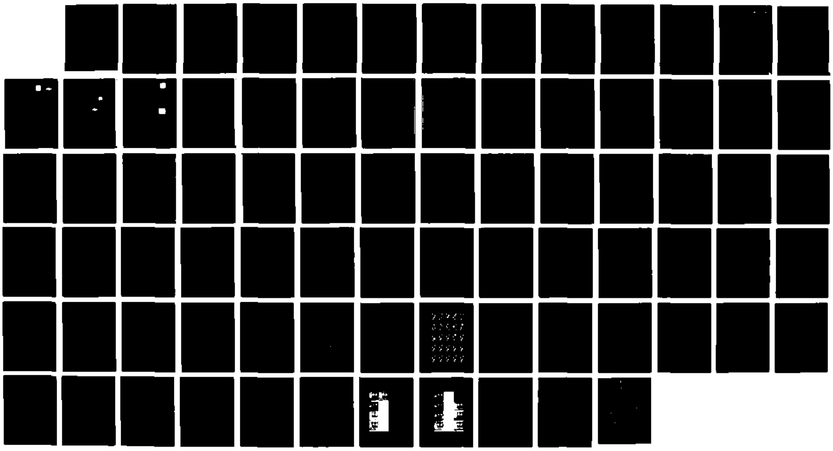
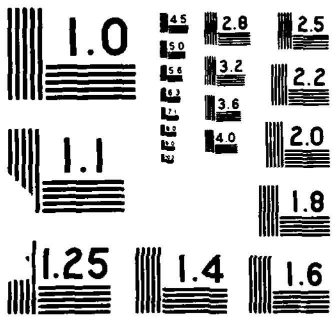
AD-A207 514 TIME-SEQUENTIAL OPTICAL DATA PROCESSING(U) 1/1
CARNEGIE-MELLON UNIV PITTSBURGH PA DEPT OF ELECTRICAL
AND COMPUTER ENGINEERIN G D CASASENT 01 FEB 89
UNCLASSIFIED N00014-86-K-0599 F/G 17/5 NL




1.0
4.5
5.0
5.6
6.3
7.1
8.0
9.0
2.8
3.2
3.6
4.0
2.5
2.2
2.0
1.1
1.8
1.25
1.4
1.6

2

Annual Report on Contract N00014-86-K-0599
(Period Covered: 1 January 1988 - 31 December 1989)

TIME-SEQUENTIAL OPTICAL DATA PROCESSING

Prepared by:
Carnegie Mellon University
Center for Excellence in Optical Data Processing
Department of Electrical and Computer Engineering
Pittsburgh, PA 15213
Dr. David Casasent, Principal Investigator
(412) 268-2464

Prepared for:
Office of Naval Research
Boston Detachment
495 Summer Street
Boston, MA 02210
Attention: Mr. William Miceli

1 February 1989

ABSTRACT

Our work has emphasized analog optical processing for three levels of multitarget tracking: target detection, track initiation and target tracking. New algorithms and new optical processing architectures have resulted and initial quantitative results have been obtained for each phase of the work. A new acousto-optic implementation of our sub-pixel target detection algorithm was devised and tested. This involved a new algorithm and architecture that achieves image registration, interpolation and subtraction. This technique is thus suitable for general image registration applications also. A 2-D Hough transform optical processor is used for track initiation. It allows accumulation of target measurements until sufficient confidence has been established that a target track exists. It provides position and velocity estimates for each target and predictions of the next vector location of each target. A new thresholded Hough transform provides improved performance and reduced false alarms in noise. Very impressive results from many multiple targets were obtained. This work is also suitable for general object feature extraction and product inspection applications. We devised a completely new realization of the Joint Probabilistic Data Association Algorithm for multitarget tracking and a most attractive new acousto-optic architecture to achieve this. This also has more general use in hypothesis testing. Finally, a new analog neural net algorithm and architecture was formulated, an optical processor to implement it was described and initial simulations of it were obtained. This system also has more general use in optimization problems involving multiple constraints. It employs a new cubic energy function and can incorporate various nonlinear optical materials being developed on other SDIO/IST ONR programs.

1. INTRODUCTION

In this chapter, we provide an overview of the various aspects of our work performed in 1988, with references to our earlier SDIO/IST ONR work. The details of the various aspects of each phase are included in separate chapters of this report.

Chapter 2 details our Hough transform (HT) track initiation work. Chapter 3 details our new acousto-optic (AO) processor to achieve this. Chapter 4 provides the mathematical basis for our new Joint Probabilistic Data Association (JPDA) multitarget tracker (MTT) algorithm and highlights the new optical architecture to realize it. Chapter 5 details the formulation of our new MTT algorithm as a cubic energy optimization problem with multiple constraints, advances a new neural net solution, optical architecture and provides initial simulation results.

1.1 OVERVIEW SUMMARY

Our earlier work [1,2,3] provided methods to detect sub-pixel targets moving on a structured correlated noise background that had various shifts in the background present between frames (due to platform instability and jitter). Chapter 2 provides a new AO architecture [4] to achieve the necessary functions. These involve 1-D correlations to estimate the sub-pixel frame background shifts, a method to shift one frame with respect to the other and perform the interpolation necessary to allow sub-pixel shifts to be achieved, and a technique to achieve the amplitude subtraction of the two frames (with one frame sub-pixel shifted and interpolated). This architecture/algorithm are very attractive since it requires only rugged 1-D AO devices. Initial tests show the algorithm's ability to extract targets that are sub-pixel from a strong background of correlated noise (background structure) and uncorrelated noise (due to the sensor). This technique is useful for non-SDI applications such as image registration (to increase SNR and for multi-sensor processing) and image differencing (for change detection) and for time-sequential image processing in general. The use of an optical processor to achieve the computationally-intensive function of interpolation is very new and most powerful. A comparison to other algorithms (showing the superior performance of our method) and extensive data on many target and noise scenarios was performed earlier [1-3].

Once candidate target measurements/detections have been made, we must assign target measurements to tracks. This is referred to as track initiation. It is a necessary preliminary step in which we first verify that a target exists, before handing this information off to a tracker (this is needed, since trackers that can initiate new tracks have enormous computational requirements). We utilize information on allowed target trajectories to assign measurements in separate frames to tracks. The algorithm used is a Hough transform performed on the composite sum of multiple frames of candidate target detections. Initial results were presented earlier [2,3]. A full description [5] is now provided and more extensive data. Our earlier work had developed distortion-invariant HT pattern recognition techniques using efficient search methods [6] and HT projections [7], associative processor realizations of a HT [8], and generalized HT methods to recognize curves and their parameters [9]. The algorithm we use allows target measurements to be accumulated until a sufficient number of measurements are obtained on one trajectory to achieve any degree of confidence desired that that trajectory is valid. New features of the algorithm are its ability to provide position and velocity estimates for each target and estimates of where (in vector space) the next measurement for each target is expected (these are necessary inputs to the MTT). A new thresholded HT algorithm was introduced that provides improved performance with significantly reduced false alarms. Extensive tests were performed on the algorithm including dither/jitter in measurements, the presence of noise and false measurements and many targets simultaneously present with different launch times and launch angles, etc. The results obtained were most impressive [5] as detailed in Chapter 3.

The track initiation (Chapter 3) provides us with high confidence target tracks for which MTT is now required. After surveying various MTT algorithms, we selected the JPDA algorithm (and is relative, the JPDAM algorithm, that can handle merged measurements such as target tracks that cross in the same time slot). This algorithm was chosen since it provides the best performance in associating measurements to targets, and it has a sound mathematical basis. Its conventional implementation requires the generation, storage and processing of an enormous number of feasibility matrices. We completely revised its implementation to avoid feasibility matrices, introduced new analog validation matrices and a new and efficient technique to search for the most feasible target/measurement pair assignments. The technique requires vector inner product operations and is thus suitable for optical realization, since it appears that these intensive computations can be performed to analog accuracy. A new AO optical processing architecture was devised to achieve these operations. Chapter 4 details the algorithm [10], expanding upon our 1988 work [3,11]. The algorithm is attractive, as it has a sound mathematical basis. It is also of more general use in hypothesis testing problems.

We recently [12-14] introduced a new and most efficient neural net MTT algorithm and analog optical realization for a MTT. This algorithm is more efficient than others since it only requires a number of neurons equal to the number of measurements (rather than one neuron per pixel in the field of view, as is typical). The algorithm and neural net are new since it utilizes a cubic energy function and since multiple constraints are imposed on the solution. The resultant optical architecture is also new, since it requires nonlinear elements and tensors (due to the cubic energy function). Initial simulation results, the development of the algorithm as an optimization problem, and a description of the required optical architecture are provided in Chapter 5.

A new concept for processing time-sequential image data was advanced in 1988 and initial results obtained [15]. A new and most efficient recursive singular value decomposition algorithm to produce a model of time-sequential data (a process) was devised. This algorithm requires only simple vector inner products (that should allow analog processing accuracy); it provides a novelty detector (this is new, as it indicates if new image frame data is sufficiently novel); it allows the type of novel data to be determined (i.e. is the object merely shifted or does it contain a different aspect view of the object); it provides a novelty filter (using an algorithm that is significantly different from others) that then allows us to estimate the rank of the process and to determine novel image data. Preliminary results and the development of the algorithm are provided in Chapter 6.

1.2 PLANNED 1989 RESEARCH

Our 1989 plans are affected by a reduction in our SDIO/IST grant funds by 50%. In 1989, we plan to:

- TASK 1:
  Extend the HT to 3-D to allow tracking in 3-D space from several 2-D sensor data sets or from range data.

- TASK 2:
  Quantification of our optical JPDA multitarget tracker will be conducted. This will involve: assessment of the analog processing accuracy required; certain algorithm improvements; tests of the full system on multitarget, data crossing targets, etc.; and encoding of a factorized extended Kalman filter algorithm with improved stability.

- TASK 3:
  Extensions of our analog optical neural net to a simpler quadratic energy formulation and a simpler optical realization.

- TASK 4:
  Our initial novelty detector and filter work on processing time-sequential image frames will be extended to include: a novelty tracker to produce a model of the object and the background, techniques to determine the type of novel data (i.e. shifts or different aspects views of the objects), the role for optics, and analog accuracy requirements.

1.3 PRESENTATIONS AND PUBLICATIONS

The 15 conference and journal paper references noted represent an extensive documentation and description of our results. They include non-optics journals and non-optical neural net conferences (to allow wide discrimination of our results). Papers 1, 2, 11, 13 and 15 were also presented at various SPIE conferences. Paper 12 was presented at a neural net conference.

Other presentations included a review of this work presented at RADC in New York, a review at Teledyne Brown Engineering in Huntsville, a lengthy review presented to the Army SDI in Huntsville, plus a presentation at the SDIO/ONR June 1988 conference in Washington, D.C.

1.4 REFERENCES

1. D. Casasent, B.V.K. Vijaya Kumar and Y.L. Lin, "Sub-pixel Target Detection and Tracking", SPIE Proc., Vol. 726, pp. 206-220, October 1986.

2. D. Casasent, Y.L. Lin and J. Slaski, "Optical Multitarget Subpixel Detection and Track Initiation", Proc. SPIE, Vol. 881, pp. 12-27, January 1988.

3. "Time-Sequential Optical Data Processing", ONR Technical Report, David Casasent, 1988.

4. D. Casasent and J.H. Song, "Optical Projection Correlations", Applied Optics, Vol. 27, pp. 4977-4984, 1 December 1988.

5. D. Casasent and J. Slaski, "Optical Track Initiator for Multitarget Tracking", Applied Optics, Vol. 27, pp. 4546-4553, 1 November 1988.

6. D. Casasent and R. Krishnapuram, "Curved Object Location by Hough Transformations and Inversions", Pattern Recognition, Vol. 20, No. 2, 1987, pp. 181-188.

7. R. Krishnapuram and D. Casasent, "Hough Space Transformations for Discrimination and Distortion Estimation", Computer Vision, Graphics, and Image Processing, Vol. 38, pp. 299-316, June 1987.

8. R. Krishnapuram and D. Casasent, "Optical Associative Processor for General Linear Transformations", Applied Optics, Vol. 26, pp. 3641-3648, 1 September 1987.

9. R. Krishnapuram and D. Casasent, "Hough Transform Detection of 3-D Curves and Target Trajectories", Applied Optics, submitted August 1988.

10. J. Fisher and D. Casasent, "A Fast JPDA Multitarget Tracking Algorithm", Applied Optics, Vol. 28, pp. 371-376, 15 January 1989.

11. J. Fisher and D. Casasent, "An Optical Technique for Measurement/Target Probability Assignments", Proc. SPIE, Vol. 881, pp. 273-289, January 1988.

12. E. Barnard and D. Casasent, "New Tracking and Matrix Inversion Optical Neural Nets", IEEE, *Annual International Conference on Neural Networks*, San Diego, CA, July 1988, Poster presentation.

13. E. Barnard and D. Casasent, "New Optical Neural System Architectures and

Applications", presented at the *OSA Toulon Optical Computing Conference 88*, August/September 1988, Toulon, France, Proc. SPIE, Vol. 963.

14. E. Barnard and D. Casasent, "Multitarget Tracking with Cubic Energy Optical Neural Nets", Applied Optics, accepted for 15 March 1989 publication.

15. P. Vermeulen and D. Casasent. "KL Techniques for Optimal Processing of Time Sequential Imagery", Proc. SPIE, Vol. 1007, November 1988.

CHAPTER 2

"Optical Projection Correlations"

Optical projection correlations

David P. Casasent and Jung-Hee Song

The correlation of several 1-D projections of a 2-D image are considered for pattern recognition. A theoretical analysis and SNR comparison to 2-D correlations are provided with successful simulated results that show that the use of two or three 1-D correlations can identify and discriminate the 26 characters in the alphabet. Several possible 1-D optical correlators to implement projection correlations are described.

I. Introduction

Two-dimensional optical correlations are conventionally used for pattern recognition. Distortion invariance can be incorporated by using various smart filters,[1-3] white light processing,[4,5] and other techniques. The computational complexity required to synthesize distortion-invariant matched spatial filters can be large (especially when many object classes are considered and when 2-D images are employed). This paper concerns the use of 1-D correlations of projections of 2-D images (i.e., the 2-D image is integrated at some angle to produce a 1-D projection image) for pattern recognition. One-dimensional projections are widely used to reconstruct a 2-D image by Radon transforms.[6] By the central slice theorem,[7] the 1-D Fourier transform (FT) of the 1-D projection along an angle θ of a 2-D image is a 1-D slice (at an angle orthogonal to θ) of the 2-D FT of the 2-D image. Thus significant information about a 2-D image exists from its 1-D projections and their 1-D FTs. Our emphasis is to use only several 1-D projections and to perform pattern recognition not image reconstruction.

The motivation for this is that 1-D acoustooptic (AO) correlators are well established and more mature, robust, faster, and less expensive than 2-D correlators requiring real-time spatial light modulators. In addition, filter synthesis is greatly simplified if 1-D data are used. Related work on the use of 1-D projections includes registration of images,[8] reconstruction of a distorted pattern by iterative processing of its known and measured horizontal and vertical projections,[9] and the use of linguistic rules (not correlation techniques) and subspace methods to recognize characters from their horizontal and vertical projections.[10] Our work differs in its attention to pattern recognition not image reconstruction,[9] multiclass problems rather than single-class ones[8,10] and the use of several (e.g., three) projections rather than only horizontal and vertical projections. Section II discusses several properties of 1-D projection correlations and provides a theoretical comparison to 2-D correlation results. Simulated results are presented in Sec. III, and candidate optical processing architectures are included in Sec. IV. A discussion of this technique and the results follows in Sec. V.

II. One-Dimensional Projection Correlation Analysis

A. Relating 1-D Projection FTs to Slices of the 2-D FT

The 1-D projection at an angle θ of a 2-D image function $f(x,y)$ is written as

$$p_\theta(r) = \int\int f(x,y)\delta(r - x\cos\theta - y\sin\theta)dxdy, \qquad (1)$$

where r is the projection axis. Figure 1(a) defines r and θ and shows the operation graphically as the integration of the 2-D function along a set of parallel beams orthogonal to the axis r (the x axis rotated by θ). The variable r has positive and negative values, and the angle θ need only vary over $(0,\pi)$ to allow a full description of the object. The 1-D FT in r of $p_\theta(r)$ yields

$$P_\theta(\rho) = \int_{-\infty}^{\infty}\int_0^{\pi} f(r_0,\theta_0)\exp[-j2\pi\rho r_0\cos(\theta_0-\theta)]r_0 d\theta_0 dr_0. \qquad (2)$$

This result is obtained by converting the image to (r_0,θ_0) polar coordinates and producing a polar coordinate description of the 2-D FT of $f(x,y)$ with the radial frequency variable ρ corresponding to r in the image. Since θ is fixed, Eq. (2) describes a 1-D slice of the 2-D Cartesian coordinate FT $F(u,v)$ at an angle θ as shown in Fig. 1(b).

The authors are with Carnegie Mellon University, Department of Electrical & Computer Engineering, Center for Excellence in Optical Data Processing, Pittsburgh, Pennsylvania 15213.

Received 12 April 1988.
0003-6935/88/234977-08$02.00/0.

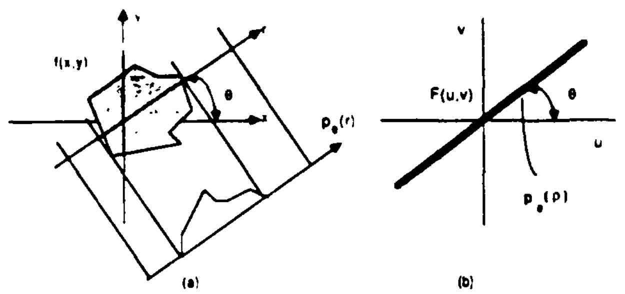


Fig. 1. Examples of (a) 1-D projection and (b) the Fourier transform slice it is equivalent to.

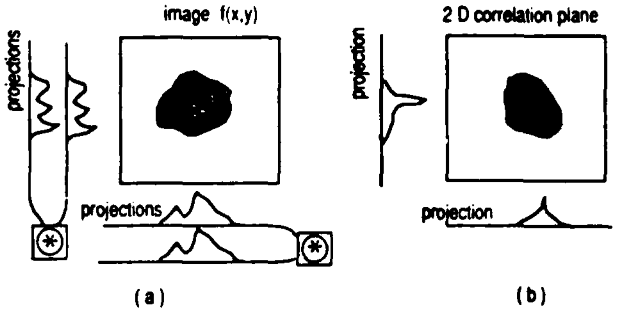


Fig. 2. Equivalence of 1-D projection correlations and projections of the 2-D correlation pattern: (a) horizontal and vertical projections of a 2-D image; (b) horizontal and vertical projections of the 2-D correlation pattern.

B. Relating 1-D Projection Correlations to Projections of 2-D Correlations

A comparison of an input f to a reference g can be made by correlating the projections of f and g at the same θ. Denoting these projections by $p_f(r)$ and $p_g(r)$ and their 1-D FTs by $P_f(\rho)$ and $P_g(\rho)$, the 1-D correlation for projections at the same θ is

$$\begin{aligned} c_\theta(r) &= \int P_f(\rho)P_g^*(\rho)\exp(j2\pi\rho r)d\rho \\ &= \int C_{fg}(\rho,\theta)\exp(j2\pi\rho r)d\rho \\ &= \int\int c_{fg}(x,y)\delta(r - x\cos\theta - y\sin\theta)dxdy. \end{aligned} \quad (3)$$

The first equation in (3) describes this 1-D correlation as the 1-D FT of the product of the FTs of the projections. This equals the 1-D FT of a slice (at angle θ) of the 2-D FT C_{fg} of the correlation pattern $c_{fg}(x,y)$ as the second equation shows. The last expression in (3) states that this 1-D correlation is the 1-D projection at angle θ of the 2-D correlation function $c_{fg}(x,y)$. Thus the result of a projection correlation is the same as integrating a projection of the 2-D correlation plane. Figure 2 shows these observations pictorially. In Fig. 2(a), the horizontal and vertical projections of the image are formed and correlated with reference horizontal and vertical projections. These two output 1-D correlation patterns are the same as the horizontal and vertical projections of the 2-D correlation plane pattern as shown schematically in Fig. 2(b). Thus 1-D projection correlations are equivalent to projections of 2-D correlations. We will use the terms projection correlations and integrated 2-D correlations to describe these two results. We will use this new result in our work. For these results to be equal, a bipolar correlation is necessary (i.e., we cannot intensity detect the correlation output). As we will see, this is possible optically and is most easily achieved with 1-D correlations.

C. SNR and Peak-to-Sidelobe Ratio (PSR) Comparisons

One might assume that some loss must occur when such 1-D processing is used. We now address this topic. We envision forming only selected 1-D integrated slices of the correlation plane rather than all integrated slices of it. The number of slices employed will depend on the performance required and the number of multiple objects simultaneously present. We use amplitude SNR (rather than the more conventional energy SNR) as our performance measure. This choice is made, since our processor with product correlation amplitude (i.e., the correlation peak value will be the product of the amplitudes of the input and reference signals and not peak intensity, as one would obtain optically without heterodyne detection). Our output is a correlation plane, and its SNR = SNR_o is the correlation peak value divided by the standard deviation in the correlation plane. Our SNR_i (in the input image plane) is thus defined as the mean square root of the signal energy per pixel divided by the standard deviation of the input noise. Using the signal energy per pixel is analogous to the SNR_o definition (at one point, the correlation peak). Averaging (via the mean) is included so that results do not depend on the input pixel chosen. (In our case, all signal and noise data are uniformly distributed.) Thus, for an $N \times N$ 2-D image with noise standard deviation σ, we have $\text{SNR}_i = 1/\sigma$ (since the full input energy over N^2 pixels is N^2 for an amplitude per pixel value of 1). The standard processing gain PG_E = SBWP (space–bandwidth product) = $(\text{SNR}_o/\text{SNR}_i)^2$ is an energy ratio. For our case, the amplitude PG_A ratio applies, i.e., $\text{PG}_A = (\text{PG}_E)^{1/2} = (\text{SBWP})^{1/2}$. Throughout our discussion, subscripts A and E refer to amplitude or energy, respectively.

With these preliminaries and definitions, we now consider and detail that there is no SNR loss if 1-D projection correlations are performed rather than a 2-D correlation. We consider the received input image to be $r(x,y) = s(x - x_0,y - y_0) + n(x,y)$, where s is the reference signal centered at input location (x_0,y_0) and $n(x,y)$ is additive noise. We form the horizontal and vertical projections of the reference and received signals. These are described by

$$s_x(y) = \int s(x,y)dx, s_y(x) = \int s(x,y)dy, \quad (4a)$$

$$r_x(y) = \int r(x,y)dx, r_y(x) = \int r(x,y)dy. \quad (4b)$$

We first consider the 2-D image with a space–bandwidth product $\text{SBWP}_E = N^2$ (and $\text{SBWP}_A = N$), and the noise is assumed to be white, Gaussian, and of zero mean with variance σ^2. The input $\text{SNR}_i^2 = 1/\sigma$ for the 2-D case (as detailed above) and $\text{PG}_A = N$. Thus we expect an output $\text{SNR}_o^2 = (\text{SNR}_i)(\text{PG}_A) = N/\sigma$. This agrees with our SNR_o definition as we now show. The correlation peak amplitude is N^2, and the output (correlation plane) noise standard deviation is $N\sigma$. To see why $N\sigma$ arises, consider the correlation plane noise variance [i.e., the contribution to the correlation plane due to input noise and a filter $s^*(x,y)$ matched to the input signal]. This is σ^2 times the energy of the filter. Its standard deviation is σ times the square root of the energy of the filter, i.e., $(N^2)^{1/2} = N$; thus the correlation plane noise standard deviation is $N\sigma$ and $\text{SNR}_o = N^2/N\sigma = N/\sigma$ for the 2-D case.

We now consider the 1-D projection correlation case. Now each projection pixel has an amplitude of N or an energy N^2. There are N projection pixels with a total energy N^3 or a square root energy per pixel value of $(N^3/N)^{1/2} = N$. (This is the SNR_i numerator.) Adding N noise pixels (in a projection image) produces a projection noise pixel variance $N\sigma^2$ or a standard deviation $N^{1/2}\sigma$. Thus the projection $\text{SNR}_i = N/N^{1/2}\sigma = N^{1/2}/\sigma$. Comparing this to the 2-D $\text{SNR}_i = 1/\sigma$, we see that the projection image has a larger input SNR_i (by a factor of $N^{1/2}$) than does the 2-D image.

Next we consider output SNR_o for the 1-D projection and the 2-D image cases. For the 2-D case, $\text{PG}_A = N = (\text{SBWP})^{1/2} = \text{SNR}_o/\text{SNR}_i$. Thus $\text{SNR}_o = (\text{PG}_A)(\text{SNR}_i) = N/\sigma$ (in agreement with our prior results). Similarly, for the 1-D projection case, $\text{PG}_A = N^{1/2}$ and $\text{SNR}_o = (\text{PG}_A)(\text{SNR}_i) = N^{1/2}(N^{1/2}/\sigma) = N/\sigma$. Thus we have just shown that SNR_o for a 2-D correlation and one 1-D projection correlation are identical. As a result, we find 1-D amplitude projection correlations to be useful and suitable. We note that even if the 1-D projection correlations are normalized (or averaged), i.e., if each pixel in the projection of an $N \times N$ pixel image is divided by N (to normalize each projection pixel value to unity), we obtain the same resultant SNR_o.

If more than one 1-D projection correlation is used and the results combined, the resultant SNR_o may be increased (since the noise for each 1-D correlation is different). However, the major result appears to be an increased confidence in the resultant correlation (rather than an improved SNR) since each projection correlation provides us with a separate output correlation sample to average over. It is also interesting to compare the performance (SNR_o) of a 1-D signal correlation to a 1-D projection correlation. For the 1-D signal case, $\text{SNR}_i = 1/\sigma$ still, $\text{PG}_A = N^{1/2}$ now, and $\text{SNR}_o = N^{1/2}/\sigma$ is less than for the 1-D projection correlation (by a factor of $N^{1/2}$).

From these brief summary theoretical analyses, it is clear that projection correlations are adequate measures of agreement of two 2-D images. The key step in this analysis is the effect of projection integration on noise. This effect is now summarized. When N noise realizations are summed, the resultant noise variance is $N\sigma^2$ (regardless of the mean of the noise), since the variance is the spread expected in the noise values. However, if the average of these N signals is formed (as is done in computing our signal measure), i.e., if we divide each pixel value by N, the variance is reduced from $N\sigma^2$ by a factor of N^2 to σ^2/N. The factor N^2 (rather than N) is seen by considering the average $y = (1/N)\Sigma n_i$ of one column of N noise pixels n_i. Since the n_i are independent, $E\{y\} = 0$ and $\text{Var}\{y\} = E\{y^2\} = (1/N^2)\Sigma\sigma^2 = (1/N^2)(N\sigma^2) = \sigma^2/N$.

The prior analysis assumed a white signal (so that all N or N^2 pixels in it contributed equally to the correlation peak value). The SBWP value (used for PG), i.e., N or N^2, assumes this type of signal distribution. In a practical (nonwhite) signal case, there will be a difference in SNR_o for 1-D projection correlations vs 2-D correlations. The difference will be a function of the shape of the spectrum of the signal. The loss factor (by which SNR_o of the projection correlation is worse than that for the 2-D correlation) is a maximum of $N^{1/2}$. To see this, consider the case when the input image has only one nonzero pixel. Here the 2-D $\text{SNR}_i = (1/N^2)^{1/2}/\sigma = 1/(N\sigma)$, and $\text{SNR}_o = 1/\sigma$. The 1-D projection has $\text{SNR}_i = (1/N^{1/2})/(N^{1/2}\sigma) = 1/N\sigma$ (which is the same as for the 2-D case). However, the 1-D $\text{PG}_A = N^{1/2}$ and the 1-D projection $\text{SNR}_o = N^{1/2}/N\sigma = 1/N^{1/2}\sigma$ is a factor of $N^{1/2}$ worse than for the 2-D case ($\text{SNR}_o = 1/\sigma$). As more input pixels (besides just one) become nonzero, SNR_i for the 1-D projections increases faster than SNR_i for the 2-D case (as shown before, SNR_i for 1-D projections is larger than for the 2-D case when all input pixels are one, and in this case SNR_o for both cases are equal). Thus the loss factor of $N^{1/2}$ is a worst case theoretical one.

Ros *et al.*[8] have shown data (averaged over sixteen images) indicating that SNR_i for projection images is larger than SNR_i for the same 2-D images. This supports our remark that generally the projection SNR_i is better than for the 2-D case. The detectability of the 1-D or 2-D correlation peak is also of concern. Using the criteria that a correlation peak is detectable if its expected value is more than 16 times its variance, it has been shown that to achieve the same detectability for projection correlations and 2-D correlations, we require that SNR_i for a 1-D projection need only be four times higher than SNR_i for the same 2-D pattern. Since SNR_i for projections is larger than for the 2-D image, projection correlations are expected to yield adequate and useful results.

In our simulation tests (Sec. III), we will use SNR and PSR as correlation plane performance and comparison measures. We define these by

$$\text{SNR} = \frac{\text{peak value}}{\text{(standard deviation away from the peak)}}, \tag{5a}$$

$$\text{PSR} = \frac{\text{peak value}}{\text{average value around the peak}}. \tag{5b}$$

Let us now advance various expected projection correlation, 2-D correlation, and integrated 2-D correlation results.

(1) We expect projection correlations and integrated correlations to yield equal outputs (by our correlation plane version of the central slice theorem).

(2) We expect the peak value for an integrated cor-

relation to be larger than the peak value in a 2-D correlation (due to integration of the sidelobes).

(3) We do not expect equal peak values for horizontal and vertical 1-D projection correlations (due to integration of different sidelobe patterns for the different projection angles).

(4) We expect input SNR for projection images to be larger than for 2-D images.

(5) We expect SNR and PSR for projection correlations to allow adequate peak detectability.

For many patterns, the peak value for the integrated 2-D correlations will increase more than the sidelobes. This occurs since most correlation functions have a circular shape, and thus more sidelobe values contribute to the peak than to the sidelobes when integrated. Thus we can expect PSR for projection correlations to be larger than for 2-D correlations. Our data in Sec. III verify and quantify these remarks. It will also address the amount by which the cross-correlation of two different patterns is affected by use of projection correlations.

III. Test Results

A. Database

To test and quantify our projection correlation concepts, the 26 upper-case letters in the Times font were used. Each letter was digitized to 32 × 32 binary pixels. Most letters occupied the central 10 × 10 pixels and had 40–75 nonzero pixels and all letters had less than 110 nonzero pixels out of the total 1024 pixels. In our PSR measure, we calculate the sidelobe value as the mean in a 11- × 11-pixel region around the correlation peak. This is approximately the 3-dB width of the correlation function, and the peak value was not included in this sidelobe average. Since the correlation function is a number of pixels wide, the average sidelobe value includes a significant amount of the peak energy, and thus PSR is a measure of the sharpness (detectability) of the correlation peak. We measured correlation plane SNR by calculating the noise variance in the entire 64 × 64 correlation plane except for the 32- × 32-pixel region around the peak. (This excludes all the energy in the correlation peak and its sidelobes.)

B. Autocorrelations

We formed the horizontal and vertical projection correlations and the 2-D correlation of various characters. Figure 3 shows the results for a binary (0,1) image of the letter *C* [Fig. 3(a)]. The 2-D correlation is shown in Fig. 3(b). The unnormalized autocorrelation peak height is 43 (there are 43 nonzero pixels on this letter) and the measured PSR is 3.8. To verify our results in Sec. II, we formed the vertical projection correlation [Fig. 3(c)] and the vertically integrated [Fig. 3(d)] 2-D correlation of Fig. 3(b). Both results are nearly identical (with the difference being of the order of $10^{-3}\%$). The peak height (215) in the projection correlation is larger than the 2-D peak (43) since sidelobes contribute to the peak when the 2-D pattern

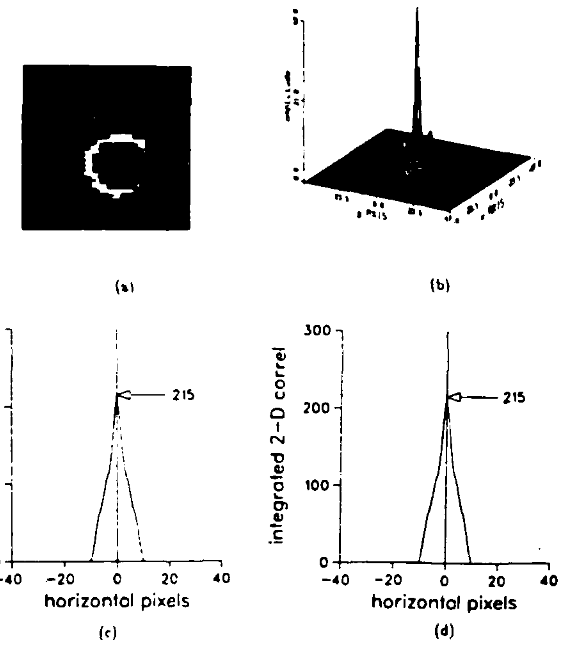


Fig. 3. Comparison of a projection correlation and an integrated 2-D correlation: (a) input character *C* image; (b) 2-D correlation; (c) vertical projection correlation; and (d) vertically integrated 2-D correlation.

in Fig. 3(b) is integrated. The measured PSR (11.9) is also larger. This occurs, since in the integration of Fig. 3(b), more sidelobes contributed to the peak than to the sidelobes around it. In this case, the peak in the integrated 2-D correlation increased by a factor of 5, and the average sidelobe level increased by a factor of ~18/11 ≃ 1.6. Because of the effect of sidelobes when integrating the 2-D correlation, we expect different results when different projection or integration angles are used. For example, the horizontal projection correlation gave a peak height of 193 and a PSR of 13.5. The full projection correlation peak is broader (since sidelobes are now integrated) than the 2-D correlation peak, but its PSR (in the 10- × 10-pixel region around the peak) is better, and thus the central portion of the projection correlation is sharper than for the 2-D correlation. This result holds for any reasonable window size used to calculate the average sidelobe value. The SNR is not calculated since no noise is present.

Next, we repeated the above tests using zero-mean images. The 2-D correlation pattern had nearly the same peak height (it is reduced because the dc value does not contribute to the peak) and a slightly larger PSR of 3.9. (This is expected since the average of bipolar, positive and negative, sidelobes will be less than before.) The vertical and horizontal bipolar projection correlations are shown in Figs. 4. The integrated 2-D correlation results are the same as expected and are not shown. The projection correlation peak values (186 and 164) were less, and the PSR values (12.9 and 15.4) were larger than when binary images were used. Again, with zero-mean data, projection correlations also gave better PSR. The shapes of the projection correlations differ due to the different sidelobe patterns integrated as explained earlier.

For the 2-D bipolar correlation, the average value of the entire image was subtracted from all image pixels or equivalently the dc spatial frequency in the 2-D FT is blocked. One can zero-mean the input or the reference or both and achieve the same results. The projections are formed from the zero-mean image described above, and a bipolar correlation is performed to yield the projection correlations shown. Noise variance is not affected by whether the data are of zero-mean or not. Zero-mean data will result in sharper autocorrelation peaks (since the spectrum of the image is whitened by dc filtering). The major advantage of using zero-mean data is that this enhances the differences between characters and thus improves discrimination. We briefly experimented with different bandpass filters and edge enhancement. This resulted in very narrow correlation peaks but sidelobes with high variance and very sensitive to noise. Further work should allow determination of the optimum cutoff frequency to reduce cross-correlations and produce sharp peaks with limited noise sensitivity.

As noted above, the ratio of the auto-to-cross-correlation peaks will be larger, and hence discrimination will be better with zero-mean data than with nonzero mean images. However, one must not form the magnitude or intensity of the output bipolar correlation, since we have found large negative correlation plane peaks with zero mean data (and these become positive when the magnitude or intensity is formed). A bipolar correlation is necessary for the integrated 2-D correlation to equal the bipolar projection correlation. If the magnitude or intensity of the 2-D correlation is taken, the integrated 2-D correlation does not equal the magnitude or intensity of the projection correlation. If we could optically form a unipolar amplitude correlation and then remove its mean, we would obtain the desired bipolar correlation. However, this is not easily achieved in 2-D. Conventional 2-D frequency plane optical correlators can produce bipolar correlations, but detection forms the intensity of the result. Bipolar 2-D output correlations are not easily obtained without heterodyne detection, and a significant increase in the number of output detectors is required. Conversely, AO systems can easily produce bipolar correlations[11] and thus better results. Furthermore, the electrical bipolar optical correlation output can easily be thresholded at zero volts, thus removing all negative values (sidelobes and noise) and yielding a preferred output.

Our final autocorrelation test included noise with standard deviation $\sigma = 0.35$ present and with the full image plus noise made to be zero mean. (This is what an optical system does.) Figure 5(a) shows the magnitude of the noisy input letter C used. The bipolar noise results in the dropout of some pixels on the letter and the addition of noise in the background. Figure 5(b) shows the magnitude of the 2-D bipolar output correlation, and Fig. 5(c) shows the horizontal projection correlation. For the horizontal projection, the

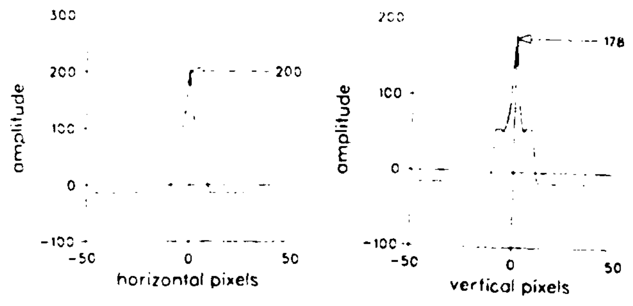
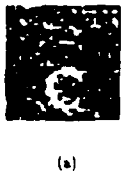


Fig. 4. Zero-mean projection correlations: (a) vertical projection correlation and (b) horizontal projection correlation.



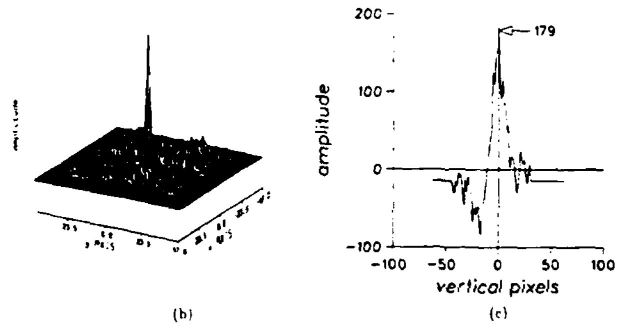


Fig. 5. Noisy projection and 2-D correlations: (a) noisy input; (b) 2-D correlation input; and (c) horizontal projection correlation.

peak is 179.3, PSR = 11.5, and SNR = 26.2 (or 14.2 dB). These values are all acceptable, and the peak is easily detected. For comparison, we note that the peak (40.2) and PSR (3.9) of the 2-D correlation are less. Although the SNR is not as useful of a correlation plane performance measure as is PSR or auto-to-cross-correlation peak ratio, we note that SNR for the 2-D correlation is 100.8 (or $\simeq$20 dB). The 1-D SNR of 14.2 dB is less than this by 5.8 dB, and this is less than the $N^{1/2} = 8$ (or 9-dB) worst-case loss predicted in Sec. II.C. We used $N = 64$ in this calculation, since the 64 lines of the correlation plane are integrated here. Since the projection correlation peak increases by a factor of ~4 (from $\simeq$40 to $\simeq$180 for projections), we would expect the SNR loss using projections to be no more than 8/4 = 2 (or 3 dB). When results for more than one noise realization are averaged, the measured SNR difference is expected to approach this 3-dB factor. We include these SNR comparisons only to note that the measured data do yield a lower SNR loss than the worst-case theory in Sec. II.C.

C. Cross-Correlations and Multiple Projections

From Sec. III.B, we showed that autocorrelation peaks using projection correlations are easily located (good PSR) and easily detected (good SNR). In pattern recognition, cross-correlations are also of concern. Specifically, with multiple filters, the autocorrelation peak for the true input must be larger than the correlation output anywhere in the cross-correlation plane with a different input. We thus tested projection cross-correlations and compared their performance with 2-D cross-correlations. We chose C as our reference object and considered its cross-correlation with P. We normalized the energy of all letters to 1 and normalized the filter energy to 1 also. All data were of zero-mean.

Figure 6(a) shows the 32×32 input data used. With a filter of C used, the correlation output has autocorrelation and cross-correlation peaks. Figures 6(b) and (c) show the horizontal and 45° projection autocorrelation and cross-correlation outputs. The horizontal projection correlations gave a normalized autocorrelation peak height of 1.0 and a large cross-correlation peak height of 0.95. Vertical correlation projections (not shown) gave autocorrelation and cross-correlation peak heights of 1.0 and 0.92, which are still very close. The 45° projection correlation [Fig. 6(c)] shows better discrimination with autocorrelation and cross-correlation peak heights of 1.0 and 0.82. This difference is sufficient to allow discrimination. Thus, by using several projections (besides just horizontal and vertical ones), we can achieve acceptable projection correlation results. This is reasonable, since the projections of two characters should be more distinguishable at some projection angles than at others. In our tests, we used only horizontal, vertical, and 45° (diagonal) projections. When compared with 2-D correlations with a normalized autocorrelation peak of 1, the cross-correlation peak was 0.55. Thus 2-D correlation provides more discrimination, but the use of only two or three projection correlations yields sufficient discrimination with a significantly simplified processor. We expect projection correlations to yield large cross-correlations since they contain only 10 pixels. With more pixels on the character, much better discrimination is expected. Our present concern is to demonstrate the concept and potential for projection correlations, and thus higher resolution inputs and preferable smart filters were not used.

As our next test, we used zero-mean input signal-plus-noise test characters with noise with $\sigma = 0.45$. The inputs are shown in Fig. 7(a). The horizontal and diagonal projection correlation results are shown in Figs. 7(b) and (c). The ratio of the autocorrelation to cross-correlation peak values was $0.52/0.39 = 1.33$ for the horizontal projection correlations and $0.57/0.38 = 1.5$ for the 45° diagonal projection correlations. These ratios are larger with noise due to the integrating effects of the noise. The projection autocorrelation to cross-correlation peak ratio of 1.5 is comparable with the ratio of 1.8 obtained from 2-D correlations with noise present. Thus no significant loss results when

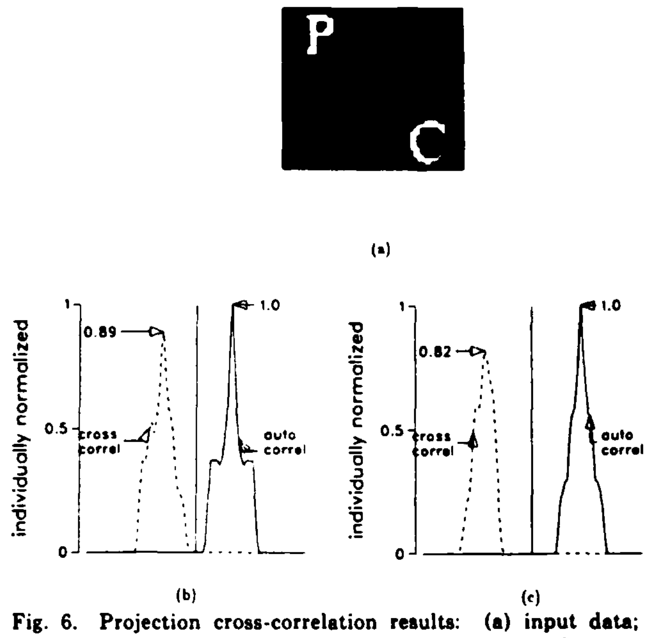


Fig. 6. Projection cross-correlation results: (a) input data; (b) horizontal projection correlations with a filter of C; and (c) 45° diagonal projection correlations with a filter of C.

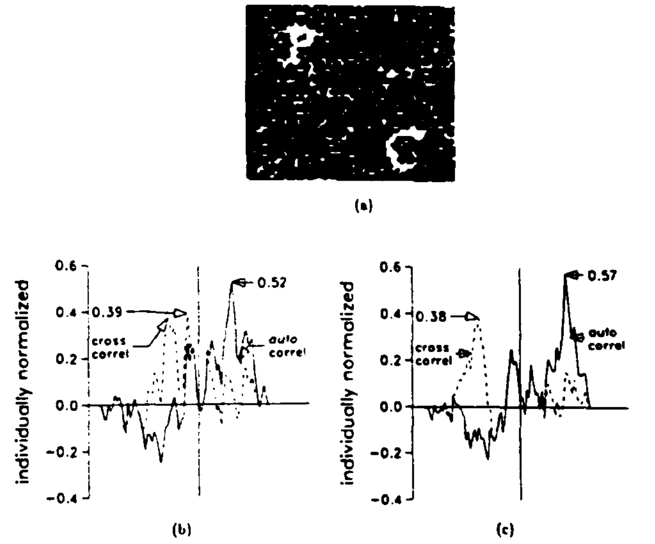


Fig. 7. Noisy projection cross-correlation results: (a) noisy input data; (b) horizontal projection correlation with a filter of C; and (c) 45° diagonal projection correlation with a filter of C.

projection correlations are used on data with noise present.

We formed a bank of 26 sets of three 1-D projection filters (horizontal, vertical, and diagonal projections) for each of the characters and tested all 26 zero-mean input characters with noise added against all 26 filter sets. Although the use of diagonal projection filters improved the difference between true and false peaks, perfect recognition resulted using only the horizontal and vertical projection correlations. For each filter pair, the largest pair of correlation peaks was measured and the peak values were summed. The filter set (from all 26) with the largest output peak sum was chosen as the class of the input test object. The system gave perfect 100% correct classification performance for all 26 characters for noise with $\sigma = 0.44$. In general, the vertical projection correlations had more discrimination than the horizontal projection correlations. At $\sigma = 0.45$, two errors resulted, and for increased noise the errors rapidly increased. We note that $\sigma = 0.44$ is a quite significant noise level.

IV. Optical Realization

Our projection correlation algorithm requires only 1-D correlations. Thus it is easily implemented using AO devices. Figure 8 shows a multichannel AO space integrating correlator. The test input is fed to an AO cell at P_1, which is imaged horizontally and expanded vertically at P_2 where N 1-D reference patterns are stored. The outputs on N detectors at P_3 as a function of time are the N 1-D correlations of the P_1 input and the N references. The filter bank at P_2 can be fixed. One can also frequency-multiplex several 1-D filters at each location in P_2 and input several 1-D signals at P_1 (several projections, each on a different carrier frequency). As proved elsewhere,[11] only the correlations of input and reference signals on the same frequency carriers will land on the detectors. In our case, this frequency selectivity allows us to form N sets of multiple correlations and to sum the results from each set of correlations. This is the discrimination method we used in our tests in Sec. III.

Should one desire adaptive filters, the fixed film mask at P_2 can be replaced by a multichannel AO cell. However, a more cost-effective adaptive filter architecture is the multichannel time-integrating correlator of Fig. 9. In this system, each point modulator at P_1 uniformly illuminates a single-channel AO cell at P_2 at different angles. P_2 is imaged horizontally onto P_3, and P_1 is imaged vertically onto P_3. The 2-D detector array at P_3 time integrates, and its N 1-D outputs are the N correlations of the input test projection fed to P_2 and the N 1-D reference filter functions fed to the N point modulators at P_1.

V. Discussion

We have advanced a simplified 2-D pattern recognition algorithm that requires only several 1-D projection correlations. Simulations have shown that projection correlations provide sufficient SNR and PSR to allow recognition and discrimination. We quantified the performance for a 26-class character recognition problem and showed that perfect recognition and discrimination was possible with a large input noise standard deviation of 0.44. The realization of such processors on multichannel AO correlators was described. Frequency multiplexing was noted to enhance the capability of such systems and to allow summation of the outputs from multiple filters in one set onto a single detector. Such AO correlators can also achieve bipolar correlations. This is essential for projection correlations to perform well. As noted, one can

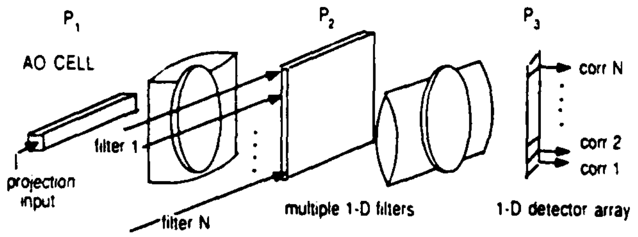


Fig. 8. Multichannel AO space-integrating projection correlator.

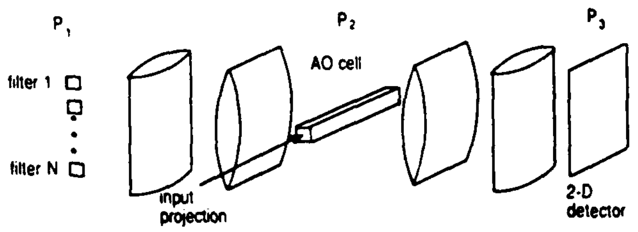


Fig. 9. Multichannel AO time-integrating projection correlator.

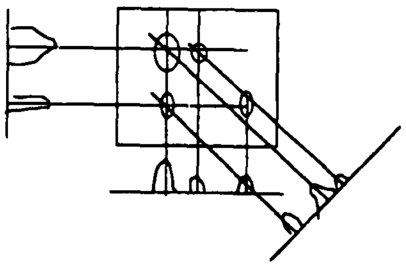


Fig. 10. Use of multiple projections to identify multiple objects and resolve ambiguities when two objects have the same horizontal and vertical coordinates.

rectify the negative portion of the correlation output, form only the positive portion of the bipolar correlation, and obtain improved results.

Although we were able to discriminate 26 classes of object, the cross-correlation values were quite close to the autocorrelation peak values. The use of various smart filters to increase the separation between classes would be recommended in practice when operating on projection images.

When multiple objects are present in the input, multiple correlation peaks occur. In this case, more than two projections are required. In general, we require $N - 1$ projections for N objects. The number of objects can be estimated from the energy in the correlation plane. Figure 10 shows the use of three projection correlations to locate uniquely four objects.

In conclusion, the possibilities for projection correlations appear to be quite significant and promising.

The support of this research by Teledyne Brown Engineering is gratefully acknowledged.

References

1. D. Casasent and W. T. Chang, "Correlation Synthetic Discriminant Functions," Appl. Opt. **25**, 2343 (1986).
2. A. Mahalanobis, B. V. K. Vijaya Kumar, and D. Casasent, "Minimum Average Correlation Energy (MACE) Filters," Appl. Opt. **26**, 3633 (1987).
3. Y. N. Hsu and H. H. Arsenault, "Optical Pattern Recognition Using Circular Harmonic Expansion," Appl. Opt. **21**, 4016 (1982).
4. K. Mersereau and G. M. Morris, "Scale, Rotation, and Shift Invariant Image Recognition," Appl. Opt. **25**, 2338 (1986).
5. F. T. S. Yu, *Optical Information Processing* (Wiley, New York, 1983), pp. 264–270.
6. S. R. Deans, *The Radon Transform and Some of Its Applications* (Wiley, New York, 1983), pp. 1–50.
7. A. Rosenfeld and A. C. Kak, *Digital Picture Processing I* (Academic, New York, 1982), pp. 365–369.
8. D. Ros, I. Juvells, S. Vallmitajana, and J. R. De F. Moneo, "A Simplified Method for the Automatic Alignment of Images Affected by a Random Noise," Opt. Acta **31**, 1151 (1984).
9. K. T. Ma and G. Kusic, "An Algorithm for Distortion Analysis in Two-Dimensional Patterns Using Its Projections," in *Proceedings, Seventh New England Bioengineering Conference* (1979), p. 177.
10. P. Breuer and M. Vajita, "Structural Character Recognition by Forming Projections," Probl. Control Inf. Theory **4**, 339 (1975).
11. D. Casasent and J. Lambert, "General I and Q Data Processing on a Multichannel AO System," Appl. Opt. **25**, 3217 (1986).

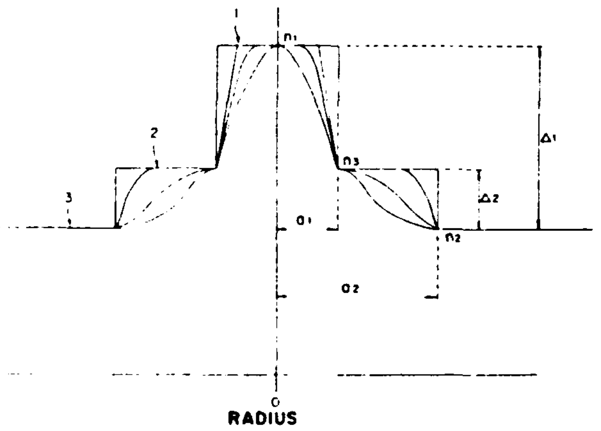
Patents *continued from page 4794*

4,755,022 5 July 1988 (Cl. 350–96.33)

Zero dispersion single mode optical fiber with center core and side core in the 1.5 μm wavelength region.

M. OHASHI, N. KUWAKI, S. SEIKAI, N. UESUGI, and C. TANAKA. Assigned to Nippon Telegraph and Telephone Corp. Filed 29 Aug. 1986 (in Japan 2 Sept. 1985).

Various index profiles are shown that minimize dispersion and bending effects for single-mode fibers. G.L.M.



4,755,038 5 July 1988 (Cl. 350–347 V)

Liquid crystal switching device using the Brewster angle.

A. P. BAKER. Assigned to ITT Defense Communications. Filed 30 Sept. 1986.

This two-input, two-output device completely switches collimated, unpolarized input light by separating it into two orthogonally polarized beams whose polarization is then either rotated by 90° or left unchanged by the action of an (unspecified) liquid crystal cell. This device is identical to the unpatented switch of R. E. Wagner and J. Cheng [Appl. Opt. **19**, 2921 (1980)], except that the polarizing beam splitters are specified to be "Brewster angle polarizers," and are arranged in an especially compact manner. M.H.

4,756,589 12 July 1988 (Cl. 350–96.15)

Optical coupler utilizing low or zero birefringence optical fibers and a method of making same.

T. BRICHENO and V. BAKER. Assigned to STC plc. Filed 15 Jan. 1986 (in United Kingdom 12 Feb. 1985).

A coupler is made by a normal twisting and fusing process. Individual legs are pulled to control polarization through the coupler. G.L.M.

4,756,601 12 July 1988 (Cl. 350–130)

Three-dimensional image-viewing apparatus.

W. SCHRODER. Assigned to Jos. Schneider Optische Werke Kreuznach GmbH & Co. KG. Filed 15 Nov. 1985 (in F. R. Germany 15 Nov. 1984).

An apparatus is described for centrally projecting the image of an object formed from two images in different lines of vision. The system consists of two field lenses located at luminous objects, such as CRTs, two objective lenses, and mirrors for folding the ray paths located between the images and the positive optical system located in front of the eyes of the viewer. The stereoscopic viewing apparatus allows a relatively large viewing distance and unusually large exit pupils. S.K.T.

4,756,604 12 July 1988 (Cl. 350–331 R)

Liquid crystal device using a Fresnel collimating lens for improving uniformity of display contrast ratio.

H. NAKATSUKA and M. KANAZAKI. Assigned to Hitachi, Ltd. Filed 2 July 1987 (in Japan 4 July 1987).

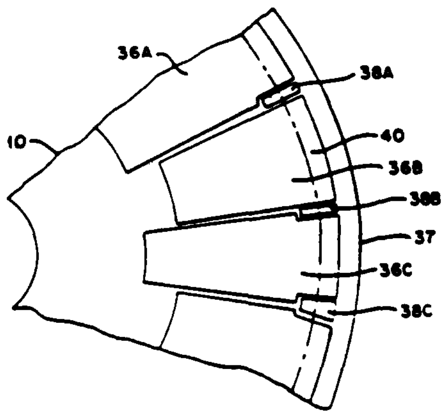
The typical liquid crystal device's variation of optical transmission with changing incident angle causes nonuniformity in the projected image when such a device is used for image generation, e.g., on an overhead projector. The present invention corrects this by collimating the illumination with Fresnel lenses. M.H.

4,758,058 19 July 1988 (Cl. 350–3.71)

Holographic disk scanner having special position-indicating holograms.

R. T. CATO, L. D. DICKSON, and R. S. FORTENBERRY. Assigned to International Business Machines Corp. Filed 12 Jan. 1987.

This patent describes a holographic scanner for point of sale applications. The scanner is a rotating glass disk carrying holograms for generating many scanned lines. The improvement in this patent is the addition of a small hologram in between the scanning hologram to generate position-indication signals. W.-H.L.



CHAPTER 3

"Optical Track Initiator for Multitarget Tracking"

Optical track initiator for multitarget tracking

David P. Casasent and James Slaski

An optical processor for multitarget track initiation is presented. It uses the Hough transform to detect target tracks in a fixed time independent of the number of tracks present. In new results, we describe how the system produces target position and velocity estimates in linear time (with the number of targets). Tests of the system are presented for multiple targets with various velocities and target detection probabilities. A new thresholded Hough transform algorithm to reduce the effects of noise and false tracks in single data frames is introduced and tested. The optical implementation of these Hough transform track initiation algorithms is discussed.

I. Introduction

Multitarget tracking algorithms have been under study and in use for a number of years for a number of applications, such as civilian air traffic control and military air defense systems. More recently, emphasis has been placed on the application of multitarget tracking procedures to situations involving large number of targets, such as the tracking intercontinental ballistic missiles. There are a number of factors which make this tracking task more difficult than those that have traditionally been addressed. The first difficulty is the large number of targets which must be tracked at one time, approximately 3000 separate targets.[1] This presents a difficulty since the computational load on the tracking algorithms increases rapidly with the number of targets tracked, and most present systems handle a much smaller number of targets, such as the U.S. Navy's Aegis ship-defense system which is designed to handle 200 targets.[1]

There is a class of tracking algorithm, such as the JPDA tracker,[2-4] whose computational load does not increase rapidly with increasing numbers of targets but which unfortunately suffers from the inability to initiate tracks on new targets by itself. For this reason, a separate track initiator is required, but most track initiators are based on standard tracking techniques which are slowed down by large computational loads when presented with many targets. In this work, we present a track initiation system that makes use of optical processing to perform track initiation on a large number of targets in linear time. The system makes use of the Hough transform to detect target trajectories (in a time independent of the number of targets) along with the use of a small amount of postprocessing (that is linear in the number of targets) to determine the target's position and velocity.

We consider the normal parametrization[5] of the Hough transform (ρ,θ) space where ρ is the normal distance of the line from the origin and θ is the angle the normal makes with the x axis. Extensions of the Hough transform to other shapes besides lines have been discussed using generalized Hough transforms[6-8] and processing in the Hough transform space followed by an inverse Hough transform.[6] We consider a portion of the target's flight over which the trajectory is nearly a straight line. Thus a straight line Hough transform is sufficient. There has been some work[6,7,9] on the application of the Hough transform to detecting moving target tracks. This work requires one object per region[9] or a high dimensional Hough space and an extensive search,[7] and none of it considers many tracks or does it provide position and velocity information.[6]

In Sec. II, we discuss the use of the Hough transform to detect target tracks in images, and we detail the Hough space parameters used. In Sec. III we show how to obtain target position and velocity estimates. Simulations of the performance of this track initiation system are then presented in Sec. IV. In Sec. V, we present a thresholded Hough transform which can be used to reduce noise in tracking applications of the Hough transform. In Sec. VI, we develop (and verify through simulations) equations for estimating the covariance of the position and velocity measurements. In Sec. VII, we present an optical architecture for performing the Hough transform and the correlations required by the track initiator.

The authors are with Carnegie Mellon University, Center for Excellence in Optical Data Processing, Department of Electrical & Computer Engineering, Pittsburgh, Pennsylvania 15213.

Received 28 December 1987.
0003-6935/88/214546-08$02.00/0.

II. Hough Transform

As a first step in track initiation, the Hough transform is used to locate lines in a composite image containing the superposition of candidate target detections in a series of frames. The targets we will consider, boost phase ballistic missiles, have curved trajectories which can be approximated as straight lines over reasonable time intervals. The straight line Hough transform produces a mapping of lines in the input image to points in a Hough parameter space. The Hough coordinates (ρ,θ) of peaks in this parameter space give the (ρ,θ) parameters of all lines in the input image, while the intensity of the Hough peaks indicates the number of input points on each line.

A. Hough Transform Definition

The Hough transform is produced by mapping each input point (x,y) into the Hough space (ρ,θ) according to

$$\rho = x \cos\theta + y \sin\theta. \tag{1}$$

This mapping produces a sinusoid. All sinusoids for points on a line accumulate at the same point in the Hough space. The HT can also be defined[10] as the integral

$$H(\rho,\theta) = \int_{-\infty}^{\infty} f(\mathbf{r})\delta(\rho - \mathbf{r}\cdot\hat{n})d^2\mathbf{r}, \tag{2}$$

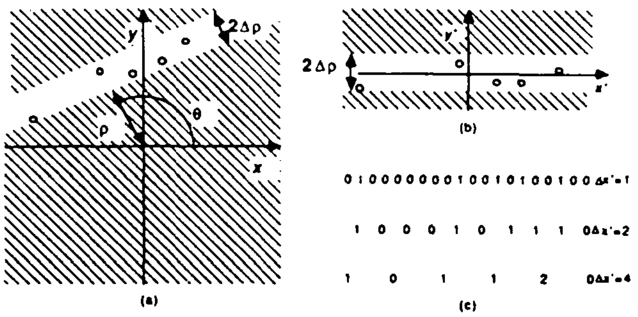
where $\mathbf{r}$ is a vector to the (x,y) point in the image being transformed and $\hat{n}$ is a unit vector making an angle θ with the x axis. The ρ projection lies along $\hat{n}$. Each θ slice of the Hough transform is a projection of the input image to a 1-D function in ρ. This is seen by writing Eq. (2) as

$$H(\rho,\theta) = \int_{-\infty}^{\infty} f(x,y)dy', \tag{3}$$

where $\rho = x \cos\theta + y \sin\theta$ and $y' = -x \sin\theta + y \cos\theta$. The HT at a single θ value is thus simply the integral of the input image along the direction y' given by the value of θ. The fact that the Hough transform is composed of a number of projections of the input image at different angles θ is central to most optical HT implementations.

B. HT Parameter Selection

We now consider the total time T_T over which we will collect measurements for our composite image. This relates to the number of points on a line or the length of a line. It is best to choose T_T so that the target tracks will cover a large portion of the input image frame, since the Hough transform performs best when this is true. This occurs because, if the line is short, the points on the line will produce sinusoids as in Eq. (1) in the Hough plane, which are almost exactly the same, and the peak that is produced will be spread out over a considerable length of the sinusoid. This produces a shape in the Hough transform which is a plateau rather than a peak, thus making detection difficult. If the target track is longer, the sinusoids for each of its points are more different and the Hough transform peak they produce will be sharper. We chose T_T so that the highest velocity targets cover half of the input image. This allows the Hough transform to detect targets easily while insuring that they stay within the field of view long enough to be detected. The images used in simulations were 256 × 256 pixels in size with image frames at sample times of $T = 0.25$ s. The composite frame we consider covers a period of $T_T = 5.25$ s and produces 21 points/track.



y
$2\Delta\rho$
ρ
θ
x
(a)
$2\Delta\rho$
y'
x'
(b)
0 1 0 0 0 0 0 0 0 1 0 0 1 0 1 0 0 1 0 0 $\Delta x'=1$
1 0 0 0 1 0 1 1 1 0 $\Delta x'=2$
1 0 1 1 2 0 $\Delta x'=4$
(c)

Fig. 1. Projection of track lines for velocity and position estimation.

We now consider selection of the $\Delta\rho$ and $\Delta\theta$ used in the Hough space resolution so that the HT peaks are not spread. Since the actual target tracks are curves, we select $\Delta\rho$ so that it exceeds the size of the linearity error for the line. The linearity error is the perpendicular distance of target measurements from the straight line through the endpoints of a set of trajectory points over T_T. This $\Delta\rho$ value is obtained off-line from sample data from representative trajectories from given sensor platforms. This technique places no constraints on $\Delta\theta$, while other methods do.[11] Based on the linearity error in the trajectory, a $\Delta\rho$ value of 2 was chosen and $\Delta\theta$ was set to 1°. The full range of ρ, $\pm128\sqrt{2}$ or ±180 (length of the diagonal), dictates the use of 180 Hough transform samples in ρ. Since we are sampling the Hough transform over $\pm\rho$ we need to sample θ over only 0–180° due to the dual line descriptions possible with the normal parametrization. If only positive values of ρ were considered, it would be necessary to sample θ over 0–360°.

III. Target Position and Velocity Estimation

We now consider the use of HT data to determine both the position and velocity of the target. This information is needed to better initialize a tracking filter. Consider one detected target track. We overlay the line and the original data. We expect the target points to lie within $\Delta\rho$ of this line as shown in Fig. 1(a). In terms of the coordinate system (x',y') with x' aligned with the line detected by the Hough transform, we write

$$\begin{aligned} x' &= x \sin\theta + y \cos\theta, \\ y' &= -x \cos\theta + y \sin\theta - \rho. \end{aligned} \tag{4}$$

This strip of the composite target image along the x' axis with extent $\pm\Delta\rho$ in y' is shown in Fig. 1(b). We integrate this strip in the y' direction and store this integral or projection in a 1-D memory array. The first line in Fig. 1(c) shows the target samples obtained (1 and 0 indicate measurements and no measurements) for the line with $\Delta x' = 1$ resolution. From this array we can determine the target's position and its average velocity as we now detail with reference to Fig. 1(c).

For simplicity, we assume a constant target velocity. The effect of this assumption when the targets constantly accelerate is quantified in Sec. IV. To estimate the target's position and velocity, we vary the resolution $\Delta x'$ along x' as shown in Fig. 1(c). With our constant velocity assumption and with no missing target detections and no noise, the final projection in Fig. 1(c) (for the proper resolution $\Delta x'$) would result in a sequence of ones for a target track. In practice, this does not occur, and thus the algorithm we use involves increasing $\Delta x'$ until we have a sequence of adjacent projection samples P_j that are nonzero. The second line in Fig. 1(c) shows the P_j samples obtained for $\Delta x' = 2$ (by simply adding each pair of sample in the $\Delta x' = 1$ projection). We continue to increase $\Delta x'$ until the sum of adjacent nonzero P_j values is $\geq 75\%$ of the number of ones in the original ($\Delta x' = 1$) projection. For the example in Fig. 1, the $\Delta x' = 1$ projection has five ones and at $\Delta x' = 4$ we have three adjacent P_j that are nonzero and whose sum is 4 (80% of 5). This sum is obtained by correlating the P_j sequence with a pulse of width L (the track length, three here, at the given $\Delta x'$ resolution). This correlation is performed for different L choices. To insure that a contiguous positive P_j is found, $P_j = 0$ values are set to -10 when doing the correlation. We denote the correlation peak value by I (it is the number of detected targets in a range L), and I/L is the target density (number of targets in a resolution cell $\Delta x'$).

We denote the location of the correlation peak by j (the index of the present P_j sample sequence). The reference pulse we correlate with has its index j chosen so that $j + 1$ denotes its leading edge. Thus $j\Delta x'$ is an estimate of the target's position. To better estimate where the target is within the last $\Delta x'$, we use the target density I/L and the P_j value for the most recent target sample, and as our position estimate we use

$$x' = \Delta x'[j + (P_j - 0.5)L/I], \tag{5}$$

where $P_j/(I/L)$ denotes the edge of the region where the target lies and $(P_j - 0.5)/(I/L)$ denotes its center.

The velocity of the target is determined from the average target density I/L. This denotes how many frames it takes the target to travel a distance $\Delta x'$ (on the average). For example, if $I/L = 5$ and $\Delta x' = 2$, the target travels two pixels in five frames and $\Delta x'L/I = 0.4$ pixels/frame. Dividing by T, we obtain the target's velocity in pixels per second. Thus our velocity estimate is

$$v_t = \Delta x' \frac{L}{I} \frac{1}{T} \frac{\text{pixels}}{\text{second}}. \tag{6}$$

IV. Target Trajectory Tests for Target Position and Velocity Estimates

We now quantify the use of the HT track initiator for target detection and position and velocity estimation. To test this algorithm we produced a set of fifty missile launches with different trajectories and launch positions. Only six missile trajectories are present in the 256- × 256-pixel image frame we used. Twenty-one frames of data (0.25 s between frames) were produced to constitute 5.25-s trajectories. Each image frame included a Gaussian-distributed motion jitter with a standard deviation of $\sigma_j = 0.5$ pixels. This varied the location of each of the six single pixel binary target points in each frame. Five noise pixels were added, uniformly distributed within each frame. Thus each frame is a binary image with 6 single pixel targets and 5 noise pixels. We produced the composite of twenty-one frames and form the Hough transform of this image, detect lines in it, and obtain the position and velocity of each target by the method presented in Sec. III. In our data results (Figs. 2 and 3), squares denote true targets and × denotes a false target measurement. In all composite frame displays, such as Figs. 2 and 3, only every fourth frame is included to make the composite shown clearer (although all frames are included in processing). In the field of view for the case in Fig. 2, five of the target trajectories are nearly parallel and the sixth one is more vertical with more closely spaced target samples (corresponding to a more recently launched missile). The dashed lines in Figs. 2 and 3 denote target tracks selected. The tip of the arrow denotes the target's estimated position 2 s (8 frames) later. The target's actual position 2 s later is plotted as a blackened square for comparison. As seen, all target tracks were found with no false tracks, and the estimated position and velocity for each target are close to the actual position and velocity. In all cases, we find that the estimate of the future target position is less than the actual position. This occurs because the targets are constantly accelerating and our estimate assumes a constant velocity. An analysis of the accuracy of the track estimates is presented in Sec. VI.

Figure 3 shows the same results, with the probability of target detection reduced from unity to $P_D = 0.9$. This was accomplished by using the value of a random number between 0 and 1 generated for each target point to determine if that target point was included. If the random number was above 0.9, the target point was omitted. From the results in Fig. 3, we see that the estimated velocity is higher than before. This is because the missing target points (caused by reducing P_D) make it appear that the target is moving faster than it actually is (since the detected track length is the same and less target measurements are detected). Three missing data points (frames 4, 16, and 20) are present in the track in the top center of the figure. Point 0 is missing from the track left of center. The other missing points are not apparent since they are present in the frames of data not plotted (target points for only six of the twenty-one frames are plotted). If an estimate of the probability of detection is available, we can compensate for $P_D < 1$ by modifying the velocity estimate to be

$$v_t = \Delta x' P_D \frac{L}{I} \frac{1}{T}. \tag{7}$$

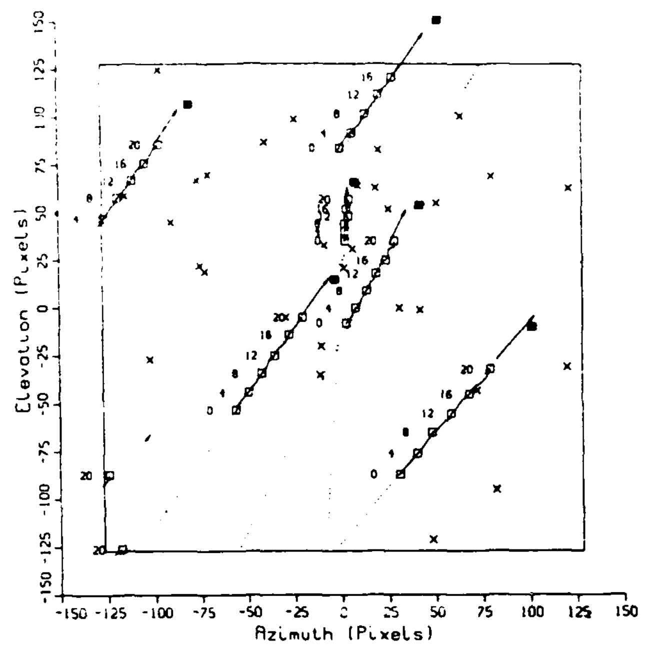


Fig. 2. Hough transform track initiation results for twenty-one data frames at 0.25 s/frame with 5 noise pixels/frame and target detection probability $P_D = 1.0$.

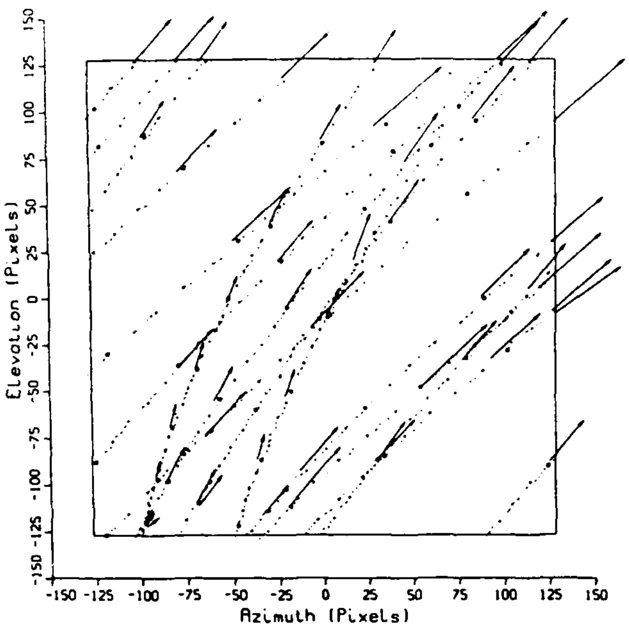


Fig. 4. Continuous track initiation results over 120 s in 5-s segments with 5 noise pixels/frame and target detection probability $P_D = 1.0$.

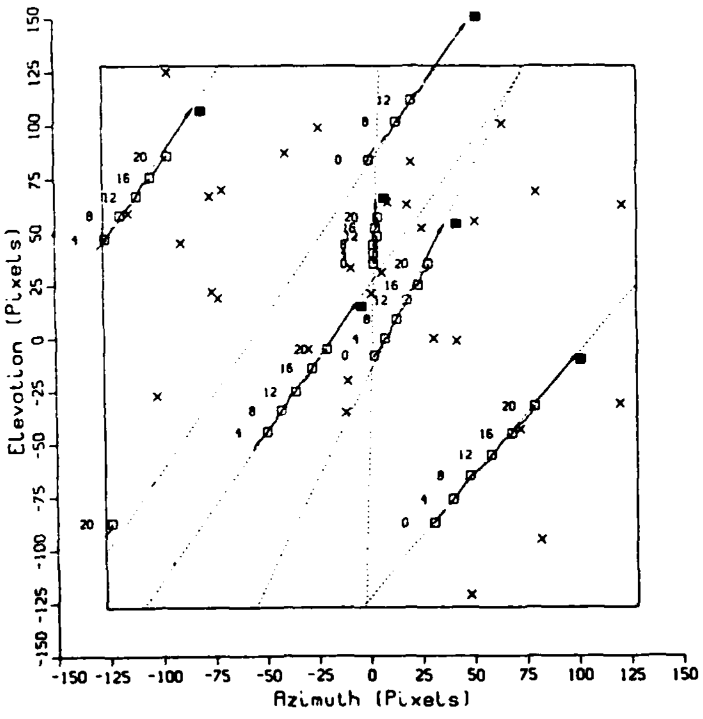


Fig. 3. Hough transform track initiation results for twenty-one data frames at 0.25 s/frame with 5 noise pixels/frame and target detection probability $P_D = 0.9$.

The track initiator requires a fixed amount of time to form the Hough transform and detect the target tracks. Thus target detection is independent of the number of targets. The total time required to obtain the position and velocity estimates is linear in the number of targets. In Fig. 4, we show the results of applying the track initiator to 120 s of target data. The dotted lines show the actual target trajectories with the dark points denoting the target locations at 1-s intervals. The spacing between dark spots indicates the velocity and aspect of the trajectory. The frame rate is $T = 0.25$ s, and 5 noise pixels (which are not shown) were added to each frame. The Hough transform track initiation was applied to the input for a 5-s interval and then repeated for subsequent 5-s intervals (a total of 120/5 = 24 times). The detected targets are shown by the arrows in the figure. The target's velocity and time of launch determine the number of 5-s intervals (and hence the number of arrows expected on each target trajectory). We ideally expect an arrow for each five dark spots per trajectory. As seen, all fifteen targets which traversed the image were detected by the track initiator at least once. Out of forty-nine total possible detections (one for each 5 s), only four were missed and each track has at least one detection.

V. Thresholded Hough Transform

The data in Figs. 2–4 merit further discussion. Since we base track initiation and confirmation on the presence of a number of candidate target hits in a line, the algorithm is not significantly affected by several random observations close to the actual observations. Similarly, when the tracks from two targets cross, our algorithm extracts two straight lines and is not affected by crossing targets. This occurs since it is a track initiator and not a tracker, with its outputs (target state estimates) fed to a multitarget tracker (see Sec. I references). When two target tracks are very closely spaced, this track initiator will assume one target (and pass this information onto the tracker), and when the target is separate, it will notice the presence of two tracks (and pass the new track information to the tracker). Figures 2–4 (especially Fig. 4) show examples of these cases. We now consider the effect of very high noise levels and a modified Hough transform to improve results in this case.

The Hough transform for a composite image can yield false detections of target tracks in one frame. Such false tracks can be due to cloud edges, targets in one frame that fall in a line, etc. Keeping a time tag on the frame number of each data would reduce this effect (at a considerable increase in bookkeeping). To achieve this at a reduced computational load, we form the Hough transform for each image frame of potential targets. We threshold this Hough transform frame and add such successive frames. A threshold of the Hough transform for each data frame reduces false target response without increasing the processing load. This technique reduces (and effectively removes) the contribution of straight line tracks in one frame to only one unit of weight in the final Hough transform space. The Hough transform thresholding can be performed by various optical light modulators.

We demonstrate the results of this technique in Figs. 5 and 6. In Fig. 5, we show the output from the Hough transform of a composite set of twenty-one image frames with five targets per frame and twenty noise points distributed uniformly in each frame. We see that all five true target tracks (shown by the solid lines) were detected, but that nine lines (target tracks) due to noise were also detected (shown by the dashed lines). Our projection analysis algorithm (Sec. III) to estimate target position and velocity was applied to these nine false tracks. The largest $\Delta x'$ resolution value used was 16 pixels. [This was set by the highest velocity (7.5-pixel/frame) target expected and corresponds to target points that are the farthest apart.] With this $\Delta x'$ limit and the requirement that the projection correlation peak value I exceed 75% of the number of ones in the initial $\Delta x' = 1$ projection sequence, seven of these nine false tracks were rejected and only two were treated as confirmed target tracks. These two occur on the right center edge of the figure and are denoted by arrows on the ends of the dashed lines. In Fig. 6, we show results for the same data when the threshold Hough transform was used. In this case, the number of detected false target tracks was reduced from nine to four and the number of false target tracks confirmed was reduced to one. This represents a reduction in false detections of 50%.

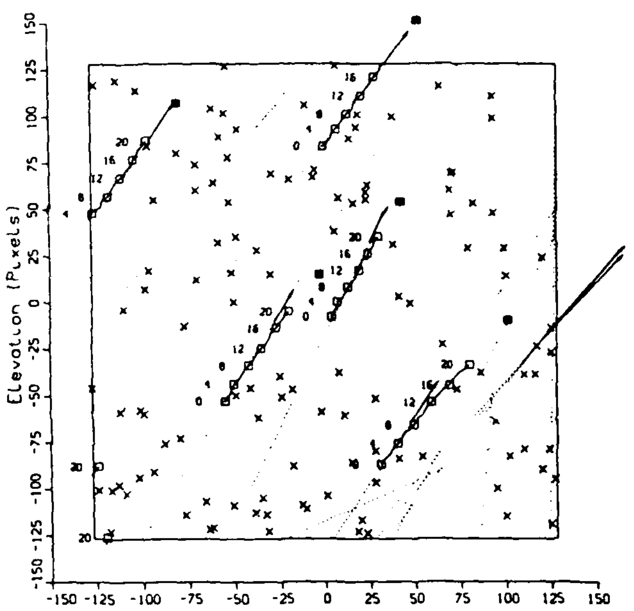


Fig. 5. Hough transform track initiation results for twenty-one data frames at 0.25 s/frame with 20 noise pixels/frame and target detection probability $P_D = 1.0$.

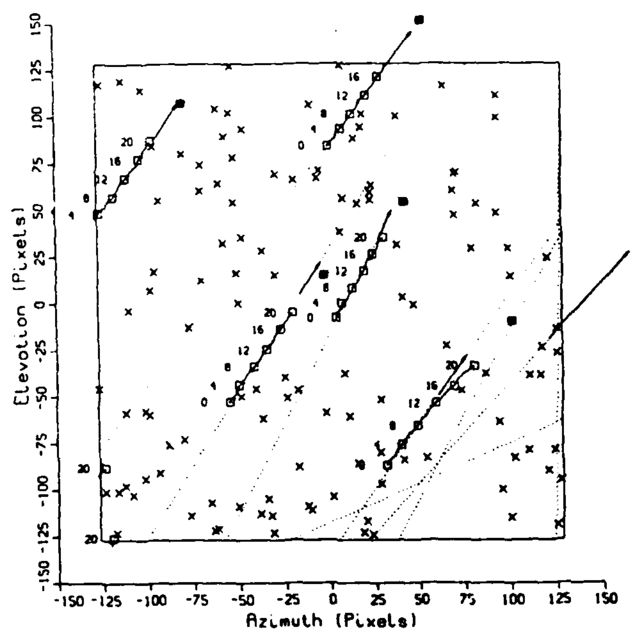


Fig. 6. Thresholded Hough transform track initiation results for twenty-one data frames at 0.25 s/frame with 20 noise pixels/frame an target detection probability $P_D = 1.0$.

VI. Track Accuracy

To use these track state estimates as input to a tracking filter, we require an estimate of the state covariance. In this section, we approximate the covariance based on the major error sources due to the Hough transform resolution $\Delta\rho$, quantization errors due to the $\Delta x'$ projection resolution, missing measurements, and platform jitter. In each case, we give the major error sources affecting each estimate and a covariance estimate that will be close to or larger than the actual covariance; i.e., we assume a larger error than actual.

We first consider error effects on the target's x' position. To obtain x', the track initiator performs a correlation on the target track projection and fixes the location of the target peak as described in Sec. III. This value includes an estimate (using the target's average speed) of how far into the next projection sample the target has moved. This gives us the target's position to within a distance $v_{x'}T$ (the amount the target would move over the sample time T). Any angular error between the track and projection does not enter into the x' error, since we are measuring x' directly. These errors do affect the y' measurement. Our formula assigns the target's position to the center of the window shown. The standard deviation associated with this x' position estimate is plus or minus half of the distance $v_{x'}T$, is distributed uniformly, and contributes a variance equal to the square of the error divided by 3. There is also an error term σ_j^2 due to sensor jitter giving a total variance of

$$\sigma_{x'}^2 = \frac{v_{x'}^2}{4(3)} + \sigma_j^2. \tag{8}$$

Missing measurements are included in $P_D \neq 1$, and P_D effects are included in the $v_{x'}$ estimate in Eq. (8) as in Eq. (7). We do not include the statistics for this here.

The error in the y' measurements is easy to bound, since we assume that the target must fall into the strip of width $\pm\Delta\rho$. The y' position becomes offset when the Hough parameter θ does not match the actual track angle. However, we only need to know the projection width in y' to estimate the y' error. With a uniform distribution, the y' estimate has a variance

$$\sigma_{y'}^2 = \frac{\Delta\rho^2}{3}. \tag{9}$$

We now consider error source effects on velocity estimates. The velocity of the target is determined by dividing the length L of the detected target track by the number of target points N_p that fall in this track. Due to the quantization of the projection, the track length could be in error by $\pm v_{x'}T$, the distance the target could travel in one sampling interval. Assuming that this error is uniformly distributed, the estimated track length $\Delta x'L$ in Eq. (6) becomes $\Delta x'L \pm v_{x'}T + e_j$, where e_j is the position error due to sensor jitter. This gives an x' velocity estimate

$$v_{x'} = \frac{\Delta x'L \pm v_{x'}T + e_j}{N_p}\frac{1}{T}. \tag{10}$$

We assume a point target for each sample. This gives a velocity error of

$$e_{v_{x'}} = \frac{\pm v_{x'}T}{N_p}\frac{1}{T} + \frac{e_j}{N_pT} = \frac{\pm v_{x'}}{N_p} + \frac{e_j}{N_pT}. \tag{11}$$

The variance of this error is

$$\sigma_{v_{x'}}^2 = \frac{v_{x'}^2}{3N_p^2} + \frac{\sigma_j^2}{N_p^2T^2}. \tag{12}$$

Next, we consider the target's velocity $v_{y'}$ perpendicular to the track. This is taken to be zero, since it is assumed that the Hough transform line (which is the basis for the x' axis) is exactly aligned with the target's motion. Since any error in the Hough transform invalidates this assumption, we need to know what the possible $v_{y'}$ error is. We first find the maximum y' velocity that the target can have and still remain in the image strip of width $2\Delta\rho$ for the entire time T_T used to form the composite image. The maximum y' velocity is this width divided by the time T_T. Assuming that the velocity error is uniformly distributed between plus and minus this maximum velocity, we obtain a variance

$$\sigma_{v_{y'}}^2 = \frac{4\Delta\rho^2}{3T_T^2}. \tag{13}$$

Since these variance estimates are only approximations, we checked how closely they represent the actual variances. We did this by running fifty simulations and calculating an experimental covariance based on the postion and velocity errors for each run of the simulation. Specifically, we simulate fifty targets with initial positions uniformly distributed in x and y within the lower left quadrant of the image and moving in a straight line at angles evenly distributed between θ = 90 and 180° with a specified constant velocity. We performed a Hough transform on each of the fifty target tracks, obtained the line parameters from each Hough transform, and then obtained the target position and velocity from the projection of each Hough transform line. The position and velocity values obtained were compared with the actual values, and an average error and squared error were calculated over all the simulation runs for different velocity and position variables. These were then used to calculate the covariances. The Hough transform was produced over T_T = 5 s at four sampling rates (T = 0.25, 0.1, 0.05, and 0.025 s). This was a total of N_p = 20, 50, 100, and 200 Hough transform times samples for each of the fifty cases.

In Figs. 7–10, we compare the experimental standard deviation results to the expected values calculated from our equations. We see that in all but one case the standard deviations are within the expected values. The $\theta_{x'}$ position error in Fig. 9 for V_t = 5 pixels/s is above the expected value for T = 0.025 and 0.05 s and is the only nonconsistent result. These results are useful in estimating the worst-case errors expected in the track initiator. For example, from the expected errors for the highest target velocity ($v_{\max}$ = 30 pixels/s), a frame time of T = 0.25 s, and a total time of T_T = 5 s for N_p = 20 frames, we obtain a combined x' and y' position error of 2.5 pixels and a maximum velocity error of 1 pixel/s.

VII. Optical Implementation

Now that we have shown the feasibility of using the Hough transform to perform track initiation, we discuss an optical architecture to produce the Hough transform and form the track projections shown in Fig. 1 and the correlations required on the projection data for target position and velocity estimates. This system uses the rotating prism Hough transform architecture.[10,12] In this system at the top of Fig. 11, light is projected through a spatial light modulator (SLM) at $P1$, on which the composite track image is written. The image is then passed through a Dove prism (which rotates the input image) and imaged onto a second SLM at $P2$. We ignore this SLM at $P2$ for now and assume that an image of $P1$ is incident on the spherical/cylindrical lens set $L2$, which form images in the x

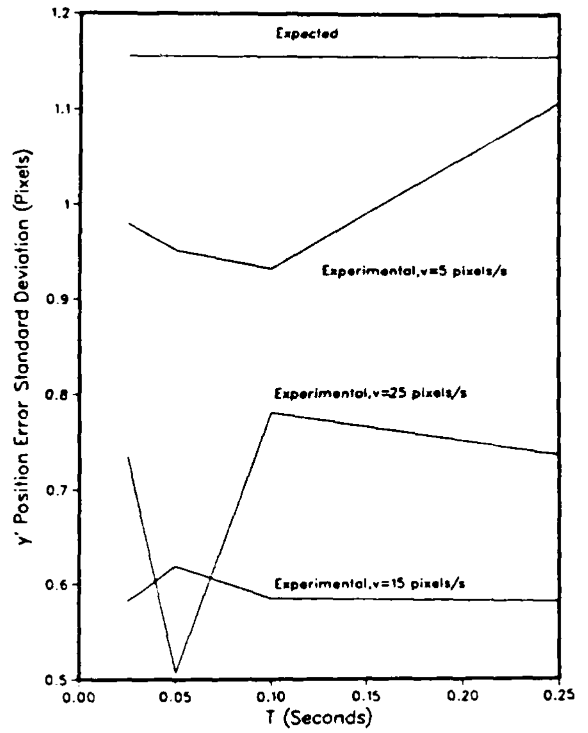


Fig. 7. Experimental results of the standard deviation $\sigma_{y'}$ of the track y' position.

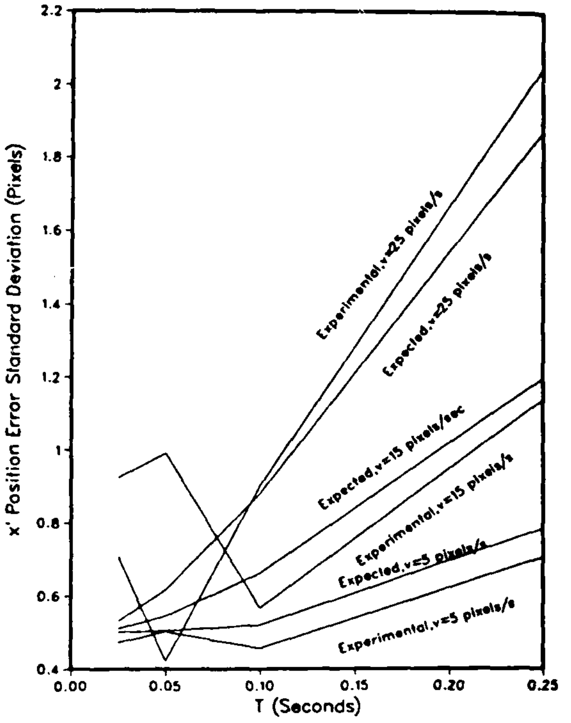


Fig. 9. Experimental results of the standard deviation $\sigma_{x'}$ of the track x' position.

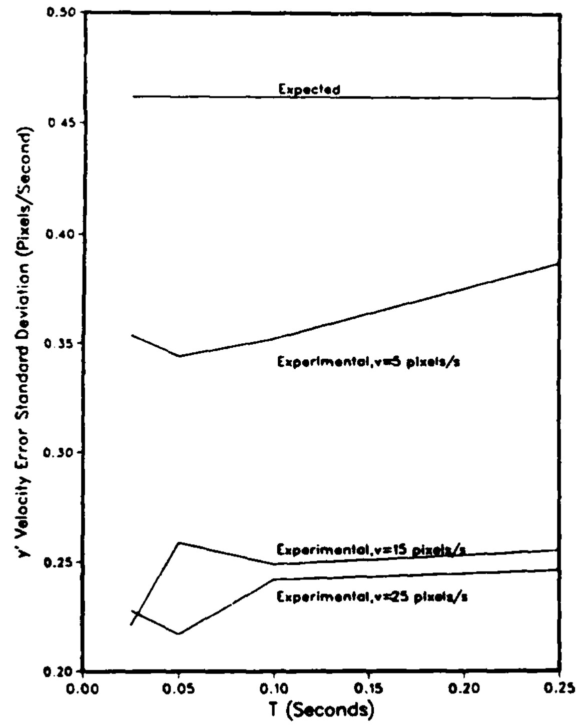


Fig. 8. Experimental results of the standard deviation $\sigma_{v_{y'}}$ of the track y' velocity.

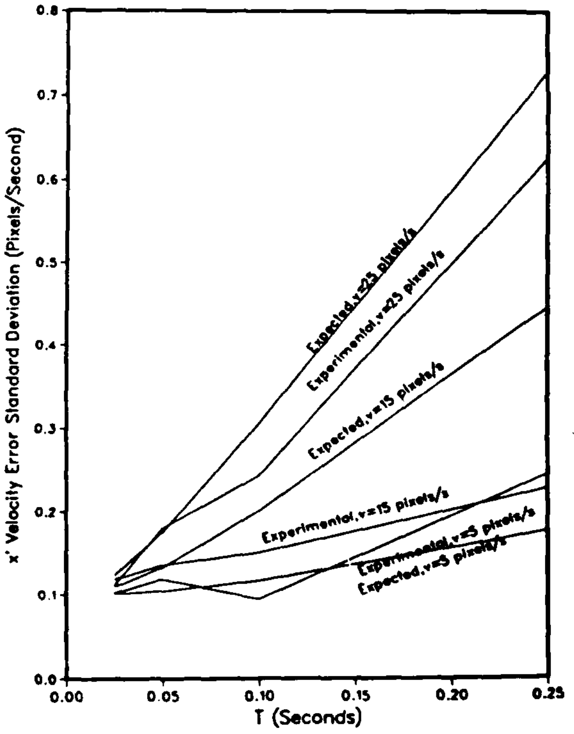


Fig. 10. Experimental results of the standard deviation $\sigma_{v_{x'}}$ of the track x' velocity.

direction and integrates in the y direction onto a linear detector array at $P3$. This detector output is the Hough transform at a single value of θ. The Hough transform for other values of θ are obtained by rotating the Dove prism.

The algorithm we found to be useful (and necessary) in detecting peaks in the Hough transform plane required a blurred Hough transform with Hough transform samples averaged over local 5×5 regions of the Hough space. This was necessary since the Hough

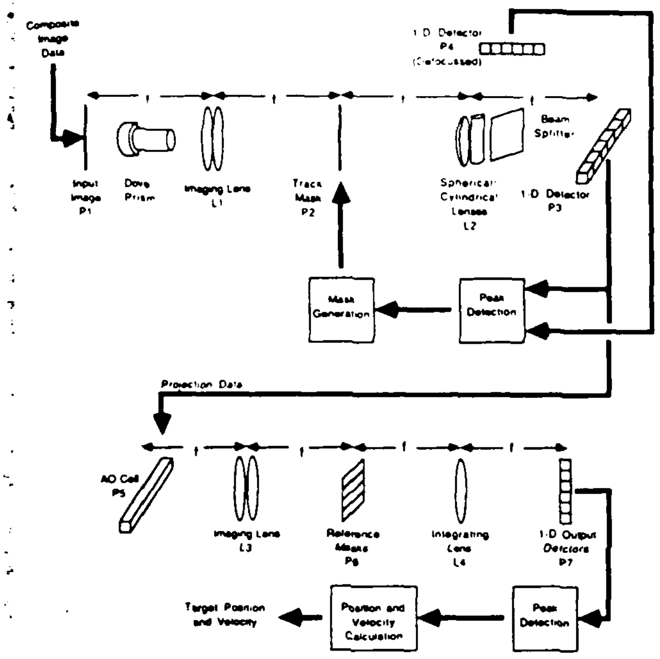


Fig. 11. Optical Hough transform architecture for track initiation.

transform peaks were found to consist of several peaks in a local neighborhood on a large bias. This blurred Hough transform pattern is produce at $P4$ by a beam splitter (BS) with the local 5×5 averaging achieved by displacing $P4$ from the exact back focal plane of $L2$. The Hough transform patterns at $P3$ and $P4$ are combined to allow the true Hough transform peaks to be detected in the peak detector box. Combining the two Hough transform outputs involves comparing values in the two Hough transform outputs. If the ratio of the exact to the defocused Hough transform value exceeds a threshold, the point is considered to be part of a peak. The threshold that we found to be useful was 3. Adjacent peak points are grouped together, and the center of the peak is calculated by simple centroid methods. In cases where the peak splits into seemingly individual peaks, a lower threshold (1.5) is used to fill the gap between the peaks. Even is this case, we require that at least one point be above the threshold of 3.

As noted earlier, the Hough transform is a collection of 1-D integrals of the input image at angles θ with respect to the y axis. The target track projections used in Fig. 1 are 1-D integrals of the masked composite image at an angle θ with respect to the x axis or an angle of $\theta - 90°$ with respect to the y axis, where θ is the Hough transform parameter of the detected line. In our optical realization, we rotate the target track data at $P1$ (by an amount determined from the θ value of the Hough transform peaks), and we write a horizontal line on the SLM at $P2$ (at a vertical position given by the ρ value of the Hough transform peak). With the dove prism at an orientation of $\theta - 90°$, the $P3$ output is now the desired projection data from which target position and velocity estimates can be obtained.

The correlations required on the projection data to obtain the target's position and velocity are easily realized on the space integrating AO correlator in the bottom of Fig. 11. This system correlates the projection data in an AO cell at $P5$ with image plane masks $M3$ at $P6$ (which contain 1-D reference pulses of different lengths L) on a linear detector array at $P7$. The detector with a peak denotes the pulse width. The time of occurrence of the peak denotes the location of the center of the set of target pulses. With proper modulation of the input data, the bipolar correlation required can be obtained. Simple postprocessing yields the target position and velocity information.

VIII. Conclusion

We have presented an optical track initiation system for multitargets based on the Hough transform. This system detects any number of targets in a fixed amount of time and provides target position and velocity in linear time. Calculations of the accuracy of the track initiation were given, and the performance and accuracy of the system were verified through simulation. An optical implementation of this system was presented. A new thresholded Hough transform was also presented which reduces the number of false target detections in noise.

This work was supported by a contract from the Strategic Defense Initiative Office of Innovative Science and Technology monitored by the Office of Naval Research.

References

1. J. A. Adam and P. Wallich, "Part 1—Mind-Boggling Complexity," IEEE Spectrum 22 (9), 36 (1985).
2. S. Tsuji and F. Matsumoto, "Detection of Ellipses by a Modified Hough Transformation," IEEE Trans. Comput. C-27, 777 (1978).
3. K. C. Chang and Y. Bar-Shalom, "Joint Probabilistic Data Association for Multitarget Tracking with Possibly Unresolved Measurements and Manuevers," IEEE Trans. Autom. Control AC-29, 585 (1984).
4. K. C. Chang, C. Y. Chong, and Y. Bar-Shalom, "Joint Probabilistic Data Association in Distributed Sensor Networks," IEEE Trans. Autom. Control AC-31, 889 (1986).
5. M. L. Padgett, S. A. Rajala, W. E. Snyder, and W. H. Ruedger, "Detection of Maneuvering Target Tracks," Soc. Photo-Opt. Instrum. Eng. Proc. (Aug. 1985).
6. T. E. Fortmann, Y. Bar-Shalom, and M. Scheffe, "Sonar Tracking of Multiple Targets Using Joint Probabilistic Data Association," IEEE J. Oceanic Eng. OE-8, 173 (1983).
7. G. R. Gindi and A. F. Gmitro, "Optical Feature Extraction via the Radon Transform," Opt. Eng. 23, 499 (1984).
8. W. H. Steier and R. K. Shori, "Optical Hough Transform," Appl. Opt. 25, 2734 (1986).
9. R. O. Duda and P. E. Hart, "Use of the Hough Transform to Detect Lines and Curves in Pictures," Commun. Assoc. Comput. Mach. 15, 11 (1972).
10. D. Casasent and R. Krishnapuram, "Detection of Target Trajectories Using the Hough Transform," Carnegie-Mellon U., Department of Electrical and Computer Engineering.
11. A. E. Cowart and W. H. Ruedger, "The Detection of Unresolved Targets Using the Hough Transform," Comput. Vision Graphics Image Process. 21, 222 (1983).
12. P. M. Merlin and D. J. Farber, "A Parallel Mechanism for Detecting Curves in Pictures," IEEE Trans. Comput. 24, 96 (1975).

CHAPTER 4

"A Fast JPDA Multitarget Tracking Algorithm"

Fast JPDA multitarget tracking algorithm

James L. Fisher and David P. Casasent

Multitarget tracking requires assigning probabilities that measurements are associated with particular targets. A new and efficient algorithm to achieve this is detailed. Its implementation on an analog optical processor using a new and cost-effective frequency-multiplexing technique is discussed.

I. Introduction

The joint probabilistic data association (JPDA) class of algorithms[1] provides excellent ability to maintain track on multiple targets. Currently, they are not easily implemented in real time because of the large number of matrices that are required and the resulting high computational load (Sec. II). We present a new computationally efficient algorithm using predominantly analog vector inner product operations (Sec. III) and a new optical processor architecture for its realization (Sec. IV). A summary of the notation used appears in the Appendix.

II. JPDA Class of Algorithms

The JPDA algorithms have five basic steps: formation of the validation matrix (a record of coarse association information denoting, which subset of measurements could be associated with each target); generation of feasible event matrices (with each matrix representing a different combination of noncompeting events, i.e., sets of single measurement/target pairs); calculation of the probabilities $\beta_{jt}(k)$ for each measurement/target pair at time step k (the probability that measurement $j = 1,\ldots,m$ is associated with target $t = 1,\ldots,T$ based on the probabilistic occurrence of all possible measurement/target pairings); updating the state estimate error covariance matrix; and updating the state estimate vector for each target with all plausible measurements, each multiplied by the appropriate scalar weighting coefficient $\beta_{jt}(k)$. We discuss the first three steps (for background and to show the superiority of our new algorithm). The error covariance and state estimate updating are standard linear algebra operations and are not detailed here. The major goal is to determine the $\beta_{jt}(k)$.

A. Validation Matrix Formation

The binary elements of the validation matrix[1] denote which measurement/target pairs are statistically reasonable to consider as being true. Consider the example in Fig. 1 with $T = 2$ targets in track (denoted by an X) and five measurements (denoted by a dot). Each target has a validation gate associated with it (shown by an ellipse). These are obtained by standard procedures[2] using the error covariance matrix of the state estimate and a predetermined acceptable probability of detection $P_{D,t}$ for each target t. If any of the measurements j lie within the validation gate (or confidence ellipsoid) of any target, they are in the set of m validated measurements. For the example shown in Fig. 1, $m = 3$ measurements are validated for the $T = 2$ targets.

The validation matrix for this simple example is 3×3 for $m = 3$ validated measurements and $T + 1 = 3$ targets. (The target $t = 0$ is included to represent the source of a false measurement.) For each measurement j lying within the validation gate of (the real) target t, we enter a 1 in the associated element of the validation matrix. We also consider the probability that each measurement is false, and, therefore, all measurements are also associated with target $t = 0$. Thus all entries in column $t = 0$ are 1's. For example, consider row 1 in Fig. 1, for which measurement $j = 1$ falls within the validation gate of target $t = 1$ only. Thus row 1 is 110 (since only the false measurement target $t = 0$ and target $t = 1$ can be associated with measurement $j = 1$). Row 2 contains all 1's since measurement $j = 2$ could be false and since it falls in

The authors are with Carnegie Mellon University, Department of Electrical & Computer Engineering, Center for Excellence in Optical Data Processing, Pittsburgh, Pennsylvania 15213.

Received 16 May 1987.
0003-6935/89/020371-06$02.00/0.

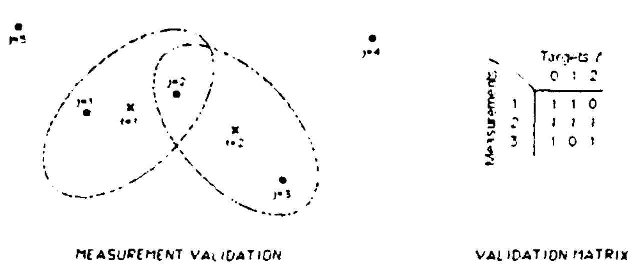


Fig. 1. Measurement validation example with five measurements and two targets with their validation gates (shown to the left) and the corresponding validation matrix (shown to the right).

$$\underset{\Omega_1}{\begin{bmatrix}1&0&0\\1&0&0\\1&0&0\end{bmatrix}}\quad\underset{\Omega_2}{\begin{bmatrix}0&1&0\\1&0&0\\1&0&0\end{bmatrix}}\quad\underset{\Omega_3}{\begin{bmatrix}1&0&0\\0&1&0\\1&0&0\end{bmatrix}}\quad\underset{\Omega_4}{\begin{bmatrix}1&0&0\\0&0&1\\1&0&0\end{bmatrix}}$$

$$\underset{\Omega_5}{\begin{bmatrix}0&1&0\\0&0&1\\1&0&0\end{bmatrix}}\quad\underset{\Omega_6}{\begin{bmatrix}1&0&0\\1&0&0\\0&0&1\end{bmatrix}}\quad\underset{\Omega_7}{\begin{bmatrix}0&1&0\\1&0&0\\0&0&1\end{bmatrix}}\quad\underset{\Omega_8}{\begin{bmatrix}1&0&0\\0&1&0\\0&0&1\end{bmatrix}}$$

Fig. 2. All possible feasibility matrices for the validation matrix of Fig. 1.

the intersection of the validation gates of both established targets. And finally, row 3 is 101 since measurement $j = 3$ could be false or it could have come from target $t = 2$, but it is statistically too far from target $t = 1$ to be associated with target $t = 1$.

B. Feasibility Matrices

As an intermediate step in computing the $\beta_{jt}(k)$ coefficients, the conventional JPDA algorithm[1] produces and then interprets feasibility matrices. These matrices exhaustively represent every possible combination of feasible (noncompeting) events. Noncompeting events are collections of events in which a measurement is associated with only one target t and a target can be responsible for at most only one measurement j (except for the target $t = 0$ case). In one feasibility matrix, all m measurements are associated with a target $t = 0,1,\ldots,T$.

The purpose of the feasibility matrices is to provide a format in which we can examine every possible (statistically plausible) combination of measurements and targets, i.e., individual events. We denote a specific event, i.e., one (j,t) pair by $\chi_{j,t}$. For example, the event that measurement $j = 2$ is associated with target $t = 1$ is denoted as $\chi_{2,1}$. For illustration, consider only measurements $j = 1$ and $j = 2$ in the example depicted in Fig. 1. In one scenario, the events $\chi_{1,0}$ (i.e., measurement $j = 1$ is associated with $t = 0$) and $\chi_{2,1}$ occur simultaneously. In another scenario, $\chi_{1,1}$ and $\chi_{2,2}$ occur together. However, $\chi_{1,1}$ and $\chi_{2,1}$ cannot occur simultaneously (i.e., measurements $j = 1$ and $j = 2$ cannot both be assumed to have come from target $t = 1$ at the same time), and we call $\chi_{1,1}$ and $\chi_{2,1}$ competing events. A collection of noncompeting events $\chi_{j,t}$ wherein each measurement is associated with a target is called a feasible event χ_i. (There is no correlation between the index i and indices j and t.) Feasible events are expressed as feasibility matrices Ω_i.

Each feasibility matrix Ω_i is obtained from the validation matrix by selecting only one 1 per row so that there is at most one 1 per column, except for column $t = 0$, which may have any number of 1 entries (i.e., any number of measurements may be false simultaneously). Figure 2 shows all eight feasibility matrices Ω_i (one for each χ_i) for the validation matrix in Fig. 1. Case 1 (χ_1) corresponds to no targets detected. Case 5 (χ_5) corresponds to the case when target $t = 1$ produced measurement $j = 1$ (the first row of Ω_5), target 2 produced measurement 2 (row 2), and measurement 3 was a false measurement (row 3), i.e., when individual events $\chi_{1,1}$, $\chi_{2,2}$, and $\chi_{3,0}$ occur simultaneously.

The number of feasibility matrices for one validation matrix depends on the structure of the validation matrix. For the case of a validation matrix that is all 1's, we have derived an expression for the number of feasibility matrices to be formed:

$$1+\sum_{k=0}^{T-1}\binom{T}{k}\left[\prod_{i=0}^{T-1-k}(m-i)\right]=1+\sum_{k=0}^{T-1}\frac{T!m!}{k!(T-k)!(m-T+k)!}\,. \tag{1}$$

Equation (1) holds for $m \geq T$, for $m < T$, we interchange m and T. Table I lists Eq. (1) quantified for different numbers of measurements m and targets T. As seen, the number of possible feasibility matrices

Table I. Number of Possible Feasibility Matrices for an m-Measurement T-Target Validation Matrix Consisting of all Ones

Number of Measurements	Number of Targets (T) 1	2	3	4	5	6	7	8	9	10
1	2	3	4	5	6	7	8	9	10	11
2	3	7	13	21	31	43	57	73	91	111
3	4	13	34	73	136	229	358	529	748	1021
4	5	21	73	209	501	1045	1961	3393	5509	8501
5	6	31	136	501	1546	4051	9276	19081	36046	63591
6	7	43	229	1045	4051	13327	37633	93289	207775	424051
7	8	57	358	1961	9276	37633	130922	394353	1047376	2501801
8	9	73	529	3393	19081	93289	394353	1441729	4596553	12975561
9	10	91	748	5509	36046	207775	1047376	4596553	17572114	58941091
10	11	111	1021	8501	63591	424051	2501801	12975561	58941091	234662231

$$\begin{bmatrix} a0 & b0 & c0 & d0 & e0 & f0 \\ a1 & b1 & c1 & d1 & e1 & f1 \\ a2 & b2 & c2 & d2 & e2 & f2 \\ a3 & b3 & c3 & d3 & e3 & f3 \\ a4 & b4 & c4 & d4 & e4 & f4 \end{bmatrix}$$

Fig. 3. Modified validation matrix for the case of four measurements and six targets.

rapidly increases (235 million for $m = T = 10$). Each feasibility matrix must then be interpreted (Sec. II.C) requiring approximately m multiplications per feasibility matrix. This enormous computational load due to both generation and interpretation is the major problem with this feasibility matrix approach to the calculation of the $\beta_{jt}(k)$.

C. Calculation of $\beta_{jt}(k)$ in JPDA

Each feasibility matrix Ω_i is interpreted to obtain the probability $P\{\chi_i\}$ of the feasible event χ_i. The probability $P\{\chi_i\}$ is the product of the probabilities $P\{\chi_{j,t}\}$ of the composite individual events. The individual probabilities are of one of two forms: if the target t is undetected (i.e., measurement j is associated with target $t = 0$), then $P\{\chi_{j,t}\} = (1 - P_{D,t})$; otherwise $P\{\chi_{j,t}\} = f_t[\mathbf{z}_j(k)]P_{D,t}$, where $P_{D,t}$ is the probability that target t was detected, and $f_t[\mathbf{z}_j(k)]$ is the probability density of measurement j when associated with target t. A scaled normal distribution $f_t(\mathbf{z}_j) \sim N[\mathbf{z}_j(k);\hat{\mathbf{z}}_t(k|k-1),\mathbf{S}_t(k)]$ is used, where $\mathbf{S}_t(k)$ is the error covariance matrix of the innovations $[\mathbf{z}_j(k) - \hat{\mathbf{z}}_t(k|k-1)]$.

The final step in the conventional approach to calculating a single coefficient $\beta_{jt}(k)$ is to sum the probabilities of only those feasibility matrices in which the individual event $\chi_{j,t}$ occurred and normalize by dividing by the sum of the probabilities of all feasibility matrices. For every 1 in the validation matrix in columns $t = 1,\ldots,T$ [representing a statistically plausible (j,t) pair] there exists a nonzero $\beta_{jt}(k)$ weighting coefficient. The *a posteriori* probability $\beta_{0,t}(k)$ that no measurements from target t were detected is

$$\beta_{0,t}(k) = 1 - \sum_{j=1}^{m} \beta_{jt}(k). \qquad (2)$$

Therefore, there are at most $(m + 1)T$ coefficients $\beta_{jt}(k)$ to be calculated at each time step k.

While the method of feasibility matrices is intuitively appealing, it is computationally wasteful. In Sec. III, we detail our new algorithm, which is mathematically equivalent to the conventional method and computationally more efficient.

III. New JPDA Algorithm

In our new JPDA algorithm, we adopt the same modeling assumptions as in the conventional JPDA algorithm,[1] but we derive a very different method for calculating the weighting coefficients $\beta_{jt}(k)$. In our new approach, instead of the binary validation matrix, we employ an analog modified validation matrix (Sec. III.A), and feasibility matrices are not formed or used. The method described in Secs. III.B through III.D iteratively calculates only T^2 vectors of combinations of single-measurement/single-target pair probabilities $P\{\chi_{j,t}\}$. The vectors are stored *in situ* (i.e., we need to retain only the latest vector) in a vector of reasonable length $\sum_{j=\max(1,m-T+1)}^{m} \binom{m}{j}$, especially when compared to the number of feasibility matrices in the conventional JPDA algorithm.

A. Modified Validation Mark

To record the coarse association measurement/target information we form a new modified validation matrix (MVM) with $m + 1$ rows (for the m validated measurements plus the new $j = 0$ case) and T columns (for the T established target tracks). We omit the $t = 0$ column of the conventional validation matrix and include the new $j = 0$ row in the MVM. (When we consider the conventional validation matrix and MVM in their entirety, the information is preserved.) Thus our attention is shifted from false measurements to undetected targets, the latter being more desirable since we require the corresponding coefficient $\beta_{0,t}(k)$ as defined in Eq. (2). The analog entries in the MVM are the individual probabilities $P\{\chi_{j,t}\}$. We introduce a shorthand notation wherein we denote the T established targets t by **a**,**b**,**c**, etc. and the m validated measurements j by 0,1,2,3, etc. Given this notation, the entries for the undetected target row $j = 0$ are

$$(1 - P_{D,t}) = \begin{cases} \text{a0 for target } t = 1, \\ \text{b0 for target } t = 2, \\ \text{c0 for target } t = 3, \text{ etc.} \end{cases} \qquad (3)$$

The entries in the other rows are

$$f_t(\mathbf{z}_j)P_{D,t} = \begin{cases} \text{a}j \text{ for target } t = 1 \\ \text{b}j \text{ for target } t = 2 \\ \text{c}j \text{ for target } t = 3, \text{ etc.} \end{cases} \qquad (4)$$

The analogous MVM for a general $4 \times 6 = m \times T$ case ($m = 4$ validated measurements and $T = 6$ targets) thus has potential entries as shown in Fig. 3. The MVM has zero entries where the conventional matrix did, and the analog entries in Eqs. (3) and (4) for the nonzero entries.

B. Vector T Formation

We now consider how to determine all possible noncompeting events and the probability of each. [We will then use these values to determine the $\beta_{jt}(k)$.] We use the MVM of Fig. 3 for a case study. We first form all possible sums of the last column of the MVM (individual event or measurement association probabilities for target f for the example in Fig. 3). It can be shown that the number of probabilities summed (excluding those from row $j = 0$) varies from max $(m - T + 1,1)$ to m. For our present case, this includes the sums of all combinations of 4, 3, 2, and 1 individual event probabilities from target f. To each of these sums, we add f0 (since target T could always have not been detected). Table II shows the sixteen possible target f measurement combinations, the elements f_n of the vector **f** (column 2) grouped by bit counts, i.e., combinations of 4, 3, 2, 1, and no measurements (column 1). The number of elements (dimensionality) of this vector **f** is given by the sum of the number of combinations possible for $m = 4$ elements taken four at a time, then three at a time, etc., i.e.,

$$\sum\left[\binom{4}{4}+\binom{4}{3}+\binom{4}{2}+\binom{4}{1}+\binom{4}{0}\right] = 16. \qquad (5)$$

Table III shows the size (dimension) of the last target vector for different m and T combinations. As seen, it does not increase appreciably as m and T grow. In general, this vector is denoted by **T** for the last target T. The measurements used (MU) binary word in column 4 denotes the measurement association probabilities **f**4 to **f**1 included in each f_n. The last column of Table II is discussed below.

C. Generation of Probability Vector for all Feasible Events for Targets 2 to T

Each element f_n of **f** represents a set of target **f** measurement association or individual event probabilities that represents a single event (for target **f**). We now include target **e** individual event probabilities into this set of target **f** data. This involves multiplying the different f_n by different elements ej for target **e** (i.e., by individual event probabilities for target **e** in column **e** of Fig. 3) and forming their sums. The allowable (noncompeting) sums are indicated in the last column of Table II. They comprise a new vector **e** (including feasible events for both targets **f** and **e**) with elements e_n.

One can precompute the pattern of these possible target **e** and **f** combinations (given m and T). However, the storage required can become quite large, and thus we consider ways to produce **e** (and subsequent vectors including other target data) on-line. We achieve this with the MU data word as we now detail. For example, to obtain those elements of vector **e** with a bit count of 3, we exclusive OR (XOR) each MU bit count = 3 word with all MU bit count = 2 words and retain only those results containing one 1. Those bit count = 2 words that result in one 1 denote which f_n is used, and the location of the 1 in the XOR result denotes which ej multiplies the f_n element. This forms the new bit count = 3 elements of **e**. This procedure is executed for bit counts of 4, 3, 2, and 1, thus determining all elements of **e**. Each element e_n of **e** also contains the additive term **e**0f_n, since each target could not have been detected as before.

Our idea of and use of the control vector MU to guide selection of the **e**j measurement elements that multiply the f_n elements are quite novel. We note that the same MU vector also describes the **e**j measurements included. The **e** vector produced can be stored in the **f** vector location in memory, since **f** is no longer needed once **e** has been produced.

The aforementioned procedure is repeated for targets $T - n$ (where $n = 1,2,\ldots,T-2$), i.e., for targets **d** through **b** in our example in Fig. 3. The same MU control vector describes each target t measurement set combination. This procedure can be used to determine the target vectors **d**, **c**, and **b**, and as they are calculated they can be stored in the location of the prior vector. All the target vectors are of the same length, which is given in Table III. For our example, this procedure results in a final target vector **b** whose elements denote the probability of all feasible events for targets **b** through **f** (or targets 2 through 6).

D. Generation of the $\beta_{jt}(k)$ Coefficients

The aforementioned procedure results in a vector **b**. To obtain $\beta_{j,t}$ for target 1, i.e., a vector of the $m + 1$ scalar weighting coefficients $\beta_{j,1}(k)$ for each measurement j that could be assigned to target 1, we multiply the reordered vector $\mathbf{b}' = [b_1,b_{m+1},b_m,\ldots,b_2]^T$ by a diagonal matrix A with elements given by the target **a** measurement entries in the MVM (i.e., $A = \text{diag}[\mathbf{a}0,\mathbf{a}1,\ldots,\mathbf{a}m]$). The basic operation required to compute $\beta_{jt}(k)$ is thus

$$\beta_{j,t} = \mathbf{A}\mathbf{b}'. \qquad (6)$$

The calculation of **e** from **f** involves a vector inner product (VIP) for each element e_n of **e**. The **f** vector is the vector **T**. The elements of the new **e** vector (or in general vector **T** − 1) are each given by a VIP of the elements f_n of **f** and elements **e**j of column **e** of the MVM. We denote the elements of column **e** of the MVM (or in general column **T** − 1) by $\mathbf{p}_{T-1}^{m}$ (the m individual event probabilities $P\{\chi_{j,t}\}$ for target $t = T -$

Table II. Example of Generation of Vector e for the MVM of Fig. 3

4	f0+f1+f2+f3+f4	f_1	1111	e0(f_1)+e4(f_2)+e3(f_3)+e2(f_4)+e1(f_5)
3	f0+f1+f2+f3	f_2	0111	e0(f_2)+e3(f_6)+e2(f_7)+e1(f_8)
	f0+f1+f2+f4	f_3	1011	e0(f_3)+e4(f_6)+e2(f_9)+e1(f_{10})
	f0+f1+f3+f4	f_4	1101	e0(f_4)+e4(f_7)+e3(f_9)+e1(f_{11})
	f0+f2+f3+f4	f_5	1110	e0(f_5)+e4(f_8)+e3(f_{10})+e2(f_{11})
2	f0+f1+f2	f_6	0011	e0(f_6)+e2(f_{12})+e1(f_{13})
	f0+f1+f3	f_7	0101	e0(f_7)+e3(f_{12})+e1(f_{14})
	f0+f2+f3	f_8	0110	e0(f_8)+e3(f_{13})+e2(f_{14})
	f0+f1+f4	f_9	1001	e0(f_9)+e4(f_{12})+e1(f_{15})
	f0+f2+f4	f_{10}	1010	e0(f_{10})+e4(f_{13})+e2(f_{15})
	f0+f3+f4	f_{11}	1100	e0(f_{11})+e4(f_{14})+e3(f_{15})
1	f0+f1	f_{12}	0001	e0(f_{12})+e1(f_{16})
	f0+f2	f_{13}	0010	e0(f_{13})+e2(f_{16})
	f0+f3	f_{14}	0100	e0(f_{14})+e3(f_{16})
	f0+f4	f_{15}	1000	e0(f_{15})+e4(f_{16})
0	f0	f_{16}	0000	e0(f_{16})

Table III. Size of Vectors Processed in the New Algorithm

Number of Measurements	Number of Targets				
	3	4	5	6	7
3	7x1	8x1	8x1	8x1	8x1
4	11x1	15x1	16x1	16x1	16x1
5	16x1	26x1	31x1	32x1	32x1
6	22x1	42x1	57x1	63x1	64x1
7	29x1	64x1	99x1	120x1	127x1

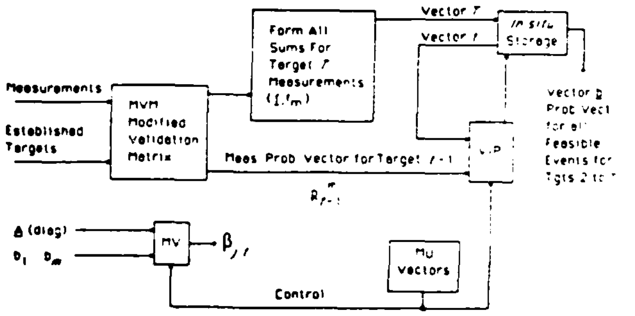

Measurements
Established Targets
MVM Modified Validation Matrix
Form All Sums For Target T Measurements (f_1–f_m)
Vector T
Vector t
In situ Storage
Vector β Prob Vect for all Feasible Events for Tgts 2 to T
VIP
Meas Prob Vector for Target T-1
p_{T-1}^m
Δ (diag)
b_1 b_m
MV
$\beta_{j,t}$
Mu Vectors
Control

Fig. 4. Block diagram of the computational steps in our new $\beta_{jt}(k)$ direct generation algorithm.

1). We thus require the VIP of $\mathbf{p}_{T-1}^m$ and T, which results in vector T − 1 or vector e in our example.

Figure 4 shows the block diagram of our algorithm. The MVM is formed first and then the vector **f** (or in general the vector T) as shown in the top center of Fig. 4. The VIP of f and $\mathbf{p}_{T-1}^m$ is then formed (both vectors are analog). A different VIP is formed with different $\mathbf{p}_{T-1}^m$ weighting elements to product each element e_n of the new vector e (or in general the vector T − 1) as shown on the right side of Fig. 4. The *in situ* storage box in Fig. 4 denotes the fact that each new vector T,T − 1, etc. is stored in the location of the prior vector. The bottom portion of Fig. 4 shows the single matrix–vector (MV) multiplication required to produce the final $\beta_{jt}(k)$ according to Eq. (6).

The steps outlined above (Fig. 4) produce $\beta_{j,1}$ for target $t = 1$. To obtain $\beta_{j,2}$ etc., we interchange columns 1 and t in the MVM and repeat the aforementioned procedure. Similar steps yield all $\beta_{j,t}$ vectors.

Normalization so that $\sum_{j=0}^{m} \beta_{jt}(k) = 1$ for all targets t is straightforward. We simply calculate the normalization constant $c'' = \sum_{j=0}^{m} \beta_{jt}(k)$ for any one target t and divide all $(m + 1)T$ unnormalized coefficients $\beta_{jt}(k)$ by c''.

To reduce further the number of operations required to produce $\beta_{jt}(k)$ coefficients for subsequent targets t, one can save intermediate **f**,**e**, etc. However, even if this is not done, the savings in our algorithm over the conventional JPDA algorithm[1] are significant. Tables IV and V list the number of operations required to compute the $\beta_{jt}(k)$ coefficients via the conventional JPDA algorithm[1] and our new algorithm, respectively. (Table V does not reflect the potential savings realizable by saving intermediate vectors **f**,**e**, etc.) Table VI notes the factor by which the number of addition and multiplications are reduced in our algorithm. These factors are denoted as additions/multiplications in Table VI. As seen, the savings is a factor of 52 (for additions) or 67 (for multiplications) for the case of $m = T = 7$, and the improvement factor increases significantly as m and T increase. When intermediate **f** etc. vectors are stored, the improvement exceeds a factor of 110 (thus making our algorithm even more attractive).

IV. Optical Realization

Optical processors can be used to achieve many of the functions or operations noted in Fig. 4. However,

Table IV. Operation Count for the Conventional JPDA. Entry Syntax Is Additions/Multiplications

Number of Measurements	Number of Targets				
	3	4	5	6	7
3	54/68	144/219	300/544	540/1145	882/2148
4	144/146	528/627	1440/2004	3240/5225	6384/11766
5	300/272	1440/1503	5200/6148	15000/20255	36540/55656
6	540/458	3240/3135	15000/16204	55620/66635	170100/225798
7	882/716	6384/5883	36540/37104	170100/188165	652974/785532

Table V. Operation Count for our Fast Algorithm as Number of Additions/Multiplications

Number of Measurements	Number of Targets				
	3	4	5	6	7
3	63/51	132/144	225/280	342/456	483/672
4	132/78	304/260	540/560	840/960	1204/1456
5	240/111	620/432	1175/1045	1890/1920	2765/3024
6	396/150	1152/672	2370/1830	3996/3654	6006/6048
7	609/195	1988/992	4480/3030	8022/6624	12495/11655

Table VI. Savings Factor (Additions/Multiplications) for Our New Algorithm vs the Conventional JPDA

Number of Measurements	Number of Targets				
	3	4	5	6	7
3	0 9/1 3	1 1/1 5	1 3/1 9	1 6/2 5	1 8/3 2
4	1 1/1 9	1 7/2 4	2 7/3 6	3 9/5 4	5 3/8 1
5	1 3/2 5	2 3/3 5	4 4/5 9	7 9/10 5	13 2/18 4
6	1 4/3 1	2 8/4 7	6 3/8 9	13 9/18 2	28 3/37 3
7	1 4/3 7	3 2/5 9	8 2/12 2	21 2/28 4	52 3/67 4

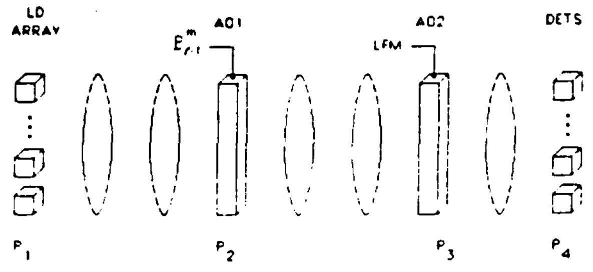


Fig. 5. Optical processor for parallel calculation of m vector inner products.

the most time-consuming and repetitive function is the analog VIP. Initial studies show that this operation (to generate subsequent target probability vectors and eventually **b**) need be performed only to analog accuracy. Thus an analog VIP optical processor is the system realization element given major attention here. This operation can be performed quite easily on the new architecture of Fig. 5.

In Fig. 5, one vector **t** is applied to a linear laser diode (LD) array at P_1, and a second vector $\mathbf{p}^m_{t-1}$ is input to an acoustooptic (AO) cell $AO1$ at P_2. P_1 is imaged onto P_2, and the resulting point-by-point products leaving P_2 are imaged onto a second AO cell $AO2$ at P_3. Consider the $AO1$ and $AO2$ system. This system images the vector fed to $AO1$ onto a linear frequency modulated (LFM) signal at $AO2$. This has the effect that every possible element of the $\mathbf{p}^m_{t-1}$ vector is multiplied by all frequencies in the LFM at some time. The $AO2$ frequency determines which P_3 detector receives the data at the given spatial location in $AO1$ at a given time. This produces the desired $\mathbf{p}^m_{t-1}$ multiplicative factors for the **t** vector elements to produce the new **t** − 1 vector elements on separate P_3 detectors. At each time, the proper P_1 laser diodes are pulsed on with the proper t vector elements, and one set of scalar products is formed (by imaging P_1 onto P_2) with each product directed to the proper P_3 detector (by the LFM control signal to $AO2$). These results (partial products) are accumulated (by time integration) on the P_3 detectors to produce the VIPs required to calculate each element of the new **t** − 1 vector. This space and time-integrating hybrid architecture uses a single LFM control signal to produce the required VIPs by frequency multiplexing (plus space and time integration).

V. Summary and Conclusion

A new JPDA multitarget tracking algorithm has been devised. It is very efficient in terms of calculations, bookkeeping, and data flow. Only VIPs are predominantly required, and these can be implemented with analog accuracy. An efficient optical realization with a new and cost-effective frequency-multiplexed control has been described. The basic use for generating all noncompeting feasible pairs of data from a large combination of data has many other attractive and general applications.

The support of this research by a grant from the Strategic Defense Initiative Office of Innovative Science and Technology, monitored by the Office of Naval Research, is gratefully acknowledged.

Table VII. Summary of Terms

Symbol	Description
d	scalar validation threshold constant
$f_t(\underline{z}_j)$	probability density of the single measurement $\underline{z}_j$ from target t at time k
$\underline{H}_t(k)$	coordinate transformation matrix relating the state space of $\hat{\underline{x}}_t(\cdot)$ and the *measurement* space of $\underline{z}_t(\ )$ for target t at time k
j	index of validated measurements, $j = 1, \ldots, m$ in Section II, and $j = 0, 1, \ldots, m$ in Section III
k	discrete time index
m	total number of validated measurements at time k
$P_{D,t}$	probability of detecting target t
$\underline{P}_t(k \mid k-1)$	*a priori* state estimate error covariance matrix for target t at time k
$\underline{P}_t(k \mid k)$	*a posteriori* state estimate error covariance matrix for target t at time k
$P\{\chi_i \mid Z^k\} = P\{\chi_i\}$	probability that feasible event χ occurred, conditioned on Z^k
$\underline{S}_t(k)$	error covariance matrix associated with $\hat{\underline{z}}_t(k \mid k-1)$
T	total number of established target tracks at time k
t	index of established targets: $t = 0, 1, \ldots, T$ in Section II and $t = 1, \ldots, T$ in Section III
$\hat{\underline{x}}_t(k \mid k-1)$	*a priori* state vector estimate for target t at time k
Z^k	set of *all* validated measurements $\underline{z}_j(k)$ from time 0 to time k
$\underline{z}_j(k)$	measurement vector j (validated or unvalidated) at time k
$\hat{\underline{z}}_t(k \mid k-1)$	*a priori* predicted measurement location of target t at time k
$\beta_{jt}(k)$	measurement innovations weighting coefficient at time k; the probability that measurement j originated from target t
$\underline{\nu}_{jt}(k)$	measurement innovations vector for measurement/target pair (j,t) at time k
χ_{jt}	an event, i.e., a single measurement/target pair
χ_i	a feasible event, a collection of non-competing events χ_{jt} such that all measurements j have been considered
Ω_i	(binary) feasibility matrix
$\omega_{jt}(\chi)$	event occurrence indicator

Appendix: Summary of Notation

Table VII is a compilation of the notation used in this paper. The notation used conforms to the following conventions: plain and boldfaced lowercase letters not underlined are scalars, italicized lowercase letters with numerical subscripts are vector elements (scalars), underlined lowercase letters are vectors, and underlined uppercase letters are matrices. The symbols in Table VII are listed in alphabetical order with the letters of the Roman alphabet preceding those of the Greek alphabet.

References

1. T. E. Fortmann, Y. Bar-Shalom, and M. Scheffe, "Sonar Tracking of Multiple Targets Using Joint Probabilistic Data Association," IEEE J. Oceanic Eng. OE-8, 173 (1983).
2. Y. Bar-Shalom and E. Tse, "Tracking in a Cluttered Environment with Probabilistic Data Association," Automatica 11, 451 (1975); also in *Proceedings, Fourth Symposium on Nonlinear Estimation, Theory, and Its Applications* (U. California, San Diego, Sept. 1973), pp. 13-22.

CHAPTER 5

"Multitarget Tracking with Cubic Energy Optical Neural Nets"

Accepted for MARCH 1989 publication - Applied Optics

Multitarget tracking with cubic energy optical neural nets

Etienne Barnard and David Casasent

Carnegie Mellon University
Center for Excellence in Optical Data Processing
Department of Electrical and Computer Engineering
Pittsburgh, PA 15213

Abstract

A neural net processor and its optical realization are described for a multitarget tracking application. A cubic energy function results and a new optical neural processor is required. Initial simulation data are presented.

I. Introduction

Considerable interest currently exists in neural networks[1, 2] due to their adaptive properties, fault tolerance and high computational throughput. One can distinguish current neural processors by whether they concern pattern recognition and associative memories[3, 4, 5] or multivariate optimization[6, 7]. Our concern will be with the application of neural networks in optimization problems. As a specific case study, we consider multitarget tracking.

In Section II, we briefly review the evolution equations as used in neural minimization. Section III contains a definition of the specific problem we consider, and Section IV is a formulation of the constraints in our multitarget tracking problem as an energy function to be minimized. New optical architectures for the implementation of the equations in Section IV are then described (Section V). Simulation results are presented in Section VI. Our work contains three new ideas: the application of the Hopfield model to a multitarget tracking problem; the use of a non-quadratic energy function in the minimization problem and an optical architecture which can calculate the evolution of a system with such a non-quadratic energy function.

II. Neural Model

We use the Hopfield model[6] as a minimization network. We represent the state of the "neurons" by $X_i(t)$, where t is a time variable and i labels a particular neuron within the set of neurons. For an optimization problem, we wish to find the set of X_i that minimizes an "energy" function E, which is a function of the neural activities X_i. In this model, the evolution of the activity of each neuron (its rate of change with time) is described by

$$\frac{dX_i}{dt} = -\frac{\partial E}{\partial X_i}. \tag{1}$$

The time evolution of the energy function is

$$\frac{dE}{dt} = \sum_i \frac{\partial E}{\partial X_i} \frac{dX_i}{dt}. \tag{2}$$

To show that the model in (1) minimizes E, we substitute (1) into (2) and obtain

$$\frac{dE}{dt} = -\sum_i (\frac{\partial E}{\partial X_i})^2. \tag{3}$$

Eq. (3) shows that E is a decreasing function of time t. The energy E will converge to a local minimum as t progresses. Thus, the set of neural activities $\{X_i\}$ in the final stationary state describe a minimum energy state of the system.

In our work, this basic algorithm is modified by using discrete time[8] and by employing binary neuron activities X_i. With binary X_i, the neuron activity (neuron state) in (1) can now be replaced by

$$X_i = \begin{cases} 1 \text{ if } \dfrac{\partial E}{\partial X_i} < 0 \\ 0 \text{ if } \dfrac{\partial E}{\partial X_i} > 0 \end{cases} . \tag{4}$$

That is, the state of neuron i is binary and depends on the energy as noted. The choice in (4) insures that the energy function is approximately minimized in the stationary state, as we now show.

With unit time steps, we replace dX/dt by ΔX and dE/dt by ΔE. To find the change in energy due to a state change of neuron i, we recall the Taylor expansion of $E(\{X_i+\Delta X_i\})$ in the vicinity of $E(\{X_i\})$:

$$E(\{X_i+\Delta X_i\}) = E(\{X_i\})+\sum_i \frac{\partial E}{\partial X_i}\Delta X_i \tag{5}$$
$$+\sum_i \sum_j \frac{\partial^2 E}{\partial X_i \partial X_j}\Delta X_i \Delta X_j + \ldots$$

One can therefore calculate the change in neural energy ΔE due to the state changes ΔX_i of the neurons by $\Delta E{=}E(\{X_i+\Delta X_i\}){-}E(\{X_i\})$. From (5), keeping only the first term:

$$\Delta E{=}\sum_i \frac{\partial E}{\partial X_i}\Delta X_i + \ldots \tag{6}$$

Using (4) in (6), we see that if $\partial E/\partial X_i$ is negative, we set the state X_i of neuron i to one and if $\partial E/\partial X_i$ is positive, we set X_i to zero. Thus, ΔE is negative. In (6) the higher order terms were omitted. Thus, the prior analysis is only approximately correct, and the energy of the system actually can increase on some iterations. It turns out that this is more beneficial than harmful, since it allows the system to escape from shallow local minima.

Since we will express the multitarget-tracking problem as a constrained optimization problem, it is also possible to use conventional (non-neural) optimization techniques. However, such techniques are not suitable for optical implementation, and generally require much more computation than the Hopfield net. We therefore restrict our attention to the techniques described in this section.

III. Problem Definition and Case Study

The minimization problem we consider is a multitarget tracking problem. The scenario we consider assumes:

1. N_T targets are to be tracked with N_T known and fixed (being determined by the track initiator).
2. There are N_M measurements each time step or frame of data. N_M is fixed and is the maximum number of measurements we will accept in any time frame. This is achieved as explained below. If the number of peaks (measurements) is less than N_M, we lower the detection threshold or insert artificial measurements to insure that at each time we have $N_M \geq N_T$ measurements.
3. The targets do not accelerate appreciably during the time steps under investigation and thus that their trajectories are approximately straight lines.
4. Each target corresponds to no more than one measurement at each time.
5. Each measurement is due to no more than one target at each time. (That is, we ignore crossing targets for now.)
6. At each time step, each target must be assigned to one measurement. Our selection of N_M in item 2 insures that $N_M \geq N_T$ so that this rule can be satisfied.

The optimization problem is to assign one track to each target, i.e., for each time step one set of detected coordinates is assigned to each target. Hence, our objective is to find the N_T best straight lines in the given data.

IV. Problem Formulation

We first present our notation, introduce the distance measures we wish to minimize and then develop a neuron energy description. We denote the measured position vectors at time steps a, $a+1$ and $a+2$ by $\mathbf{r}_{\alpha}^{a}$, $\mathbf{r}_{\beta}^{a+1}$ and $\mathbf{r}_{\gamma}^{a+2}$, respectively. The subscripts α,β and γ are used to refer to a particular one of the N_M different measurements at a given time step, the time steps being indexed by the superscript. We denote the vector difference between a specific measurement (one of the α) at time step a and one of the β measurements at time step $a+1$ by

$$\mathbf{d}_{\alpha\beta}^{a} = \mathbf{r}_{\alpha}^{a} - \mathbf{r}_{\beta}^{a+1}$$

Similarly, $\mathbf{d}_{\beta\gamma}^{a+1}$ denotes the vector distance between a measurement β at time step $a+1$ and a measurement γ at $a+2$. In terms of these vector distances, the vector distance measure we wish to minimize for a sequence of three time steps for all measurements is

$$D_{\alpha\beta\gamma}^{a}=\|\mathbf{d}_{\alpha\beta}^{a} - \mathbf{d}_{\beta\gamma}^{a+1}\|. \tag{7}$$

The minimum of (7) assigns one measurement in each of time steps a, $a+1$ and $a+2$ to the same target. Note that D in (7) is the norm of a vector difference. This ensures that two successive distance vectors (for time steps a and $a+1$) should be collinear to minimize D; that is, D is minimized for straight line tracks. With equal time step increments and a straight line trajectory with no acceleration, the two distance vectors will be equal (for true target measurements). Thus D will be 0 for the case of three collinear and evenly spaced measurements in three successive frames.

The measure D will now be used as a basis for the description of an energy function

E which, when minimized, solves our problem. We label each binary neuron with three indices, such as $X_{i\alpha a}$, where i is the target index, α is the measurement index and a is the time step index. This neuron is active (i.e. $X_{i\alpha a}=1$) if the i-th target is associated with a specific position vector $\mathbf{r}_{\alpha}^{a}$ (one of the measurements α) at time step a, and otherwise $X_{i\alpha a}=0$. The energy function to be minimized for the optimization problem in Section III can be written as

$$\begin{aligned} E = {} & A_1 \sum_{\alpha} \sum_{a} \sum_{i} \sum_{j \neq i} X_{i\alpha a} X_{j\alpha a} \\ & + A_2 \sum_{i} \sum_{a} \sum_{\alpha} \sum_{\beta \neq \alpha} X_{i\alpha a} X_{i\beta a} \\ & + A_3 \sum \left(\sum_{i} \sum_{\alpha} X_{i\alpha a} - N_T \right)^2 \\ & + A_4 \sum_{a} \sum_{i} \sum_{\alpha} \sum_{\beta} \sum D^{a}_{\alpha\beta\gamma} X_{i\alpha a} X_{i\beta(a+1)} X_{i\gamma(a+2)}, \end{aligned} \tag{8}$$

where A_1 to A_4 are positive constants. Their choice is discussed in Sect. VI.

We now discuss the terms in this energy function to provide an understanding of it. We first note that all terms are positive semi-definite. Consider the first term: each term in this sum is either 0 or 1 (since binary neurons are employed). Note that $X_{i\alpha a}$ and $X_{j\alpha a}$ denote neuron states associated with targets i and j (any of the N_T targets) and some measurement α (of the N_M) at time step a. The first term contains the sum over the measurements and time steps of products of these neurons. Since only the target index (i or j) differs in the $X_{i\alpha a}X_{j\alpha a}$ product, a given term in term 1 can be one only if targets i and j are assigned to the same measurement α at the same time step a. Since we sum over α, a, i and j, term 1 is zero if and only if at each time step no measurement is assigned to more than one target. Thus, minimization of this first term occurs when each measurement is associated with no more than one target. It therefore enforces condition 5 in Section III.

The second term in (8) consists of a sum of products with different measurement indices (α and β) on the neurons. It is therefore minimized when one target is associated with no more than one measurement (α or β) at the same time step a. Thus term 2 is included so that the system satisfies condition 4 in Section III.

We next consider term 3. For a fixed time a, the set of neurons $X_{i\alpha a}$ for the various target indices i and measurement indices α can be arranged in a matrix with horizontal index α and vertical index i. When one measurement has been assigned to each target, this matrix has a single one in each row i, indicating which measurement is assigned to this target. Thus, $\sum_i \sum_\alpha X_{i\alpha a}$ for a fixed time a is the number of non-zero entries in the matrix above. This sum equals N_T when condition 5 of Section III is satisfied. Thus, term 3 is minimized when all N_T targets are each associated with one measurement at each time step; it is included so that condition 6 in Section III is satisfied. Hence, the first three terms in (8) ensure that the measurement-target matching is admissible.

In term 4, both the time step and measurement indices on the three neurons in the product differ. This term selects a measurement (from each of the sets labeled by α, β and γ) in each of three successive time frames for each target. The measurement-target pairs are selected such that these three measurements lie closest to a straight line, with the search done for each target and for each measurement α in each frame. To see how this is accomplished, recall that $D^a_{\alpha\beta\gamma}$ in (7) is calculated for three successive time steps. The three neurons in term 4 in (8) have their time indices appropriately stepped. For a fixed time frame, the three X terms can each be represented by a matrix with horizontal index α and vertical index i, as before. For a fixed target i, the neuron choices to be considered occur in the same row (row i) in each of these matrices. Each row of each matrix should have only a single "one", because of condition 5 in Section III. The best choice for the position of these "ones" is determined as follows.

Consider that there are 10 measurements in each frame. We select a measurement α in frame a. For the single measurement α chosen, there are 10 possible choices for the measurement (indexed by β) in the second frame and for each of these there are 10 possibilities for the third measurement indexed by γ. For each of these 100 combinations, the three X factors in term 4 could all be 1, but for only one set of these will D be small. Minimization of E for this term ensures that the set of 3 successive measurements chosen (for each measurement in the first frame) will be the set closest to a straight line. The summation over a implies that this minimization is repeated for each time step (using the prior two frames of data).

A possible variation to the energy function in (8) is the omission of the first term. Then, one would not be enforcing the assignment of only one target to each input measurement (this case arises if two paths of different tracks cross). In our simulations, this term was retained. We intend to do more work and the target-crossing problem in the future.

Note that the energy function in (8) enables us to tolerate both spurious measurements and the absence of measurements for some targets at some time steps. To achieve this, let N_M be the largest number of measurements at any of the N_P time steps. If a given frame has $M < N_M$ target measurements we set $N_M - M$ measurements equal to the zero vector. (In our reference frame the zero vector lies in the center. This choice minimizes the effect of missing measurements on D, for the case of a uniform spatial distribution of targets). Spurious measurements (if they have random position vectors, as they should) will not be assigned to true target tracks because of the energy minimization step. If both spurious and missing measurements are present, we have found that the spurious measurements are assigned to the same tracks as the missing measurements (zero vectors).

The time evolution of the neurons is required to result in a neural system that minimizes E in (8). This is given by the derivative of (8), i.e.

$$\begin{aligned}\frac{\partial E}{\partial X_{i\alpha a}} = {} & 2A_1 \sum_{j \neq i} X_{j\alpha a} + 2A_2 \sum_{\beta \neq \alpha} X_{i\beta a} \\ & + 2A_3 (\sum_j \sum_\beta X_{j\beta a} - N_T) \\ & + A_4 \sum_\beta \sum_\gamma \{ D^a_{\alpha\beta\gamma} X_{i\beta(a+1)} X_{i\gamma(a+2)} \\ & \qquad + D^{a-1}_{\beta\alpha\gamma} X_{i\beta(a-1)} X_{i\gamma(a+1)} \\ & \qquad + D^{a-2}_{\beta\gamma\alpha} X_{i\beta(a-2)} X_{i\gamma(a-1)} \}.\end{aligned} \tag{9}$$

Fig. 1 shows the block diagram of the neural multitarget tracker described by Eqs (4), (8) and (9). We produce $\partial E/\partial X$ in (9) from X and threshold $\partial E/\partial X$ as defined in (4) to produce the new X with E given by (8) from which $\partial E/\partial X$ is obtained in a closed loop. In this design, the activities of the neurons evolve according to (9) and (4), and thus their states will evolve to a steady state energy minimum. This minimum indicates which target should be associated with which measurement at each time step.

We now discuss how to realize the terms in (9) as linear algebra functions. We consider the case of N_T=10 targets, N_M=10 measurements and N_P=3 time steps. There are $N_T x N_M x N_P$=300 neurons that represent the different $X_{i\alpha a}$. We represent these neurons as a vector **x** with elements x_k, where each value of the index k denotes a different (i,α,a) combination. In steady state, **x** will have 30 "one" entries (ten measurement/target pairs for each of three time step values a.) Term 1 in (9) is the sum of a number of such vectors and can thus be described as a matrix-vector product $2A_1\mathbf{T}_1\mathbf{x}=\mathbf{y}_1$. The elements T_{kl} of the binary connection matrix $\mathbf{T}_1$ are described by

$$T_{kl} = T_{i\alpha a,j\beta b} = \delta_{\alpha\beta}\delta_{ab}(1-\delta_{ij}), \tag{10}$$

where the indices k and l denote the different sets of target/measurement/time parameters $(i\alpha a)$ and $(j\beta b)$, respectively. Since both k and l range over 300 values, $\mathbf{T}_1$ is a 300x300 matrix. This $\mathbf{y}_1$-term has non-zero contributions to the output only for indices corresponding to different targets $(i \neq j)$ but the same measurement $(\alpha = \beta)$ and time step $(a = b)$.

Terms 2 and 3 in (9) are other sums of vectors $\mathbf{x}$ and can likewise be written as matrix-vector products $2A_2\mathbf{T}_2\mathbf{x}=\mathbf{y}_2$ and $2A_3\mathbf{T}_3\mathbf{x}=\mathbf{y}_3$. For these first three terms, the connection matrices are fixed and thus we can form $\mathbf{Tx}=(2A_1\mathbf{T}_1+2A_2\mathbf{T}_2+2A_3\mathbf{T}_3)\mathbf{x}$ in a single step with $\mathbf{T}$ and the constants A_1 to A_4 fixed for all problems.

Term 4 in (9) is more complex. Each of the three parts of this term is similar and contains the product of two different neuron states which we can relabel as X_k and X_l times a tensor $\mathbf{D}$ of rank three. If we think of X_k and X_l as components of a vector $\mathbf{x}$, the product of two different neuron states X_kX_l for all k and l is a matrix with components X_kX_l (where k and l are the row and column indices, respectively). This is the vector outer product (VOP) matrix $\mathbf{xx}^t$, where the superscript t denotes the transpose. In terms of this new vector labeling scheme, term 4 can be written as

$$y_{4j} = \sum_{k,l} D_{jkl}X_kX_l, \tag{11}$$

where y_{4j} denotes the j-th component of term 4 in (9). We can view $\mathbf{D}$ as a number of matrices. The sum of products in (11) for a given j is the sum of the point-by-point products of the elements of the VOP matrix and one of the matrices in **D**. The result for all j is a vector $\mathbf{y}_4$, which is called[9] the tensor-matrix inner product, i.e.

$$\mathbf{y}_4 = \mathbf{D}\bullet\mathbf{x}\mathbf{x}^t. \tag{12}$$

Since this is not a simple matrix-vector product, it cannot be calculated in the same way as the other three terms in (9). In addition, **D** changes with the input data whereas the matrix **T** in the other terms is fixed. We now investigate how the linear algebra operations and thresholding described in this section can be done optically.

V. Optical architecture

As has often been noted, connectionist architectures are well-suited for optical implementation since optical systems easily achieve large numbers of interconnects[10]. In the optical design of the Hopfield net by Psaltis and Farhat[11, 12], a matrix-vector multiplier was sufficient. This optical realization is suitable for implementing time evolution equations that are linear in the "neuron activities" X (or equivalently, neural systems and applications in which the energy function is quadratic in X). Such an optical architecture is most efficient if only non-negative connection matrices are involved. Thus, the first two terms in (9) and the positive definite part ($2A_3\sum_j\sum_\beta X_{j\beta a}$) of the third term can easily be calculated by unipolar optical matrix-vector multiplications. Our optimization problem involves one energy term which is cubic (term 4) and a negative term (part of term 3). The optical realization of these terms is now discussed.

Eq. (9) can be realized on the optical system of Fig. 2 as we now detail. This optical system is best drawn in two parts: Fig. 2a (which performs a matrix-vector multiplication and implements terms 1 to 3) and Fig. 2b (which implements term 4). The data plane B1 is common to both parts of the optical system. For simplicity, only the essential lenses are shown. In Fig. 2a, the vector data on a one-dimensional (1-D) bistable device[13] B1 is the current neuron state $\mathbf{x}$. It is imaged vertically and expanded horizontally by L1 onto a 2-D spatial light modulator (SLM1) which contains the matrix interconnection data $\mathbf{T}$ in (10). This is a fixed interconnection pattern that is independent of the data. Thus, it can be recorded on film and need not be altered. The light leaving SLM1 represents the point-by point products $T_{ij}x_j$. This light is integrated vertically and input back to B1 (by 4 mirrors M in the version shown). This forms the matrix-vector product $\mathbf{T}\,\mathbf{x} = (\sum_j T_{1j}x_j, \sum T_{2j}x_j, \ \ldots, \ \sum_j T_{nj}x_j)$.

The same B1 is also present in the system of Fig. 2b. In Fig. 2b, its output is expanded horizontally by L1 and rotated by 90° and expanded vertically (by the beam splitter (BS), mirror (M) and lens L2). These two expanded patterns are superimposed on a second bistable device B2 with horizontal and vertical indices k and l. The light intensity incident on element (k,l) of B2 is $X_k + X_l$. When the threshold for B2 is set to trigger only if both inputs are active, the B2 output is the binary VOP of the B1 data. This VOP matrix is imaged onto SLM2 where it multiplies several multiplexed D-matrices element-by-element. The computer-generated hologram (CGH) behind SLM2 directs the proper element-by-element products to different portions of B1. To implement term 4 on this system, we form $\mathbf{x}$ on B1, the VOP matrix $\mathbf{x}\mathbf{x}^t$ on B2, and the tensor-matrix inner product $\mathbf{y}_4 = \mathbf{D}\bullet\mathbf{x}\mathbf{x}^t$ on B1 using SLM2 and the CGH. We now consider the multiplexing data format on SLM2 and the CGH used.

In our simplified index notation (Sect. IV), the elements of the $\mathbf{y}_4$-vector output due to term 4 are given by (11). Consider the case when j, k and l range from 1 to 3 in D_{jkl}, X_k and X_l. The neuron vector $\mathbf{x}$ then has three elements and the VOP matrix on B2 is 3 x 3. Each element X_kX_l must multiply the three different elements D_{jkl} corresponding to the three different values that j can take given the indices k and l. One possible multiplexed arrangement for the SLM2 data D_{jkl} is shown in Fig. 3, with the VOP elements in row 1 of B2 corresponding to $(k,l) = (1,1), (1,2), (1,3)$ and the elements of row 2 corresponding to $(k,l) = (2,1), (2,2), (2,3)$ etc. The spatial size of the elements on B2 and SLM2 and the imaging optics (not shown) from B2 to SLM2 are such that VOP element (1,1) illuminates the first three elements in column 1 of Fig. 3 (i.e. D_{111}, D_{211} and D_{311}), VOP element (1,2) corresponding to X_1X_2 illuminates the first three elements of column 2 (D_{112}, D_{212} and D_{312}), etc. The bold lines in Fig. 3 indicate regions of SLM2 illuminated by one element of B2. Since **D** is a tensor of rank 3, it is not possible to assign one spatial dimension (horizontal or vertical) to each rank (as is possible with a tensor of rank 2, i.e. a matrix). This arrangement in Fig. 3 multiplies each VOP element X_kX_l by the 3 possible j values in D_{jkl} and thus forms the point-by-point product of the VOP matrix and the different D-matrices. The CGH behind SLM2 focuses all products with the same j onto the same region of B1 (i.e. for the 3 x 9 example in Fig. 3, it sums the light leaving the first, fourth and seventh rows, the light leaving the second, fifth and eighth rows, etc). This forms the sum over k and l of $D_{jkl}X_kX_l$ for each j on a different region of B1. A CGH could be placed between B2 and SLM2 to replicate the B2 data onto the proper regions of SLM2 such that the CGH behind each region (3 x 3 region for the example in Fig. 3) of SLM2 could be a simple spherical lens plus a grating at the required spatial frequency and orientation. However, since the CGH is fixed and independent of the input data, it appears that it can be fabricated on film with sufficient resolution to allow one CGH to be used with improved light budget efficiency. We detail this later.

Next, we consider how the negative part of term 3 in (9) is handled. Recall that B1 is common to both parts of the system (Figs. 2a and 2b). Thus, the input to B1 contains the sum of all the non-negative terms in $\partial E/\partial X_{i\alpha a}$ in (9), i.e. the j-th element on the input side of B1 is $\partial E/\partial X_j + 2A_3N_T$ (where j corresponds to $(i\alpha a)$). Thus, we set the threshold of B1 to be $2A_3N_T$ and hence achieve the subtraction of the positive and constant $2A_3N_T$ portions of term 3 by thresholding without the need to compute negative numbers and the proper neuron vector **x** emerges from B1 We note that the contents of SLM2 need not change between iterations (in minimizing the energy E) for a given set of input measurements. Its contents change for each set of input measurements, but such distance calculations are needed for most multitarget tracking problems.

We next consider an improved version of the system of Fig. 2b with reduced space bandwidth product for B2 and SLM2. To see the significance of this, consider the case of $N_T = 6$, $N_M = 7$ and $N_P = 5$. The vector **x** has 6 x 7 x 5 = 210 components, the VOP has 210 x 210 components and SLM2 requires 210 x 210 x 210 pixels. Clearly, for large values of N_T, N_M and N_P, this architecture becomes unrealistic. Fortunately, not all of the terms in $\mathbf{xx}^t$ are required and most elements of **D** are zero. In (9), we see that only those values of $X_{i\alpha a}X_{j\beta b}$ with $i = j$ are used. To take advantage of this, we divide the vector **x** into N_T smaller vectors $\mathbf{z}_i$ (i=1,..N_T), one for each target j, with each vector of size N_MxN_P. Thus, the full vector **x** can be written as $\mathbf{x}^t = (\mathbf{z}_1{}^t, \mathbf{z}_2{}^t, .., \mathbf{z}_{N_T}{}^t)\}$. This allows us to calculate the VOP $\mathbf{z}_i\mathbf{z}_i{}^t$ of each vector separately, multiply each by the proper elements of **D** point-by-point and sum up the products as indicated in (9).

We now discuss how to efficiently separate the tensor **D** into several smaller matrices. Recall that the full tensor **D** is described by $N_TN_MN_P$ matrices, each of dimension $N_TN_MN_P$. However, the only non-zero elements of these matrices occur for $i = j$ and do not depend on the value of i or j (the entries of **D** do not depend on the target, since the calculations of the elements of **D** involve only distance calculations on the measurements). The fact that only three adjacent time steps a are included in the **D** calculation further reduces the number of non-zero entries. We chose to separate the **D** tensor into N_M matrices, each of dimension $N_M N_P$. To see how this is possible, recall that each distance measure is associated with three measurements (α, β and γ) at three different sequential time steps. A given set of pairs of two measurements at two different (not necessarily successive) time steps is described by a matrix of dimension $N_M N_P$. There are N_M possibilities for the third measurement in the other time step of the three in sequence. Thus there are N_M such matrices, each of dimension $N_M N_P$, that describe the tensor data **D**.

This division is attractive since each of the $N_T = 6$ reduced size 35 x 35 (when $N_M N_P = 35$) VOP matrices can now be multiplied by each of the $N_M = 7$ D matrices. After summation of the proper point-by-point products, the output is $N_T = 6$ vectors, each of dimension $N_M N_P = 35$, i.e. the 6 x 35 = 210 element neuron state vector. Thus, this new arrangement requires that we calculate six 35 x 35 VOPs (i.e. B2 requires only 6 x 35 x 35 = 7350 elements), and SLM2 is only of size 7 x (35 x 35). We have thus reduced the space bandwidth product of SLM2 by a factor of over 1000. B1 is still a 1-D SLM of size $N_T N_M N_P = 210$ elements (one for each element of **x**). This size for **x** is still much less than for cases when one assigns one neuron for each possible single target state (position in x and y) for every time step n (i.e. xyn neurons, where x and y are the number of pixels in the (x,y) projections of the measurement space, respectively). Our assignment of one neuron for each measurement for each target for each time results in a much smaller number (since the number of measurement points per frame is usually much less than the number of 2-D pixels in one image frame).

The N_T VOPs of the partitioned B1 data can be produced on B2, by replacing L1 in Fig. 2 by a CGH that is a set of N_T cylindrical lenses with gratings at different orientations and with different spatial frequencies. This L1 lenslet CGH array focuses the N_T sections of **x** on the vertical B1 device onto N_T different horizontal regions of B2 and replicates the vector data in each of the N_T sections horizontally at B2. As before, L2 expands **x** vertically onto B2. After thresholding, the result on B2 is the desired N_T VOPs of the $N_M N_P$ element vector in each of the N_T regions on B1. These VOP's thus emerge from B2 as N_T matrices horizontally separated, with each matrix of dimension $N_M N_P$.

The architecture in Fig. 2 (even with the reduced SLM2 and B2 resolution requirements) requires fast non-linear optical devices for B1 and B2 (since they limit the speed for one iteration of the system). Since such devices are not yet readily available, we present the alternative architecture of Fig. 4 that does not require a non-linear optical device. Rather, it computes the vector outer product by feeding the **x**-vectors to the rows and columns of an electroded SLM such as a ferroelectric liquid crystal[14] . Thus, SLM3 in Fig. 4 can be substituted for the 2-D bistable device B2 in Fig. 2b. The outer products formed by SLM3 are imaged onto SLM2, where they multiply D, and the CGH focuses the terms belonging to the same sum to the same point on the detector array D1 which now replaces B1 in Fig. 2. The thresholding is done electronically by an array of operational amplifiers fed from the detectors D1. These detector outputs provide the electronic inputs to SLM3 in Fig.4.

The architctures which we have introduced in this section are quite complicated. We do not find this surprising: it is well known that the multitarget-tracking problem is very hard. Therefore, any parallel architecture which attempts to solve this problem in real time will probably be rather sophisticated.

VI. Simulation results

The above neural net and algorithm were simulated for the case of $N_T = N_M = 4$ and $N_P = 5$. A 256 x 256 x 256 three-dimensional (x,y,z)-space was used. The initial four measurement positions were chosen from a random number generator. The velocities and directions of each target were similarly chosen. $N_P = 5$ equally spaced time steps were generated for each target. The target directions were evenly distributed over 360°. They generally ranged in length (in a 2-D projection) by a factor of 5:1 with the longest track of five time steps occupying approximately 70% of the field of view. To simulate imperfect measurements, each position was perturbed by $n\%$ of noise. This was achieved by adding a uniformly distributed random number to the measurement. This produced a random variation in the location of the measurement by at most $n\%$ of its distance from the origin (the center of the 256 x 256 x 256 space).

For each run (corresponding to a given value $n\%$ of noise), ten different initial data conditions (initial target positions, velocities and directions) were used. Thus, for each value of noise, ten sets of four target tracks were processed. The initial conditions (the neuron states at which the neurons were started in the processing) were chosen by randomly perturbing the all-equal condition (in which each coordinate is assigned to each target with equal probability). This is detailed more fully in Hopfield and Tank[6] and motivation for randomizing initial conditions is also provided there. Ten different random perturbations were used in the ten simulations in each run. The A_1 to A_4 coefficients were chosen to equally weight the first three terms in (9), i.e. $A_1 = A_2 = A_3 = 15$ with A_4 chosen to be less ($A_4 = 1.8$). Larger A_1 to A_3 values were used to give more weight to the first three terms in (9), i.e. we <u>must</u> have the proper form for the matrix $X_{i\alpha a}$. A_1 and A_2 should be chosen equal (since these terms correspond to enforcing the correct row and column structure, respectively, and are thus equivalent by symmetry). A_3 could differ from these terms since it multiplies a different type of term; on the other hand, the first three terms in (9) have similar roles and it was found that $A_3 = A_1 = A_2$ gave good results. Term 4 is given less weight ($A_4 = 1.8$) since it involves the sum of more products than do the other terms and satisfying conditions (1), (2), (4), (5) and (6) in Sec. III (terms 1 to 3) is essential. No detailed optimization of A_1 to A_4 was attempted.

The coefficients $D^a_{\alpha\beta\gamma}$ and the connection matrix (T) were calculated. The threshold $2A_3N_T$ in term 3 in (9) was slightly increased (from $2A_3N_T$ to $2A_3N_T + 0.035$). We found this to be helpful when noise is present. Such an increase compensates for the neglected higher-order terms in the Taylor expansion in (6). In the simulation, the neural activities were updated as they were calculated, i.e the state of the first neuron was calculated (using the most recent values of all other neurons) and then the state of neuron one was updated before calculating the state of neuron two, etc. This better models[8] a continuous time rather than a discrete time system. After one set of updates of the neurons, a new iteration commenced until the neural net converged. The serial mode neural updating results in faster convergence than if all new neuron states are calculated in parallel and simultaneously fed back, as is usually done[7].

Table 1 shows the results obtained. Column 1 lists the percentage noise (positional variation) introduced into the measurements. Column 2 gives the percentage of runs that converged in less than 50 iterations and column 3 gives the average number of iterations to convergence for these cases. We restricted the number of iterations to 50. If convergence was not obtained after 50 iterations, we call this an error. If the system converged to the wrong set of measurement/target pair assignments, this is also an error. As seen, the neural net converged in much less than 50 iterations on the average. Also, we found that whenever the net converged, it converged to the correct target-measurement matching. A proof of this remains to be derived (but from these initial tests, one should be able to accept results that converge with a high probability). At low noise ($n = 0\%$ or 2.5%), the neural net converged to the correct solution for all 40 target tracks (i. e. excellent performance was obtained.) As noise increased, correct convergence or two other conditions occurred (the neural net wandered with no apparent trend to convergence or it oscillated between two states, one being the correct solution and the other differing in one target/measurement pairing.)

Fig. 5 shows a representative example of the evolution of the neurons to a stable state for one set of data. In Fig. 5, each rectangle in the 5 x 5 grid shown represents the 4 x 4 matrix $X_{i\alpha a}$ for $i = 1$ to 4, $\alpha = 1$ to 4 and a fixed (the assignments at one time step). The matrices from left to right on one row correspond to different time steps $a =$ 1 to 5 and the matrices in the different rows are the neuron states after various numbers of iterations, as indicated. The horizontal axis of each matrix has four divisions, corresponding to the four different targets which are assigned to the measurements. The four different measurements in a given time frame occupy the various rows of the indicated matrix. A dark spot at position (i,α) indicates that target i has been assigned to measurement α. The neural net has converged if the matrices are unchanged between two adjacent iterations. The targets are correctly tracked if $X_{i\alpha a}$ (for a fixed) has only one entry per row and one entry per column and row and column entries are the same in all time frames. This is because the measurements for this example were generated such that the same target was given the same number in all time frames.

From Fig. 5, the target-measurement associations appear random after the first iteration (i. e. the neural net has assigned many of the measurements to only one target and vice versa). They converge and eventually reach a correct solution after 15 iterations.

VII. Summary and Conclusion

In this paper, a neural net energy minimization formulation of multitarget tracking with a reduced number of neurons was presented. Cubic energy terms were present and an optical architecture to implement this neural net was described. Initial simulations indicate that the al:orithm has desirable performance, even in the presence of random noise. We expect improved performance when the optimal values for the A-coefficients are understood, and when the characteristics of good initial neural states are known.

The formulation presented can be extended in a variety of ways. Other sensors than the simple position sensors we employed can be introduced and non-straight tracks (i.e. accelerating targets) can be considered. These extensions will not complicate the energy formulation or optical architecture that we have presented, since they require only that the D-tensor be calculated in a different manner.

The main reason why conventional tracking algorithms are so slow is because they enforce a serial structure onto a process that is essentially parallel: the various targets are being measured simultaneously, but are processed one-by-one. The parallelism of our neural architecture means that it can perform at a much higher rate. We therefore feel that this architecture is prototypical of the types of systems that will have to be used to successfully deal with a complicated multitarget scenario.

Acknowledgment: Funding for this research was provided by a contract from the Strategic Defense Initiative Office of Innovative Science and Technology, monitored by the Office of Naval Research (Contract N00014-86-K-0599).

References

1. R.P.Lippmann, "An introduction to computing with neural nets", IEEE ASSP Magazine **2**, pp. 4-22 (1987).

2. G. A. Carpenter and S. Grossberg, Eds., "Special Issue on Neural Nets", Applied Optics **26**, pp. 4909-4992 (1987).

3. B.V.K. Vijaya Kumar and Bruce L. Montgomery, "A direct storage nearest neighbor associative-memory model for optical pattern recognition", Applied Optics **25**, pp. 3759-66 (1986).

4. Gail A. Carpenter and Stephen Grossberg, "A Massively parallel architechture for a self-organizing neural pattern recognition machine", Computer Vision, Graphics, and Image Processing **37**, pp. 54-115 (1987).

5. K. Fukushima, S. Miyake, and T. Ito, "Neocognitron: A neural network model for a mechanism of visual pattern recognition", IEEE Transactions on Systems, Man, and Cybernetics **SMC-13**, pp. 826-834 (1983).

6. J.J.Hopfield and D.W.Tank, "Computing with neural circuits: A model", Science **233**, pp. 625-633 (1986).

7. M. Takeda and J. W. Goodman, "Neural networks for computation: number representations and programming complexity", Applied Optics **25**, pp. 3033-3046 (1986).

8. J.J. Hopfield, "Neural Networks and Physical Systems with Emergent Collective Computational Abilities", Proc. Natl. Acad. Sci. USA **79**, pp. 2554-58 (1982).

9. M. R. Spiegel, *Vector Analysis,* Schaum, New York, 1959.

10. L. D. Hutcheson, Ed., "Special issue on Optical Interconnections", Optical Engineering **25**, pp. 1075-1141 (1986).

11. D. Psaltis and N. Farhat, "Optical Information Processing based on an

Associative-Memory Model of Neural Nets with Thresholding and Feedback", Optics Letters **10**, pp. 98-100 (1985).

12. Nabil H. Farhat, Demetri Psaltis, Aluizio Prata, and Eung Paek, "Optical Implementation of the Hopfield model", Applied Optics **24**, pp. 1469-75 (1985).

13. J.L. Jewell, Y.H. Lee, M. Warren, H.M. Gibbs, N. Peyghambarian, A.C. Gossard, and W. Wiegmann, "3-pJ, 82 MHz Optical Logic Gates in a Room Temperature GaAs-AlGaAs Multiple-QuantumWell Etalon", Appl. Phys. Lett. **46**, pp. 918-920 (1985).

14. M. F. Bone, D. Coates, W. A. Crossland, P. Gunn and P. W. Ross, "Ferroelectric liquid crystal display capable of video line address times", Displays **1**, pp. 115-8 (1987).

List of Figure titles

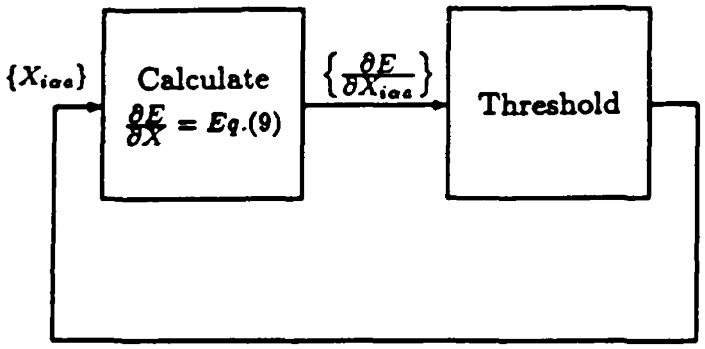
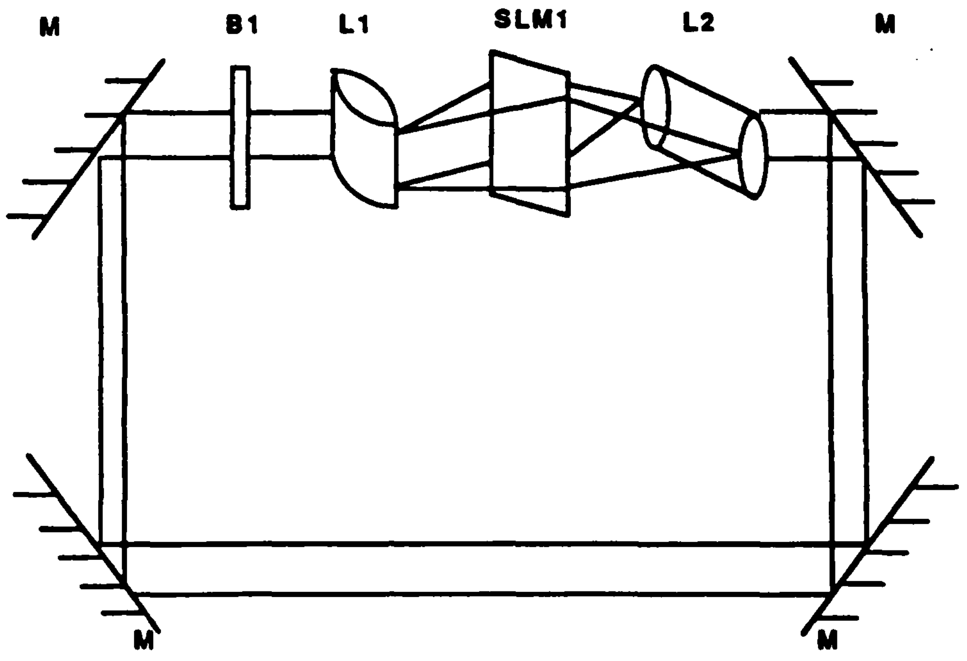
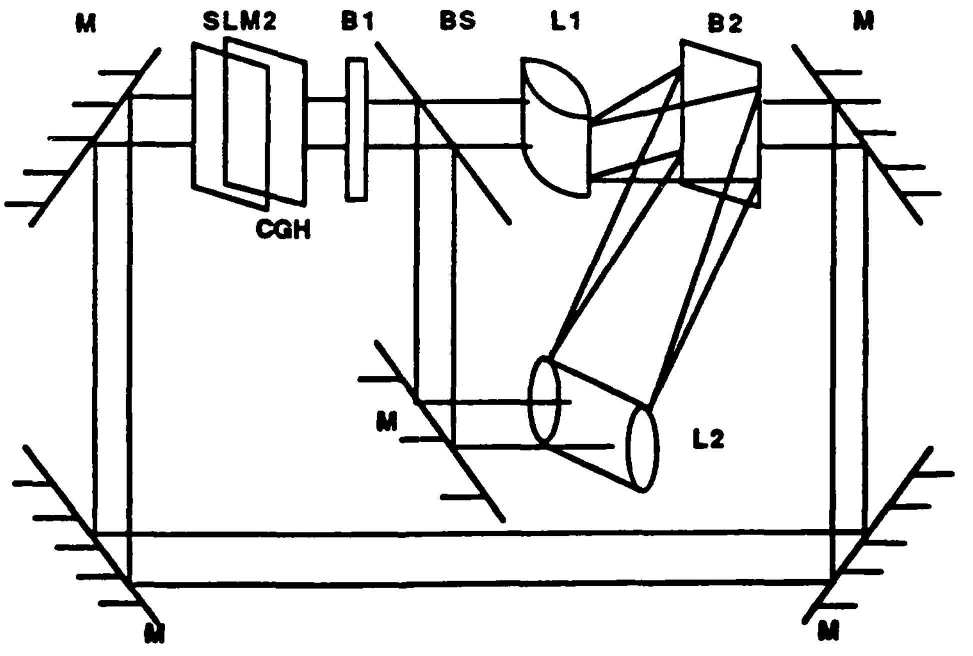

$\{X_{ia a}\}$
Calculate
$\frac{\partial E}{\partial X} = Eq.(9)$
$\left\{\frac{\partial E}{\partial X_{ia a}}\right\}$
Threshold

Fig. 1



(a)



(b)

Fig. 2

D_{111}	D_{112}	D_{113}
D_{211}	D_{212}	D_{213}
D_{311}	D_{312}	D_{313}
D_{121}	D_{122}	D_{123}
D_{221}	D_{222}	D_{223}
D_{321}	D_{322}	D_{323}
D_{131}	D_{132}	D_{133}
D_{231}	D_{232}	D_{233}
D_{331}	D_{332}	D_{333}

Fig. 3

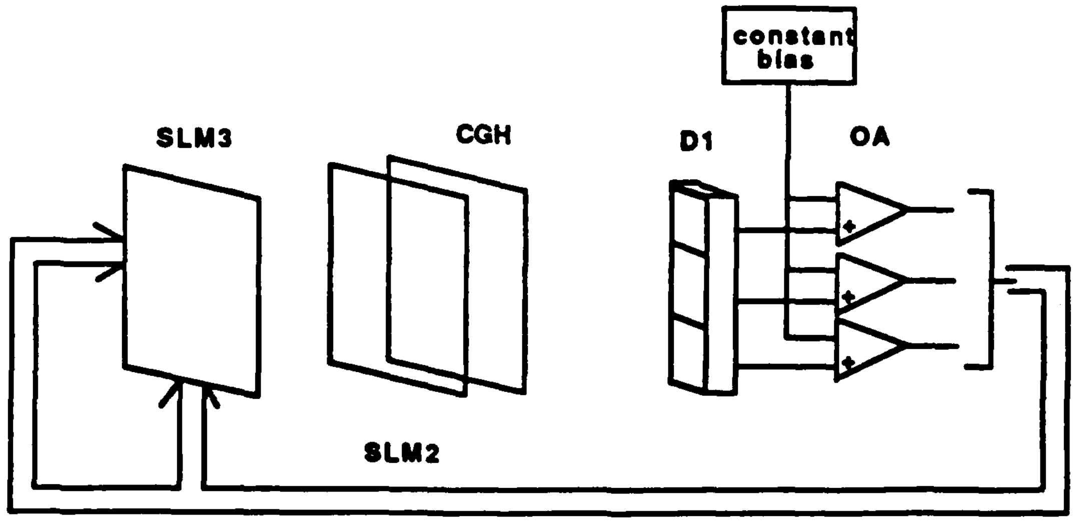


Fig. 4

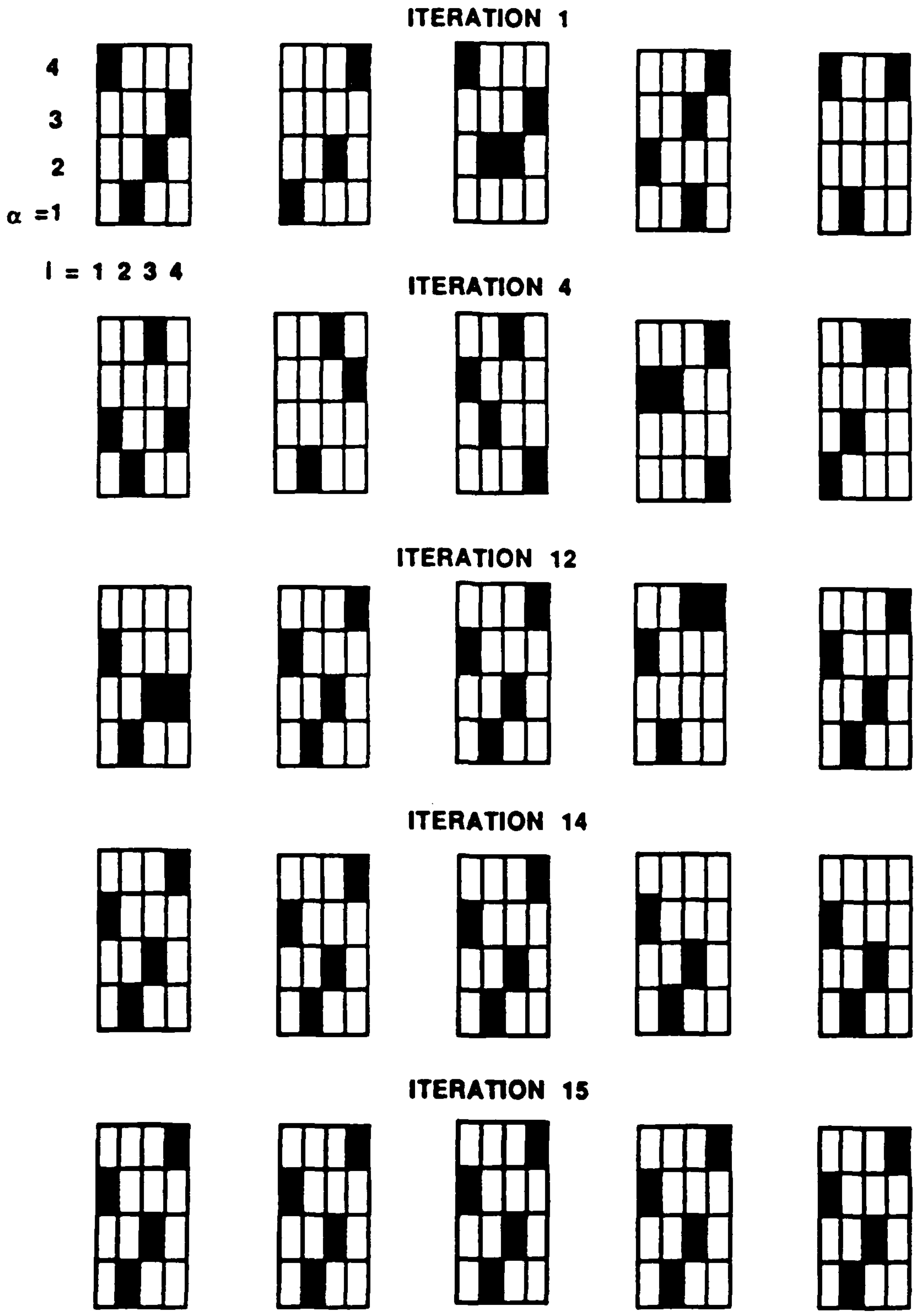

ITERATION 1
4
3
2
α =1
i = 1 2 3 4
ITERATION 4
ITERATION 12
ITERATION 14
ITERATION 15

Fig. 5

%noise	% successful	avg. no. of iterations
0	100	19
2.5	100	16
5	90	20
7.5	80	13

Table 1: Simulation results for tracking of 4 targets through 5 time steps.

CHAPTER 6

"KL Techniques for Optimal Processing of Time Sequential Imagery"

Proc. SPIE
Volume 1007
November 1988

KL Techniques for optimal processing of time sequential imagery

Pieter Vermeulen and David Casasent

Center for Excellence in Optical Data Processing
Carnegie Mellon University
Pittsburgh, PA 15213

Abstract

Time sequential imagery is difficult to analyze, because of its high dimesionality. This paper advances a new algorithm that screens input data in an intelligent way, discards data with negligible information and uses the remaining images to represent the sequence in an optimal compact form. We present data to illustrate how this algorithm can be used to do novelty filtering, novelty detection, segmentation, background independent modelling and classification.

1 Introduction

A salient aspect of visual data is the large amount that is generated in typical situations [1-3]. For video data, bit rates of 5Mb/s are typical. Thus, it would be useful to limit the amount of data that must be processed at a particular instant. Fortunately, most of an input image sequence is usually quite redundant. Consider the output of a camera mounted on an autonomous vehicle: in this case, most of the information in a frame is relatively slowly changing background. Once the background is known, only a small percentage of the information in a frame is new or *novel.* Also, most new information consists of shifts in the location of an object or different aspect views of an object that has already been seen.

Our concern is to process a time sequence of images and to model objects and/or background from a time sequential image sequence in a compact form suitable for recognition. This is closely related to bandwidth compression problems [4-6] in image transmission. However, we are not concerned with the visual quality of the result, but rather the ability of the model produced by the compression technique to retain information useful for recognition purposes. Such problems can be optimally solved by using a truncated Karhunen-Loève [7-10] (KL) expansion of the process. In this case, the process is represented by a set of basis vectors and coefficients associated with each basis vector. One such set of basis vectors and coefficients is the eigenvectors and eigenvalues of the covariance matrix of the process. This is the KL expansion of the process.

Section 2 reviews the KL expansion, several techniques to efficiently compute it for an image sequence and shortcomings associated with each. A new and efficient algorithm is then advanced (Section 3) and we discuss the use of this algorithm for various new applications (Section 4). These applications includes three dimensional object modeling, background modeling, novelty detection, novelty filtering, tracking and the ability to determine the *type* of new information in new frames of time sequential imagery.

2 KL description of time sequential imagery

Given a process with samples $\mathbf{x}_i$ (where $i = 1, \ldots, I$) which are a sequence of I lexicographically scanned images, assume that we wish to find an optimum compressed description of that sequence. One description of a process is a series expansion in terms of a set of basis vectors, the most obvious being the sequence of images itself. The optimum series expansion description of the sequence $\hat{\mathbf{x}}$ is the Karhunen Loève expansion [8]

$$\hat{\mathbf{x}} = \sum_{k=1}^{I} \lambda_k \Phi_k, \tag{1}$$

where Φ_k and λ_k are the normalized eigenvectors and eigenvalues respectively of the sample covariance matrix $\mathbf{C}$ of the sequence of input vectors. The eigenvector and eigenvalue pairs are ordered in terms of descending ei;envalues. In an optimum series expansion representation, the basis vectors must be mutually orthof,onal to ensure that the information contained in each is unique. This is obviously the case when the 'basis vectors are eigenvectors.

For an entire time sequential process, energy is a good measures of activity or information. In some recognition tasks, the mean of a process does not contain significant discrimination information and this information should not be included in the optimum description. The mean information can be ignored by using the variance instead of the energy of the process as a measure of information. The variance of the process is the sum of all the eigenvalues. If the basis vectors are orthogonal, then the amount of relative information preserved in each eigenvector is given by its eigenvalue normalized by the sum of all the eigenvalues. If the mean information is important, then we can preserve it in the optimum representation by using the correlation matrix $\mathbf{R}$ of the process instead of the covariance matrix $\mathbf{C}$. When using $\mathbf{R}$, the energy of the process is the sum of the eigenvalues, and as before we can calculate the relative amount of information in each eigenimage, where the information measure in this case is the energy of the process.

The KL expansion of a process is attractive since the set of basis vectors can easily be ordered in terms of the information content of each. (If the eigenvectors are orthonormal, then the information content is given by the eigenvalues associated with the eigenvectors.) The eigenvalues provide a measure of the information content that allows us to make decisions on the number of basis vectors and which basis vectors to retain to store a certain percentage of the information in the sequence. We will make use of this in our time sequential work. If the expansion is truncated and only the basis vectors associated with the largest eigenvalues are kept, then this truncated KL expansion is the minimum mean square error representation of the process (and is better than any other series expansion with the same number of basis vectors).

Even if we only desire the set of K eigenvectors associated with the K largest eigenvalues, this computation is generally too complex for an image sequence. To see this, consider a sequence of $N \times N$ two-dimensional images, each of which is lexicographically ordered into a one-dimensional N^2 element vector. If N is typically of order 256, the covariance matrix is large ($N^2 \times N^2 \approx 64K \times 64K$), making the direct calculation of its eigenvectors very difficult. The covariance matrix is also highly singular with a rank much less than its dimensions. Recursive algorithms, such as the simultaneous iteration method [11] and stochastic gradient ascent method [8], have been suggested to ease the computation of the eigenvectors of this large matrix. Although they allow more efficient calculation of the first few eigenvectors, the computations are still excessive (since convergence is slow and often does not occur) and both algorithms require the storage of all of the images in the sequence. This very excessive storage requirement is what we want to avoid by compression of the data.

When the number of images I in the sequence is much smaller than the dimension N^2 of each image, an economical technique to obtain the eigenvectors of $\mathbf{C}$ uses singular value decomposition (SVD) as we now note. Since the covariance matrix has a rank of at most I, there are at most only I nonzero eigenvalues and those eigenvectors corresponding to the zero eigenvalues are not of interest, since they contain zero information. The simplified [12] algorithm to compute the eigenvectors and eigenvalues using SVD is to form the (VIP) matrix $\mathbf{V}$ of the image sequence with the mean removed, calculate its eigenvectors and from them obtain the eigenimages of the covariance matrix. We now highlight this algorithm.

The VIP matrix with elements $v(l,m)$ is given by

$$v(l,m) = v(m,l) = (1/I)\,(\mathbf{x}_l - \mathbf{m})^T(\mathbf{x}_m - \mathbf{m}). \qquad l,m = 1,\ldots,I \tag{2}$$

where $\mathbf{m}$ is the process mean. The VIP matrix is $I \times I$ (which is much smaller than $\mathbf{C}$), hence calculation of its eigenvectors $\boldsymbol{\theta}_i$ and eigenvalues γ_i $(i = 1,\ldots,I)$ is simple. We use the term *eigenvector* to refer to the eigenvectors of the lower dimensionality VIP matrix. We use the term *eigenimage* in referring to the eigenvectors of the higher dimensionality covariance matrix $\mathbf{C}$. The larger dimensional normalized eigenimages $\bar{\boldsymbol{\Phi}}_i$ and their eigenvalues λ_i can be obtained from the eigenvectors $\boldsymbol{\theta}_i$, their eigenvalues γ_i and the original images using

$$\begin{aligned} \bar{\boldsymbol{\Phi}}_i &= \frac{1}{\sqrt{\gamma_i}} \sum_{k=1}^{I} \theta_i^k \mathbf{x}_i \\ \lambda_i &= \gamma_i \end{aligned} \tag{3}$$

where θ_i^k is the k-th element of the i-th eigenvector $\boldsymbol{\theta}_i$. The eigenvalues γ_i of the VIP matrix and the eigenvalues λ_i of the covariance matrix are equal. Thus, the eigenimages of $\mathbf{C}$ with nonzero eigenvalues are linear combinations of the input images with weighting coefficients given by the elements of the corresponding eigenvector of $\mathbf{V}$ as in (3).

The number of images I in a sequence is still large (1800 for only a one minute sequence of video data). Thus the eigenvector solution of the reduced dimensionality $I \times I$ eigenvalue problem described above is still quite difficult and we still need to store all of the images. However, in most time-sequential image sequences and applications, a significant number of the eigenvalues of $\mathbf{V}$ are also zero or sufficiently close to zero to be ignored (since the VIP matrix is also highly singular or numerically ill conditioned, because of the natural redundancy in an image sequence). In this case (which is typical of image sequences), we can provide a quite significant compression of many images in a sequence, thus allowing use of the recursive SVD algorithm [13,14]. As an example of when this occurs, consider a time sequence of images in which the only difference between successive frames is a slow changes in the scale, aspect or rotation of an object in the scene. To quantify this situation, we generated four sequences of four different rolling aircraft flying in a straight line with only roll differences between frames. Each sequence had 36 images at 10 degree intervals of roll, with the aircraft centered in each frame. For these data, we found that 3 eigenimages contained enough information to correctly recognize the aircraft in any roll orientation in the sequence (and could also recognize a test set of 12 images at orientations not included in the training set). We therefore concluded that the rank of the VIP matrix and hence also the rank of the covariance matrix for this process was close to 3, which is much smaller than the number (36) of training images. Since we correctly classified the aircraft at roll orientations not presented in the training set, we concluded that if more training images were included in the training set, that the rank of **V** and **C** would still have been close to 3.

For such cases when the rank K of a process is much less than I and is known in advance, a recursive SVD algorithm [13,14] can be used to estimate the K dominant eigenvectors and eigenimages of the process (with a required storage of no more than K images). When K can be $5-10$ out of a sequence of $I = 1800$ images, this is quite significant. To start the algorithm, the first $K+1$ images in the sequence are used to calculate an initial set of $K+1$ eigenimages using SVD. The last eigenimage (the one with the smallest eigenvalue) is assumed to have a small associated eigenvalue and is discarded and the dominant K eigenimages are used as an initial estimate of the first $K+1$ images in the process. As a new image arrives (images $K+2$, etc), it is combined with the K eigenimages weighed by the square root of their eigenvalues and a new set of $K+1$ eigenimages is formed. The smallest eigenimage from this set is then discarded. This process of updating a set of K eigenimages is continued until the process is completed, at which time the resultant K eigenimages are used as a description of the image sequence.

This algorithm is storage efficient, since only the K most significant eigenimage estimates are saved and since each image is seen only once. It is also computationally efficient, since the eigenvectors and eigenvalues are calculated from the smaller VIP matrix. However, it has three shortcomings. First, *a priori* information is required that the process has a small rank and a good estimate of this rank is needed (i.e. the number of eigenvectors K to be saved). Such information is often not available. Secondly, there is no clear relation between K and the amount of information retained in the final set of eigenvectors, because information is discarded at each iteration. Therefore, the final K eigenimages obtained are only approximations of the K eigenimages of **C** associated with the largest eigenvalues. The third problem is that the process must be zero-mean. If the process is not zero-mean, then all of the images have to be collected and stored before the mean vector can be estimated and subtracted from each image at each iteration step (this is not realistic).

We now discuss these issues further and present (Section 3) a new algorithm that overcomes these disadvantages and yet retains the computational and storage advantages of recursive SVD.

3 Novelty detector and filter algorithm

We do not use zero mean data (because the computation of the mean of the process requires excessive storage as noted above). In our applications the mean information is important (for example, to estimate the image background) and we have found that keeping the mean information does not influence our recognition applications. We will use the VIP matrix of the non-zero-mean data. In this case, its eigenvalues equal those of the correlation matrix **R** rather than the covariance matrix **C**. The KL expansion uses the covariance matrix and compresses information to preserve maximum covariance of the process which leads to a minimum least square error representation of the process. Calculating the eigenimages of the correlation matrix rather than those of the covariance matrix is equivalent to compressing information to preserve maximum energy rather than maximum variance of the process and also leads to a minimum least square error representation of the process [9,10]. The rank of the correlation matrix **R** is one more than the rank of the covariance matrix **C**, since it also contains the mean or background information. When we use the VIP matrix of **R**, we are not computing the KL basis set of the process, but are determining a related (and equally useful) basis set with one additional vector that contains the background information.

We refer to our new algorithm as a novelty detector and filter (these terms will be clear shortly). We monitor the information content (process energy) and ensure that a given percentage of the information in the image sequence is retained in our eigenvector representation. This is a more direct measure than retaining a fixed number K of eigenimages. To achieve this, we allow the number K of eigenimages retained to vary and to adapt to the data (i.e. we adapt the number of significant eigenimages to the given image sequence and application and do not fix K as is required in recursive SVD). We provide two novelty measures (discussed below) for use in controlling the processor. We also discard new input images (with negligible information) rather than eigenimages (as is done in recursive SVD) and we control this by a novelty detector. We now discuss our algorithm and detail these issues.

Adaptive Recursive SVD Algorithm	
Step	**Description**
1	Initialize the iteration counter $i = 1$ and the sequence counter $j = 1$
2	Initialize the covariance rank estimation $K_1 = 1$.
3	Read the first image $\mathbf{x}_1$ to form the first estimate of the eigenimages of the process.
4	Increment the iteration counter i
5	Increment the rank estimate $K_i = K_{i-1} + 1$
6	Increment the sequence counter j and read the next image in the sequence $\mathbf{x}_j$.
7	Form the $K_i \times K_i$ VIP matrix $\mathbf{V}_i$ as detailed in Eqs (4) and (5).
8	If the current image is not *novel* as indicated by the novelty detector in Eq (8), then go to step 6
9	Form the K eigenimages and eigenvalues from the eigenvectors and eigenvalues of $\mathbf{V}_i$ according to Eq (9). This updates our eigenimage set.
10	Estimate the rank of the process $\widehat{K}_i$ using Eq (11). Determine if one should add an additional eigenimage or discard the excess eigenimages. Set $K_i = \widehat{K}_i$.
11	If there are more images in the input sequence, then go to step 4

Table 1: The Adaptive Recursive SVD Algorithm

Our adaptive recursive SVD algorithm is summarized in Table 1. Steps 1–3 initiate the process and read the first image ($K_1 = 1$). This image is the initial representation of the process and is therefore used as the first eigenimage. Its eigenvalue is its squared modulus. We first provide an overview of the algorithm. The iteration counter i (step 4) denotes the number of times that the eigenimages have been updated. The sequence counter j (step 6) denotes the number of the current input image in the sequence. This can differ from the number of iterations i, since input images with negligible information are discarded, without updating the set of eigenimages and hence incrementing i. In steps 4–6, the number of iterations is incremented and so is the rank estimate K (to allow for the possibility of adding an additional eigenimage) and a new image is read. If the new image is not novel (step 8), then the algorithm returns to step 6, reads a new image and proceeds. Note that K is not incremented for this image and that K is only incremented if the input image is novel (only step 11 returns the algorithm to step 4). Also, note that at step 10 we discard negligible eigenimages and hence reduce K. Thus, K can change constantly (throughout even one iteration) and K_i denotes the present rank estimate.

Let us assume that at the end of iteration $i = 4$, we have $K_i = K_4 = 4$ (i.e. a rank estimate of 4). In the fourth iteration, we have a $K_4 \times K_4 = 4 \times 4$ VIP matrix and its 4 eigenvectors $\theta_{i,l} = \theta_{4,l}, l = 1, \ldots, 4$ (where $\theta_{i,l}$ denotes the l-th eigenvector at iteration i) and 4 eigenvalues $\lambda_{i,l} = \lambda_{4,l}, l = 1, \ldots, 4$ (where $\lambda_{i,l}$ denotes the l-th eigenvalue at iteration i). Next, in iteration 5, we enter step 4, where we increment K_i and obtain $K_5 = 5$ (step 5) and read in a new image (step 6). We compute the new 5×5 VIP matrix (step 7) in Eq. (2) and compute its eigenvalues and eigenvectors. This operation is greatly simplified as we now detail. At the present iteration, only one new image is present and thus $K_i - 1$ of the K_i vectors forming the new VIP matrix $\mathbf{V_i}$ are already orthogonal (since they are the prior $K_i - 1$ retained eigenimages). Specifically, the top-left $K_{i-1} \times K_{i-1}$ submatrix of $\mathbf{V_i}$ is diagonal and its known first $K_i - 1$ diagonal elements are,

$$v_i(l,l) = \frac{i-1}{i}\lambda_{i-1,l} \qquad l = 1, \ldots, K_i - 1. \tag{4}$$

Thus, the only new elements in the VIP matrix are the elements in the last row $v_i(K,l)$ and column $v_i(l,K)$. As in recursive SVD, the normalized eigenimages have to be weighed by the square root of the corresponding eigenvalues. From (3), we see that this weighting is done automatically in SVD. Since the VIP matrix is symmetric, we need only concern ourselves with its last row or column. These elements are given by

$$\begin{aligned} v_i(l,K) &= v_i(K,l) = (1/i)\,\mathbf{x}_j^T \Phi_{i-1,l} \quad l = 1, \ldots, K_i - 1 \\ v_i(K,K) &= (1/i)\,\mathbf{x}_j^T \mathbf{x}_j \end{aligned}\,. \tag{5}$$

They are easily calculated as the VIP of the new image $\mathbf{x}_j$ and the prior eigenimages $\phi_{i-1,k}$ and only K_i (a small number) VIP calculations are needed. This completes step 7.

In step 8, we determine if the new image should be used or not (i.e. is it novel). We now discuss the motivation for this step. When modelling a very unconstrained process such as an aircraft, one would like to select the training images carefully to avoid *cluttering* the problem with too much data. There are two advantages in reducing the sample data set. First and obvious, by limiting the number of input images, we are limiting the dimensionality of the problem and therefore its computational complexity. Second, a lower dimensionality problem ensures better conditioning of the VIP matrix and therefore reduces the computational resolution (or precision) requirements in the algorithm. In step 8, we determine whether the input image $\mathbf{x}_j$ contains enough significant information to merit its inclusion in the training set. If the image does not contain statistically significant novel information, then we discard it (step 8) and consider the next input image (step 6). We now discuss how to determine this measure of novelty.

The last column (row) of the new VIP matrix $\mathbf{V}_i$ are the VIPs of the new image $\mathbf{x}_j$ and the prior set of eigenimages $\Phi_{i-1,k}, k = 1, \ldots, K_i - 1$. The diagonal elements of $\mathbf{V}_i$ are the modulus of each eigenimage and are unchanged except for the last diagonal element which is the modulus of the newest input image $\mathbf{x}_j$. From these elements of $\mathbf{V}_i$ we can compute the angle between the $\mathbf{x}_j$ and each one of the set of eigenimages. The cosines of these angles are the projections of a unit vector along the direction of $\mathbf{x}_j$ onto each of the eigenimages. The set of eigenimages forms a normal basis set in Euclidean space and therefore the squares of these projections of the unit vector can at most sum to one. If $\mathbf{x}_j$ is contained in the set of eigenimages, then the squares of the projections will sum to one. If $\mathbf{x}_j$ is orthogonal to the set of eigenimages, then the squares of the projections will sum to zero. The sum of the squares is therefore a measure of the percentage of $\mathbf{x}_j$ that is contained in the set of eigenimages (a correlation coefficient) and can therefore be used to obtain a novelty measure n_i. We first calculate the direction cosines α_i

$$\alpha_{i,k} = \frac{\mathbf{x}_j^T \Phi_{i-1,k}}{|\mathbf{x}_j|\ |\Phi_{i-1,k}|} = \frac{v_i(k, K_i)}{\sqrt{v_i(K_i, K_i)\, v_i(k,k)}} \tag{6}$$

and subtract the sum of their squares from one to obtain the novelty measure n_i

$$n_i = 1 - \sum_{k=1}^{K_i-1} \alpha_{i,k}^2 = 1 - \sum_{k=1}^{K_i-1} \frac{v_i^2(k, K_i)}{v_i(k,k)\, v_i(K_i, K_i)}. \tag{7}$$

Note that there is no need to calculate the eigenvectors and eigenvalues of the VIP matrix $\mathbf{V}_i$ in order to calculate this novelty measure. Rather, we only require the VIPs. A *novelty detector* for image $\mathbf{x}_j$ can now be defined by comparing the novelty measure n_i in (7) to a threshold T_d

$$N\{\mathbf{x}_i\} = \begin{cases} \text{true} & \text{if } n_i > T_d \\ \text{false} & \text{otherwise} \end{cases} \tag{8}$$

When $N\{\mathbf{x}_j\}$ is true, image $\mathbf{x}_j$ is novel. If $N\{\mathbf{x}_j\}$ is false, we discard image $\mathbf{x}_j$ and return to step 6. For T_d in (8) we typically use 0.01–0.05. This concludes step 8.

If we determine (step 8) that $\mathbf{x}_j$ is novel, we now include it in our data (eigenimages). We next determine if we should add an additional eigenimage and increment K_i. We first calculate (step 9) the new set of K_i eigenimages and eigenvectors from the VIP matrix $\mathbf{V}_i$ and the prior $K_i - 1$ eigenimages. Since $\mathbf{V}_i$ is small, real and symmetric, its eigenvectors and eigenvalues are easily calculated in real-time by techniques such as the Jacobi algorithm or the QR-method [15-18]. The set of K_i new eigenimages $\mathbf{\Phi}_{i,k}$ (eigenimage k at iteration i) can be obtained from the eigenvectors $\theta_{i,k}$ of the new VIP matrix (where $\theta_{i,k}^l$ is the l-th element of the k-th eigenvector of $\mathbf{V}_i$ at iteration i), the $K_i - 1$ prior eigenimages $\mathbf{\Phi}_{i-1,k}$ and the new input image $\mathbf{x}_j$ by

$$\mathbf{\Phi}_{i,k} = \sum_{l=1}^{K-1} \theta_{i,k}^l \mathbf{\Phi}_{i-1,l} + \theta_{i,k}^K \mathbf{x}_j. \tag{9}$$

From (9), we see that computing the eigenimages from the eigenvectors of $\mathbf{V}_i$ is simple. It involves only additions of the eigenimages $\mathbf{\Phi}_{i-1,k}$ and the new input image $\mathbf{x}_j$, weighed by the eigenvector elements $\theta_{i,k}^l$. All pixels of each image are weighed by the same value. Calculation of the eigenimages and their eigenvalues completes step 9.

In step 10 we now determine how many of the new eigenimages (step 9) to keep. During each iteration, we increment the rank estimate (step 5) and compute a new set of eigenimages (step 9). Since the new image information is included (potentially) in all the new eigenimages, we have reorganized and redistributed the present data by computing the new set of eigenimages. The rank of the new sample correlation matrix $\mathbf{R}_i$ (made up of all novel images thus far) does not necessarily increase by one, even if the new image contained a significant amount of new information. We now detail how we form an estimate of the true rank of $\mathbf{R}_i$. Each eigenvalue of $\mathbf{V}_i$ when normalized by the sum of all the eigenvalues indicates the fractional information in the corresponding eigenvector. Since the eigenvalues of $\mathbf{V}_i$ equal those of $\mathbf{R}_i$, this measure of fractional information also holds for the $k = K_i$ eigenimages $\mathbf{\Phi}_{i,k}$. (We assume that the information discarded at each iteration is negligible.) The fraction $F_{i,k}$ of information in eigenimage k is

$$F_{i,k} = \left| \frac{\lambda_{i,k}}{\sum_{k=1}^{K_i} \lambda_{i,k}} \right|. \tag{10}$$

where $\lambda_{i,k}$ is the k-th eigenvalue at the i-th iteration and where the denominator normalizes this by the sum of all K_i eigenvalues at iteration i. If the fractional information in the last new eigenimage F_{i,K_i} is close to zero, then the rank of $\mathbf{R}_i$ is less than K_i. An estimate of the rank $\widehat{K_i}$ of $\mathbf{R}_i$ is given by

$$\widehat{K_i} = \min m \quad \text{such that} \quad \sum_{k=1}^{m} F_{i,k} \geq T_I, \tag{11}$$

where T_I is a threshold that determines the *information capacity* of the filter (algorithm). When $T_I = 1$, we are computing the true set of eigenimages of **R** and are retaining all information in the process in the final set of eigenimages. With $T_I < 1$, we are limiting the information capacity of the set of eigenimages, by purposefully discarding statistically insignificant information. T_I has to be chosen such that K_i in (11) remains small, while maintaining a high information capacity. Typical values for T_I are 0.95 in the case of segmentation and 0.99 in the case of novelty filtering (see section 4).

After we have made the new rank estimate $\widehat{K}_i$, we discard the $K_i - \widehat{K}_i$ excess eigenimages and set $K_i = \widehat{K}_i$ (step 10). This is different from the recursive SVD algorithm [13,14] which always removes an eigenimage. We allow for not removing any eigenimages at all or for removing several eigenimages, should this be necessary. We then form our ***novelty filter*** $\widehat{\mathbf{x}}_j$ (our present description of the process) as a weighted sum of the retained eigenimages.

Our main concern is to first check the novelty of the input image (step 8), before initiating the calculation of the eigenvectors and eigenimages and to allow for an increase in K_i in such cases (step 5). However, we do not recursively increase the number of eigenimages (K_i), unless the storage capacity of K_i eigenimages has been proven to be too low, as indicated by the fractional information in the last eigenimages. There are many variations in the use of the parameters in this algorithm (several are addressed and quantified in section 4). For example, we can monitor n_i and when it remains small and constant, we can feel comfortable that we have modeled the process and that the process is repeating (this could, for example, correspond to a repetition of prior aspect views of an aircraft). In this case, the *learning phase* of the process modeling is complete, no more vectors are added and no further eigenvector and eigenimage calculations are performed.

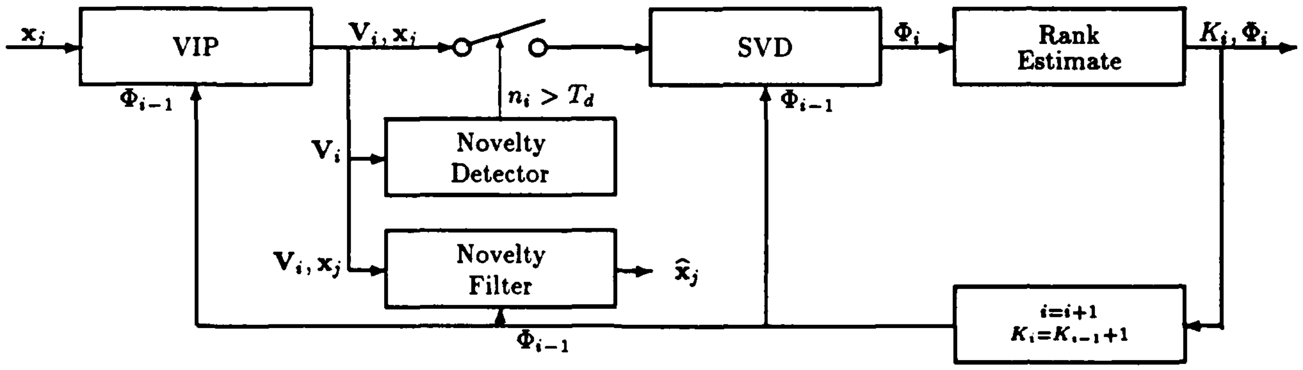


Figure 1: Adaptive Recursive SVD — Block Diagram

4 Applications and case studies

Figure 1 shows the block diagram of our novelty processor. The novelty filter $\widehat{\mathbf{x}}_j$ is the truncated series expansion of the current image in terms of the eigenimage basis set (the present model of the process):

$$\widehat{\mathbf{x}}_j = \sum_{j=1}^{K_i-1} \left(\frac{\mathbf{x}_j^T \Phi_k}{\lambda_k} \right) \Phi_k = \sum_{j=1}^{K_i-1} \frac{V(k, K_i)}{V(k,k)} \Phi_k. \tag{12}$$

The novelty detector with a threshold T_d (≈ 0.001 typically) declares each frame novel if it contains enough new data. We form the difference $|\mathbf{x}_j - \hat{\mathbf{x}}_j|$ of the input frame and the novelty filter to show the new data in the present frame. In this novel image frame we show pixels for which the difference $|\mathbf{x}_j - \hat{\mathbf{x}}_j|$ is more than T_f (typically 8 out of 256 gray levels). Note that the new image is partly present in the first and other eigenimages and not necessarily in just the last eigenimage (with the least significant eigenvalue). We use $F_{i,k}$ (the fractional amount of information present in each eigenimage) to decide whether to increase the number of eigenimages necessary to describe the process. The amount of fractional information in the retained eigenimages should be more than the information threshold T_I.

Figure 2 shows a selection from a sequence of 24 frames (128×128 pixels with 8 bits of gray scale) with two people moving in the background and one in the foreground. The sequence spans 24 seconds – one video frame per second. This sequence was used as input to our novelty processor with $T_d = 0.002$ and $T_I = 0.95$. In Figure 2 the left figure is the present input image and the center figure is the novel data (with present image pixels different from $\hat{\mathbf{x}}_j$ by more than 8 gray levels). Miscellaneous data are shown on the right. For Figure 2a, frame 1, $\mathbf{V}_1$ denotes its energy; $n_1 = 1$ (it is all new, since it is the first frame); $F_{1,1} = 1$ (since there is only one eigenimage) and it becomes eigenimage 1. Frame 2 (Figure 2b) has negligible new information ($n_2 = 0.0018$). The $v_2(1,1)$ entry is less than $v_2(2,2)$ since the energy in frame 2 is slightly larger than that in frame 1. The novel image shows new data (people, movement, their old and new positions), but it is not sufficiently new. In frame 4 (Figure 2c) the data is now sufficiently new ($n_4 = 0.0025$) with respect to the eigenimage (frame 1) since the people have moved more. The process model (eigenimage 1) is updated, but no second eigenimage is needed since the amount of information in it is small ($F_{4,2} = 0.0006$). Every few frames, the process model is updated, but only one eigenimage is sufficient to store all the information (since $T_I = 0.95$). This continues until the person enters the foreground in frame 8 (Figure 2d). Now and hereafter each frame is new, the process model (one eigenimage) is updated and only one eigenimage is sufficient to satisfy $T_I = 0.95$.

In frame 2 (Figure 2b) several background pixels appear in the novel image because of camera synchronization differences. The background behind the people and the people are both novel as seen in the novel data image. The present image is declared novel when enough pixels differ by enough to exceed $T_d = 0.002$. No background pixels appear in the novel image of frame 3 (not shown) since the camera was synchronized with frame 1 and therefore also with the current and only eigenimage (which is the first frame). When frame 4 enters it is found to be novel and is combined with the current eigenimage to form two new eigenimages (both contain information from frame 1 and frame 4). The dominant eigenimage contains more than $T_I = 95\%$ ($F_{4,1} = 0.9994$) of the information and therefore the other eigenimage is discarded. In the next several frames (e.g. frame 8, Figure 2d) the energy in the first eigenimage increases (due to the added data) and thus $v_8(1,1)$ increases. The VIP values $v_i(l,m)$ shown are all divided by the number of updates performed (i.e. the number of novel frames found thus far). The people moving in the background occlude the scene behind them. These people, as seen in the first frame (Figure 2a, left) are also in the eigenimage when frame 2 enters. The pixels where they were in frame 1 (background in frame 2) as well as those where they are now are declared novel (Figure 2b, center). These novel areas overlap in Figure 2 (center). In frame 4 (versus frame 1) the novel areas overlap much less and thus more pixels are novel; the new information exceeds T_d and triggers an update. The background behind the people has been seen once in both of these areas and thus the new first eigenimage contains both background and people information in these areas. In subsequent frames the people in the background are not new enough to trigger an update (since only the people are new and not the background behind them). It is not until frame 8 when the large person enters the foreground that another update is done. While this person walks across the foreground each frame is considered novel, but only one eigenimage was sufficient.

The data obtained (Figure 2, center) shows the new information per frame and its location. When all frames are seen we find that the system (the eigenimages) remembers moving parts with high contrast (e.g. parts of the person in the foreground) for several novel frame updates, but it remembers lower contrast moving parts (e.g. the background people) for fewer frames. This occurs because the eigenimage is a linear combination of the previous eigenimage and the input image when it is novel. These moving parts are averaged out after a few updates since they are replaced by background in the new update frames.

In Figure 3 we show the results when we increase $T_I = 0.9995$ and decrease $T_d = 0.001$, using the same input sequence as in Figure 2. This larger T_I forces the novelty filter to retain more information – more than one eigenimage. Figure 3 shows selected image frames and the output of the algorithm for this sequence. The left image is the input frame. The center image is the novelty date and the right image (if present) is the first eigenimage; it is shown when an update occurs. Frame 2 (Figure 3a) shows the moving persons (center) and as before the first eigenimage is the first frame. In subsequent frames the first eigenimage is updated, but only one eigenimage is retained. Figure 3b (frame 5) shows that the dominant eigenimage now contains only a little of the background people. The foreground person enters in frame 8. In frame 10 the first eigenimage (Figure 3c, right) contains only background. In all subsequent frames (frame 8 etc.) a second image is added; this eigenimage will contain the person and the first eigenimage will contain the background.

We now detail why the first eigenimage in the novelty filter contains the background information (and thus its novelty data is the moving object information). As subsequent novel frames are added to the process model, only a fraction of the new full image is added and thus the background is reinforced in each new image and the person is not reinforced and becomes lost (in the first eigenimage). Thus, as the Figure 3 data shows, the first eigenimage gradually forgets the moving objects (Figure 3, right) since when new frames are added the background is reinforced much more than are the moving persons. We could (if desired) modify the algorithm (using the centered novelty image) to add new data and not background to the process model. In the sequence in Figure 3 up to three eigenimages are kept. The lower T_d causes more updating than in Figure 2. This results in a cleaner first eigenimage (since more updates, dominated by additions, reinforces the background earlier in the sequence) and better novelty data (a cleaner image of the moving person, not smeared by the background) results.

Our third test used five aircraft in different roll orientations and no background. The algorithm keeps 2–3 eigenimages per class when a novelty detector n_i was formed for each class (with $T_I = 0.97$ and $T_d = 0.06$). These five novelty filters were then used on three new input images of each aircraft (at orientations not present in the training set). The outputs of the five novelty detectors (one per class) are shown (Table 2, columns 1–5) and the class estimate (smallest n_i) is shown in column 6. As seen 100% recognition was obtained.

Acknowledgments

The support of this research by a grant from the Strategic Defense Initiative Office of Innovative Science and Technology, monitored by the Office of Naval Research, is gratefully acknowledged, as is the support of NASA Ames Research Center. We also acknowledge Etienne Barnard for fruitful technical discussions.

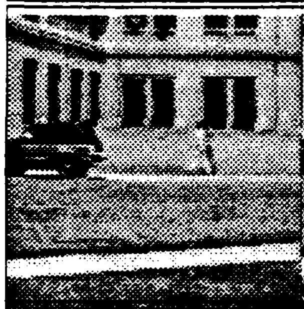
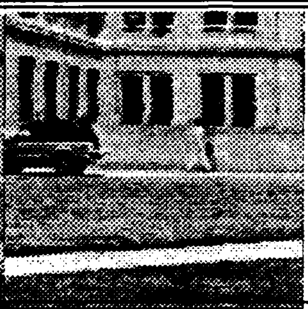
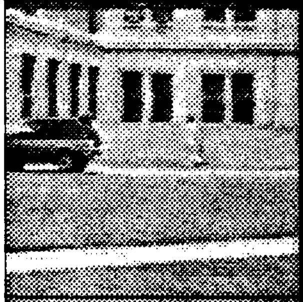
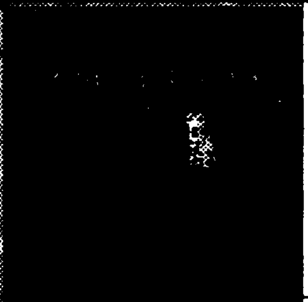
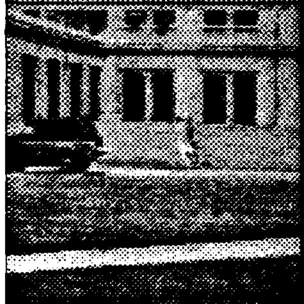
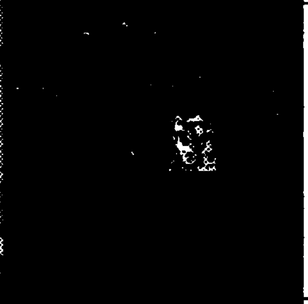
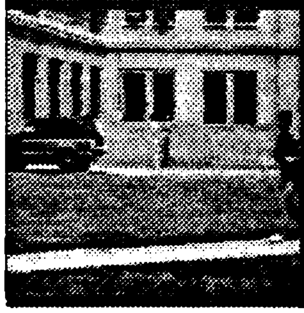
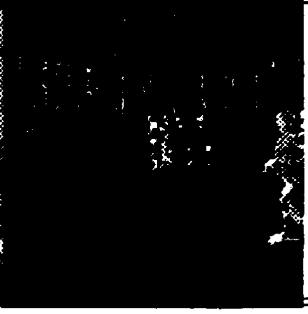
INPUT	NOVEL DATA	REMARKS
		(a) Frame 1 V_i = [73544016] =E in image n_i = 1. all novel first image. expected $F_{i,k}$ = 1 (FRACTION NEW) ALL
		(b) Frame 2 $V_i = \begin{bmatrix} 36772008 & 36772052 \\ 36772052 & 36838816 \end{bmatrix}$ n_i = 0.001811. not novel no update
		(c) Frame 4 n_i = 0.002498. novel update eigenimages $F_{i,k}$ = [0.999375 0.000625] keep one e-image only Frame 4 info in new one e-image
		(e) Frame 8 n_i = 0.005841. novel update e-images $F_{i,k}$ = [0.998910 0.001090] keep one e-image only FRAMES 1.3.8 IN ONE E-IMAGE

Figure 2: Sequence 1. low T_I = 95% (one eigenimage)

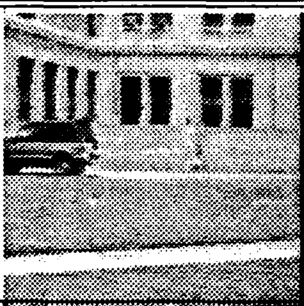
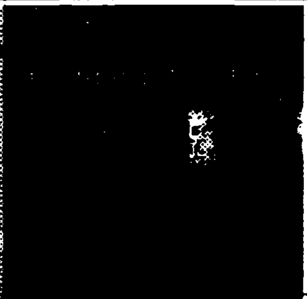
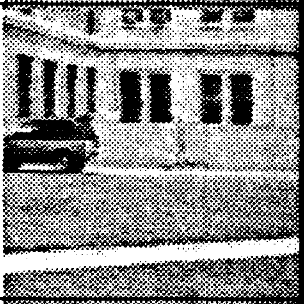
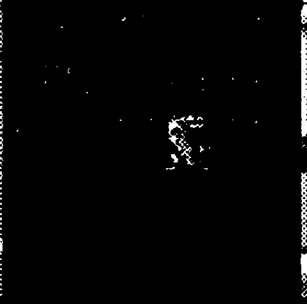
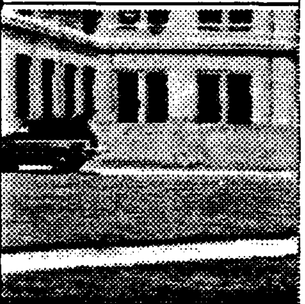
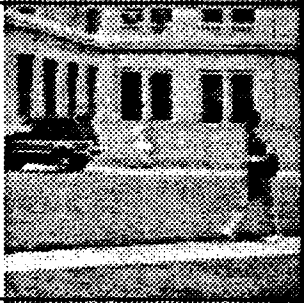
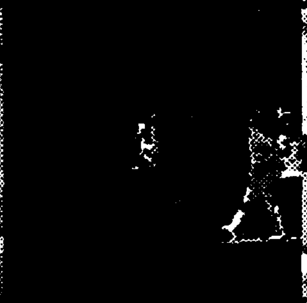
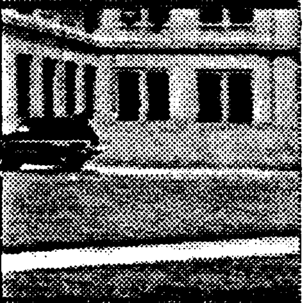
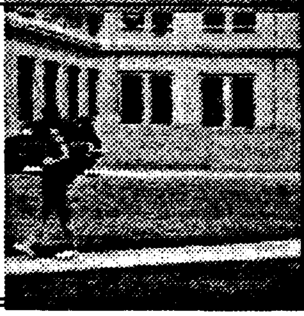
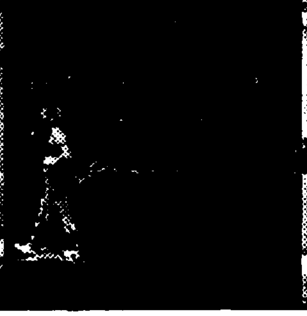
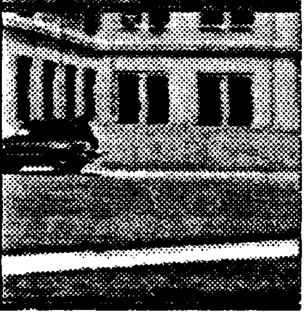
Frame	Input Frame	Novel Data	First E-Image
(a) 2			
(b) 5			
(c) 10			
(d) 20			

Figure 3: Sequence 2. Higher $T_I = 0.9995$

n_i					Classification	Description
Mig	DC-10	Mirage	Phantom	Boeing-737		
0.7579	0.5555	0.7667	0.6644	0.0609	Boeing-737	Boeing-737 5°
0.7169	0.5044	0.7136	0.6060	0.1721	Boeing-737	Boeing-737 45°
0.6393	0.4457	0.6852	0.6003	0.3981	Boeing-737	Boeing-737 85°
0.6772	0.0606	0.3645	0.5108	0.5122	DC-10	DC-10 5°
0.5963	0.1530	0.3524	0.4036	0.5087	DC-10	DC-10 45°
0.5542	0.3015	0.5787	0.5303	0.4350	DC-10	DC-10 85°
0.0694	0.5239	0.3765	0.2945	0.6935	Mig	Mig 5°
0.1397	0.5319	0.3870	0.3180	0.6768	Mig	Mig 45°
0.1007	0.5990	0.3501	0.4777	0.6921	Mig	Mig 85°
0.5957	0.3074	0.0441	0.4167	0.7267	Mirage	Mirage 5°
0.4594	0.4066	0.0936	0.2789	0.7124	Mirage	Mirage 45°
0.2981	0.5884	0.1925	0.4499	0.6893	Mirage	Mirage 85°
0.4821	0.4190	0.2982	0.0533	0.6097	Phantom	Phantom 5°
0.3135	0.4267	0.3154	0.1657	0.6124	Phantom	Phantom 45°
0.2451	0.4591	0.3275	0.2337	0.5567	Phantom	Phantom 85°

Table 2: Aircraft Classification Results

References

[1] D. Ballard and C. Brown, *Computer Vision.* Englewood Cliff NJ: Prentice-Hall Inc, 1982.

[2] R. B. Macdonald, "A summary of the history of the development of automated remote sensing for agricultural applications," *IEEE Trans. Geoscience and Remote Sensing*, vol. GE-22, pp. 463–482, November 1984.

[3] A. F. H. Goetz, G. Vane, J. E. Solomon, and B. N. Rock, "Image spectrometry for earth remote sensing," *Science*, vol. 228, pp. 1147–1153, June 1985.

[4] E. L. Hall, *Computer Image Processing and Recognition.* New York: Academic Press, 1979.

[5] A. Habibi and P. A. Wintz, "Image coding by linear transformation and block quantization techniques," *IEEE Trans. Comm. Technol.*, vol. COM-19, pp. 948–956, 1971.

[6] W. K. Pratt, *Digital Image Processing.* New York: John Wiley and Sons, 1978.

[7] K. Fukunaga and W. L. G. Koontz, "Representation of random processes using the finite Karhunen-Loéve expansion," *Inform. Contr*, vol. 16, pp. 85–101, 1970.

[8] E. Oja, *Subspace Methods of Pattern Recognition.* New York: John Wiley and Sons Inc, 1983.

[9] S. Watanabe, "Karhunen Loève expansion and factor analysis – theoretical remarks and applications," in J. Sklansky, ed., (*PATTERN RECOGNITION: Introduction and Foundations*), Benchmark Papers in Electrical Engineering and Computer Science, (Stroudsburg, PA), Dowden, Hutchinson & Ross, Inc., 1973, pp. 146–171.

[10] S. Watanabi, P. F. Lambert, C. A. Kulikowski, J. L. Buxton, and R. Walker, "Evaluation and selection of variables in pattern recognition," in J. T. Tou, ed., (*Computer and Information Sciences - 2*), (New York), Academic Press, 1967, pp. 91–122.

[11] F. L. Bauer, "Das verfahren der treppeniteration und verwandte verfahren zur losung algebraischer eigenwertprobleme," *Z. Angew. Math. Phys.*, vol. 8, pp. 214–235, 1957.

[12] A. Albert, *Regression and the Moore-Penrose pseudoinverse.* New York: Academic Press, 1972.

[13] B. V. K. V. Kumar, D. Casasent, and H. Murakami, "Principal-component imagery for statistical pattern recognition correlators," *Optical Engineering*, vol. 21, pp. 43–47, January/February 1982.

[14] H. Marukami and B. V. Kumar, "Efficient calculation of primary images from a set of images," *IEEE Trans. Pattern Anal. Mach. Intel.*, vol. PAMI-4, pp. 511–515, September 1982.

[15] J. H. Wilkinson and C. Reinsch, *Linear Algebra*, Vol. II of Handbook for Automatic Computation. New York: Springer-Verlag, 1971.

[16] B. T. Smith et. al., *Matrix Eigensystem Routines — EISPACK Guide*, Vol. 6 of Lecture Notes in Computer Science. New York: Springer-Verlag, 1976.

[17] *IMSL Library Reference Manual.* 7500 Bellaire Boulevard, Houston TX 77036, 1984.

[18] W. H. Press et. al., *Numerical Recipes in C.* Cambridge: Cambridge Univ. Press, 1988.

END
6-89
DTIC