frame is digitized such that there are 512 lines with 512 pixels in each line. This is equivalent to over a quarter of a million pixels in a frame to be processed, each represented by 8 bits. A histogram measurement and several filtering operations are performed concurrently, as well as table look-ups and general arithmetic operations. The filtering operations involve manipulating pixels in the neighborhood of a given pixel. Most of the computation is in the form of simple integer arithmetic operations, but the amount of data to be processed is quite large. The total computation rate required is approximately 900 million arithmetic operations per second. This rate is equal to the computation rate of several supercomputers, although it involves low-precision integer arithmetic. The images are processed by special-purpose boards optimized for specific image processing functions. These boards are capable of performing at this very high rate only for these given functions, and are in fact very limited in regard to other functions. That is, it is the application-specific nature of these boards, where the design is tailored for a given

function, that leads to such high performance.

The computer system currently consists of several special-purpose boards, and is not suitable for a production vehicle at this time. However, with the continued rapid advancement in integrated circuit technology we believe that it will soon be possible to implement all of these functions in a small chip set or chip package with reasonable cost for production, sufficient performance, and low power consumption.

SYSTEM ARCHITECTURE

The block diagram of the system is shown in Figure 1. The system consists of several boards on a VMEbus, housed in a standard full-height VME rack. There are two sections to this system: the general-purpose computer and the real-time image processor.

The main task of the general-purpose computer section is to function as the master controller for the real-time image processor. It consists of an M68030 processor running at a clock speed of 25 MHz, 4 Mbytes of dynamic random access memory (DRAM), a disk controller, a 20 Mbyte hard disk, and a floppy diskette drive. The M68030 processor communicates with all the peripherals through the VMEbus. The real-time image processing boards are interfaced to the VMEbus and to the video bus. The processor is capable of accessing any of these boards to send commands, to read status, or to access data. The processor periodically sends commands to the image processing boards at specific times so that they perform their intended functions in synchronization.

The general-purpose computer runs under a real-time operating system [4]. This is a multi-tasking operating system with fast interrupt handling and system utilities. There is a screen oriented text editor for developing and modifying programs. An assembler enables the programs developed on the system to be assembled and loaded into the memory for execution. A debugger on the system is an aid for developing the software while it is running in real-time.

The real-time image processor consists of a camera, a set of processing boards and a display. Each of the steps in processing the image is performed at video rate. After an image is acquired by the camera, pixels are passed from board to board in serial form. The boards are synchronized to the pixel times and operate concurrently with each other. Each board performs a particular process on the image by operating on the pixels sequentially as they arrive. Shortly after the last pixel from the input image to a process is encountered, the process reaches completion.

Information flow among the boards is accomplished through high-speed video data paths, as shown in Figure 1. Each data path conveys 8- or 16-bit data and shares common timing and control information with the other data paths. The separate data paths provide for concurrent data transfers among the processing modules. This increases the overall data transfer rate significantly compared to a conventional bus, in which only one transfer at a time can occur. The network of boards and data paths is said to act as a pipeline since information appears at various process stages throughout the network and the pieces of information flow through the network synchronously.

The detailed block diagram of real-time image processor is shown in Figure 2. Each of the blocks in this diagram performs a specific function, and most of them reside on a single board. A charge-coupled device (CCD) camera is used to acquire the image. This is a monochrome image in RS-170 format [5]. The digitizer is an 8-bit flash analog-to-digital (A/D) converter with a sampling rate of 10 MHz. The output of the digitizer is a 512x512 image. The digitized image is simultaneously fed into a framestore and the masking board. The masking board passes only a portion of the image whose location is chosen to be representative of the road intensity. The output of the masking board is the input to a histogram board, which generates the intensity histogram of the unmasked image portion. The M68030 processor uses this histogram to compute the mean intensity over this portion.

Once the mean from the histogram is computed, the M68030 processor generates a linear function for intensity normalization and updates a look-up table with this function. As the original image is passed through this look-up table, its intensity is modified according to this linear function so that the average intensity over the road region is nearly constant. This helps to remove undesired intensity variation in an image caused by, for example, an underpass, a bright or cloudy day, shadows from other vehicles, etc..

The output of the look-up table is presented to two paths in which the image is processed concurrently by different boards. In the first path, the normalized image is passed through a convolver that computes a 5-by-5 local average of the pixels. The output of the convolver is stored in a framestore as a processed image for later analysis.

The normalized image is concurrently fed into a Sobel edge detector. The edge detector consists of two convolvers and a look-up table. The convolvers operate on a 3x3 window. The look-up table combines the edges in the two perpendicular directions by adding their absolute values described by the formula:

$$S(x,y) = |H| + |V|$$

where the first absolute-value term corresponds to a vertical edge and the second to a horizontal edge.

The output of the edge detector is the input to another convolver, which also computes a 5-by-5 local average. The output of this convolver is stored in a framestore as a second processed image for later analysis

There are six framestores in the system. Each framestore is capable of holding one full image, and is accessible from the VMEbus and the video bus. The real-time image processor accesses the framestores through the video bus at video

rates. The CPU accesses them from the VMEbus at a slower rate because the framestore is implemented as a single-port memory, and the VMEbus has a lower priority.

One framestore is allocated for the original image, shown as "raw" in Figure 2. The two processed images generated from the special processing boards are stored in two separate framestores. These are labeled "averaged raw" image and "averaged edge" image. The rest of the lane sensing algorithm is executed directly by the M68030 processor in software. The processor uses the two processed images as input and estimates the lane boundaries. Finally, the processor executes an algorithm which uses the lane boundaries and vehicle dynamics information as inputs to estimate the curvature of the road, offset of the vehicle from the center of the lane, and the heading of the vehicle. Once all of this information is computed, it is placed in a framestore called "graphics overlay". As the image for the next process cycle is being acquired and stored, the contents of the "raw" framestore and the "graphics overlay" framestore are transferred to two other framestores which function as display buffers. These two images are displayed on the monitor as processing begins on the newly acquired image.

SYSTEM TIMING

The timing diagram of the lane sensing function is shown in Figure 3. The timing signals are generated by the video digitizer. All of the operations are synchronized to the beginning of image frames which are indicated by markers in the figure. Each frame consists of two fields and the digitizer generates a program interrupt to the processor at the beginning of each field. The frame markers in the figure correspond to the program interrupts for just the even field. The lane sensing function digitizes and processes the road images once every six frames which yields a total processing time of 0.2 seconds.

The timing diagram indicates the processing performed on each acquired image during different frame periods. The image which is acquired at frame number 0 is shown as image #n in the timing diagram. As soon as the even field interrupt is received, the processor instructs a framestore to accept the raw digitized image from the camera. Simultaneously, it instructs the other framestore to get the raw image of the previous processing cycle for display. The processor activates the histogram board so that the histogram of the pixel intensity of the image to be processed is generated. This process continues until the next even field interrupt arrives. At this time the processor freezes the raw image in a framestore and stops the histogram operation. During the second frame period the processor computes a linear transformation of the raw image for intensity normalization. The transformation function for this specific image is loaded into the look-up table by writing the computed values into 256 memory locations where each corresponds to a grey level of the image. The processor waits for the next even field interrupt which arrives at the beginning of the third frame period. At this time the raw image is passed through the look-up table, then through two independent process chains, and finally the results are placed into two separate framestores which were instructed by the processor to accept this data at the beginning of this frame period. At the beginning of the next even field interrupt, the processor freezes the two processed images already stored in the framestores. No further processing is performed by the image processing boards until the beginning of the next process cycle, which is shown as frame #6 in the timing diagram.

During the remaining three frame periods the M68030 processor uses these two partially processed images that reside in the framestores. The program in the processor determines the lane boundaries, executes the Kalman filter algorithm, and places the graphics information related to this computation into the designated framestore. During this processing, program interrupts still arrive from the digitizer, but the processor uses this information only to keep track of time relative to the processing cycle. In some cases, it is possible that the total processing is completed earlier than six frame periods. In such a case the program will wait before starting the next processing cycle in order to maintain the overall system timing. If processing is not completed in six frame periods, the current computation cycle is aborted, so that the next cycle can be started on time. The results of the previous cycle are used to substitute for those of the aborted cycle. In any case, the original image and the computed graphics information are transferred to the display buffers during the first frame period for the next computation cycle.

REAL-TIME SOFTWARE

The image processing section produces two images to be further processed by the real-time software. Real-time software is an assembly language program executing on the M68030 processor. This task is divided into five main tasks: Search, Line Fit, Decision Making, Estimation, and Graphics. A search is performed within the two processed images for points corresponding to lane markers or lane edges for each of the two lane boundaries. A list of such points is formed and is presented to the line fit procedure. The line fit procedure determines the slope and offset of a line that best represents the points in the list. Decisions are made throughout the task, such as to search the edge processed image when insufficient points have been found in the non-edge processed image and to reject either or both lane boundaries if certain angle and width criteria are not met. Estimation uses a Kalman filter to compute values for road curvature, vehicle heading and vehicle offset based on sensed lane position, measured vehicle speed and measured steering angle. Finally, graphics procedures display the results in the form of lane boundaries and estimation values superimposed on the road scene. Various parts of the real-time software are described below.

Main Loop:
This task initializes the image processing section, and various data and conditions. The software enters a continuous loop that is repeated every six frames. Depending on lane boundary candidates provided by processed image, one path of a four way branch is taken. If there is no candidate for ten successive cycles, then the default lane boundary is used with appropriate indication.

Search:
The search process produces a list of points from a processed image that correspond to a lane marker or lane edge. It establishes an area within the image over which the search is to be performed. The area is trapezoidal in shape and is centered around the previously found lane boundary. The search is carried out by scanning each horizontal line in the search area looking for at most one point that corresponds to a lane edge. This step is repeated for all horizontal lines in the search area.

Line Fit:
The search process supplies a list of points which are fitted in a line using the least mean square deviation process. This process is performed in combination of fixed- and floating-point format to compromise between speed, accuracy, and dynamic range.

Lane Change:
Each time new boundaries are determined, the condition for a possible lane change is checked. If one is detected the boundaries are replaced with those projected for the new lane. This process is repeated for both left and right lane change.

Estimation:
Three road/vehicle variables are estimated: road curvature, vehicle heading error and vehicle lateral offset from lane center. A vehicle/road interaction model has been developed for the estimation of the above variables. This model is divided into the cascade of two submodels each represented in linear state-space form. The first submodel involves the dynamics of the vehicle including the interaction of steering angle, yaw rate, lateral acceleration, roll angle, vehicle speed, etc. The second submodel involves the interaction of the vehicle motion and the geometry of the roadway. A Kalman filter is used with the second submodel to produce the three output variables.

Computational Burden:
The time allotted for the real-time software task is 0.1 second. Table I summarizes the worst-case processing time for each of the tasks and shows that the requirement is met.

Table I. Task Processing Times

Task	Processing Time (milliseconds)
Search	72.00
Line Fit	0.17
Decision Making	0.15
Estimation	2.10
Graphics	12.00
Miscellaneous	5.00
Total	91.42

SUMMARY AND CONCLUSION

A computer architecture has been developed to implement lane sensing in real-time on a vehicle. This system has been evaluated in the laboratory and successfully operated on a vehicle. The system has a 0.200 second processing time with 0.233 second delay, which translates into five images processed per second.

Inspection of the timing diagram provides information about the bottlenecks in the system. Updating the look-up table for intensity normalization takes one full frame period or two fields. This time could be reduced by generating a histogram only for the first field of the image during the first frame period, then updating the look-up table during the second field of that frame period. This will eliminate one full frame period from the total processing time.

The M68030 microprocessor spends three frame periods for processing the image. Most of the time is spent in searching the frame-stores for lane boundary information. This part of the algorithm can be made faster by developing custom electronics tailored for such searching. As a result, we can either run the same algorithm faster, or execute a more complex algorithm in the same time.

REFERENCES

[1] Kenue, S. K., "LANELOK: Detection of Lane Boundaries and Vehicle Tracking Using Image Processing Techniques - Part II: Template-Matching Algorithms", Proceedings of SPIE Conference on Mobile Robots IV. Vol 1195, pp. 234-245, Philadelphia, PA, November 1989.

[2] Duda, R.O. and Hart, P.E., Pattern Classification and Scene Analysis, Wiley, New York, 1973.

[3] Motorola Inc., The VMEbus Specifications, Rev C.1, Oct 1985.

[4] Eyring Research Institute Inc., PDOS Real-Time Operating System, System and User's Reference Manual, Oct 1987.

[5] EIA Standard EIA-170, Electrical Performance Standards - Monochrome Television Studio Facilities, November 1957.

Computer architecture and implementation of vision-based real-time lane sensing

3 Author(s)

O.D. Altan ; H.K. Patnaik ; R.P. Roesser

View All Authors

Abstract:
Lane sensing is the detection of lane boundaries from a video image of the roadway and the determination of the vehicle's position and orientation with respect to the lane. The detection of lane boundaries involves the digital processing of a roadway image to bring out points comprising the lane edge and a search of the processed images to find those points. Lane sensing supports several other functions, including collision warning, collision avoidance, driving performance monitoring and lane control. This paper describes the implementation of the two algorithms developed to achieve lane sensing in real-time on board a vehicle. This required developing the computer architecture, electronics and software to perform the processing steps in the algorithms. Attention was given to multitasking, multiprocessing and interprocess communication. In particular, use of the VME bus provides an open architecture and the disc-based real-time operating system provides multitasking and a flexible environment. The lane sensing system was installed in a vehicle and the real-time implementation was demonstrated on a test track and limited access highways.

Published in: Intelligent Vehicles '92 Symposium., Proceedings of the

Date of Conference: 29 June-1 July 1992

INSPEC Accession Number: 4509307

Date Added to IEEE *Xplore*: 06 August 2002

DOI: 10.1109/IVS.1992.252257

Print ISBN: 0-7803-0747-X

Publisher: IEEE

Keywords

IEEE Keywords
Computer architecture, Image edge detection, Multitasking, Real time systems, Vehicle detection, Road accidents, Collision avoidance, Computerized monitoring, Communication system control, Vehicles

INSPEC: Controlled Indexing
road vehicles, computer vision, computerised navigation, edge detection, position control, real-time systems

INSPEC: Non-Controlled Indexing

lane sensing system, lane boundary detection, computer vision, edge detection, navigation, road vehicles, roadway image, collision avoidance, driving performance monitoring, lane control, computer architecture, multiprocessing, interprocess communication, VME bus, open architecture, disc-based real-time operating system

Authors

O.D. Altan
Dept. of Vehicle Syst. Res., Gen. Motors Res. Lab., Warren, MI, USA

H.K. Patnaik
Dept. of Vehicle Syst. Res., Gen. Motors Res. Lab., Warren, MI, USA

R.P. Roesser
Dept. of Vehicle Syst. Res., Gen. Motors Res. Lab., Warren, MI, USA

Related Articles

A real-time computer vision system for measuring traffic parameters
D. Beymer; P. McLauchlan; B. Coifman; J. Malik

A massively parallel approach to real-time vision-based road markings detection
A. Broggi

A lane departure warning system based on a linear-parabolic lane model
C.R. Jung; C.R. Kelber

Automatic car plate detection and recognition through intelligent vision engineering
L. Salgado; J.M. Menendez; E. Rendon; N. Garcia

A stereo vision system for real time obstacle avoidance in unknown environment
F. Ferrari; E. Grosso; G. Sandini; M. Magrassi

Real-time detection of moving vehicles
R. Cucchiara; M. Piccardi; A. Prati; N. Scarabottolo

Real-Time Vision-Based Stop Sign Detection System on FPGA
Tam Phuong Cao; Guang Deng

A robust lane detection using real-time voting processor
A. Takahashi; Y. Ninomiya; M. Ohta; K. Tange

Preliminary investigation of real-time monitoring of a driver in city traffic
M. Betke; W.J. Mullally

A fusion system for real-time forward collision warning in automobiles
N. Srinivasa; Yang Chen; C. Daniell

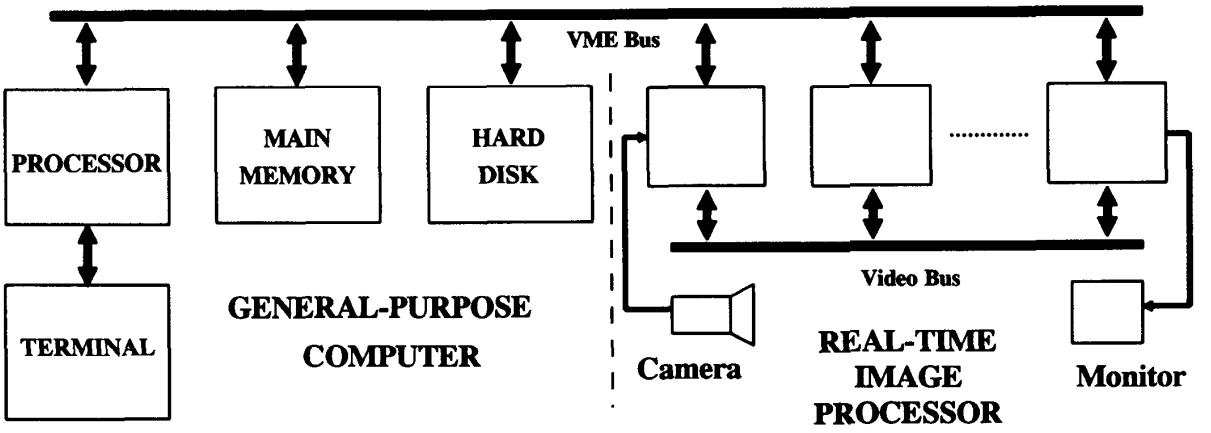


Figure 1. Block Diagram of the Lane Sensing System

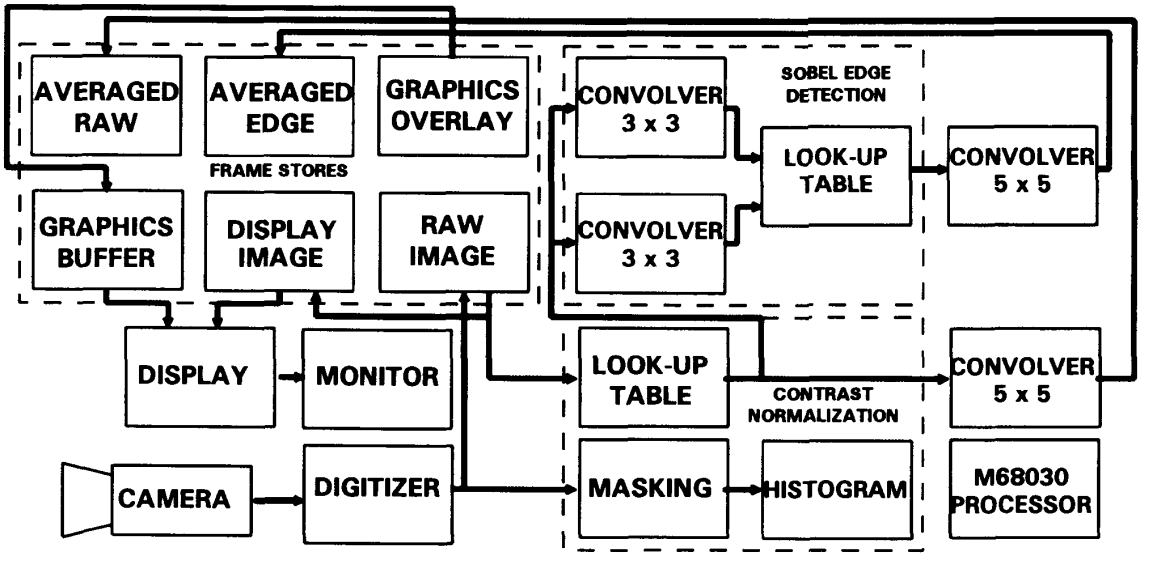


Figure 2. Architecture of Real-Time Image Processing System

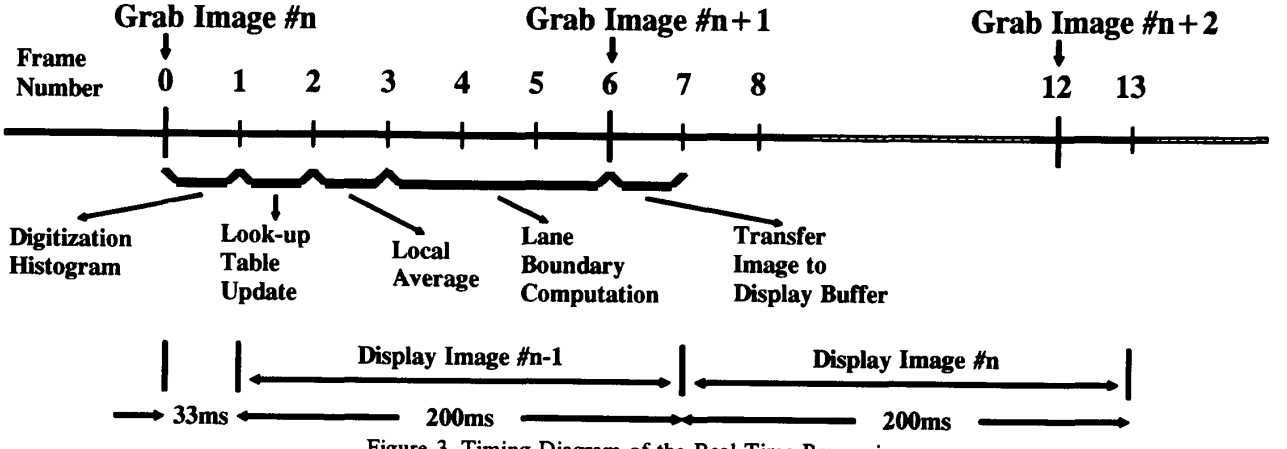


Figure 3. Timing Diagram of the Real-Time Processing