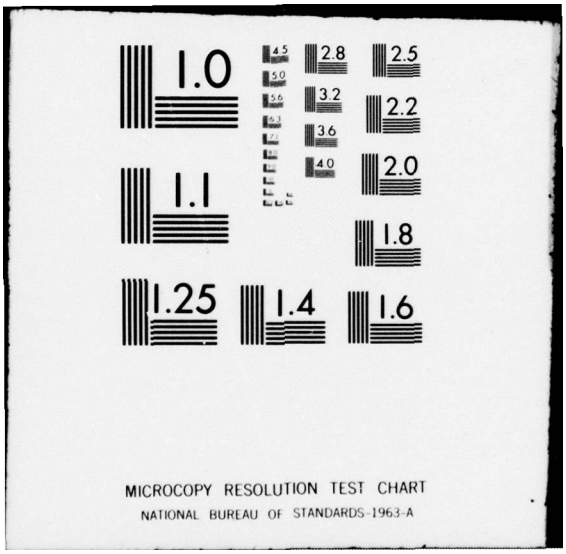
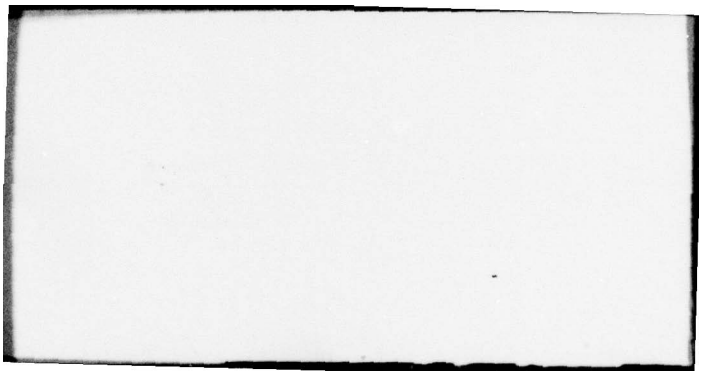
AD-A064 191 AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OHIO SCH--ETC F/G 17/5
AN EXTENDED KALMAN FILTER FOR USE IN A SHARED APERTURE MEDIUM R--ETC(U)
DEC 78 D E MERCIER

UNCLASSIFIED AFIT/GA/EE/78D-3 NL

1 OF 2
AD
A064191


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0
5.6
6.3
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A



AFIT/GA/EE/78D-3

(1)

LEVEL II

AD A064191

DDC FILE COPY

AN EXTENDED KALMAN FILTER
FOR USE IN A SHARED
APERTURE MEDIUM RANGE TRACKER

THESIS

AFIT/GA/EE/78D-3 Daniel E. Mercier
Captain USAF

Approved for public release; distribution unlimited

AFIT/GA/EE/78D-3

AN EXTENDED KALMAN FILTER FOR USE IN A SHARED APERTURE MEDIUM RANGE TRACKER

Master's thesis,

Presented to the Faculty of the School of Engineering

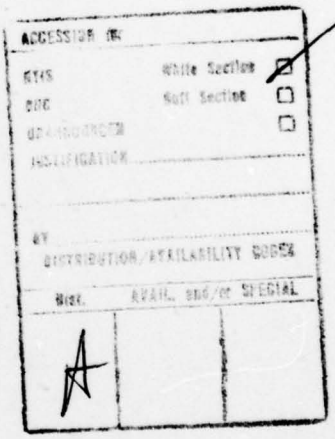
of the Air Force Institute of Technology

Air Training Command

in Partial Fulfillment of the

Requirements for the Degree of

Master of Science



by

Daniel E. Mercier, B. S.

Captain USAF

Graduate Astronautical Engineering

December 1978

157 p.

Approved for public release; distribution unlimited

012225

Preface

Since the early 1960's, science has made great strides in the development of high powered laser systems and in the theory of optimal estimation and control. This report attempts to combine these areas of technology by presenting an extended Kalman filter that can be used, together with the outputs of a forward looking infrared sensor, in an open-loop tracking problem. This subject was chosen because of the great significance of the high energy laser as a potential weapon system and because of the need for research in this highly technical area. Hopefully, the topic presented here will benefit the personnel at the Air Force Weapons Laboratory and aid them in designing a better weapons system.

My thanks go out to all those people whose assistance and guidance was given to me and without whose help this would have been impossible. Mr. Peter Maybeck was exceptionally helpful as my advisor. His many comments were beneficial and his door was always open for advice. Additional gratitude must be expressed to Captains Stan Robinson, Gary Reid, and William Wiesel for their interest and support in completing this thesis. A sincere thanks also goes to Captains Stan Lewantowicz and Felix Morgan at the Air Force Weapons Laboratory for their helpful suggestions and for the time they spent digging up letters and reports pertenent to this study.

The final debt of thanks I owe is to my wife Diane for her incredible understanding, her enormous love, her gentle prodding, and for the many hours she spent alone during this period. We were in this together.

Daniel E. Mercier

Contents

List of Tables

List of Figures

List of Symbols

Ap	Area of one picture element (pixels^2)
C	Correlation coefficient
E(•)	Expected value
f	FLIR noise term
$\underline{F}$	System plant matrix
$\underline{G}$	System input matrix
$\underline{H}$	Linearization of the intensity measurements
Hz	Cycles per second
I_{max}	Maximum glint reflected from the target
$\underline{K}$	Matrix of Kalman filter gains
n	background noise term
$\underline{P}$	Kalman filter covariance matrix
$\underline{Q}$	Strength of disturbance processes matrix
R_F	Strength of the measurement noise
$\{R_i\}$	Discretized set of reference samples
S/N	Signal-to-noise ratio
S(x)	Continuous time radiation intensity function
$\{T_i\}$	Discretized set of target samples
v	Measurement noise
w	White noise driving a shaping filter
$\hat{\underline{x}}$	Vector of filter state estimates
$\hat{x}_A$	Estimate of horizontal motion due to atmospheric jitter
$\hat{x}_D$	Estimate of horizontal motion due to target
x_{peak}	Horizontal coordinate of the center of the Gaussian intensity function

$\hat{y}_A$	Estimate of vertical motion due to atmospheric jitter
$\hat{y}_D$	Estimate of vertical motion due to target
y_{peak}	Vertical coordinate of the center of the Gaussian intensity function
z	Measurement vector
$\underline{\Gamma}$	Vector of actual measurements
$\underline{\Phi}$	State transition matrix
σ_A	RMS value of atmospheric jitter (pixels)
σ_D	RMS value of target motion (pixels)
σ_g	Dispersion of the Gaussian intensity function (pixels)
τ	Correlation time of a shaping filter (sec^{-1})

Abstract

An extended Kalman filter algorithm, using outputs from a forward looking infrared (FLIR) sensor as measurements, is used to track a point source target in an open loop tracking problem. The filter estimates the translational position changes of the target in the FLIR field of view due to two effects: actual target motion, and apparent motion caused by atmospheric turbulence. Sixteen cases are examined to determine the performance of the filter as a function of signal-to-noise ratio, Gaussian beam size, the ratio of RMS target motion to RMS atmospheric jitter, target correlation times, and mismatches between the true shape and the shape assumed by the filter. The performance of the extended Kalman filter is compared to the performance of an existing correlation tracker under identical initial conditions. A one sigma tracking error of .2 and .8 picture elements is obtained with signal-to-noise ratios of 20 and 1 respectively. No degradation in performance is observed when the beam size is decreased or when the target correlation time is increased over a limited range, when filter parameters are adjusted to reflect this knowledge. Sensitivity analysis shows that the filter is robust to minor changes in target intensity size.

AN EXTENDED KALMAN FILTER FOR USE IN A SHARED APERTURE MEDIUM RANGE TRACKER

I. Introduction

Background

Since the early 1960's, there has been an impressive growth in laser technology throughout the world. Now, only fifteen years later, the laser beam has become a prime candidate as a potential weapon system and is currently being tested against airborne targets. As with all new systems, however, many problems must be resolved before the high energy laser can be used as an effective weapon.

Problem

One of these problems is the precision pointing of the laser at a given target. The Air Force Weapons Laboratory is currently examining several methods of tracking a target for the purpose of depositing high power laser energy on the target in the presence of several disturbances. These disturbances include any effect that can cause relative motion between the beam and the target, such as target motion, mirror vibration, and atmospheric jitter.

One tracking method currently being used employs a forward looking infrared sensor (FLIR) together with a correlation algorithm to provide relative target position information to the laser pointing system. The algorithm provides this information by first storing a complete set of

target intensity data from the outputs of the FLIR, and then correlating that data with new information at a later time. Through the use of cross correlation techniques, the algorithm can estimate the relative position offset from one set of data to the next. Using this relative position information, commands can then be sent to the gimbals controlling the line of sight of the infrared system to keep it centered on the target. This centering of the FLIR on the target also physically points the laser towards the target since the high energy laser is servoed to the FLIR system. This type of tracker needs no prior information about the type of target to perform the tracking function, and therefore, it is well suited to many general applications.

In many practical tracking problems, however, the type of target being tracked will be known, if even in a very general sense. This implies that certain target parameters such as shape, size, and acceleration characteristics, will either be known or could be estimated. Also, the statistical effects of atmospheric distortion on radiated wavefronts are known and could supply information to a tracker that would aid in separating the true target relative motion from the apparent motion due to the atmospheric disturbance. This separation is important since the wavefront of the high energy laser will not undergo the same distortion in the atmosphere as the infrared wavefronts emanating from the target. This readily available information, not used by the correlation tracker, could be used, together with the outputs of the FLIR system, in an algorithm which would optimally estimate relative target position information. One way to do this would be to use an extended Kalman filter in the tracking loop in place of the present correlation algorithm.

Research Goals

This report presents the use of an extended Kalman filter in an open loop tracking problem for the eventual use in the precision pointing of a laser beam at a target. The study was performed by simulating the filter algorithm and the FLIR outputs on a CDC-6600 computer. A variety of simulations were performed, examining scenarios with varying signal-to-noise ratios (S/N), target intensity pattern sizes, and ratios of target dynamics to atmospheric jitter.

As a basis for comparison, a correlation tracking algorithm, devised by the Aeronutronic Ford Corporation, using single-degree-of-freedom correlation techniques in two independent directions, was also simulated. Both algorithms were run with identical inputs and with identical information from the FLIR. The extended Kalman filtering algorithm, however, made use of some apriori knowledge of the size, shape, and typical dynamics of the target along with some knowledge of the jitter characteristics of the atmosphere. Outputs for the two algorithms were then compared to evaluate the performance of each under varying conditions. Several simulations were also made to investigate the sensitivity of the extended Kalman filter algorithm to incorrectly made assumptions. In order to limit the problem to one which could be accomplished in a specified allotment of time, only the open loop tracking problem was addressed. In this way, controller design could be left as a follow-on study.

Assumptions

Target. Most applications for which the laser system is being considered require the acquisition and tracking of targets at long ranges.

Because of these distances, even very large targets appear as point sources of infrared radiation and can be accurately modelled as such. This was the only type of target considered since a variation in target shape would require a major rewrite of those subroutines simulating the FLIR outputs.

FLIR Sensor. The outputs of a typical forward looking infrared sensor are continuous time averages of an array of infrared detectors as they are mechanically scanned through a limited field of view (FOV). As the detectors are scanned in both the horizontal and vertical directions, they emit electrical current proportional to the number of photons entering the face of the detectors. In order to get an accurate representation of the average number of photons present at a specific point in space, the currents from each detector as it passes that point are combined, electronically averaged, and converted to digital signals for processing. This electronic averaging is performed in real time by processing the output of each detector through a delay line and into a summing junction. This summation is then divided by the number of detectors in the array to obtain the final averaged output. This averaging technique, as shown in Figure 1, compensates for the time difference that each detector is scanned across a specific point in space. The output of the averaging process can then either be stored or displayed on a cathode ray tube (CRT) in real time. Each output will correspond to one picture element on the CRT, usually termed pixel. The horizontal and vertical scanning of the detectors through the FLIR field of view will result in an array of pixels called a frame of data. Normal frame rates are on the order of 30 Hz. For the purposes of this study, one frame of data will be considered as one measurement array. The scan

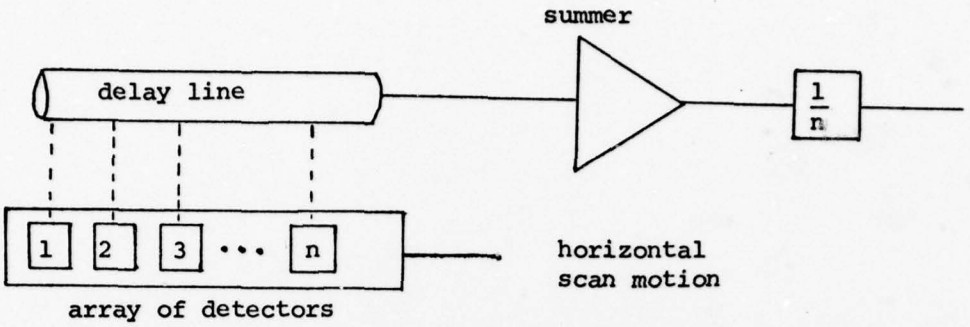
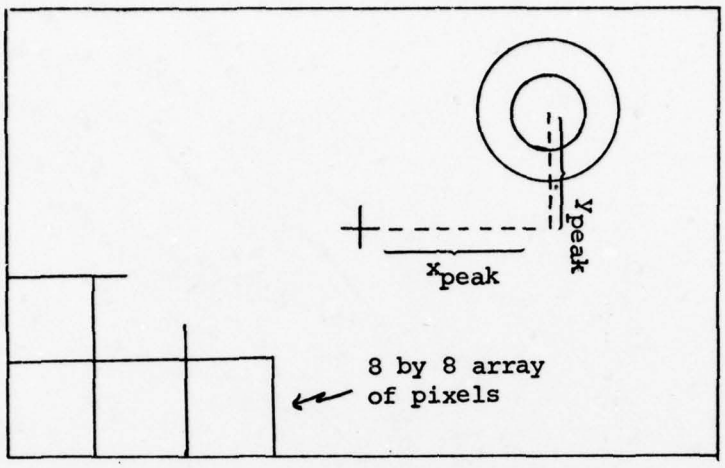


Figure 1. Electronic Averaging of Detector Outputs

time, therefore, predetermines the upper bound on the measurement rate. The measurement array can be as large as the FLIR field of view, but because of the fast measurement rates, the target will generally not move more than a few pixels between measurements. This permits looking at a small pixel array for tracking purposes. In this study, an 8 by 8 array of picture elements was used as the tracking window, resulting in a tolerable amount of computer storage area.

Glint. As mentioned above, the target is assumed to be a point source with a time invariant radiation intensity pattern, normally termed glint. Due to the physics of wave propagation, this glint has a Gaussian distribution. This distribution is the function of two Euler angles which describe the orientation of the boresight of the FLIR with respect to the path of maximum glint emanating from the target. The Euler angles are the actual pointing errors in the horizontal and vertical directions. These pointing errors translate, in general, to two linear pointing errors in the FLIR field of view as shown in Figure 2.

The glint reflected from the target can now be modelled as a bivariate Gaussian function with a two-dimensional pointing error argument. The intensity anywhere in the measurement plane can be described



x_{peak} = apparent horizontal pointing error

y_{peak} = apparent vertical pointing error

Figure 2. Linear Pointing Errors in the FLIR Field of View

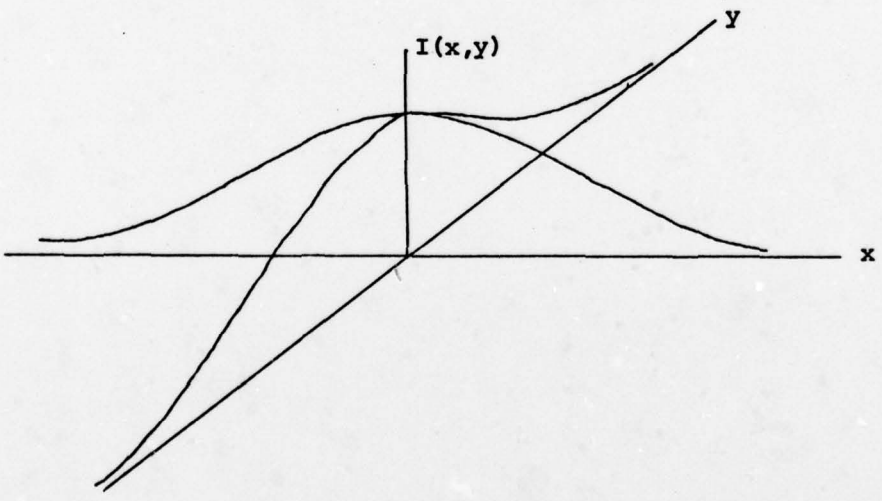


Figure 3. Glint Model for a Point Source Target

as

$$I(x,y) = I_{max}\exp\{-1/2\sigma_g^2 \left[(x-x_{peak})^2 + (y-y_{peak})^2\right]\} + b(x,y) \qquad (1)$$

where

I_{max} = the maximum target glint

σ_g = the dispersion of the Gaussian function

(x,y) = the coordinates of any point in the measurement array

(x_{peak}, y_{peak}) = the coordinates of the center of the Gaussian intensity function

$b(x,y)$ = background noise as a function of position in the measurement plane

Figure 3 shows this two-dimensional glint model without the background noise term. Points of constant intensity for this ideal function appear as circles in the FLIR field of view as shown in Figure 2.

Algorithm. Because this is an open loop tracking problem, the real task becomes creating an algorithm which will accurately estimate the position of the point of maximum intensity with respect to the center of the 8 by 8 array of pixels. The intensity function is, of course, moving around in the field of view due to the dynamics of the target and also due to the atmospheric jitter which is introduced as the radiated waves propagate through the atmosphere. The center of the intensity function on the FLIR image plane is, therefore, described by:

$$\begin{bmatrix} x_{peak}(t) \\ y_{peak}(t) \end{bmatrix} = \begin{bmatrix} x_D(t) + x_A(t) \\ y_D(t) + y_A(t) \end{bmatrix} \qquad (2)$$

where x_D and y_D are due to target dynamics and x_A and y_A are due to atmospheric jitter. An ideal tracking algorithm would accurately estimate x_D and y_D in order to eliminate the effects of the jitter.

Vibrations of the FLIR system could also cause relative pointing errors, but in this study, the FLIR system was assumed to be on the surface of the earth or some other stable platform so that the vibration effects were neglected.

Overview

The following three chapters will describe, in detail, the mathematical models used in the computer simulation. Chapters II, III, and IV will present the truth model, filter model, and correlation algorithm respectively. Chapter V will describe the computer simulation and the error analysis techniques. Chapter VI will show the results of all the test cases. Chapter VII will present the conclusions and recommendations.

II. Truth Model

Introduction

The problem, as stated in Chapter I, concerns the tracking of a moving target from a stabilized platform, using the outputs of a FLIR sensor as measurements of target radiation. The imaging tracking control loop for the high energy laser has a bandwidth on the order of 2 Hz. This control loop will, therefore, compensate for tracking error only in this range of interest. Higher order aberrations are controlled separately and will not be considered here. The two processes causing movement of the intensity function in the low frequency range are target dynamics and atmospheric jitter, caused by the constantly changing atmospheric conditions.

Target Dynamics

The purpose of any tracking algorithm is to predict, as accurately as possible, the movement of the target with respect to the center of the field of view of the sensor. Ideally, when the control loop is closed, the target should remain centered. Imperfections and noise terms, however, will combine to produce errors to this perfect tracker. In many cases, these errors can be well-described as outputs of first order shaping filters driven by zero mean, white, Gaussian noise. This type of model takes into account the time-correlated properties of the error and has been used successfully in the past to describe this type of error. Although there may be better models to describe the errors of specific targets, this model is generally applicable and simple to implement while still yielding some of the basic characteristics of trajectories. The outputs of the first order filters can be expressed

mathematically as

$$\dot{x}_D(t) = -1/\tau_T x_D(t) + x_1(t) \tag{3}$$

$$\dot{y}_D(t) = -1/\tau_T y_D(t) + w_2(t) \tag{4}$$

where

$$E\{w_1(t)\} = E\{w_2(t)\} = 0 \tag{5}$$

$$E\{w_1(t)w_1(s)\} = E\{w_2(t)w_2(s)\} = 2\sigma_d^2/\tau_T \delta(t-s) \tag{6}$$

$$E\{W_1(t)W_2(s)\} = 0$$

and

τ_T = correlation time

$w_1(t), w_2(t)$ = independent, white, Gaussian noise processes

σ_d^2 = the desired variance on the outputs x_D and y_D

Atmospheric Jitter Model

The process used to model the translational position changes of the intensity distribution due to disturbances in the atmosphere, was based on a study by the Analytic Sciences Corporation (TASC) (Ref 1:29, 30). As part of this study, TASC generated a power spectral density representation for the effects of atmospheric turbulence using a program supplied by the Air Force Weapons Laboratory (Ref 2). This function was then approximated by a third order shaping filter driven by white, Gaussian noise to arrive at a simple, yet adequate, model.

Figure 4 presents both the power spectral density and the approximation as presented in the TASC report. Using this same approximation, the target intensity jitter in each independent direction, is modeled as the output of the filter shown in Figure 5. The white noise process driving the filter, was given the following characteristics:

$$E\{w(t)\} = 0 \tag{7}$$

$$E\{w(t)w(s)\} = 1\ \delta(t-s) \tag{8}$$

The value of gain, K, can now be adjusted to obtain a desired root mean squared (RMS) jitter characteristic on the output. This allows studying the effects of differing amounts of jitter corresponding to a wide range of atmospheric conditions. A detailed description of how the actual value of gain, K, is determined is shown in Appendix C.

State Space Model

Transferring these models into state space notation yields a time-invariant vector differential equation of the form

$$\underline{x}_T(t) = \underline{F}_T\ \underline{x}_T(t) + \underline{G}_T\ \underline{w}_T(t) \tag{9}$$

where

$\underline{F}_T$ = the truth model plant matrix

$\underline{x}_T(t)$ = the truth model state vector

$\underline{G}_T$ = the truth model input matrix

$\underline{w}_T(t)$ = a vector of white Gaussian noise inputs

and

$$E\{\underline{w}_T(t)\} = 0 \tag{10}$$

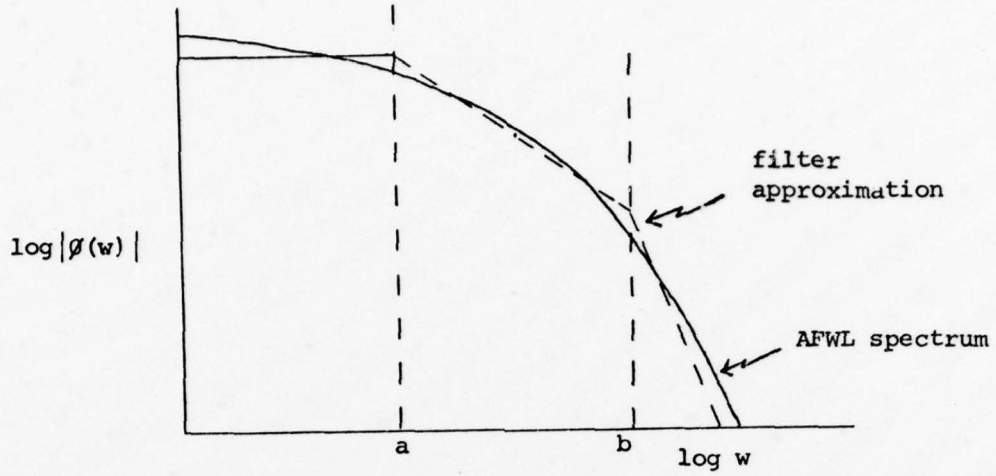
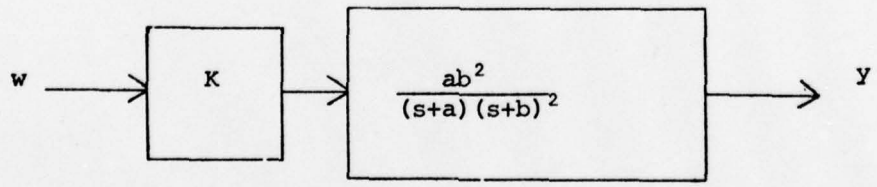


Figure 4. Atmospheric Turbulence Spectral Representation



a = 14.14 rad/sec b = 659.5 rad/sec

Figure 5. Third Order Shaping Filter

$$E\left[\underline{w}_T(t)\underline{w}_T^T(s)\right] = \underline{Q}_T\delta(t-s) \tag{11}$$

$$\underline{Q}_T = \begin{bmatrix} \frac{2\sigma_d^2}{\tau_T} & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{2\sigma_d^2}{\tau_T} & \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{12}$$

At first attempt, a standard phase variable form of the equation was used. This form, however, was found to be numerically imprecise in the calculation of certain desired quantities and the Jordon canonical form was used instead. Detailed information concerning the numerical problems encountered can be found in Appendix A.

Using the Jordan canonical form, as developed in Appendix B for the atmospheric turbulence model, equation (9) becomes

$$\underline{x}_T(t) = \begin{bmatrix} \frac{-1}{\tau_T} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -a & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -b & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -b & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{-1}{\tau_T} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -a & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -b & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -b \end{bmatrix} \underline{x}_T(t) + \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & G_1 & 0 & 0 \\ 0 & G_2 & 0 & 0 \\ 0 & G_3 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & G_1 \\ 0 & 0 & 0 & G_2 \\ 0 & 0 & 0 & G_3 \end{bmatrix} \underline{w}_T(t) \tag{13}$$

where

$$G_1 = Kab^2/(a-b)^2$$

$$G_2 = -G_1$$

$$G_3 = Kab^2/(a-b)$$

and

$$x_D = x(1)$$

$$x_A = x(2) + x(3)$$

$$y_D = x(5)$$

$$y_A = x(6) + x(7)$$

The output matrix is then

$$\underline{y}_T(t) = \begin{bmatrix} x_{peak}(t) \\ y_{peak}(t) \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \end{bmatrix} \underline{x}_T(t) \qquad (14)$$

It is apparent in this assumed formulation, that the horizontal and vertical channels are independent with identical process characteristics.

Propagation Equations

The solution to equation (9) is

$$\underline{x}_T(t) = \underline{\Phi}_T(t,t_i)\underline{x}_T(t_i) + \int_{t_i}^{t} \underline{\Phi}_T(t,\tau)\underline{G}_T(\tau)\underline{w}_T(\tau)\, d\tau \qquad (15)$$

where $\underline{\Phi}_T(t,t_i)$ is the state transition matrix and satisfies the matrix differential equation

$$\dot{\underline{\Phi}}(t,t_i) = \underline{F}_T\underline{\Phi}_T(t,t_i) \qquad (16)$$

and where $\underline{\Phi}(t_i,t_i) = \underline{I}$ (the identity matrix).

Since the truth model plant matrix is a constant, the state transition matrix is a function of $(t-t_i)$, and therefore, is also constant for a fixed sampling time. Because of the independent channels, the state

transition matrix is block diagonal with both 4 by 4 blocks equal to

$$\underline{\Phi}_{T_x}(t_{i+1},t_i) = \underline{\Phi}_{T_y}(t_{i+1},t_i) = \begin{bmatrix} e^{-\Delta t/\tau} & 0 & 0 & 0 \\ 0 & e^{-a\Delta t} & 0 & 0 \\ 0 & 0 & e^{-b\Delta t} & te^{-b\Delta t} \\ 0 & 0 & 0 & e^{-b\Delta t} \end{bmatrix} \quad (17)$$

where $\Delta t = (t_{i+1} - t_i)$.

The integral term shown in equation (15) is a stochastic integral. This requires the probabilistic characterization of the integral in order to describe the characteristics of the process $\underline{x}_T(t)$. Knowing that the integral has a zero mean Gaussian distribution, and letting

$$\underline{w}_{T_d}(t_i) = \int_{t_i}^{t_{i+1}} \underline{\Phi}_T(t_{i+1},\tau)\underline{G}_T(\tau)\ \underline{w}_T(\tau)\ d\tau \quad (18)$$

it can be determined that (Ref 4:174,181)

$$E\left[\underline{w}_{T_d}(t_i)\right] = 0 \quad (19)$$

$$E\left[\underline{w}_{T_d}(t_i)\underline{w}_{T_d}^T(t_i)\right] = \int_{t_i}^{t_{i+1}} \underline{\Phi}_T(t_{i+1},\tau)\underline{G}_T(\tau)\ \underline{Q}_T^T(\tau)\underline{G}_T^T(\tau)\underline{\Phi}_T^T(t_{i+1},\tau)d\tau \quad (20)$$

and

$$E\left[\underline{w}_{T_d}(t_i)\underline{w}_{T_d}^T(t_j)\right] = \underline{0} \qquad (t_i \neq t_j) \quad (21)$$

An equivalent discrete time model to portray the evolution of the $\underline{x}_T(t)$ process can now be given as

$$\underline{x}_T(t_{i+1}) = \underline{\Phi}_T\underline{x}_T(t_i) + \underline{W}_d(t_i) \quad (22)$$

where

$$E\left[\underline{w}_d(t_i)\right] = \underline{0} \tag{23}$$

$$E\left[\underline{w}_d(t_i)\underline{w}_d^T(t_j)\right] = \underline{Q}_{T_d}\,\delta_{i_j} \tag{24}$$

(note that $\underline{Q}_{T_d} = \underline{Q}_{T_d}(t_i)$ for all t_i because of stationary inputs and a time invariant system) and

$$\underline{Q}_{T_d} = \int_{t_i}^{t_{i+1}} \underline{\Phi}_T(t_{i+1},\tau)\underline{G}_T(\tau)\underline{Q}_T(\tau)\underline{G}_T^T(\tau)\underline{\Phi}_T^T(t_{i+1},\tau)\,d\tau \tag{25}$$

Another way of writing this equation is

$$\underline{x}_T(t_{i+1}) = \underline{\Phi}_T(t_{i+1},t_i)\underline{x}_T(t_i) + {}^c\sqrt{\underline{Q}}_{T_d}\ \underline{W}_n(t_i) \tag{26}$$

where ${}^c\sqrt{\underline{Q}}_{T_d}$ is a lower triangular matrix known as the Cholesky square root of $\underline{Q}_{T_d}$ and satisfies the equation

$${}^c\sqrt{\underline{Q}}_{T_d}\ {}^c\sqrt{\underline{Q}}_{T_d}^T = \underline{Q}_{T_D} \tag{27}$$

and

$$E\left\{\underline{W}_n(t_i)\underline{W}_n^T(t_j)\right\} = \underline{I}\,\delta_{ij} \tag{28}$$

Equation (26) has the same properties as equation (22) since

$$E\left\{\underline{W}_{T_d}(t_i)\underline{W}_{T_d}^T(t_j)\right\} = E\left\{{}^c\sqrt{\underline{Q}}_{T_d}\ \underline{W}_n(t_i)\underline{W}_n^T(t_j)\ {}^c\sqrt{\underline{Q}}_{T_d}^T\right\}$$

$$= {}^c\sqrt{\underline{Q}}_{T_d}\ E\left\{\underline{W}_n(t_i)\underline{W}_n^T(t_j)\right\}{}^c\sqrt{\underline{Q}}_{T_d} = {}^c\sqrt{\underline{Q}}_{T_d}\,\underline{I}\,{}^c\sqrt{\underline{Q}}_{T_d}^T = \underline{Q}_{T_d} \tag{29}$$

Equation (26) propagates the truth model states in time and accounts for the stochastic nature of the problem.

Measurements

The truth model measurements, by definition, are the best

representations possible for the outputs of the FLIR sensor. These outputs, as discussed earlier, are a measure of the average intensity of target radiation over the detector surfaces as they are swept across the image plane. Sweep rates are on the order to 30 Hz which means that measurements are available at discrete time points every 1/30 seconds.

The input radiation consists of apparent target radiation and FLIR generated noise term to account for noisy electronics, thermal radiation, and dark current noises internal to the detector array. Given all this, the output of the i-j-th pixel at any time t, can be described as

$$Z_{ij}(t) = \frac{1}{A_p} \iint_{\substack{\text{i-j-th} \\ \text{pixel}}} I_{max} \exp\{-\frac{1}{2\sigma_g^2}\left[(x-x_{peak}(t))^2 + (y-y_{peak}^{(t)})^2\right]\}dxdy + n_{ij}(t) + f_{ij}(t) \quad (30)$$

where

$Z_{ij}(t)$ = output of i-j-th pixel at time t

A_p = area of pixel under consideration

(x,y) = coordinates of any point in the pixel

$(X_{peak}(t), Y_{peak}(t))$ = coordinates of the center of the intensity distribution with respect to the center of the FLIR field of view at time t (see Eq. (2))

I_{max} = maximum target intensity

σ_g = dispersion of the Gaussian intensity function

$n_{ij}(t)$ = background noise term for the i-j-th pixel

$f_{ij}(t)$ = FLIR noise term for the i-j-th pixel

The two noise terms, $n_{ij}(t)$ and $f_{ij}(t)$, are, in general, independent wide band noises that can be accurately modeled, according to the Air

Force Weapons Laboratory, as zero mean white, Gaussian noise terms with the following statistics:

$$E\left[n_{kl}(t_i)\right] = E\left[f_{kl}(t_i)\right] \approx 0 \qquad \text{for all } i,k,l \qquad (31)$$

$$E\left[n_{kl}(t_i)n_{kl}(t_j)\right] = \sigma_B^2\delta_{ij} \qquad \text{for all } i,k,l \qquad (32)$$

$$E\left[f_{kl}(t_i)f_{kl}(t_j)\right] = \sigma_f^2\delta_{ij} \qquad \text{for all } i,j,k,l \qquad (33)$$

$$E\left[n_{kl}(t_i)n_{mn}(t_j)\right] = E\left[f_{kl}(t_i)f_{mn}(t_j)\right] = 0 \quad (kl \neq mn) \qquad (34)$$

i.e., n_{kl} and f_{kl} are assumed to be spatially uncorrelated as well. Since one complete scan is a measurement array, there are effectively 64 measurements available at each sample time.

III. Extended Kalman Filter Model

Introduction

The approach used in the design of the Kalman filter was to keep the models for target motion and atmospheric jitter as simple as possible while still accounting for a time correlation in the dynamics and a bandwidth characteristic of the jitter. This led to using first order Gauss-Markov processes along each independent axis direction, to describe both the target dynamics and the atmospheric turbulence. This choice results in a four state filter model which, with the proper choice of noise strengths and correlation times, will be descriptive of a wide range of targets.

State Space Model

The general form of a stationary, first order, Gauss-Markov process can be expressed as

$$\dot{x}(t) = -1/\tau\, x(t) + w(t) \tag{35}$$

with

$$E\left[w(t)\right] = 0 \tag{36}$$

$$E\left[w(t)w(s)\right] = \frac{2\sigma}{\tau}\,\delta(t-s) \tag{37}$$

where

τ = correlation time

$w(t)$ = white noise process

σ = RMS value of the output x

Expressing the four filter states (x_D, x_A, y_D, y_A) in this manner and putting them into the general form of

$$\underline{\dot{x}}_F(t) = \underline{F}_F\underline{x}_F(t) + \underline{w}_F(t) \tag{38}$$

where

$\underline{F}_F$ = the filter plant matrix

$\underline{x}_F(t)$ = the filter state vector

$\underline{w}_F(t)$ = the input white noise vector

gives the following equation:

$$\underline{\dot{x}}_F(t) = \begin{bmatrix} -1/\tau_D & 0 & 0 & 0 \\ 0 & -1/\tau_A & 0 & 0 \\ 0 & 0 & -1/\tau_D & 0 \\ 0 & 0 & 0 & -1/\tau_A \end{bmatrix} \underline{x}_F(t) + \underline{w}_F(t) \tag{39}$$

where

$$E\left[\underline{w}_F(t)\right] = \underline{0} \tag{40}$$

and

$$E\left[\underline{w}_F(t)\underline{w}_F^T(s)\right] = \underline{Q}_F\delta(t-s) = \begin{bmatrix} q_1 & 0 & 0 & 0 \\ 0 & q_2 & 0 & 0 \\ 0 & 0 & q_1 & 0 \\ 0 & 0 & 0 & q_2 \end{bmatrix} \delta(t-s) \tag{41}$$

State Propagation

The filter state equations shown above are linear, therefore, standard propagation equations can be used to propagate the filter states between measurement times. These equations are

$$\underline{\hat{x}}(t_{i+1}^-) = \underline{\Phi}_F(t_{i+1},t_i)\underline{\hat{x}}(t_i^+) \tag{42}$$

$$\underline{P}(t_{i+1}^{-}) = \underline{\Phi}_F(t_{i+1},t_i)\underline{P}(t_i^{+})\underline{\Phi}_F^{T}(t_{i+1},t_i) +$$

$$\int \underline{\Phi}_F(t_{i+1},\tau)\underline{Q}_F(\tau)\underline{\Phi}_F^{T}(t_{i+1},\tau)\, d\tau \qquad (43)$$

where the state transition matrix $\underline{\Phi}_F(t\ ,t_i)$ satisfies the following differential equation

$$\dot{\underline{\Phi}}_F(t\ ,t_i) = \underline{F}_F\underline{\Phi}_F(t,t_i) \qquad (44)$$

with the boundary condition

$$\underline{\Phi}_F(t_i,t_i) = \underline{I} \qquad \text{(the identity matrix)} \qquad (45)$$

The - and + signs refer to before and after a measurement is taken respectively.

In this case, the plant matrix $\underline{F}_F$ is a time-invariant matrix, therefore, the $\underline{\Phi}_F(t,t_i)$ matrix also becomes a constant for a given sampling period.

$$\underline{\Phi}_F(t,t_i) = e^{\underline{F}(t-t_i)} = \begin{bmatrix} e^{\frac{-(t-t_i)}{\tau_D}} & 0 & 0 & 0 \\ 0 & e^{\frac{-(t-t_i)}{\tau_A}} & 0 & 0 \\ 0 & 0 & e^{\frac{-(t-t_i)}{\tau_D}} & \\ 0 & 0 & 0 & e^{\frac{-(t-t_i)}{\tau_A}} \end{bmatrix} \qquad (46)$$

The $\underline{Q}_F$ matrix is also a constant matrix for all time, and in order to get the proper RMS values on the corresponding processes (Ref 4:193), it becomes

$$\underline{Q}_F = \begin{bmatrix} \frac{2\sigma_D^{\,2}}{\tau_D} & 0 & 0 & 0 \\ 0 & \frac{2\sigma_A^{\,2}}{\tau_A} & 0 & 0 \\ 0 & 0 & \frac{2\sigma_D^{\,2}}{\tau_D} & 0 \\ 0 & 0 & 0 & \frac{2\sigma_A^{\,2}}{\tau_A} \end{bmatrix} \qquad (47)$$

where

σ_D^2 = variance on the output of the target dynamics model

σ_A^2 = variance on the output of the atmoshperic jitter model

The integral term in equation (43) now becomes

$$\begin{bmatrix} \sigma_D^2(1-e^{-2\Delta t/\tau_D}) & 0 & 0 & 0 \\ 0 & \sigma_A^2(1-e^{-2\Delta t/\tau_A}) & 0 & 0 \\ 0 & 0 & \sigma_D^2(1-e^{-2\Delta t/\tau_D}) & 0 \\ 0 & 0 & 0 & \sigma_A^2(1-e^{-2\Delta t/\tau_A}) \end{bmatrix} \quad (48)$$

which is also a constant matrix for a given sampling time Δt . Substituting equations (46) and (48) into equations (42) and (43) gives the following

$$\underline{\hat{x}}(t_{i+1}^-) = \begin{bmatrix} e^{-\Delta t/\tau_D} & 0 & 0 & 0 \\ 0 & e^{-\Delta t/\tau_A} & 0 & 0 \\ 0 & 0 & e^{-\Delta t/\tau_D} & 0 \\ 0 & 0 & 0 & e^{-\Delta t/\tau_A} \end{bmatrix} \underline{\hat{x}}(t_i^+) \quad (49)$$

$$\underline{P}(t_{i+1}^-) = \begin{bmatrix} e^{-\Delta t/\tau_D} & & \underline{0} & \\ & e^{-\Delta t/\tau_A} & & \\ & & e^{-\Delta t/\tau_D} & \\ \underline{0} & & & e^{-\Delta t/\tau_A} \end{bmatrix} \underline{P}(t_i^+)$$

$$\begin{bmatrix} e^{-\Delta t/\tau_D} & & & 0 \\ & e^{-\Delta t/\tau_A} & & \\ & & e^{-\Delta t/\tau_D} & \\ 0 & & & e^{-\Delta t/\tau_A} \end{bmatrix}$$

$$+ \begin{bmatrix} \sigma_D^2(1-e^{-2\Delta t/\tau_D}) & & & 0 \\ & \sigma_A^2(1-e^{-2\Delta t/\tau_A}) & & \\ & & \sigma_D^2(1-e^{-2\Delta t/\tau_D}) & \\ 0 & & & \sigma_A^2(1-e^{-2\Delta t/\tau_A}) \end{bmatrix} \quad (50)$$

These are the equations which will propagate the filter states between measurements.

Measurements

The measurement history used by the filter to help predict the pointing directions, will be the outputs from the forward looking infrared sensor (FLIR). These outputs, as described in the first chapter, are average intensity values as shown in Figure 1. The output of the k-1-th pixel can be described mathematically as

$$z_{kl}(t_i) = \frac{1}{A_p} \iint I_{max} \exp\{\frac{-1}{2\sigma_{g_F}} \left[(x-x_{peak}^{(t_i)})^2 + (y-y_{peak}^{(t_i)})^2\right]\}dxdy + v_{kl}(t_i) \quad (51)$$

where

$z_{kl}(t_i)$ - the actual output of the k-l-th pixel at time t_i

A_p = the area of one pixel

I_{max} = the maximum intensity received from the target

$\sigma^2_{g_F}$ = the dispersion of the Gaussian intensity function as assumed by the filter

(x,y) = coordinates of any point in the pixel (variables of integration)

$(x_{peak}(t_i)$

$y_{peak}(t_i))$ = the center of the intensity pattern with respect to the center of the FLIR field of view at time t_i

$v_{kl}(t_i)$ = additive noise term

The additive noise term is assumed to be a spatially and temporarily white Gaussian noise with the following statistics:

$$E\left[v_{kl}(t_i)\right] = 0 \quad (52)$$

$$E\left[v_{kl}(t_i)v_{kl}(t_j)\right] = \begin{cases} R_f & (\text{for } t_i=t_j) \\ 0 & (\text{for } t_i \neq t_j) \end{cases} \quad (53)$$

$$E\left[v_{mn}(t_i)v_{kl}(t_j)\right] = 0 \quad (\text{for all } t_i, t_j) \quad (54)$$

$$(mn) \neq (kl)$$

It is used to account for any uncertainties in the system, such as background noise, FLIR noises, and integration approximations.

In general state space form, equation (51) is of the form

$$z_{kl}(t_i) = h_{kl}(\underline{x}(t_i), t_i) + v_{kl}(t_i) \quad (55)$$

where $h_{kl}(\underline{x}(t_i),t_i)$ is a nonlinear function of the states at time t_i. Because of the 8 by 8 tracking array, there will be 64 independent measurements available at each discrete measurement time. These measurements can be grouped into a 64-dimensional vector of the form

$$\underline{z}(t_i) = \underline{h}(\underline{x}(t_i),t_i) + \underline{v}(t_i) \tag{56}$$

where

$$E\left[\underline{v}(t_i)\right] = \underline{0} \tag{57}$$

$$E\left[\underline{v}(t_i)\underline{v}^T(t_j)\right] = \begin{cases} R_F\underline{I} & (\text{for } t_i=t_j) \\ \underline{0} & (\text{for } t_i\neq t_j) \end{cases} \tag{58}$$

Not all of these measurements need to be used in the estimation for the peak intensity. In fact, a smaller number of measurements could significantly reduce computer time while still maintaining good tracking accuracy. The number of measurements used would depend on two factors. First, it would depend on the dispersion of the Gaussian intensity function σ_{g_F}. For a small dispersion, on the order of one pixel, most of the useful information is contained in an area of about 5 pixels square. A smaller dispersion would require fewer pixels to get the same amount of information. The second factor is the tracking accuracy needed. For most laser applications, high accuracy tracking is desired. The best accuracy would be obtained when all the available information is used in the prediction process. The smaller the amount of information, the less accurate the prediction, even with very low signal-to-noise ratios. A third factor that may be considered, is the trade-off between estimation performance and computer time and storage.

Measurement Update Equations

In order to handle the nonlinearities in the measurement equation, a special form of the Kalman filter, called the extended Kalman filter, was used. This form of the Kalman filter accounts for the nonlinearities by linearizing the measurement equation about the most recent estimate of the states. This linearization process is reaccomplished prior to every update, thereby maintaining a current set of equations for updating the states.

The usual form of the measurement update equations for the extended Kalman filter (Ref 3:233) are

$$\underline{K}(t_i) = \underline{P}(t_i^-)\underline{H}^T(t_i)\left[\underline{H}(t_i)\underline{P}(t_i^-)\underline{H}^T(t_i) + \underline{R}(t_i)\right]^{-1} \tag{59}$$

$$\hat{\underline{x}}(t_i^+) = \hat{\underline{x}}(t_i^-) + \underline{K}(t_i)\{\underline{\Gamma}(t_i) - \underline{h}(\hat{\underline{x}}(t_i^-),t_i)\} \tag{60}$$

$$\underline{P}(t_i^+) = \underline{P}(t_i^-) - \underline{K}(t_i)\underline{H}(t_i)\underline{P}(t_i^-) \tag{61}$$

where

$$\underline{H}(t_i) = \left.\frac{\partial \underline{h}(\underline{x},t)}{\partial \underline{x}}\right|_{\underline{x}=\hat{\underline{x}}(t_i^-)} \tag{62}$$

and $\Gamma(t_i)$ is the 64 dimensional vector of actual measurements.

Notice, however, that calculating the Kalman filter gains, $\underline{k}(t_i)$, requires the inversion of a 64 by 64 matrix every update time. This would require a large amount of computer time, and would probably be unfeasible for most on-line applications. Another form of the update equations, known as the inverse covarience form (Ref 3:257), eliminates this problem. Using this form, the equations become

$$\underline{P}^{-1}(t_i^+) = \underline{P}^{-1}(t_i^-) + \underline{H}^T(t_i)\underline{R}^{-1}(t_i)\underline{H}(t_i) \tag{63}$$

$$\underline{P}(t_i^+) = \left[\underline{P}^{-1}(t_i^+)\right]^{-1} \tag{64}$$

$$\underline{K}(t_i) = \underline{P}(t_i^+)\underline{H}^T(t_i)\underline{R}^{-1}(t_i) \tag{65}$$

$$\hat{\underline{x}}(t_i^+) = \underline{x}(t_i^-) + \underline{K}(t_i)\{\underline{\Gamma}(t_i) - \underline{h}(\hat{x}(t_i^-), t_i)\} \tag{66}$$

where the $\underline{H}(t_i)$ matrix and the $\underline{\Gamma}(t_i)$ vector are defined the same as above. Using this new set of update equations, only two 4 by 4 inverses are needed, the $\underline{P}(t_i^-)$ matrix, and the $\underline{P}^{-1}(t_i^+)$ matrix. Normally, this form of the update equation would also involve the offline inversion of the $\underline{R}(t_i)$ matrix, but in this case, the pixel noises were assumed independent which means that

$$\underline{R}(t_i) = R_F\underline{I} \tag{67}$$

and therefore,

$$\underline{R}_F^{-1}(t_i) = (1/R_F)\underline{I} \tag{68}$$

Now this inversion can be computed once offline and the $\underline{R}^{-1}(t_i)$ matrix will remain constant for all time.

The partial derivative matrix shown in equation (62) can be easily computed knowing the fact that the integral is a linear operator. For this reason,

$$\left.\frac{\partial \underline{h}}{\partial \underline{x}}\right|_{\underline{x}=\hat{\underline{x}}(t_i^-)} = \left.\frac{\partial}{\partial \underline{x}}\left[\iint \frac{I_{max}}{A_p} \exp\{\frac{-1}{2\sigma_g^2}[(\zeta-x_{peak})^2+(\rho-y_{peak})^2]\}d\zeta d\rho\right]\right|_{\underline{x}=\hat{\underline{x}}(t_i^-)} \tag{69}$$

$$= \iint \frac{I_{max}}{A_p} \left.\frac{\partial}{\partial \underline{x}}\left[\exp\{\frac{-1}{2\sigma_g^2}\left[(\zeta-x_{peak})^2+(\rho-y_{peak})^2\right]\}\right]d\zeta d\rho\right|_{\underline{x}=\hat{\underline{x}}(t_i^-)} \tag{70}$$

where the partial derivative of the double integral term is equal to the double integral of the partial. Expanding this into matrix form and evaluating at $\underline{x} = \hat{\underline{x}}(t_i^-)$ results in a 64 by 4 matrix, whose ith row is

$$\begin{bmatrix} \iint \left\{(\zeta-\hat{x}_{peak})\cdot C_F\right\} d\zeta d\rho \\ \iint [(\zeta-\hat{x}_{peak})\cdot C_F] d\zeta d\rho \\ \iint [(\rho-\hat{y}_{peak})\cdot C_F] d\zeta d\rho \\ \iint [(\rho-\hat{y}_{peak})\cdot C_F] d\zeta d\rho \end{bmatrix}^T \tag{71}$$

for $\underline{x}=(x_D, x_A, y_D, y_A)^T$, $\hat{x}_{peak}=\hat{x}_D+\hat{x}_A$, and $\hat{y}_{peak}=\hat{y}_D+\hat{y}_A$ and where

$$C_F = \frac{I_{max}}{A_p \sigma_{g_F}^2} \exp\{\frac{-1}{2\sigma_{g_F}^2}\left[(\zeta-x_{peak})^2 + (\rho-y_{peak})^2\right]\} \tag{72}$$

Summary of Filter Equations

The following sets of equations represent the extended Kalman filter algorithm used for this study.

Propagation Equations.

$$\hat{\underline{x}}(t_{i+1}^-) = \underline{\Phi}_F(t_{i+1}) \hat{\underline{x}}(t_i^+) \tag{73}$$

$$\underline{P}(t_{i+1}^-) = \underline{\Phi}_F(t_{i+1},t_i)\underline{P}(t_i^+)\underline{\Phi}_F^T(t_{i+1},t_i) + \underline{Q}_D(t_i) \tag{74}$$

where

$$\underline{\Phi}_F(t_{i+1},t_i) = \begin{bmatrix} e^{-\Delta t/\tau_D} & 0 & 0 & 0 \\ 0 & e^{-\Delta t/\tau_A} & 0 & 0 \\ 0 & 0 & e^{-\Delta t/\tau_D} & 0 \\ 0 & 0 & 0 & e^{-\Delta t/\tau_A} \end{bmatrix} \tag{75}$$

and

$$\underline{Q}_D(t_i) = \begin{bmatrix} \sigma_D^2(1-e^{-2\Delta t/\tau_D}) & 0 & 0 & 0 \\ 0 & \sigma_A^2(1-e^{-2\Delta t/\tau_A}) & 0 & 0 \\ 0 & 0 & \sigma_D^2(1-e^{-2\Delta t/\tau_D}) & 0 \\ 0 & 0 & 0 & \sigma_A^2(1-e^{-2\Delta t/\tau_A}) \end{bmatrix} \tag{76}$$

Update Equations.

$$\underline{P}^{-1}(t_i^+) = \underline{P}^{-1}(t_i^-) + \underline{H}^T(t_i)\underline{R}^{-1}(t_i)\underline{H}(t_i) \tag{77}$$

$$\underline{P}(t_i^+) = [\underline{P}^{-1}(t_i^+)]^{-1} \tag{78}$$

$$\underline{K}(t_i) = \underline{P}(t_i^+)\underline{H}^T(t_i)\underline{R}^{-1}(t_i) \tag{79}$$

$$\hat{\underline{x}}(t_i^+) = \hat{\underline{x}}(t_i^-) + \underline{K}(t_i)\{\underline{\Gamma}(t_i) - \underline{h}(\hat{\underline{x}}(t_i^-), t_i)\} \tag{80}$$

where the $\underline{R}^{-1}(t_i)$ and $\underline{H}(t_i)$ matrices are defined as in equations (68) and (69) respectively.

Now that the truth model and extended Kalman filter model have been described, the next chapter will outline the correlation algorithm to be used for comparison purposes.

IV. Correlation Algorithm

Introduction

In order to judge the effectiveness of the extended Kalman filter algorithm in tracking a target, its performance was compared to that of a correlation algorithm presently being tested. Although two types of correlation algorithms are currently being examined for use in operational software (Refs 4 and 5), one algorithm was readily available in FORTRAN source code (Ref 6), and has been tested against Hawk missile test flight data. This algorithm, therefore, was selected as the benchmark for comparison.

This chapter focuses on the method used by the selected algorithm to calculate the position errors needed for tracking. A more detailed analysis can be found in Reference 4.

Purpose

The purpose of any tracking algorithm is to provide estimates of the position of the target with respect to a reference point, usually, the line of sight of the sensor tracking the target. Knowing this information, the sensor can then keep the target centered in its field of view by appropriate commands to a positioning controller. The correlation tracker estimates these relative position offsets by using correlation coefficients derived from the cross-correlation of a target data set $\{T_i\}$, and a reference data set $\{R_i\}$. This correlation coefficient is defined as

$$C = \frac{\sum_{i=1}^{n} T_i R_i}{\sum_{i=1}^{n} T_i^2 \sum_{i=1}^{n} R_i^2} \qquad (81)$$

where n is the number of sampled data points in each of the data sets. This correlation coefficient is a magnitude-independent measure of the degree of resemblance between the data sets. This means that the correlation coefficient is a function of the shape of the two data sets and is independent of their actual magnitudes.

The correlation technique does not exploit any knowledge of target dynamics, nor does it use any knowledge of disturbances that would cause apparent offsets of the target, so that these effects could be filtered out.

One-Dimensional Correlation

The following discussion will be a brief overview of correlation in one dimension as presented in an Aeronutronic Ford Corporation report (Ref 4). Let $\{R_i\}$ be a set of uniformly spaced samples of a continuous time radiation intensity function, received from a one-dimensional scan of a FLIR sensor across a point target. Let $\{T_i\}$ be a new sample set taken a short time later, where the intensity data is assumed to have the same shape, but may have been translated relative to the first data set because of target motion in the dimension of interest. The $\{R_i\}$ and $\{T_i\}$ data sets can be expressed as

$$\{R_i\} = \text{sampled}\,[S(x)] \tag{82}$$

$$\{T_i\} = \text{sampled}\,[S(x + \Delta x)] \tag{83}$$

where $S(x)$ is the continuous time intensity function and Δx is the intensity offset on the FLIR image plane. Expanding $S(x + \Delta x)$ in a taylor series gives

$$S(x + \Delta x) = S(x) + \frac{\partial S(x)}{\partial x} \Delta x + \text{H.O.T.} \tag{84}$$

Now by dropping the higher order terms, $\{T_i\}$ can be approximated as

$$\{T_i\} = \text{sampled}\left\{ S(x) + \frac{\partial S(x)}{\partial x} \Delta x \right\} \tag{85}$$

$$= \{R_i\} + \frac{\partial S_i}{\partial x} \Delta x \tag{86}$$

Letting

$$\{U_i\} = \{\frac{\partial S_i}{\partial x} \Delta x\} \tag{87}$$

and recognizing that equation (81) can be rearranged, assuming orthogonality of $\{R_i\}$ and $\{U_i\}$ (Ref 4:6), so that

$$C^2 \simeq 1 - \frac{\sum_{i=1}^{n} U_i^2}{\sum_{i=1}^{n} S_i^2} \tag{88}$$

Equation (87) and (88) can be combined to give

$$C^2 \simeq 1 - \frac{\sum_{i=1}^{n} (\frac{\partial S_i}{\partial x})^2 \Delta x^2}{\sum_{i=1}^{n} (S_i)^2} \tag{89}$$

This equation can be futher simplified by letting

$$P = \frac{\sum_{i=1}^{n} (\frac{\partial S_i}{\partial x})^2}{\sum_{i=1}^{n} (S_i)^2} \tag{90}$$

Now,

$$C^2 = 1 - P^2 \Delta x^2 \tag{91}$$

This equation represents one equation in three unknowns (P, Δx, sign of Δx). Two more equations can be established by shifting one of the sampled data sets left and right by one sample. This results in the following three equations:

$$C_2^2 = 1 - P^2\Delta x^2 \tag{92}$$

$$C_1^2 = 1 - P^2(1 - \Delta x)^2 \tag{93}$$

$$C_3^2 = 1 - P^2(1 + \Delta x)^2 \tag{94}$$

where C_2 represents the correlation coefficient produced when the two data sets are aligned to the closest sample. C_1 and C_3 represent the correlation coefficients obtained when the data set is shifted left and right by one sample respectively. Solving these three equations simultaneously gives

$$\Delta x = \frac{C_3^2 - C_1^2}{2(2C_2^2 - C_1^2 - C_3^2)} \tag{95}$$

This equation represents the offset, to a fraction of a sample, between the data sets, and is a direct measure of the positional change of the target over the sampling time.

The actual equation for Δx used for this study was

$$\Delta x = \frac{1/C_2^2 - 1/C_3^2}{2\left(\frac{1}{C_1^2} + \frac{1}{C_3^2} - \frac{2}{C_2^2}\right)} \tag{96}$$

This equation was developed at the Air Force Weapons Laboratory after the equations leading up to equation (95) were rederived, eliminating

the simplifying assumption

$$\frac{1}{1 + P^2\Delta x^2} = \frac{1 + P^2\Delta x^2 - P^2\Delta x^2}{1 + P^2\Delta x^2} \simeq 1 - P^2\Delta x^2 \qquad (97)$$

The derivation of equation (96) can be found in a memo from the Air Force Weapons Laboratory to the Ford Corporation (Ref 7).

In order to present a clearer representation of what is taking place, let Figure 6 represent a one-dimensional Gaussian intensity profile of a point source target moving horizontally to the right. At time t_1, the target is centered in the field of view of eight horizontal pixels, and at time t_2, the target has moved some distance r. Assuming ideal conditions (no change of shape and perfect measurement data), the intensity profile will also move to the right some distance d. Figures (7a) and (7b) represent the discretized value of the intensity profile at time t_1 and t_2. Each discretized value represents the averaged sampled output of the detectors as they are scanned across the target. These discrete values become the reference and target data sets $\{R_i\}$ and $\{T_i\}$ respectively. The center of the intensity function, to the nearest pixel in each data set, is estimated by the simple mean approximation.

$$J(t) = \text{INT}\left[\frac{\sum_{i=1}^{8} I_i(t) \cdot i}{\sum_{i=1}^{8} I_i(t)}\right] \qquad (98)$$

where

$J(t)$ = an integer from one to eight representing the pixel, from left to right, closest to the center of the intensity function

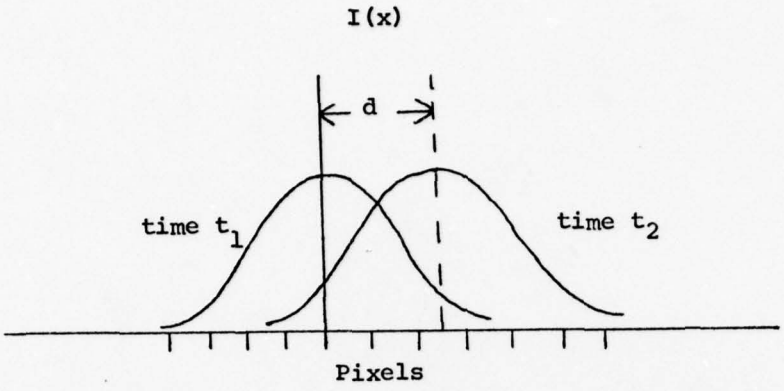
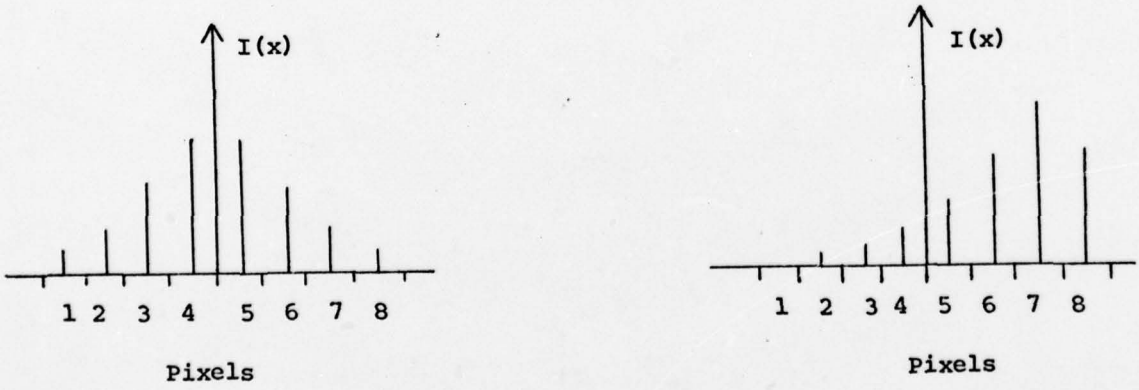


Figure 6. One-Dimensional Gaussian Intensity Profile



a. Discrete Representation at t_1

b. Discrete Representation at t_2

Figure 7. Discrete Representations of Intensity Profile

$I_i(t)$ = the intensity value in the ith pixel at time t ;

and

INT(•) = a function which rounds the argument to the nearest integer.

The offset of the intensity profile to the nearest pixel can now be estimated as the quantity $J(t_2) - J(t_1)$. The reference array $\{R_i\}$, which is the array of values at time t_1 , is now shifted by $J(t_2 - J(t_1)$ pixels. Those values that are shifted out of the array are dropped, and the values at the opposite end of the array are replaced by zeros. An example of this shifting process is shown in Figure 8. The three correlation coefficients C_1 , C_2 , and C_3 , can now be computed for the shifted reference array $\{R_i\}_s$ and the target array $\{T_i\}$. Equation (96) is then used to estimate the remaining offset to a fraction of one pixel. The total offset, d , is then estimated by the equation

$$d = J(t_2) - J(t_1) + \Delta x \tag{99}$$

Two-Dimensional Correlation

For a two-dimensional intensity function, the above process is performed for every row and then every column in the 8 x 8 arrays. This will result in eight, generally different, horizontal offsets, and eight vertical offsets. Because of this structure, the image position offsets in the horizontal and vertical directions are arrived at independently. The row correlation determines the horizontal offset, while the column correlation determines the vertical offset. If the maximum correlation coefficient C_2 for each row or column is less than some preset value, that row or column is considered as containing poor

a. Reference Array Before Shift

b. Reference Array After Shifting Two Pixels Right

Figure 8. Example of an Array Shift

information, and the computed offset is discarded. The remaining row and column offset values are then averaged separately. The average of the offset values for the rows is the overall estimate of target position offset in the horizontal direction, and the average of the offset values for the columns is the estimated target position offset in the vertical direction.

This simple averaging technique is a simplification of the method used by the Rod Corporation in their report (Ref 8). The following discussion explains the reason for this simplification. The Aeronutronic Ford Corporation algorithm used a least squares line fit of the values in each direction. One line was fit to the individual row offsets, and one to the column offsets. The total image offset in each direction was then computed in a way unspecified in the available sources. One method would be to determine the intersection of each line with the appropriate x or y axis. Another might be to determine the intersection of the two regression lines. The slope of these lines is then a measure of the target rotation over that sample period.

In this study, however, the Gaussian intensity function is not directly affected by target rotation. For this reason, the slope will contain no useful information and would, in fact, be either zero

or ninety degrees under ideal conditions,where the offsets in each direction will all be equal. This entire method was, therefore, simplified to simply finding the center of mass of the offset points in each direction, defined as the average point since all points are equally weighted.

Summary

The correlation algorithm described above, is only one of several algorithms that could be used in a correlation tracker. This algorithm was readily available, and was suggested for use by the Air Force Weapons Laboratory. No attempt was made to determine if it was the most efficient or optimal in any sense. Care was taken only to make certain that both this algorithm, and the extended Kalman filter algorithm received the same information from the FLIR sensor model. In this way, the filter algorithm could be compared for accuracy, in a statistical sense, against an algorithm already in use, to examine the utility of an extended Kalman filter in the tracking loop.

V. Computer Simulation

Introduction

The computer used to simulate the models of the previous three chapters was the CDC-6600. As with any computer simulation, the time and memory requirements vary directly with the complexity of the problem. The key, of course, is to minimize these requirements and still achieve an algorithm that will perform well in a real world environment. This chapter will discuss the methods used to arrive at this balance.

Program Modularity and Efficiency

An attempt was made to keep the program as modular as possible by using subroutines for major calculations. The purpose of the study was not, however, to create a generalized program to work under all possible variations of filter design. For this reason, changing any of the models discussed previously would probably necessitate at least some modification to the program modules. The FORTRAN source code of the program used, is listed in Appendix D.

The large word size and relative speed of the CDC-6600, compared to a machine used for on-line implementation, also compensates for many inefficiencies in programming techniques. Although the algorithms developed earlier are meant to be implemented on-line eventually, the program used would have to be restructured to make it as efficient and numerically precise as possible.

Approximations

One of the most important aspects of the computer simulation was

determining a method for evaluating the outputs of the FLIR sensor. Mathematically, this would involve evaluating the expression

$$\frac{1}{A_p} \int\int_{\substack{\text{i-j-th} \\ \text{pixel}}} I_{max} \exp\left\{\frac{-1}{2\sigma g^2}\left[(x - x_{peak})^2 + (y - y_{peak})^2\right]\right\} dxdy \qquad (100)$$

as described in equation (30) for the truth model. Similar expressions would have to be evaluated for the filter model to arrive at the filter's estimate of the measurements, equation (51), and to evaluate the partial matrix found in equation (71).

Equation (100) represents the average value of the two-dimensional Gaussian intensity function

$$I_{max} \exp\left\{\frac{-1}{2\sigma g^2}\left[(x - x_{peak})^2 + (y - y_{peak})^2\right]\right\} \qquad (101)$$

within the pixel of interest. A reasonable approximation, for on-line usage, to equation (100) would be simply to evaluate the intensity function at the midpoint of the pixel of interest. A better approximation would be to subdivide the pixel into smaller areas, perform the same midpoint evaluations for each of these smaller areas, and to average the result. Obviously, the more subdividions that are made, the better the approximation.

Because the truth model is designed to be a good representation of the real world, an accurate evaluation of the integral is needed. To meet this requirement, each pixel was split into sixteen equal subdivisions. The filter, on the other hand, used the intensity level at a single midpoint to approximate the integral.

White Noise Terms

Throughout the simulation, there was a need to produce discrete

values of a Gaussian distributed, white sequence with a given variance. In order to create these values, the RAND function of the FORTRAN library was used. This function is a random number generator with a uniform probability density function between 0 and 1. Twelve independent random samples from this function were processed with the algorithm

$$v(k) = \sigma_w \{ \sum_{i=1}^{12} u[12(k-1) + i] - 6 \} \tag{102}$$

for k = 1, 2, 3, ...

where

v(k) = a discrete realization of the white, Gaussian sequence;

σ_w = the standard deviation of the desired white noise;

u(k) = random number from the RAND function;

to generate a single sample of the desired white, Gaussian sequence. The above algorithm produces a sequence of numbers that is very nearly Gaussian distributed with zero mean and the desired variance. This is true because of the concepts set forth in the Central Limit Theorem (Ref 3:103).

Monte Carlo Analysis

Determining how any computer algorithm will perform in a real world environment is a difficult process. Unlike the carefully controlled computer simulation, the actual environment is constantly changing. No two experiments are ever the same, and to test all possible cases would be an impossible task. For this reason, a Monte Carlo analysis is used to accomplish a performance evaluation. This method actually constructs a sample-by-sample simulation using random

number generators to produce the outputs of random noise sources in the simulation. The noisy system measurements are then processed by the extended Kalman filter, or the correlator algorithm, to generate state estimates. Many of these simulations are performed and the sample statistics of the error between the state estimates and the states of the truth model are then computed. The trend of these error statistics shows how the estimation algorithm performs in a general sense. Ideally, a larger number of simulations should be performed to make sure that the error sample statistics are indeed accurate representations of the true statistical characteristics. Realistically, however, computer time requirements dictate a smaller number. One means to estimate the number of runs needed is to plot the standard deviation of the error at a number of chosen times versus the number of simulations. As the number of simulations increase, the standard deviation should converge to a steady value. This convergence shows that the error statistics are being adequately described. An example of this convergence is shown in Figure 9. It shows the variance characteristics of a representative error at four different times. It also shows that 20 simulations were adequate for this case. These types of plots were produced for every case studied to make sure the statistics converged. Twenty simulations were found to be adequate in all cases.

Tuning the Extended Kalman Filter

Tuning the filter is an off-line process to determine the proper noise strengths and correlation times so the filter will perform well in any situation. This is necessary because the filter is purposely

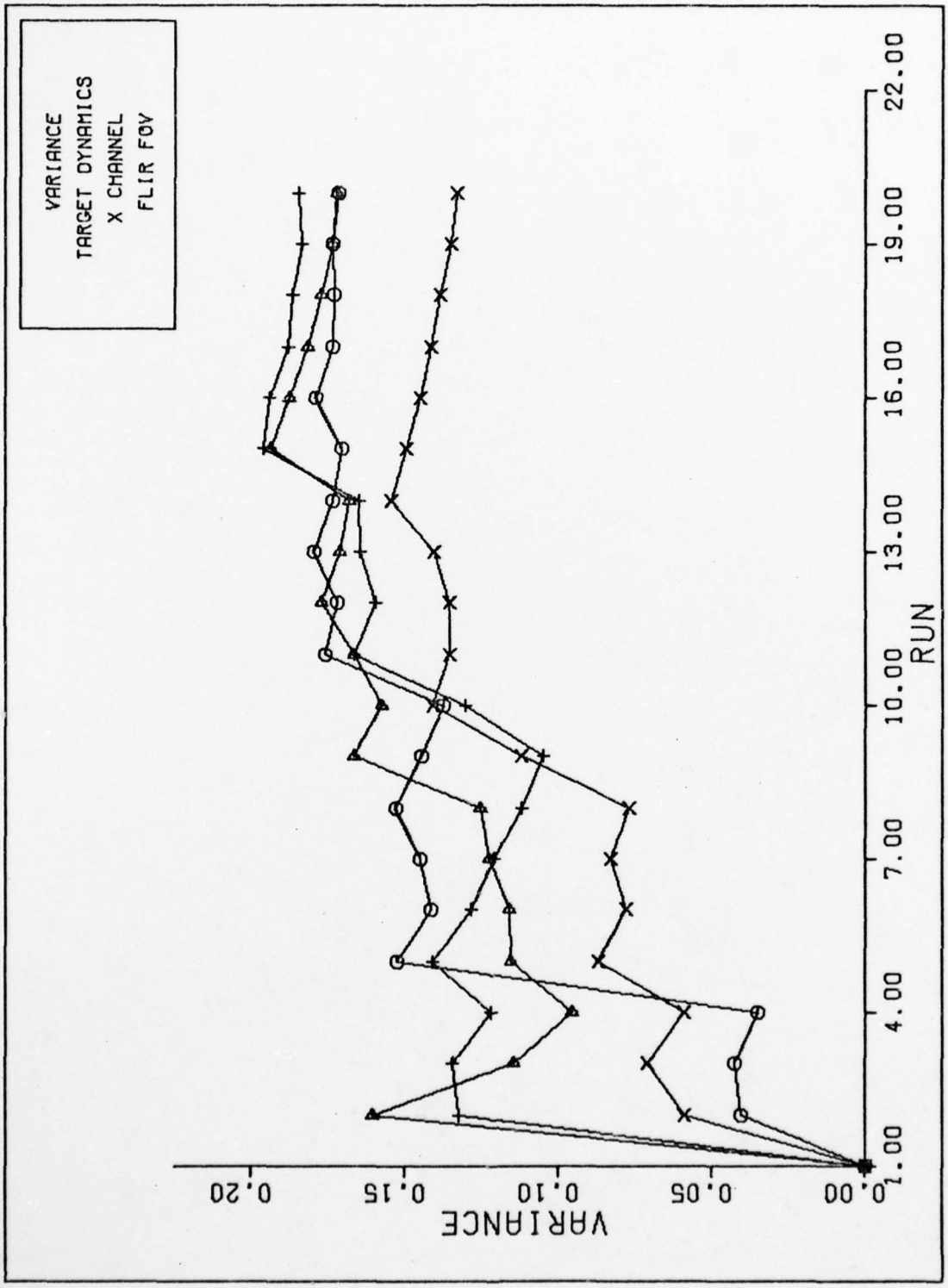


Figure 9. Example of VARIANCE CONVERGENCE

designed to be a reduced order, simplified model of the real world, and must overcome the effects of non-whiteness, non-Gaussian noises, nonlinearities, and approximations to a true environment.

The tuning process is done by varying the values of the strengths of the white noise terms in both the differential state equation and the filter measurement equation. In this study, this corresponds to altering the values of the $\underline{Q}_F$ matrix given in equation (47) and the value of R_F in equation (58). Changing these values, along with the initial conditions, will cause the statistics of the error committed by the filter in estimating the states to vary. The objective of tuning is to vary these strengths in order to minimize the estimation error in some appropriate manner. Here, the object is to minimize the error on all filter states simultaneously. This minimization is totally problem dependent, however, and other problems may call for minimizing only a given number of states, using a weighted average, or using some type of cost function to measure the minimum.

One method used, in many cases, to aid the tuning process, is to compare the filter's estimate of the variance of the error with the actual error variance for each state. When these values match well over the time interval examined, then the actual error committed by the filter should be minimized. If the filter is underestimating or overestimating the true variance, the filter gains will not be optimal, and will result in a larger true error. It is important, therefore, that the truth model be as accurate as possible. The filter will work well on-line only if the truth model is an accurate representation of the real world.

VI. Results

Introduction

This chapter presents the results of several computer simulations designed to evaluate the performance of the extended Kalman filter algorithm in the open-loop tracking problem. As requested by the Air Force Weapons Laboratory, the performance of the extended Kalman filter was compared to the performance of the correlation tracker in five specific areas:

- performance as a function of signal-to-noise ratio where

$$\frac{S}{N} = \frac{I_{max} - \text{mean background noise}}{\text{RMS value of background noise}}$$

- performance as a function of intensity pattern size (beam width);
- performance as a function of the ratio of RMS target motion to RMS atmospheric jitter;
- performance as a function of target correlation time;
- performance as a function of mismatches between the true intensity pattern and that assumed by the filter.

Due to the limitations, all of the above areas were covered with the first three areas receiving the most detailed analysis.

Correlation Algorithm

The correlation tracking algorithm (Ref 6) was modified slightly after several problems were discovered during the study. The first problem occurred in the LOCATE subroutine (Appendix D) when a line of reference data was shifted by more than 8 pixels, resulting in an entire line of zeros and an indefinite correlation coefficient. To solve this problem, a check was put into the subroutine to assure that each row or column was limited to a shift of less than 8 pixels when trying to

align the target and reference data sets. If the target and reference rows or columns were not aligned after 7 pixel shifts in the same direction, then the data from that row or column was considered "poor" and was discarded.

The second modification involved a control in the DRIVER program designed to reject low quality offset data by checking the maximum correlation coefficient for each row and column against a preset cut-off value. In the routine obtained from the Weapons Laboratory, the row correlation loop eliminated offset data with a maximum correlation coefficient less than 0.95, while the column loop used 0.90 as the cutoff value. There was no documented reason for this difference, nor was there any justification for these seemingly arbitrary cutoff values. It was discovered, however, that changing this cutoff value led to drastic differences in the performance of the correlation algorithm. These differences are documented and discussed later in this chapter. For these reasons, this study used 0.95 as the cutoff for both row and column correlation. Similar controls in the LOCATE subroutine were eliminated since they were redundant.

Cases Studied

In order to investigate the five specified areas in the time available, the sixteen scenarios listed in Table I were simulated. As instructed by the Air Force Weapons Laboratory, the specific values used in each simulation were combinations of the following:

- signal-to-noise (S/N) ratios of 20, 10, 1
- RMS target/atmospheric motion ratios (σ_D/σ_A) of 5, 1, .2
- Gaussian beam dispersions (σ_{g_T}) of 3, 1 pixels

Table I

List of Cases Studied

Case	τ_T	S/N	σ_D	σ_A	σ_D/σ_A	σ_{g_T}	σ_{g_F}
1	1.	20.	1.0	0.2	5.0	3.	3.
2	1.	20.	1.0	1.0	1.0	3.	3.
3	1.	20.	0.2	1.0	0.2	3.	3.
4	1.	20.	1.0	0.2	5.0	1.	1.
5	1.	10.	1.0	0.2	5.0	3.	3.
6	1.	10.	1.0	1.0	1.0	3.	3.
7	1.	10.	0.2	1.0	0.2	3.	3.
8	1.	10.	1.0	0.2	5.0	1.	1.
9	1.	1.	1.0	0.2	5.0	3.	3.
10	1.	1.	1.0	1.0	1.0	3.	3.
11	1.	1.	0.2	1.0	0.2	3.	3.
12	1.	1.	1.0	0.2	5.0	1.	1.
13	5.	20.	2.0	0.4	5.0	3.	3.
14	1.	20.	2.0	0.4	5.0	3.	3.
15	1.	20.	1.0	0.2	5.0	3.	1.
16	1.	20.	3.0	0.6	5.0	1.	1.

- Target correlation times (τ_T) of 1, 5 seconds

The value specified by σ_{g_F} in the table is the value for the Gaussian dispersion assumed by the filter.

The sixteen cases shown in Table I were chosen to allow a general framework in which direct comparisons could be made. Cases 1 through 12 compare four basic scenarios using the three signal-to-noise ratios and the two values of beam dispersion shown above. These four basic scenarios allow direct comparison of the performance of the Kalman filter under various atmospheric conditions. The remaining four cases were used to show the effects of using a large value for target correlation time, determining the effects of filter mismatch, and determining the effects of using a large RMS target motion combined with a small target pattern size. The results of the simulations performed are presented in Appendix E as mean $\pm 1\sigma$ error plots for the correlator, the filter estimate of target dynamics, and the filter estimate of atmospheric jitter. The mean and sigma values are sample statistics generated from a Monte Carlo simulation using 20 runs. Because of the similarity between the horizontal (x) and vertical (y) channels, only the mean $\pm 1\sigma$ error plots for the horizontal channel are presented.

Tuning

Because of the similarity of the truth and filter models used in this study, tuning of the extended Kalman filter was relatively easy. The only difference between the models was the set of equations used to simulate atmospheric disturbance. Whereas the truth model used a third order Markov process to simulate the atmospheric jitter, the extended Kalman filter used a first order process with a break frequency

set equal to the lower break frequency of the higher order model. Similarly, the variance for the white noise process used in the filter measurement equation (53) was set equal to the sum of the variances of the processes simulating the FLIR and background noises in equations (32) and (33). This is appropriate since all of the above processes are simulated as independent, zero mean, and Gaussian. Since the value of R_F is now fixed, adequate tuning results when the strengths of the white noise terms and the correlation times for both the truth and filter models for dynamics are set equal, and when the RMS values for atmospheric jitter for both models are set equal. For the third order atmospheric model, this means that the correct value of the gain, K, must be calculated to achieve the desired RMS value on the output process.

Figures 10, 11, and 12 show plots of the actual versus the filter-computed value of the RMS error committed in estimating target motion in the horizontal direction for cases 2, 6, and 10 after the above tuning was performed. Figures 13, 14, and 15 show the same type of plots for the atmospheric jitter. The smoother curve represents the filter estimate of the RMS error. The plots show typical results for signal-to-noise ratios of 20, 10, and 1 respectively. As shown in Figures 12 and 15, the filter begins to underestimate the RMS error as the signal-to-noise ratio gets low. This is due to the effects of the small mismatch between the truth model and the filter finally becoming apparent. Because of the way the Q_F matrix and the value for R_F are chosen in the above tuning process, the tuning was independent of the signal-to-noise ratio and the Gaussian beam dispersion. This allows comparison of the outputs of identically tuned filters for

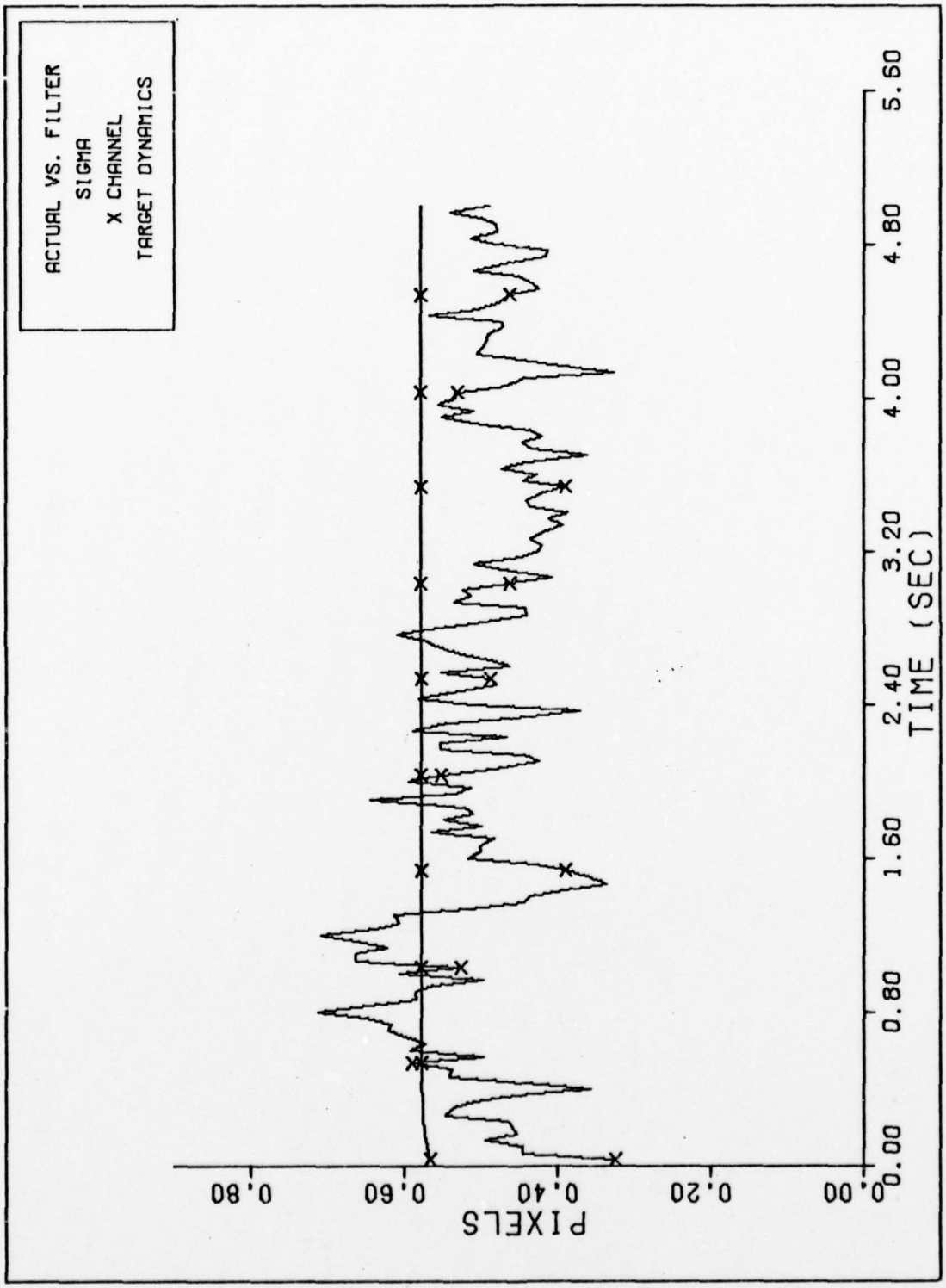


Figure 10. FILTER VS. ACTUAL SIGMA PLOT (S/N = 20)

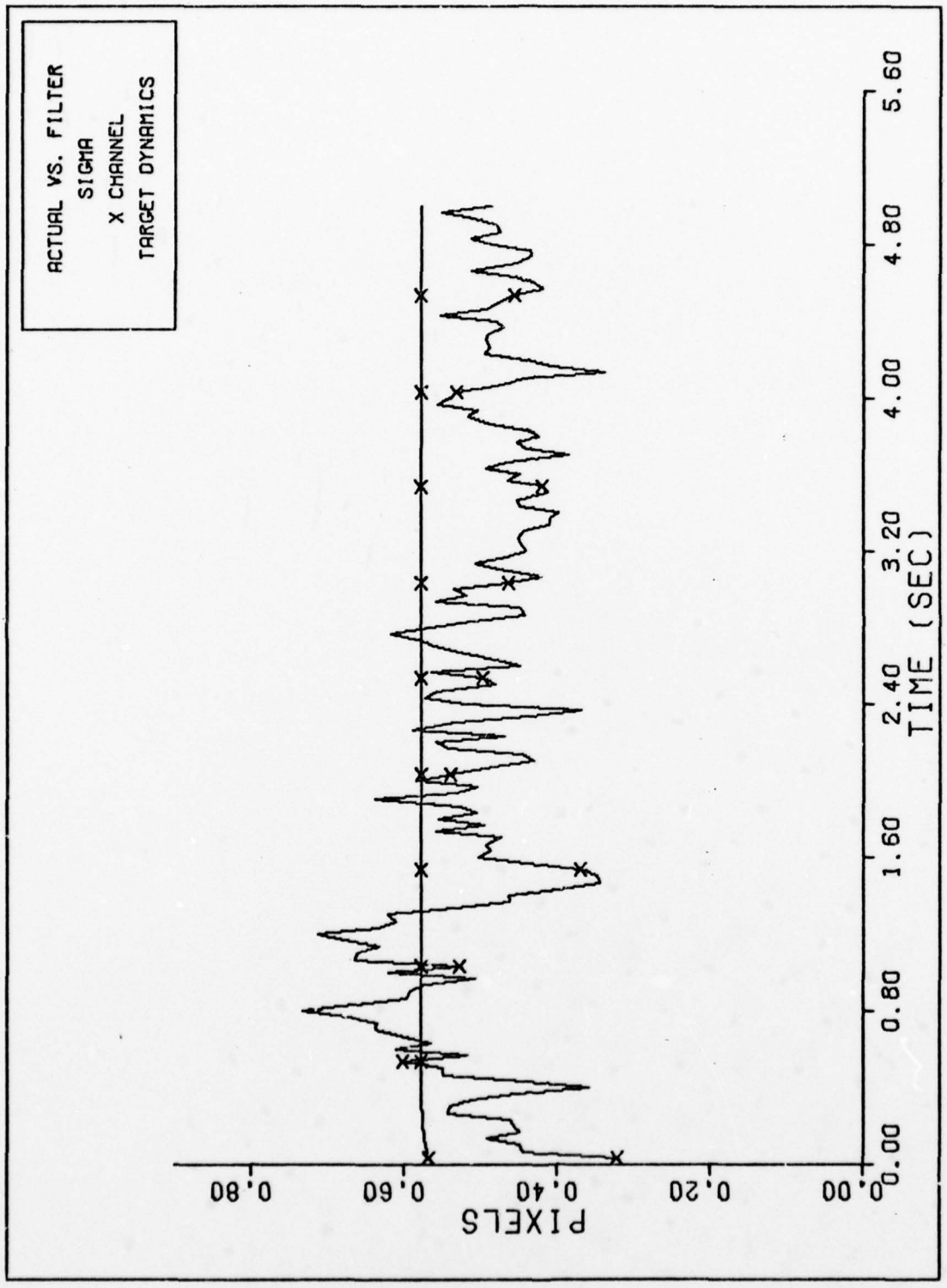


Figure 11. FILTER VS. ACTUAL SIGMA PLOT (S/N = 10)

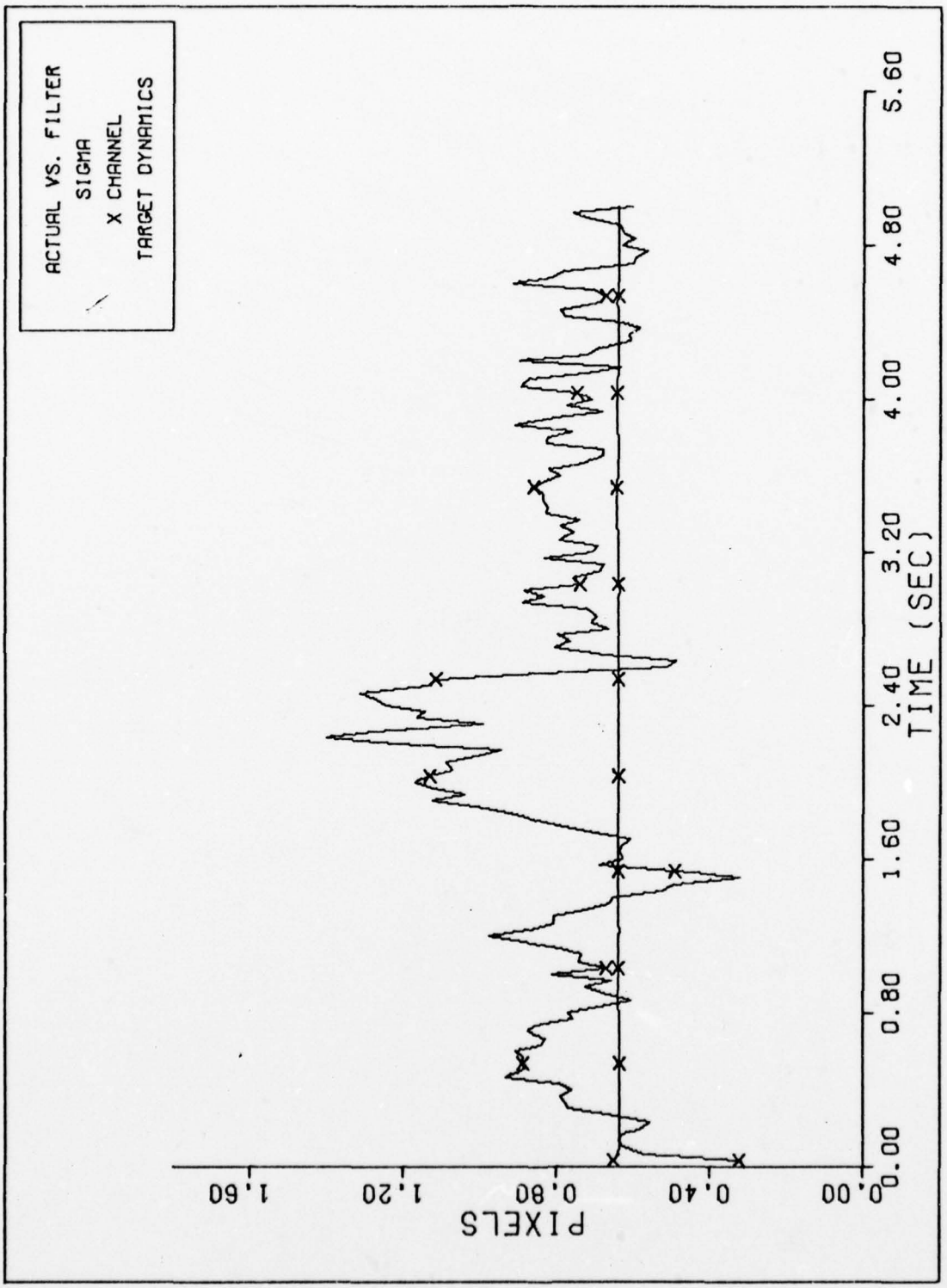


Figure 12. FILTER VS. ACTUAL SIGMA PLOT (S/N = 1)

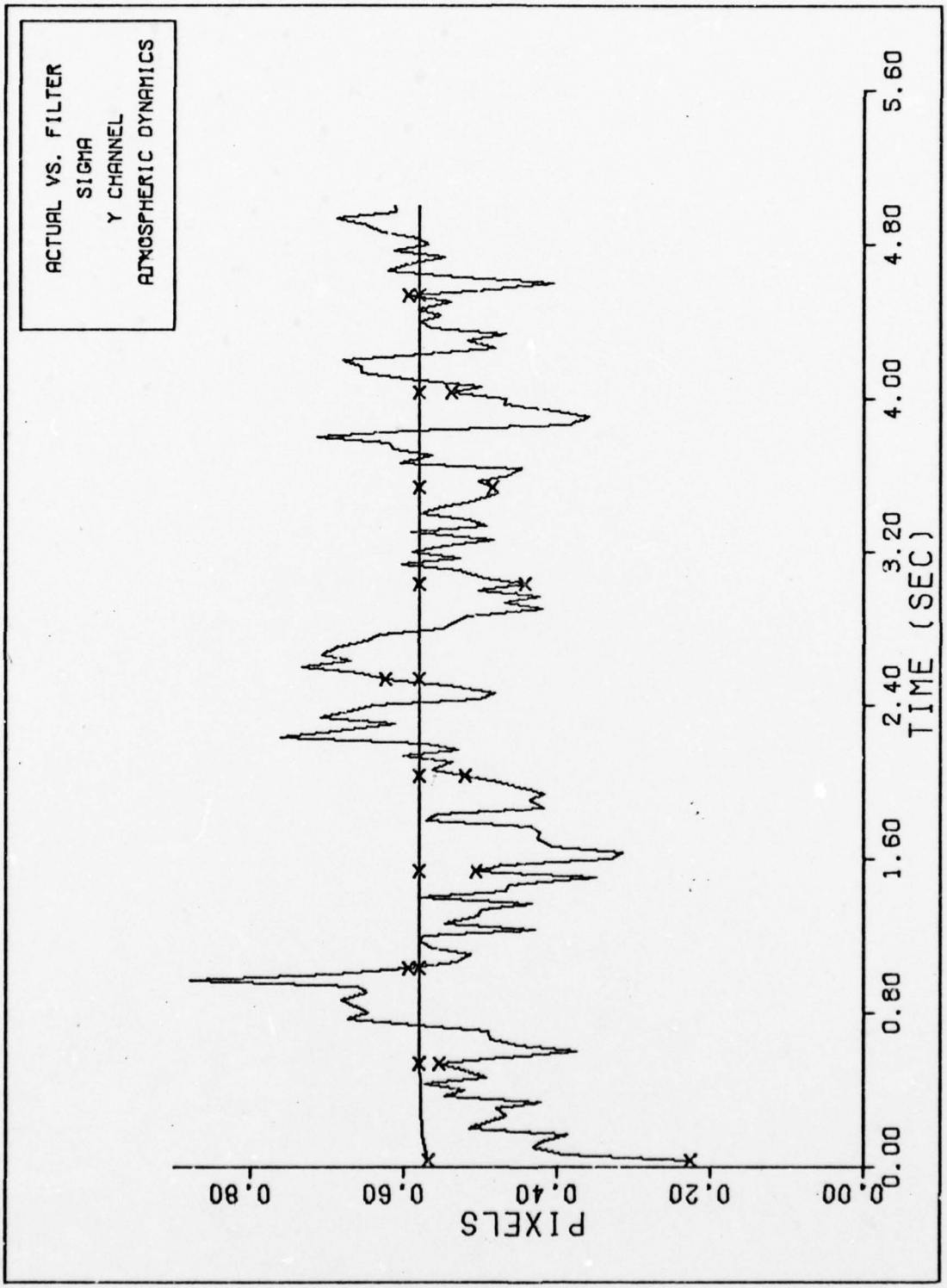


Figure 13. FILTER VS. ACTUAL SIGMA PLOT (S/N = 20)

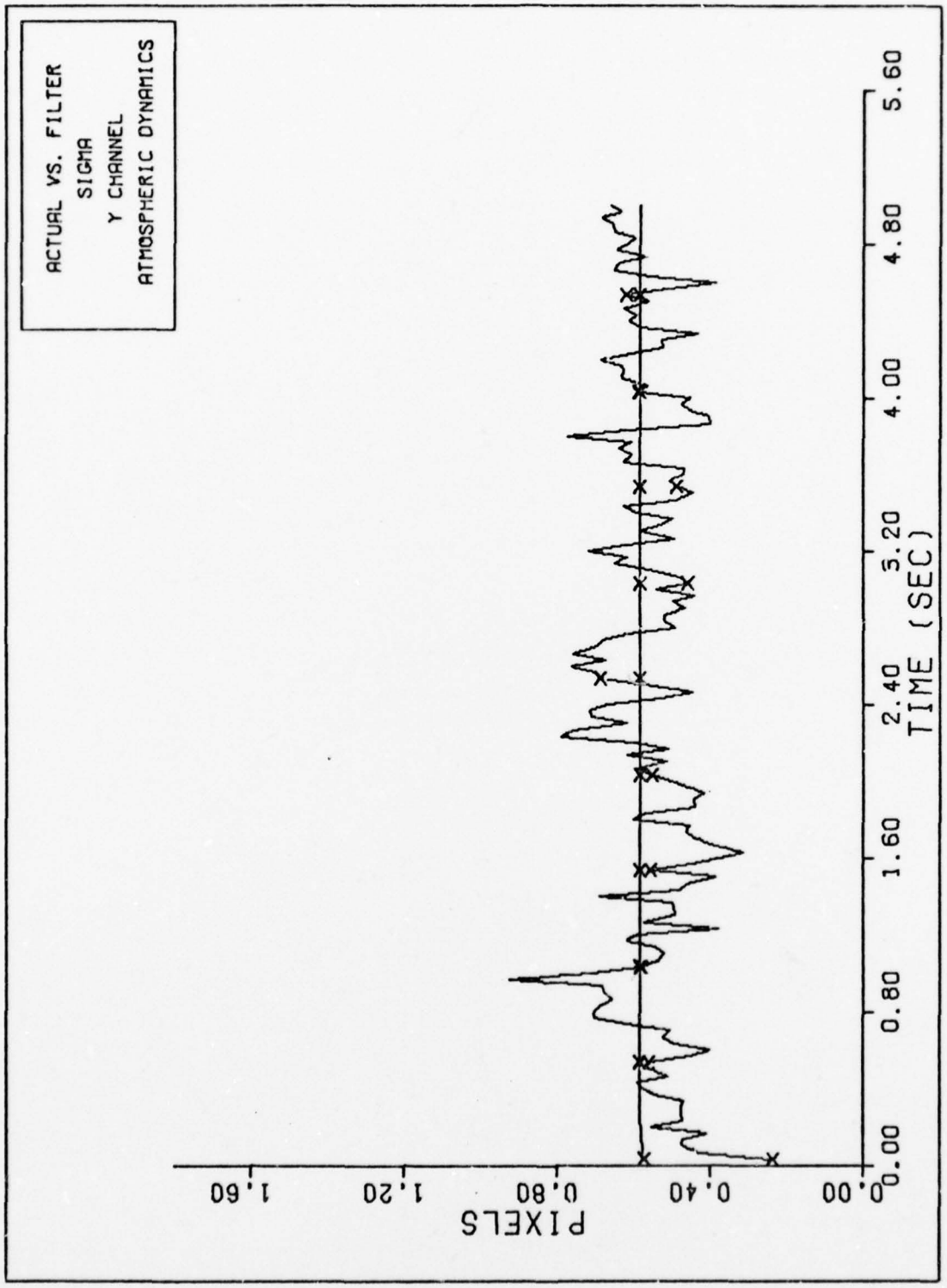


Figure 14. FILTER VS. ACTUAL SIGMA PLOT (S/N = 10)

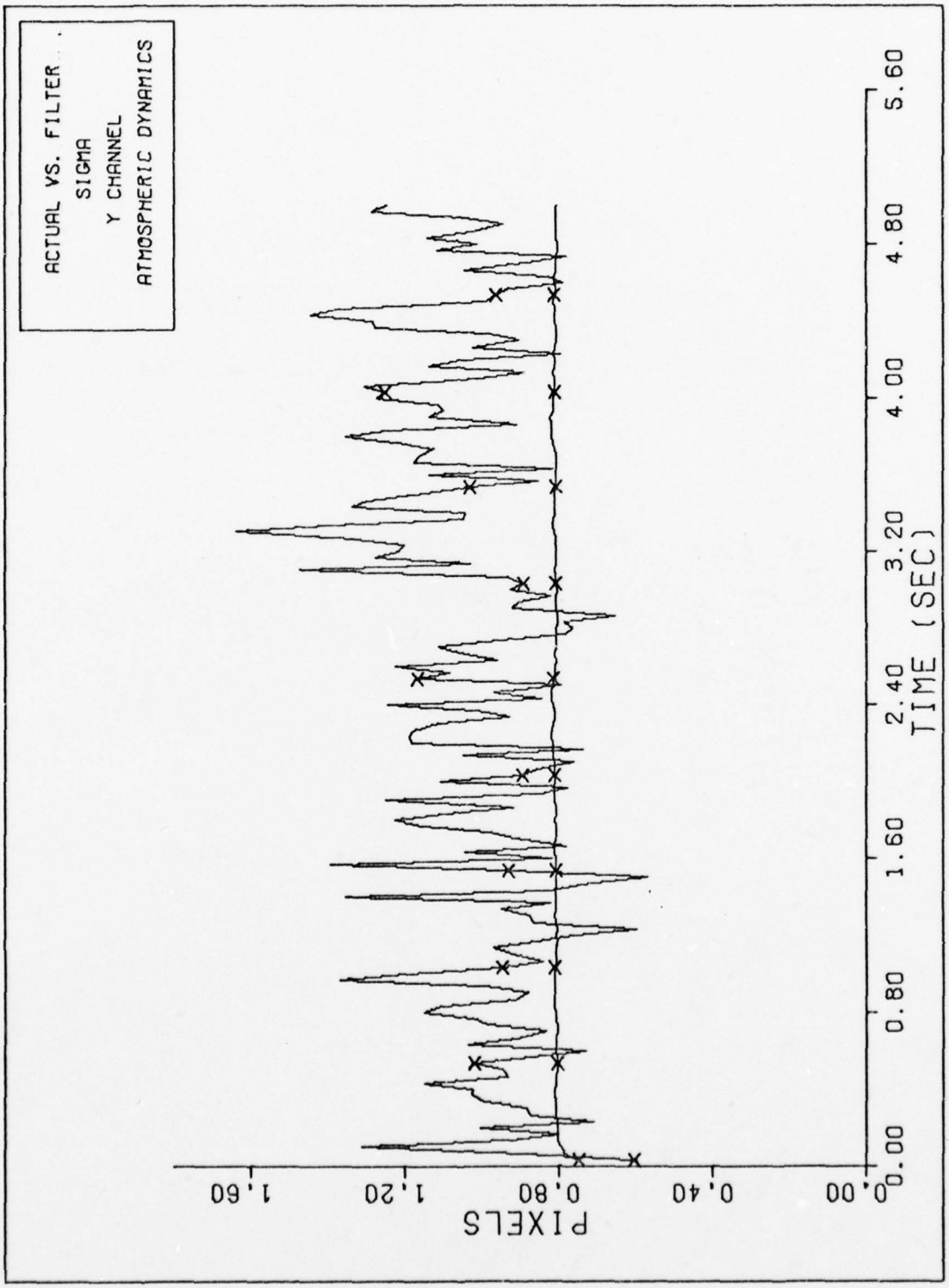


Figure 15. FILTER VS. ACTUAL SIGMA PLOT (S/N = 1)

different signal-to-noise ratios and different beam dispersions. These comparisons will show how the performance of the filter will change as the tracking environment changes.

Discussion

In comparing the performance of the extended Kalman filter to the correlation algorithm, the five specific areas mentioned earlier were examined. The extended Kalman filter algorithm performed extremely well as the signal-to-noise ratio was lowered, with no noticeable change between the ratios of 20 and 10, and only a slight degradation in performance shown at a signal-to-noise ratio of 1. Tables II and III present a comparison of the performance of the two algorithms in mean and 1 sigma tracking error for the three signal-to-noise ratios examined. The numbers in the tables are values that were estimated from the applicable plots and are only good to 1 or 2 significant figures. Table II represents the average of the estimated values for cases 1 through 12 where the beam dispersion was set to 3 pixels, while Table III represents the estimated values when the beam dispersion was set to 1 pixel. The tables also represent values estimated at the end of the 5 second run time since this is where the maximum errors occurred in most cases. As shown by the plots for cases 1 through 12, both algorithms started out with low errors due to the chosen initial conditions of zero position error. The correlation algorithm quickly diverged in almost all cases, resulting in significant errors after only a few seconds. The extended Kalman filter, on the other hand, had mean errors of close to zero for all signal-to-noise ratios, and although the 1 sigma error value rose from .2 to .8

Table II

Mean Error and 1σ Error Comparisons for the Correlation Tracker and the Extended Kalman Filter with σ_g=3 Pixels

S/N	Correlation Tracker mean error (pixels)	Correlation Tracker 1σ error (pixels)	Extended Kalman Filter mean error (pixels)	Extended Kalman Filter 1σ error (pixels)
20	0.5	1.5	0.0	0.2
10	3.0	3.0	0.0	0.2
1	15.0	30.0	0.0	0.8

Table III

Mean Error and 1σ Error Comparisons for the Correlation Tracker and the Extended Kalman Filter with σ_g=1 Pixel

S/N	Correlation Tracker mean error (pixels)	Correlation Tracker 1σ error (pixels)	Extended Kalman Filter mean error (pixels)	Extended Kalman Filter 1σ error (pixels)
20	7.0	8.0	0.0	0.2
10	8.0	10.0	0.0	0.2
1	15.0	30.0	0.0	0.8

pixels as the signal-to-noise ratio decreased, this difference is not significant. When the dispersion of the Gaussian intensity function was decreased to 1 pixel, the performance of the correlation tracker was significantly degraded with errors ranging from 7 to 10 pixels. This same change had essentially no effect on the performance of the extended Kalman filter algorithm. This difference in performance was probably due to the a priori knowledge of beam width and target motion assumed by the Kalman filter. As the size of the beam width is decreased, the useful tracking information is spread over a smaller number of pixels. This knowledge can be exploited by the filter to give better performance, especially at the lower signal-to-noise ratios. This suggests that using data from a smaller region about the projected target location may be more efficient than using the entire 8 by 8 array. Even the filter performance, however, is slightly degraded at a low signal-to-noise ratio since the approximations used in calculating the $\underline{H}$ matrix and in performing the integration are not as accurate.

When the ratio of RMS target motion to RMS atmospheric jitter was decreased from 5 to 1, the filter showed a corresponding increase in actual error with a mean still close to zero. Comparing cases 1 and 2 shows that the 1 sigma value on the error jumps from about .2 pixels to about .5 pixels. This trend, however, is reversed as the ratio is decreased further as shown by the output of case 4, where the ratio was equal to .2. Here, the 1 sigma value for the error goes back down to about .2 pixels again. This seems to imply that the filter can distinguish between the true and apparent motion easier when there is

a large difference between the two. This type of trend was harder to distinguish in the outputs of the correlation algorithm due to the divergent characteristics of the statistics. A comparison of the same cases, however, seem to present no apparent difference in accuracy due to a change in ratio. In all three cases, the correlation algorithm had a 1 sigma error value of around 1. pixel. Even in the worst case, therefore, the extended Kalman filter algorithm performed better.

Increasing the correlation time for the target in both the truth model and the filter from 1 to 5 seconds had no discernable effect on the performance of either the extended Kalman filter algorithm or the correlation algorithm. A comparison of the output for cases 13 and 14 show almost no change in the statistics. This is probably due to the small difference in correlation times used. If the correlation time were changed to a large value, on the order of 1 minute or more, the tracking accuracy should improve since the high frequency jitter term could be filtered more easily from the overall motion apparent in the target plane. A much shorter correlation time, on the other hand, would make the actual target motion appear more like a white noise term and could introduce a larger error.

Case 15 examined the effects of mismatching the shape of the intensity function between the truth model and the filter. The filter assumed a dispersion on the Gaussian intensity function of 1 pixel and the truth model value was set at 3 pixels. This scenario was designed to test the sensitivity of the filter to poor assumptions in target shape. In this case the filter appeared to be rather robust, resulting in a mean error of about zero with a 1 sigma value of only

.3 pixels. When compared to case 1, where the filter assumed the correct value of 3 pixels for the dispersion, there was only a slight degradation in performance. Case 1 resulted in a mean error of about 0. with a 1 sigma value of .2 pixels. Although results based on one case are inconclusive, this shows that the filter could be relatively insensitive to poor assumptions and further testing, such as mismatches in target shape, target motion, or signal-to-noise ratios, would be warranted.

The worst performance of the extended Kalman filter resulted when the value of RMS target motion was very high, as in case 16 where it was set to 3 pixels. This resulted in RMS tracking errors of 2 to 3 pixels after 5 seconds of run time. This is not surprising, however, since this magnitude of target motion forced the target entirely out of the field of view of the FLIR on many passes. This, of course, resulted in extremely poor measurement data which would tend to drive the tracking errors up, and suggests a larger field of view for update.

One interesting result is the divergence characteristic of the correlation tracker. This divergence was noted in almost all cases, even with high signal-to-noise ratios. It was discovered, however, that in many cases this divergence could be reduced or even eliminated by increasing the cutoff control values which reject the offset data from poorly correlated rows and columns. This reduction is distinctly shown in Figures 16 and 17 corresponding to case 14, and using cutoff values of .90 and .95 respectively, and again in Figures 18 and 19 corresponding to case 5, and using cutoff values of .95 and .975 respectively. This implies that the tracking accuracy increases with a higher cutoff control value, for the environment simulated here, and

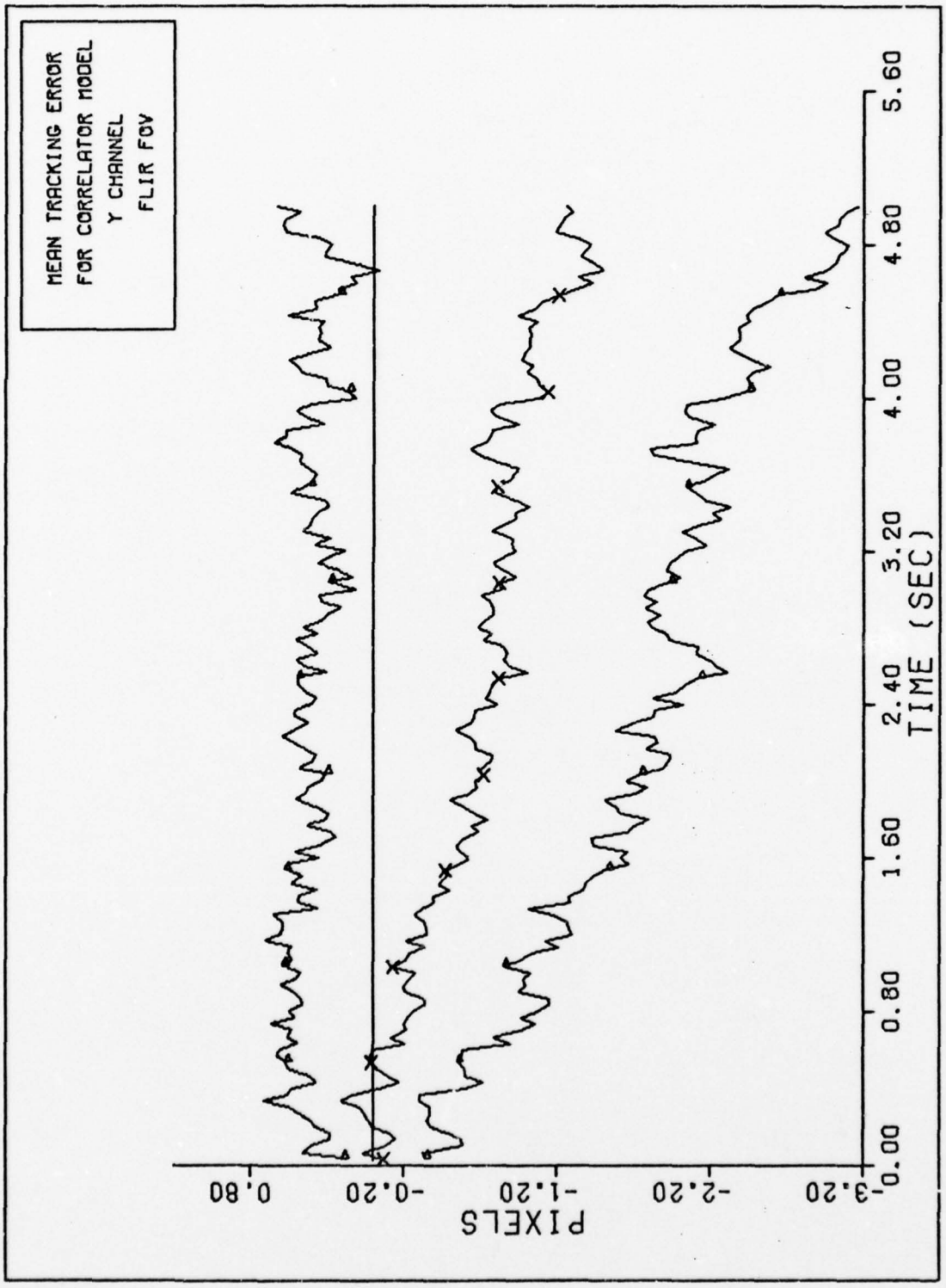


Figure 16. Y CHANNEL CORRELATOR TRACKING ERROR (S/N = 20)

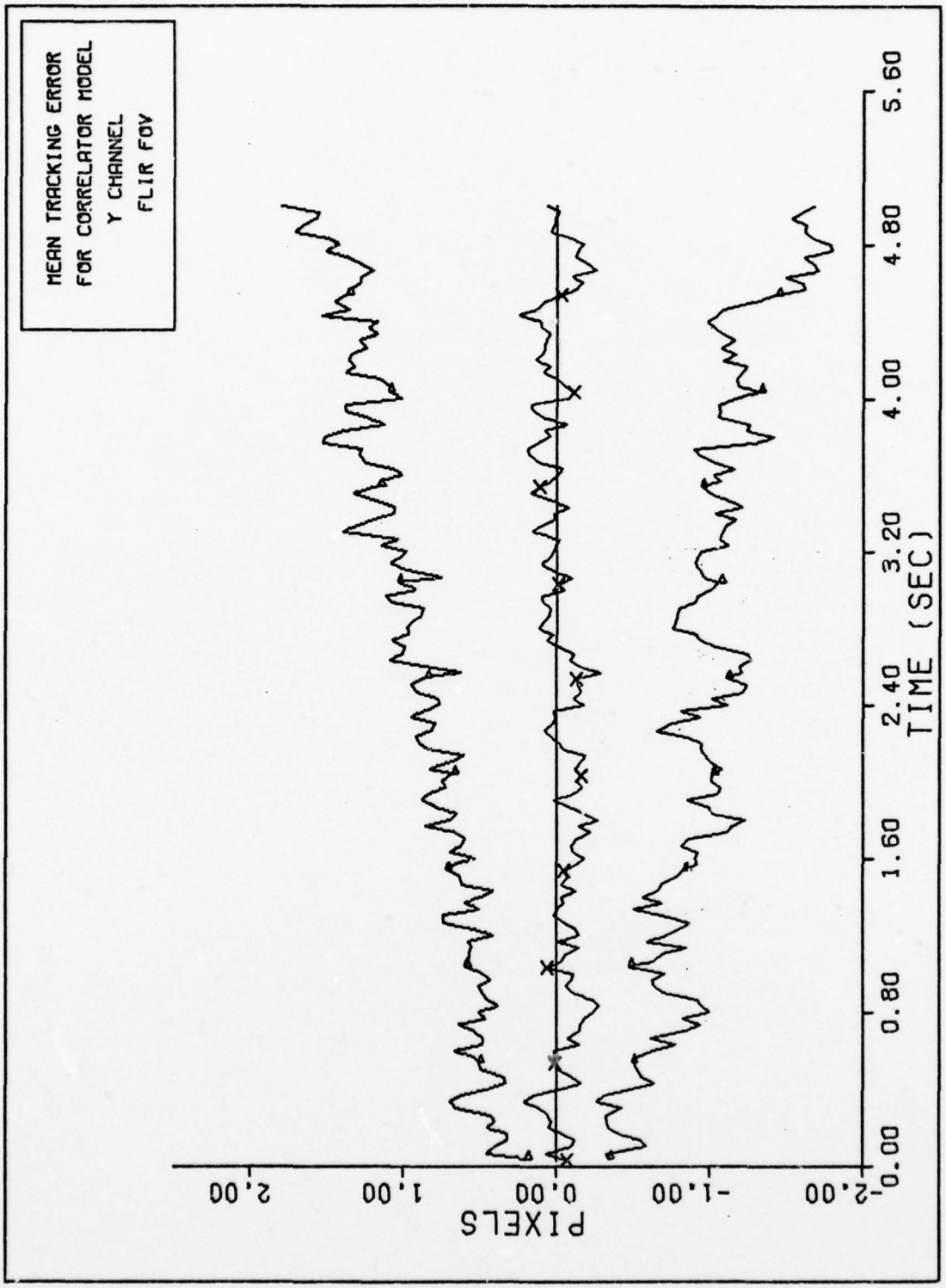


Figure 17. Y CHANNEL CORRELATOR TRACKING ERROR (S/N = 20)

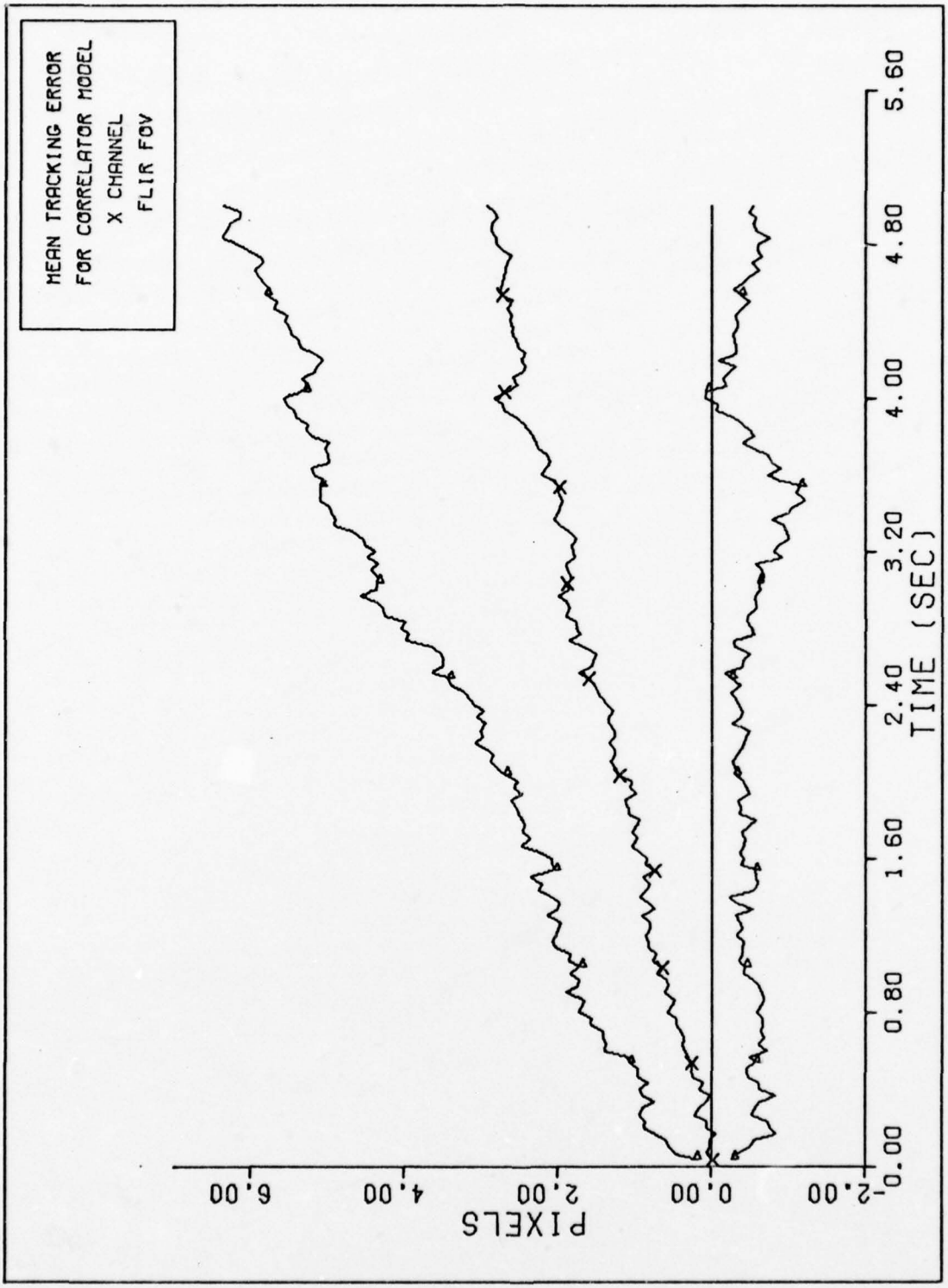


Figure 18. X CHANNEL CORRELATOR TRACKING ERROR (S/N = 10)

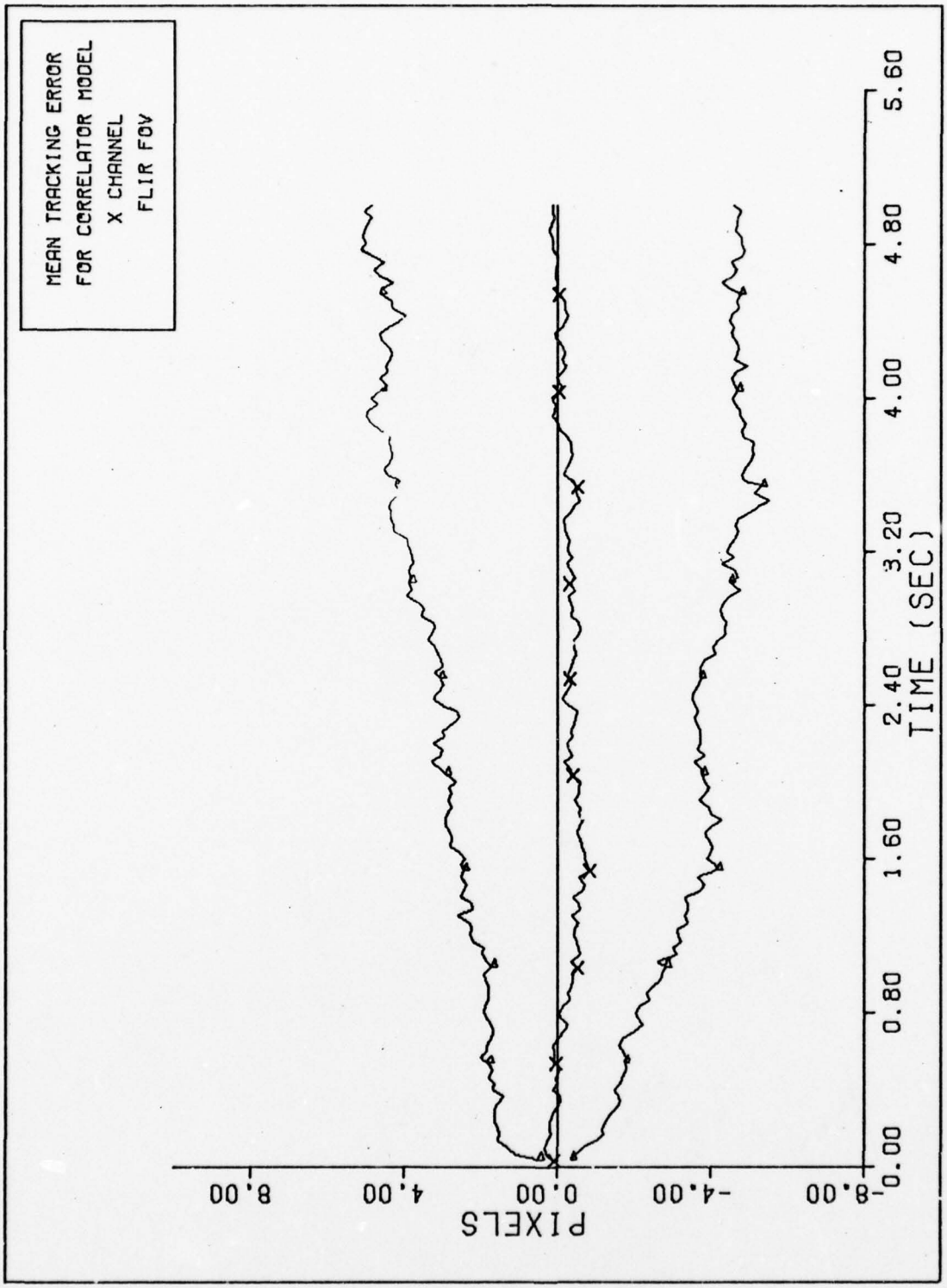


Figure 19. X CHANNEL CORRELATOR TRACKING ERROR (S/N = 0)

that even correlation coefficients as high as .95 do not guarantee accuracy. The direct effect of raising the cutoff control value is to reject the data in more rows and columns specifically, those rows or columns that did not correlate sufficiently high. Hopefully, this results in a more careful selection of "good" data, and will result in better tracking accuracy. Since the choice of a good cutoff value is very problem dependent, with noisy data leading to lower correlation coefficients, this value would have to be changed on-line for the best results.

VII. Conclusions and Recommendations

Conclusions

It is apparent in all the cases studied that the extended Kalman filter outperforms the correlation tracker under the same set of input conditions. As shown in Tables II and III, the performance difference is shown most dramatically when the signal-to-noise ratio is low, and again when the Gaussian beam dispersion is small compared to the pixel size. This result was expected, however, since when the signal-to-noise ratio is low, there is little definition to the target intensity pattern and the tracking algorithm is receiving mostly noisy data. Similarly, when the beam dispersion is small, all of the useful information is concentrated in the area of a few pixels and is easily masked by noise. This masking is especially strong at low signal-to-noise ratios. Since the correlation algorithm is given no a priori information about either the motion of the target or what the intensity pattern looks like, it does a poor job of picking out the target. Nevertheless, this type of tracker could be effectively used in an environment where the signal was strong, or where the shape of the object being tracked is completely unknown or constantly changing. These advantages could, in fact, lead to a design that is even more robust than the extended Kalman filter.

The extended Kalman filter performed well in all cases, even with a very low signal-to-noise ratio. Prior knowledge of the size, shape, and motion of the target was used to track the target effectively under a variety of conditions. The filter algorithm also appeared to be relatively insensitive to small changes in the size of the

assumed Gaussian intensity function. This implies that the filter design may be rather robust and, therefore, could be used in an environment where parameters are constantly varying or are not well known a priori. It is apparent, however, that an extremely poor assumption, such as assuming an entirely false shape, could significantly degrade the performance of this algorithm. This implies that this type of filter could be used effectively only when at least one aspect of the target were known with some degree of certainty.

Recommendations

Further research is recommended to assess the robustness of the basic design of the filter presented here. These tests should include cases where there are errors in the assumed model of atmospheric turbulence as well as poor assumptions of the size, shape, and motion of the target. If warranted, this analysis could lead into the study or design of an adaptive filter model that could estimate parameters that are critical to the filter's performance and robustness, such as target pattern size or shape, on-line. It could even be used to describe the motion of the target or the atmospheric jitter in order to adapt to changing conditons or targets.

Another area of suggested research is to attempt tracking under a wide variety of background conditions. In this way, performance of the filter could be examined as a function of background clutter as would be found in a "look down" mode.

In order to improve the computational feasibility of the filter for on-line applications, research should be performed on ways to improve the speed and memory requirements of this type of algorithm.

One way to improve the speed would be to incorporate measurements using only a small number of pixels. This would involve determing which pixels contain the most information so only these would be exploited. One way to do this would be to use $\underline{\hat{x}}(t_i^-)$ as the best guess of the position of the target at the measurement time and use only those pixels that lie in some locus centered around $\underline{\hat{x}}$. Other methods, such as square root filtering techniques or U-D implementation would reduce the computer wordlength requirements but would result in increased memory and time requirements. These techniques would probably be of little benefit here since the extended Kalman filter only has four states. With an adaptive filter, however, these techniques may be helpful since several more parameters would have to be estimated.

A last possibility would be to explore how an extended Kalman filter of this type would operate when there is more than one target in the FLIR field of view. The goal here would be to see if the filter could adequately distinguish and track targets in a combat environment with a short time constraint.

Bibliography

1. The Analytic Sciences Corporation. Advanced Adaptive Optics Control Techniques. TR-996-1. Prepared for the Air Force Weapons Laboratory, Kirtland Air Force Base, New Mexico, 6 January 1978.

2. Hogge, C. B. and Butts, R. R. "Frequency Spectra for the Geometric Representation of Wavefront Distortions Due to Atmospheric Turbulence," IEEE Transactions on Antennae and Propagation, Vol. AP-24, No. 2, March 1976. (Program supplied by authors).

3. Maybeck, P. S. Stochastic Estimation and Control Systems Part I (Course notes for EE 7.65). Wright-Patterson Air Force Base, Ohio: Air Force Institute of Technology, February 1975.

4. Richards, C. L. Correlation Tracking Algorithm. SAMRT-76-0076 Engineering Data Release. Aeronutronic Ford Corporation, Newport Beach, California, 2 July 1976.

5. -----. Precision Line and Point Reticle Location. SAMRT-77-0006 Technical Data Release. Ford Aerospace and Communications Corporation, 8 March 1977.

6. -----. Correlation Tracking Software. SAMRT-76-0088 Engineering Data Release. Aeronutronic Ford Corporation, Newport Beach, California, 17 August 1976.

7. Morgan, F. E. Chief, Systems Technology Section, Air Force Weapons Laboratory (Personal Correspondence), Kirtland Air Force Base, New Mexico, 13 February 1978.

8. Richards, C. L. Results of HAWK Image Tracking Experiment. SAMRT-76-0087 Engineering Data Release. Aeronutronic Ford Corporation, Newport Beach, California, 17 August 1976.

9. Maybeck, P. S. Stochastic Models, Estimation and Control. Unpublished text. School of Engineering, Air Force Institute of Technology, Wright-Patterson Air Force Base, Ohio, 1978.

Appendix A

Numerical Precision Problems

In the early stages of this study, a great deal of time was spent searching for a reasonable way to evaluate the truth model $\underline{Q}_{T_d}$ matrix, defined in equation (25), and repeated here as

$$\underline{Q}_{T_d} = \int_{t_i}^{t_i+1} \underline{\Phi}_T(t_{i+1},\tau)\underline{G}_T(\tau)\underline{Q}_T(\tau)\ \underline{G}_T^T(\tau)\underline{\Phi}_T^T(t_{i+1},\tau)d\tau \qquad \text{(A-1)}$$

Initially, the truth model state equations were put into phase variable form, since it could be done from inspection of the system transfer functions.

Because the truth model is a time-invariant system, the integral in equation (A-1) can be solved exactly, and will be a constant for all sampling periods. This means that it need be calculated only once off-line. Solving equation (A-1) exactly, however, would require a significant amount of manual calculation, even after reducing the problem because of the two-channel symmetry. In order to save both time and effort, approximation techniques were examined.

The simplest approximation would be to let

$$\underline{Q}_{T_d} \simeq \left[\underline{G}_T\underline{Q}_T\underline{G}_T^T\right]\Delta t \qquad \text{(A-2)}$$

This results in a good approximation when the sampling period is small compared to the response time of the system (Ref 9:Ch 6). In this case, however, using the phase variable form of the state equations, where

$$\underline{G}_T = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & Kab^2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & Kab^2 \end{bmatrix}^T \tag{A-3}$$

and

$$\underline{Q}_T = \begin{bmatrix} \frac{2\sigma_D^2}{\tau} & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{2\sigma_D^2}{\tau} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{A-4}$$

this approximation results in a diagonal matrix. This would effectively eliminate the effects that would normally be present due to any cross correlation terms in the fully evaluated $\underline{Q}_{T_d}$, and the approximation was, therefore, judged inadequate.

The second approximation, was a trapezoidal integration of equation (A-1) yielding

$$\underline{Q}_{T_d} = \frac{\Delta t}{2}\left[\underline{\Phi}_T(t_{i+1},t_i)\underline{G}_T\underline{Q}_T\underline{G}_T^T + \underline{G}_T\underline{Q}_T\underline{G}_T^T\underline{\Phi}_T^T(t_{i+1},t_i)\right] \tag{A-5}$$

This approximation to $\underline{Q}_{T_d}$ appeared to be adequate until the matrix was passed through the Cholesky square root algorithm, as described in Chapter II, where it was found to contain a negative eigenvalue. Knowing that the matrix should always be positive semi-definate (Ref 9), this was also found to be an inadequate approximation.

On the next attempt, the equation

$$\underline{Q}_{T_d} = \underline{F}_T\underline{Q}_{T_d} + \underline{Q}_{T_d}\underline{F}_T^T + \underline{G}_T\underline{Q}_T\underline{G}_T^T \tag{A-6}$$

where $\underline{Q}_T$ is defined in (A-4), was integrated using an Euler integration with ten steps per time period. This again resulted in a matrix with a negative eigenvalue when passed through the Cholesky square root algorithm. Trying to gain accuracy by using twenty steps gave almost identical results and this method was also abandoned.

At this point, the phase variable form for the state equations was reexamined to check the validity, and then transformed into the Jordan canonical form to see if better results could be obtained.

The approximations in equations (A-5) and (A-6) were again tried with the canonical form, shown in equation (13). Once again the results were similar.

As a last effort, the exact integration of equation (A-1) was attempted, using the canonical form because of the ease in calculating the $\underline{\Phi}_T(t_{i+1},t_i)$ matrix. This effort was successful, and the results are listed in the program listing found in Appendix D.

There is no apparent reason why the approximations of equation (A-3) and (A-4) resulted in matrices that were not positive semi-definate. The 60 bit word size of the CDC-6600 computer system would normally be large enough to compensate for any round-off errors that would occur. This probably means that this particular system is ill-conditioned, and extremely sensitive to minor errors.

Appendix B

Transforming the Atmospheric Model from Phase Variable to Jordan Canonical Form

The third order atmoshperic disturbance model as shown in Figure 5 has a transfer function described as

$$G(s) = \frac{Kab^2}{(s + a)(s + b)^2} \tag{B-1}$$

$$= \frac{Kab^2}{s^3 + (2b + a)s^2 + (b^2 + 2ab)s + ab^2} \tag{B-2}$$

Transferring this into a vector differential equation gives

$$\dot{\underline{x}}(t) = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -ab^2 & -(b^2+2ab) & -(2b+a) \end{bmatrix} \underline{x}(t) + \begin{bmatrix} 0 \\ 0 \\ Kab^2 \end{bmatrix} w(t) \tag{B-3}$$

$$y(t) = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \underline{x}(t) \tag{B-4}$$

Which is the phase variable form of the differential equation. This equation can be transformed into the Jordan canonical form through the use of a similarity transformation (Ref 3:13). This is accomplished by using the following equations

$$\underline{F}^* = \underline{V}^{-1} \underline{F} \underline{V} \tag{B-5}$$

$$\underline{G}^* = \underline{V}^{-1} \underline{G} \tag{B-6}$$

$$\underline{H}^* = \underline{H} \underline{V} \tag{B-7}$$

where $\underline{V}$ is a matrix made up of the eigenvectors and generalized eigenvector corresponding to the eigenvalues of the third order system.

In this case, the eigenvalues are easily definable from the transfer function as $-a, -b, -b$. The corresponding eigenvectors $\underline{\alpha}_1$, $\underline{\alpha}_2$ and the generalized eigenvector $\underline{g}$ can then be determined from the following equations:

$$(\underline{F} + a\underline{I})\underline{\alpha}_1 = 0 \tag{B-8}$$

$$(\underline{F} + b\underline{I})\underline{\alpha}_2 = 0 \tag{B-9}$$

$$(\underline{F} + b\underline{I})\underline{g} = \underline{\alpha}_2 \tag{B-10}$$

Solving these equations for the eigenvalues gives

$$\underline{v} = \underline{\alpha}_1 \quad \underline{\alpha}_2 \quad \underline{g} = \begin{bmatrix} 1 & 1 & 0 \\ -a & -b & 1 \\ a^2 & b^2 & -2b \end{bmatrix} \tag{B-11}$$

and

$$\underline{v}^{-1} = \begin{bmatrix} b^2 & 2b & 1 \\ (a^2-2ab) & -2b & -1 \\ (a^2b-ab^2) & (a^2-b^2) & (a-b) \end{bmatrix} \frac{1}{(a-b)^2} \tag{B-12}$$

Preforming the transformation shown in equations (B-5) through (B-7) gives

$$\underline{F}^* = \begin{bmatrix} -a & 0 & 0 \\ 0 & -b & 1 \\ 0 & 0 & -b \end{bmatrix} \tag{B-13}$$

$$\underline{G}^* = \begin{bmatrix} \frac{Kab^2}{(a-b)^2} \\ -\frac{Kab^2}{(a-b)^2} \\ \frac{Kab^2}{(a-b)} \end{bmatrix} \tag{B-14}$$

$$\underline{H}^* = \begin{bmatrix} 1 & 1 & 0 \end{bmatrix} \tag{B-15}$$

The transformed vector differential equation in Jordan canonical form is now

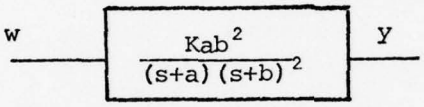
$$\dot{\underline{x}}(t) = \begin{bmatrix} -a & 0 & 0 \\ 0 & -b & 1 \\ 0 & 0 & -b \end{bmatrix} \underline{x}(t) + \begin{bmatrix} \frac{Kab^2}{(a-b)^2} \\ -\frac{Kab^2}{(a-b)^2} \\ \frac{Kab^2}{(a-b)} \end{bmatrix} w(t) \tag{B-16}$$

$$y(t) = \begin{bmatrix} 1 & 1 & 0 \end{bmatrix} \underline{x}(t) \tag{B-17}$$

Appendix C

Gain Calculation for Atmospheric Jitter Model

The atmospheric jitter model used in this study was based on a report written by the Analytic Sciences Corporation (TASC) (Ref 1). The report showed that the jitter could be adequately described as the output of a third order filter driven by zero mean, white, Gaussian noise. This model, as presented in Figure 5, is



(C-1)

where

w = an input white noise process with zero mean and variance equal to 1;

a,b = break frequencies for the process (a=14.14 rad/sec, b=659.5 rad/sec;

y = output of the system;

K = system gain.

Since the strength of the white noise term is specified as 1, the gain, K, must be adjusted to obtain the desired RMS jitter characteristics on the output. In mathematical form, this means that K is adjusted to obtain

$$E\left[y^2(t_i)\right] = \sigma_D^2 \qquad \text{(C-2)}$$

where σ_D is the desired RMS jitter.

The Jordan canonical form of the system in (C-1) was derived in Appendix B to be

$$\dot{\underline{x}}(t) = \begin{bmatrix} -a & 0 & 0 \\ 0 & -b & 1 \\ 0 & 0 & -b \end{bmatrix} \underline{x}(t) + \begin{bmatrix} \frac{Kab^2}{(a-b)^2} \\ -\frac{Kab^2}{(a-b)^2} \\ \frac{Kab^2}{(a-b)} \end{bmatrix} w(t) \qquad \text{(C-3)}$$

$$y(t) = \begin{bmatrix} 1 & 1 & 0 \end{bmatrix} \underline{x}(t) \qquad \text{(C-4)}$$

Using this form of the equation, the desired RMS jitter can now be expressed as

$$\sigma_D^2 = E\left[y^2(t)\right] \qquad \text{(C-5)}$$

$$= E\left[(x_1(t) + x_2(t))^2\right] \qquad \text{(C-6)}$$

$$= E\left[x_1^2(t) + 2x_1(t)x_2(t) + x_2^2(t)\right] \qquad \text{(C-7)}$$

$$= E\left[x_1^2(t)\right] + 2E\left[x_1(t)x_2(t)\right] + E\left[x_2^2(t)\right] \qquad \text{(C-8)}$$

The variance of the desired jitter characteristics is equal to the sum of the variances of the first two states in equation (C-3) plus twice the covariance between the states, all at time t.

The continuous time model for the covariance matrix can be written as

$$\dot{\underline{P}}(t) = \underline{F}\underline{P}(t) + \underline{P}(t)\underline{F}^T + \underline{G}\underline{Q}\underline{G}^T \qquad \text{(C-9)}$$

for this case where

$\underline{P}(t)$ = the covariance matrix

$\underline{F}$ = the system plant matrix

$\underline{G}$ = the system input matrix

Q = the variance of the white noise process driving the system

When the system has reached steady state, the $\dot{\underline{P}}(t)$ matrix will equal zero, and equation (C-9) can be written as

$$\underline{o} = \underline{F}\underline{P}(t) + \underline{P}(t)\underline{F}^T + \underline{G}\underline{Q}\underline{G}^T \tag{C-10}$$

Substituting the appropriate $\underline{F}$, $\underline{G}$, and Q into equation (C-10) yields the equation

$$\underline{o} = \begin{bmatrix} -2aP_{11}+c & -P_{12}(a+b)+P_{13}-c & -P_{13}(a+b)+E \\ -P_{21}(a+b)+P_{13}-c & -2bP_{22}+P_{32}+P_{23}+c & -2bP_{23}+P_{33}-E \\ -P_{31}(a+b)+E & -2bP_{32}+P_{33}-E & -2bP_{33}+D \end{bmatrix} \tag{C-11}$$

where

$$c = K^2a^2b^4/(a-b)^4$$

$$E = K^2a^2b^4/(a-b)^3$$

$$D = K^2a^2b^4/(a-b)^2$$

Making use of the fact that the covariance matrix is always symmetric, equation (C-11) can be solved to give the steady state values of the covariance matrix. Solving for the values of interest yields

$$E\left[x_1(t)\right] = P_{11} = \frac{K^2ab^4}{2(a-b)^4} \tag{C-12}$$

$$E\left[x_1(t)x_2(t)\right] = P_{12} = \frac{K^2a^2b^4}{(a+b)(a-b)^3}\left[\frac{1}{(a+b)} - \frac{1}{(a-b)}\right] \tag{C-13}$$

$$E\left[x_2(t)\right] = P_{22} = \frac{K^2a^2b}{4(a-b)^2} - \frac{K^2a^2b^2}{2(a-b)^3} + \frac{K^2a^2b^3}{2(a-b)^4} \tag{C-14}$$

These values can now be substituted into equation (C-8) along with the true values of a and b to give

$$K \simeq .382109544\ \sigma_D \qquad \text{(C-15)}$$

This equation will result in the correct value for gain given a desired RMS jitter in units of pixels.

Appendix D

Program Listing

This appendix contains the FORTRAN source code for the computer program used in this study. The program was written for implementation on a CDC-6600 computer system which has a 60 bit wordlength.

```
      PROGRAM THESIS(INPUT=/80,OUTPUT,TAPE5=INPUT,TAPE6=OUTPUT,TAPE8)
      COMMON/FLIR/ XFOV,YFOV,IMAX,NPIX,SIGMS,SIGMF,SIGMAB,SIGFLR,RF
      INTEGER ONE
      REAL IMAX
      DIMENSION Z(64),PHI(8,8),Q(3,3),PI(8,8),TA(8,8),
     * WORK(3,3),W(8),TEMP2(8,8),TEMP1(8,8),SAVE(14),
     *QD(8,8),PHIF(4),PFD(4),SQQD(8,8),XS(8),H(64,8),PFP(4,4)
     *,XFP(4),PFM(4,4),XFM(4),HF(64),EXPBA(4,4),RIH(64,1),
     *HT(4,64) ,WKAREA(30,30)
      NAMELIST/INPUTS/ DTAU,SIGS1,SIGM1B,SIGFLR,SN,SOSA,NRUN,TFINAL,
     *SIGF2,SIGMS,SIGMF,TCORR,RF
      READ(5,INPUTS)
      SIGF1=SIGS1
      FTAU1=DTAU
      XFOV=8.
      YFOV=8.
      NPIX=8
      ATAU1=14.14
      ATAU2=659.5
      FTAU2=1./ATAU1
      DT = 1./30.
      IREF=1
C
C        INITIALIZE TRUTH MODEL VARIABLES
C
      CALL RANSET(12345)
      ONE = 1
      NFS = 8
      NMS = NPIX**2
      NFS=4
      NIS = 3
      R=1./RF
      IMAX = SIGMAB*SN
      AGAIN = .351006534 * SIGS1/SOSA
      IFILE=6
      DELT = -1.*DT
      DO 5 I=1,8
      DO 5 J=1,8
      QD(I,J) = 0.
      SQQD(I,J) = 0.
      PHI(I,J) = 0.
    5 CONTINUE
      FACT=(AGAIN**2)*(ATAU1**2)*(ATAU2**4)
      FACT1 = ATAU1-ATAU2
      FACT2 = ATAU1+ATAU2
      FACT3 = 2.*ATAU2
      G1 = FACT/(FACT1**4)
      G2 = FACT/(FACT1**3)
      G3 = FACT/(FACT1**2)
      P1 =  1.- EXP(2.*ATAU1*DELT)
      P2 =  1.- EXP(FACT2*DELT)
      P3 =  1.- EXP(2.*ATAU2*DELT)
      P4 = DT*EXP(DELT*FACT2)
      P5 = DT*EXP(2.*ATAU2*DELT)
      PHI(1,1) = EXP(DELT/DTAU)
      PHI(2,2) = PHI(1,1)
      PHI(3,3) = EXP(ATAU1*DELT)
```

```
      PHI(4,4) = EXP(ATAU2*DELT)
      PHI(4,5) = DELT*PHI(4,4)
      PHI(5,5) = PHI(4,4)
      PHI(6,6) = PHI(3,3)
      PHI(7,7) = PHI(4,4)
      PHI(7,8) = PHI(4,5)
      PHI(8,8) = PHI(5,5)
      WRITE(6,11)
   11 FORMAT(2X,"THE TRUTH MODEL STATE TRANSITION MATRIX IS:"//)
      CALL MOUT(PHI,NPS,NPS)
C
C        FILL THE QD MATRIX WITH VALUES USING EXACT INTEGRATION
C
      QD(1,1) = (SIGS1**2)*(1.-EXP(-2.*DT/DTAU))
      QD(2,2) = QD(1,1)
      QD(3,3) = (G1*R1)/(2.*ATAU1)
      QD(3,4) = R2*(G2/FACT2**2-G1/FACT2)-R4*G2/FACT2
      QD(3,5) = G2*R2/FACT2
      QD(4,3) = QD(3,4)
      QD(4,4) = R3*(G1/FACT3-2.*G2/FACT3**2+2.*G3/FACT3**3)-
     * R5*(G2/ATAU2+G3*DT/FACT3-2.*G3/FACT3**2)
      QD(4,5) = R3*(G3/FACT3**2-G2/FACT3)-R5*G3/FACT3
      QD(5,3) = QD(3,5)
      QD(5,4) = QD(4,5)
      QD(5,5) = R3*G3/FACT3
      DO 20 I=3,5
      DO 20 J=3,5
      QD(I+3,J+3) = QD(I,J)
      Q(I-2,J-2) = QD(I,J)
   20 CONTINUE
      WRITE(6,30)
   30 FORMAT(2X,"THE TRUTH MODEL QD MATRIX IS:"//)
      CALL MOUT( QD,NPS,NPS)
      SQQD(1,1) = SQRT(QD(1,1))
      SQQD(2,2) = SQQD(1,1)
      CALL CHOLESK(Q,WORK,NIS)
      DO 33 I=1,NIS
      DO 33 J=1,NIS
      SQQD(I+2,J+2) = WORK(J,I)
      SQQD(I+5,J+5) = WORK(J,I)
   33 CONTINUE
      WRITE(6,35)
   35 FORMAT(2X,"THE CHOLESKY SQUARE ROOT OF QD IS:"//)
      CALL MOUT(SQQD,NPS,NPS)
C
C        SET UP FILTER TIME INVARIANT MATRICES.
C
      PHIF(1) = EXP(DELT/FTAU1)
      PHIF(2) = PHIF(1)
      PHIF(3) = EXP(DELT/FTAU2)
      PHIF(4) = PHIF(3)
      QFD(1) = (SIGF1**2)*(1.-EXP(2.*DELT/FTAU1))
      QFD(2) = QFD(1)
      QFD(3) = (SIGF2**2)*(1.-EXP(2.*DELT/FTAU2))
      QFD(4) = QFD(3)
      WRITE(6,40)
   40 FORMAT(2X,"THE FILTER STATE TRANSITION MATRIX DIAGONAL TERMS ARE")
```

```
      CALL MOUT(PHIF,NFS,ONE)
      WRITE(6,45)
   45 FORMAT(2X,"THE FILTER OD MATRIX DIAGONAL TERMS ARE:"//)
      CALL MOUT( OFD,NFS,ONE)
C
      PRINT *,"********** BEGIN THE MONTE CARLO SIMULATION ******"
C
      DO 99 L=1,NRUN
      TIME = 0.
      IFLAG = 0
      YNEW=0.
      XNEW=0.
      IUPDAT=0.
C
C        RESET INITIAL CONDITIONS FOR NEW RUN
C
      DO 46 I=1,NPS
      XS(I) = 0.
   46 CONTINUE
      DO 47 I=1,NFS
      DO 47 J=1,NFS
      XFP(I) = 0.
      PFP(I,J) = 0.
      IF(I.EQ.J) PFP(I,J) = .5
   47 CONTINUE
      IF(ICORR.EQ.1)CALL MEAS (XNEW,YNEW,NFS,RA)
   50 TIME = TIME + DT
      IF(TIME.GT.TFINAL) GO TO 99
      IFLAG = IFLAG+1
C
C        PERFORM TRUTH MODEL SIMULATION
C
      CALL NOISE (NPS,W)
      CALL MMPY(TEMP,SQRD,W,NPS,NPS,ONE)
      CALL MMPY(TEMP1,PHI,XS,NPS,NPS,ONE)
      CALL MADD(XS,TEMP,TEMP1,NPS,ONE,ONE)
      XPEAK = XS(1) + XS(3) + XS(4)
      YPEAK = XS(2) + XS(5) + XS(7)
      CALL MEAS(XPEAK,YPEAK,NFS,7)
C
C        PERFORM CORRELATION TRACKING
C
      IF(ICORR.NE.1)GO TO 5000
      IUPDAT=IUPDAT+1
C
C         FILL THE TARGET ARRAY
C
      CALL SHIFTA(Z,TA,NMS,NMS)
      CALL CORR(RA,TA,NPS,NPS,XCORR,YCORR,DX,DY)
      XCENTR=XNEW+XCORR
      YCENTR=YNEW+YCORR
      IF(IUPDAT.NE.IREF)GO TO 5000
      CALL SHIFTA(Z,RA,NMS,NMS)
      XNEW=XCENTR
      YNEW=YCENTR
      IUPDAT=0
      PRINT*,"XCENTR= ",XCENTR," YCENTR= ",YCENTR," XPEAK=",XPEAK,
```

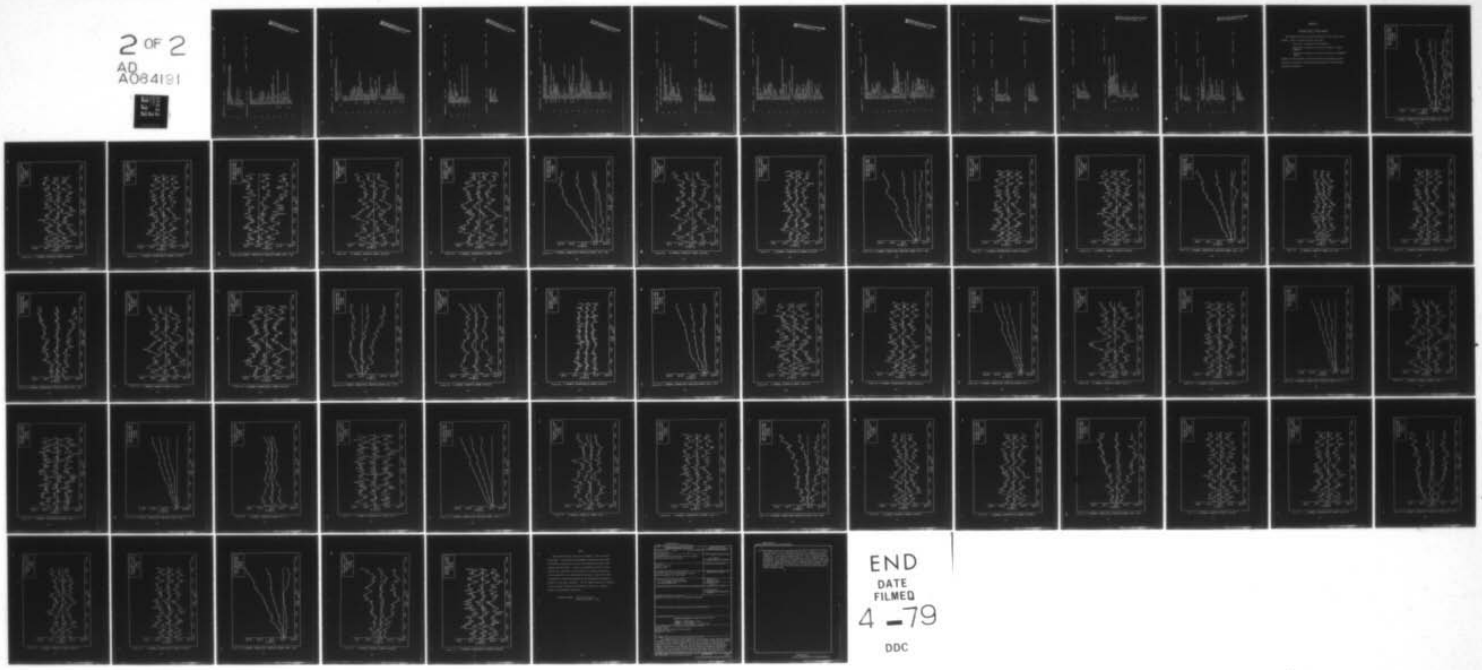
```
     *" YPEAK=",YPEAK," CX=",CX," CY= ",CY
 5000 CONTINUE
C
C        PERFORM FILTER STATE AND COVARIANCE PROPOGATION
C
      DO 55 I=1,NFS
      DO 55 J=1,NFS
      XFM(I) = PHIF(I)*XFP(I)
      PFM(I,J) = PFP(I,J)*PHIF(I)*PHIF(J)
      IF(I.EQ.J) PFM(I,J) = PFM(I,J) + QFD(I)
   55 CONTINUE
C
C        PERFORM MEASUREMENT UPDATE FOR THE FILTER
C
      XPEAK = XFM(1) + XFM(3)
      YPEAK = XFM(2) + XFM(4)
      CALL MEASF(XPEAK,YPEAK,ONE,HF,H)
      DO 60 I=1,NMS
      DO 60 J=1,NFS
      HT(J,I) = H(I,J)
   60 CONTINUE
      CALL MMPY(PFP,HT,H,NFS,NMS,NFS)
      IDGT = 0
      CALL LINV2F(PFM,NFS,NFS,EXTRA,IDGT,WKAREA,IER)
      DO 65 I=1,NFS
      DO 65 J=1,NFS
      PFP(I,J)=PFP(I,J)*R + EXTRA(I,J)
   65 CONTINUE
      IDGT = 0
      CALL LINV2F(PFP,NFS,NFS,EXTRA,IDGT,WKAREA,IER)
      DO 70 I=1,NFS
      DO 70 J=1,NFS
      PFP(I,J) = EXTRA(I,J)
   70 CONTINUE
      CALL MADD(RIH,Z,HF,NMS,ONE,NFS)
      CALL MMPY(EXTRA,HT,RIH,NFS,NMS,ONE)
      CALL MMPY(XFP,PFP,EXTRA,NFS,NFS,ONE)
      DO 75 I=1,NFS
      XFP(I) = XFP(I)*R + XFM(I)
   75 CONTINUE
      SAVE(1) = XS(1)
      SAVE(2) = XS(3) + XS(4)
      SAVE(3) = XS(2)
      SAVE(4) = XS(6) + XS(7)
      SAVE(5) = XFP(1)
      SAVE(6) = XFP(3)
      SAVE(7) = XFP(2)
      SAVE(8) = XFP(4)
      SAVE(9) = XCENTR
      SAVE(10) = YCENTR
      SAVE(11) = PFP(1,1)
      SAVE(12) = PFP(2,2)
      SAVE(13) = PFP(3,3)
      SAVE(14) = PFP(4,4)
      WRITE(8) SAVE
      GO TO 50
   10 CONTINUE

      STOP
      END
```

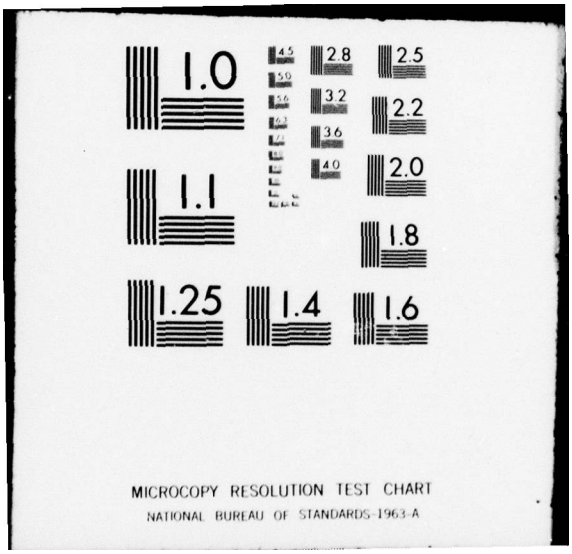
AD-A064 191 AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OHIO SCH--ETC F/G 17/5
AN EXTENDED KALMAN FILTER FOR USE IN A SHARED APERTURE MEDIUM R--ETC(U)
DEC 78 D E MERCIER

UNCLASSIFIED AFIT/GA/EE/78D-3 NL

2 OF 2
AD
A064191



END
DATE FILMED
4 -79
DDC


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0
5.6
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

```
SUBROUTINE CENTR      74/74   OPT=1                    FTN 4.6+446      10/12/78  12.22.22      PAGE     1

      1                 SUBROUTINE CENTR(A,X,J)
                      C     THIS SUBROUTINE COMPUTES THE CENTROID OF THE ARRAY A
                      C     THE OUTPUT VARIABLE IS X, AND THE ARRAY DIMENSION IS J
                        DIMENSION A(J)
      5                 SUM = 0.
                        S = 0.
                        X = 1.
                        DO 100 I=1,J
                        SUM = SUM + FLOAT(I)*A(I)
     10           100 S=S+A(I)
                        IF(S.LE.0.) GO TO 200
                        X=SUM/S
                    200 RETURN
                        END

SUBROUTINE CHOLESK    74/74   OPT=1                    FTN 4.6+446      10/12/78  12.22.22      PAGE     1

      1                 SUBROUTINE CHOLESK(A,S,N)
                        DIMENSION A(1),S(1)
                        NDIM=3
                        NDIM1=4
      5                 TOL=1.E-6
                        MR=0
                        NN=N*NDIM
                        TOL1=0.
                        DO 1 I=1,NN,NDIM1
     10                 R=ABS(A(I))
                      1 IF(R.GT.TOL1)TOL1=R
                        TOL1 = TOL1*1.E-12
                        II=1
                        DO 50 I=1,N
     15                 IM1 = I-1
                        DO 5 JJ=I,NN,NDIM
                      5 S(JJ) = 0.
                        IO = II+IM1
                        P=A(IO)-DOT(IM1,S(II),S(II))
     20                 IF(ABS(P).LT.(TOL*A(IO)+TOL1)) GO TO 50
                        IF(P) 15,50,20
                     15 MR=-1
                        WRITE(6,1000)
                     1000 FORMAT(37H TRIED TO FACTOR AN INDEFINATE MATRIX    )
     25                 RETURN
                     20 S(IO) = SQRT(P)
                        MR=MR+1
                        IF(I.EQ.N) RETURN
                        L=II+NDIM
     30                 DO 2! JJ=L,NN,NDIM
                        IJ=JJ+IM1
                     25 S(IJ) = (A(IJ)-DOT(IM1,S(II),S(IJ)))/S(II)
                     50 II=II+NDIM
                        RETURN
     35                 END
```

```
      SUBROUTINE CORR(RA,TA,I1,J1,XCORR,YCORR,CX,CY)
      DIMENSION RA(I1,J1),TA(I1,J1),R(8),T(8),TARG(8),C(3),RD(15),
     * TD(15),REF(8)
C
C        PERFORM ROW CORRELATION
C
      CX=.95
      CY=.95
    1 COUNT=0.
      SUMX=0.
      N=J1
      K=2**N-1
      EE=1.E-6
      DO 1200 II=1,I1
      DO 1100 JJ=1,J1
      R(JJ) = RA(II,JJ)+EE
      T(JJ) = TA(II,JJ)+EE
 1100 CONTINUE
      CALL RDEN(R,RD,N,K)
      CALL CENTR(R,XR,N)
      CALL LOCATE(R,RD,REF,XR,XTR,T,TD,C,E,N,K)
      IF(C(2).GT.CX) GO TO 2100
      TARG(II) = 1000.
      GO TO 2200
 2100 CONTINUE
      TARG(II) = E
 2200 CONTINUE
 1200 CONTINUE
      DO 3000 I=1,J1
      IF(TARG(I) .EQ.1000.) GO TO 3000
      SUMX=SUMX+TARG(I)
      COUNT=COUNT+1.
 3000 CONTINUE
      IF(COUNT.EQ.0.0) GO TO 3500
      XCORR=SUMX/COUNT
C
C        PERFORM CORRELATION ON COLUMNS
C
    2 COUNT=0.
      SUMY=0.
      N=I1
      K=2**N-1
      DO 4200 JJ=1,J1
      DO 4100 II=1,I1
      R(II) = RA(II,JJ) +EE
      T(II) = TA(II,JJ) +EE
 4100 CONTINUE
      CALL RDEN(R,RD,N,K)
      CALL CENTR(R,XR,N)
      CALL LOCATE(R,RD,REF,XR,XTR,T,TD,C,E,N,K)
      IF(C(2).GT.CY) GO TO 3100
      TARG(JJ) = 1000.
      GO TO 3200
 3100 CONTINUE
      TARG(JJ) =E
 3200 CONTINUE
 4200 CONTINUE
```

```
SUBROUTINE CORR      74/74   OPT=1                    FTN 4.6+446    10/12/78  12.22.22    PAGE     2

                      DO 4000 I=1,I1
                      IF(TARG(I).EQ.1000.) GO TO 4000
60                    SUMY =SUMY + TARG(I)
                      COUNT=COUNT+1.
                 4000 CONTINUE
                      IF(COUNT.EQ.0.0) GO TO 5000
                      YCORR = SUMY/COUNT
65                    YCORR = -YCORR
                      RETURN
                 5000 CONTINUE
                      CX=CX-.05
                      IF(CY.LT.0.5) PRINT*,"********** CX<0.5  **********"
70                    GO TO 1
                 5000 CONTINUE
                      CY=CY-.05
                      IF(CY.LT.0.5) PRINT*,"********** CY<0.5  **********"
                      GO TO 2
75                    END
```

87

```
FUNCTION DOT      74/74   OPT=1                    FTN 4.6+446    10/12/78  12.22.22    PAGE     1

 1                    FUNCTION DOT(NR,A,B)
                      DIMENSION A(1),B(1)
                      DOT=0.
                      DO 1 I=1,NR
 5                  1 DOT = DOT+A(I)*B(I)
                      RETURN
                      END
```

```
      SUBROUTINE LOCATE(R,RD,REF,XR,XT,T,TD,C,E,KK,KD)
      DIMENSION R(KK),RD(KD),T(KK),C(3),TD(KD),REF(KK)
C     CALCULATE THE TARGET CENTROID POSITION
      KT=KK
      KS=KD
      CALL CENTR(T,XT,KT)
C     CALCULATE A ROUGH ESTIMATE OF TARGET POSITION OFFSET
      XTR=XT-XR
C     ROUND OFF ANSWER TO NEAREST INTEGER
      CALL RNDF(XTR,MARK)
C     COMPUTE TARGET DENOMINATOR
      CALL RDEN(T,TD,KT,KS)
      DO 100 JJ=1,3
  100 C(JJ)=0.
      J=0
      IS=MARK-2
      N=KK
C     START MAIN LOOP HERE          **********
  200 CONTINUE
      J=J+1
      IS=IS+1
      IF(IS.GE.8.OR.IS.LE.-8) GO TO 525
C     SHIFT REFERENCE DATA
      CALL SHIFT(R,REF,IS,N)
C     CALCULATE THE SQUARE OF THE CORRELATION COEFFICIENTS
      SUM=0.
      DO 300 JJ=1,N
      SUM=SUM+T(JJ)*REF(JJ)
  300 CONTINUE
      CA=SUM**2
      DENOM=TD(N-IS)*RD(N+IS)
      IF (DENOM.LT.1.E-8)PRINT*,"N=",N,"IS=",IS,"TD=",TD(N-IS),
     *"RD=",RD(N+IS),"SUM=",SUM,"MARK=",MARK
      C(J)=CA/DENOM
      IF(C(J).GT.1.) C(J) = 1.
C     TEST TO SEE IF C(2) HAS BEEN COMPUTED
      IF(C(2).LE.0.)GO TO 200
      IF(C(2).GE.C(1)) GO TO 400
C     MISALIGNMENT, SHIFT LEFT)
      C(3)=C(2)
      C(2)=C(1)
      J=0
      MARK=MARK-1
      IS=IS-3
      GO TO 200
  400 CONTINUE
      IF(C(3).LE.0.) GO TO 200
      IF(C(3).LE.C(2)) GO TO 500
C     MISALIGNMENT SHIFT RIGHT
      C(1)=C(2)
      C(2)=C(3)
      MARK=MARK+1
      J=2
      GO TO 200
  500 CONTINUE
C     END OF MAIN LOOP SECTION
C
```

```
SUBROUTINE LOCATE     74/74   OPT=1                    FTN 4.6+446        10/12/78  12.22.22       PAGE     2


          COMPUTE TARGET-REFERENCE PRECISION OFFSET ESTIMATE
          C
60              DX=0.
                IF(C(1).EQ.0.0.OR.C(2).EQ.0.0.OR.C(3).EQ.0.) GOTO 550
                DN=2.*(1./C(1) +1./C(3) -2./C(2))
                IF(DN.GT.0.) GO TO 800
            525 F=1000.
65              GO TO 700
            550 PRINT*,"WARNING ********** CORRELATION  COEFF = 0. "
                DN=2.*(2.*C(2)-C(1)-C(3))
                DX = (C(3)-C(1))/DN
                E = FLOAT(MARK)+DX
70              GO TO 700
            800 CONTINUE
                DX=(1./C(1) -1./C(3))/DN
                F=FLOAT(MARK)+DX
            700 RETURN
75              END
```

```
SUBROUTINE MADD    74/74   OPT=1                    FTN 4.6+446        10/12/78  12.22.22       PAGE     1

1               SUBROUTINE MADD(C,A,B,J,K,IFLAG)
                DIMENSION A(J,K),B(J,K),C(J,K)
                IF(IFLAG.EQ.1) GO TO 6
                DO 5 N=1,J
5               DO 5 M=1,K
                C(N,M) = A(N,M) - B(N,M)
              5 CONTINUE
                RETURN
              6 DO 10 N=1,J
10              DO 10 M=1,K
                C(N,M) = A(N,M) + B(N,M)
             10 CONTINUE
                RETURN
                END
```

```
      SUBROUTINE MEAS(XPEAK,YPEAK,ISUB,Z)
      COMMON/FLIR/ XFOV,YFOV,IMAX,NPIX,SIGMS,SIGMF,SIGMAB,SIGFLR,RF
      REAL IMAX
      DIMENSION Z(8,8)
      ZMIN = 0.
      SIGMA=SIGMS
      SIG = -1./(2.*SIGMA**2)
      I = (NPIX*ISUB)
      IDIV = ISUB**2
      XINCR = XFOV/FLOAT(I)
      YINCR = YFOV/FLOAT(I)
      X = -1.*XFOV/2. + XINCR/2.
      Y = YFOV/2. - YINCR/2.
      X0 = X
      DO 20 K=1,NPIX
      DO 15 J=1,NPIX
      TOTAL = 0.
      XN = X
      YN = Y
      DO 10 N=1,ISUB
      DO 5 M =1,ISUB
      TOTAL = TOTAL+EXP(SIG*((XN-XPEAK)**2+(YN-YPEAK)**2))*IMAX
      XN = X +FLOAT(M)*XINCR
    5 CONTINUE
      YN = Y- FLOAT(N)*YINCR
      XN = X
   10 CONTINUE
      Z(K,J) = TOTAL/FLOAT(IDIV)
C
C        ADD BACKGROUND AND FLIR NOISE BOTH ZERO MEAN
C
      GAUSS=0.
      DO 100 LL=1,12
      GAUSS=GAUSS+RANF(DUM)
  100 CONTINUE
      B=(GAUSS-6.)*SIGMAB
      GAUSS=0.
      DO 110 LL=1,12
      GAUSS=GAUSS+RANF(DUM1)
  110 CONTINUE
      F=(GAUSS-6.)*SIGFLR
      Z(K,J) = Z(K,J) + B + F
      IF(Z(K,J).LT.ZMIN) ZMIN=Z(K,J)
      X = XN +FLOAT(ISUB)*XINCR
   15 CONTINUE
      Y = Y - FLOAT(ISUB)*YINCR
      X=X0
   20 CONTINUE
      IF(ZMIN.EQ.0.)RETURN
      DO 25 I=1,NPIX
      DO 25 J=1,NPIX
      Z(I,J) = Z(I,J)-ZMIN+.1
   25 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE MEASE(XPEAK,YPEAK,ISUB,Z,M)
      COMMON/FLIR/ XFOV,YFOV,IMAX,NPIX,SIGMS,SIGMF,SIGMAB,SIGFLR,RF
      DIMENSION Z(A,A),M(54,4)
      REAL IMAX
      ZMIN = 0.
      SIGMA=SIGMF
      SIG = -1./(2.*SIGMA**2)
      I = (NPIX*ISUB)
      IDIV = ISUB**2
      XINCR = XFOV/FLOAT(I)
      YINCR = YFOV/FLOAT(I)
      X = -1.*XFOV/2. + XINCR/2.
      Y = YFOV/2. - YINCR/2.
      X0 = X
      DO 20 K=1,NPIX
      NUM = K
      DO 15 J=1,NPIX
      TOTAL = 0.
      SUM1=0.
      SUM2=0.
      XN = X
      YN = Y
      DO 10 N=1,ISUB
      DO 5 M =1,ISUB
      PART = EXP(SIG*((XN-XPEAK)**2+(YN-YPEAK)**2))*IMAX
      TOTAL = TOTAL+PART
      SUM1 = SUM1 + PART*((XN-XPEAK)/SIGMA**2)
      SUM2 = SUM2 + PART*((YN-YPEAK)/SIGMA**2)
      XN = X +FLOAT(M)*XINCR
    5 CONTINUE
      YN = Y-FLOAT(N) * YINCR
      XN = X
   10 CONTINUE
      Z(K,J) = TOTAL/FLOAT(IDIV)
C
C        ADD NOISE TERM TO THE MEASUREMENTS
C
      GAUSS=0.
      DO 100 LL=1,12
      GAUSS=GAUSS + RANF(DUM3)
  100 CONTINUE
      V=(GAUSS-6.) * SQRT(RF)
      Z(K,J) = Z(K,J) + V
      IF(Z(K,J).LT.ZMIN)ZMIN=Z(K,J)
      M(NUM,1) = SUM1/FLOAT(IDIV)
      M(NUM,2) = SUM2/FLOAT(IDIV)
      M(NUM,3) = M(NUM,1)
      M(NUM,4) = M(NUM,2)
      X = XN +FLOAT(ISUB)*XINCR
      NUM = NUM + NPIX
   15 CONTINUE
      Y = Y - FLOAT(ISUB)*YINCR
      X=X0
   20 CONTINUE
      IF(ZMIN.EQ.0.)RETURN
      DO 25 I=1,NPIX
      DO 25 J=1,NPIX
```

```
SUBROUTINE MESSF      74/74   OPT=1                    FTN 4.6+446      10/12/78  12.22.22      PAGE     2

                    Z(I,J) = Z(I,J)-ZMIN+.1
                 25 CONTINUE
60                  RETURN
                    END
```

```
SUBROUTINE MMPY       74/74   OPT=1                    FTN 4.6+446      10/12/78  12.22.22      PAGE     1

1                   SUBROUTINE MMPY(C,A,B,K,M,N)
                    DIMENSION C(K,N),A(K,M),B(M,N)
                    DO 1 I=1,K
                    DO 1 J=1,N
5                   C(I,J)=0.
                  1 CONTINUE
                    DO 5 L=1,K
                    DO 5 J=1,N
                    DO 5 I=1,M
10                  C(L,J) = C(L,J) + (A(L,I)*B(I,J))
                  5 CONTINUE
                    RETURN
                    END
```

```
SUBROUTINE MOUT       74/74   OPT=1                    FTN 4.6+446      10/12/78  12.22.22      PAGE     1

1                   SUBROUTINE MOUT(A,NR,NC)
                    DIMENSION A(NR,NC)
                    DO 10 I=1,NR
                    WRITE(6,5) (A(I,J),J=1,NC)
5                 5 FORMAT(2X,9(F13.7,3X))
                 10 CONTINUE
                    RETURN
                    END
```

```
SUBROUTINE NOISE      74/74   OPT=1                    FTN 4.6+446      10/12/78  12.22.22      PAGE     1

1               SUBROUTINE NOISE(N,W)
                DIMENSION W(N)
                DO 15 J=1,N
                TOTAL=0.
5               DO 5 I=1,12
                TOTAL =TOTAL + RANF(DUM)
              5 CONTINUE
                W(J) = TOTAL - 6.
             15 CONTINUE
10              RETURN
                END
```

```
SUBROUTINE RDEN       74/74   OPT=1                    FTN 4.6+446      10/12/78  12.22.22      PAGE     1

1               SUBROUTINE RDEN(R,D,N,K)
          C        THIS SUBROUTINE COMPUTES THE CORRELATION DENOMINATORS
          C        FOR THE REFERENCE FUNCTION.
          C        R= INPUT REFERENCE ARRAY
5         C        D = OUTPUT DENOMINATOR ARRAY
          C        N IS THE INPUT ARRAY DIMENSION AND ALSO THE OUTPUT ARRAY
          C        MARKER POSITION.
          C        K IS THE OUTPUT ARRAY DIMENSION
                DIMENSION R(N),D(K)
10              SUM = 0.
                J=1
                JJ=K
            200 SUM = SUM+R(J)**2
                D(JJ) = SUM
15              IF (JJ.EQ.J )GO TO 300
                JJ=JJ-1
                J=J+1
                GO TO 200
            300 J=K
20          400 JJ=JJ-1
                D(JJ) = SUM-D(J)
                J=J-1
                IF(JJ.NE.1) GO TO 400
                RETURN
25              END
```

```
SUBROUTINE RNDF       74/74   OPT=1                    FTN 4.6+446        10/12/78  12.22.22        PAGE     1

1                SUBROUTINE RNDF(A,K)
               C    SUBROUTINE TO ROUNDOFF A,TO NEAREST INTEGER,K
               C
                 K=IFIX(A)
5                R=A-K
                 IF(R.LT.0.5)GO TO 100
                 K=K+1
               100 RETURN
                 END
```

```
SUBROUTINE SHIFT      74/74   OPT=1                    FTN 4.6+446        10/12/78  12.22.22        PAGE     1

1                SUBROUTINE SHIFT(A,B,IS,N)
                 DIMENSION A(N),B(N)
               C    A=INPUT ARRAY,B=OUTPUT ARRAYN=ARRAY SIZE,IS=AM,T OF SHIFT
               C
5                NN=N
                 DO 300 I=1,N
               300 B(I)=0.
               C    TEST FOR LEFT OR RIGHT SHIFT
                 IF(IS.LE.0) GO TO 100
10             C    EXECUTE RIGHT SHIFT
                 MM=1+IS
                 M=N-IS
                 CALL SHIFTA(A(1),B(MM),M,NN)
                 GO TO 200
15             C    EXECUTE LEFT SHIFT
               100 MM=1-IS
                 M=N+IS
                 CALL SHIFTA(A(MM),B(1),M,NN)
               200 RETURN
20               END
```

```
SUBROUTINE SHIFTA     74/74   OPT=1                    FTN 4.6+446        10/12/78  12.22.22        PAGE     1

1                SUBROUTINE SHIFTA(A,B,M,N)
                 DIMENSION A(N),B(N)
                 DO 100 K=1,M
               100 B(K) = A(K)
5                RETURN
                 END
```

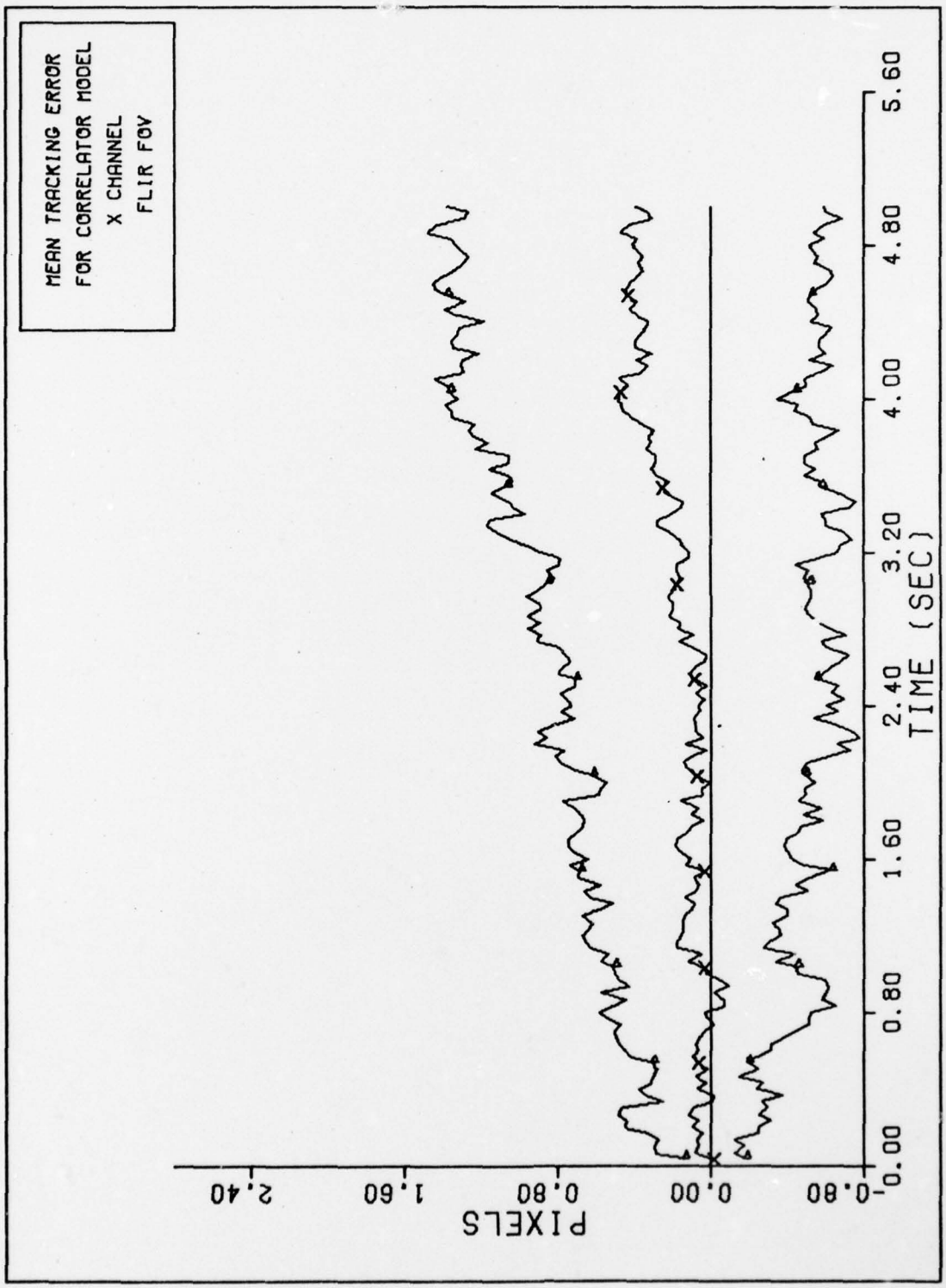
Appendix E

Plotted Outputs of Cases Studied

This appendix contains the plotted outputs for the sixteen cases studied. There are three plots for each case:

- mean error ±1 sigma for the correlator
- mean error ±1 sigma for the filter estimate of target dynamics
- mean error ±1 sigma for the filter estimate of atmospheric jitter

Because of the similarity in results between the horizontal (x) and vertical (y) channels, only the mean error plots for the horizontal channel are presented.



X CHANNEL CORRELATOR TRACKING ERROR (S/N = 20)

Figure 20a.

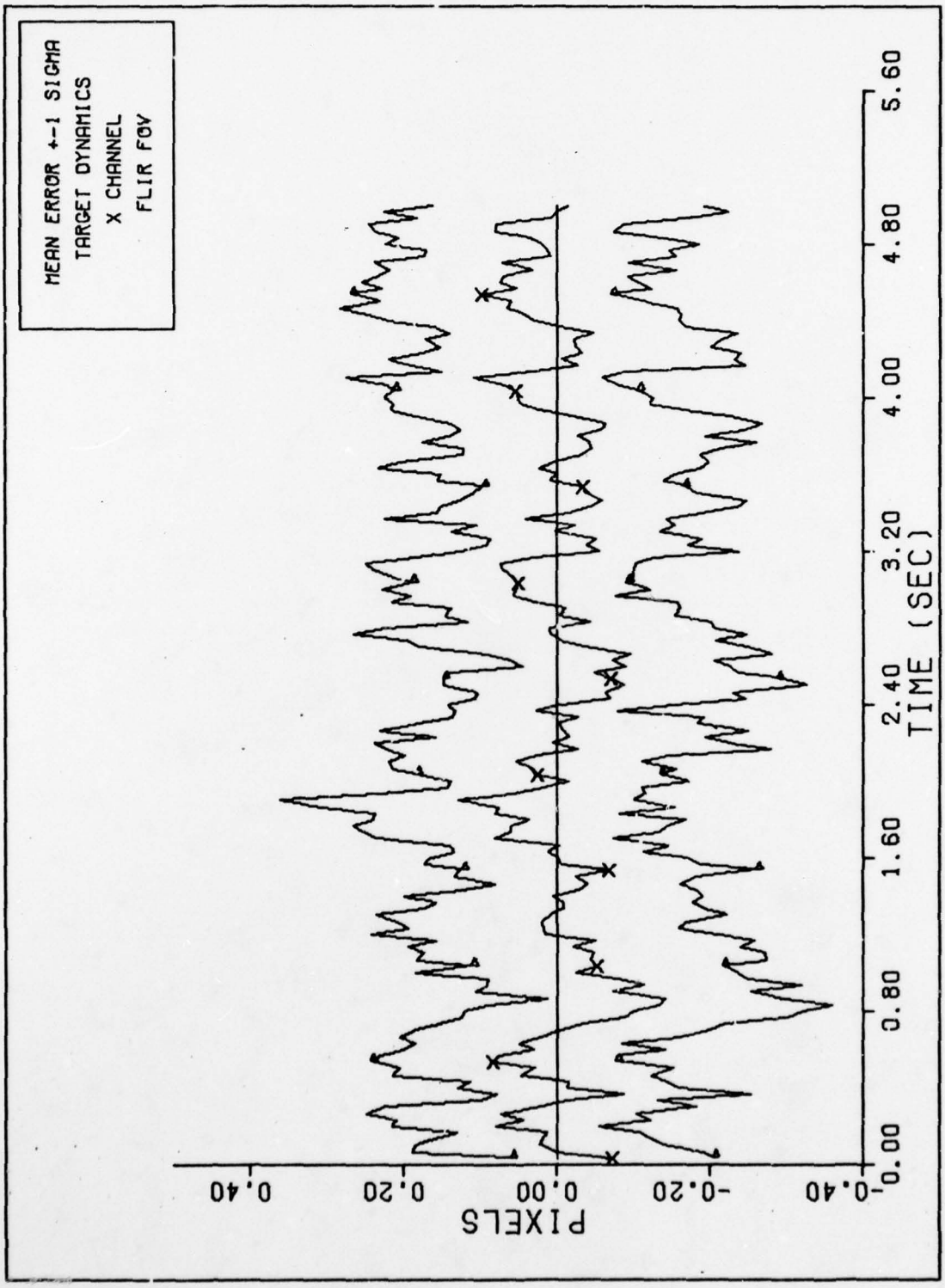


Figure 20b. X CHANNEL DYNAMICS ERROR (S/N=20)

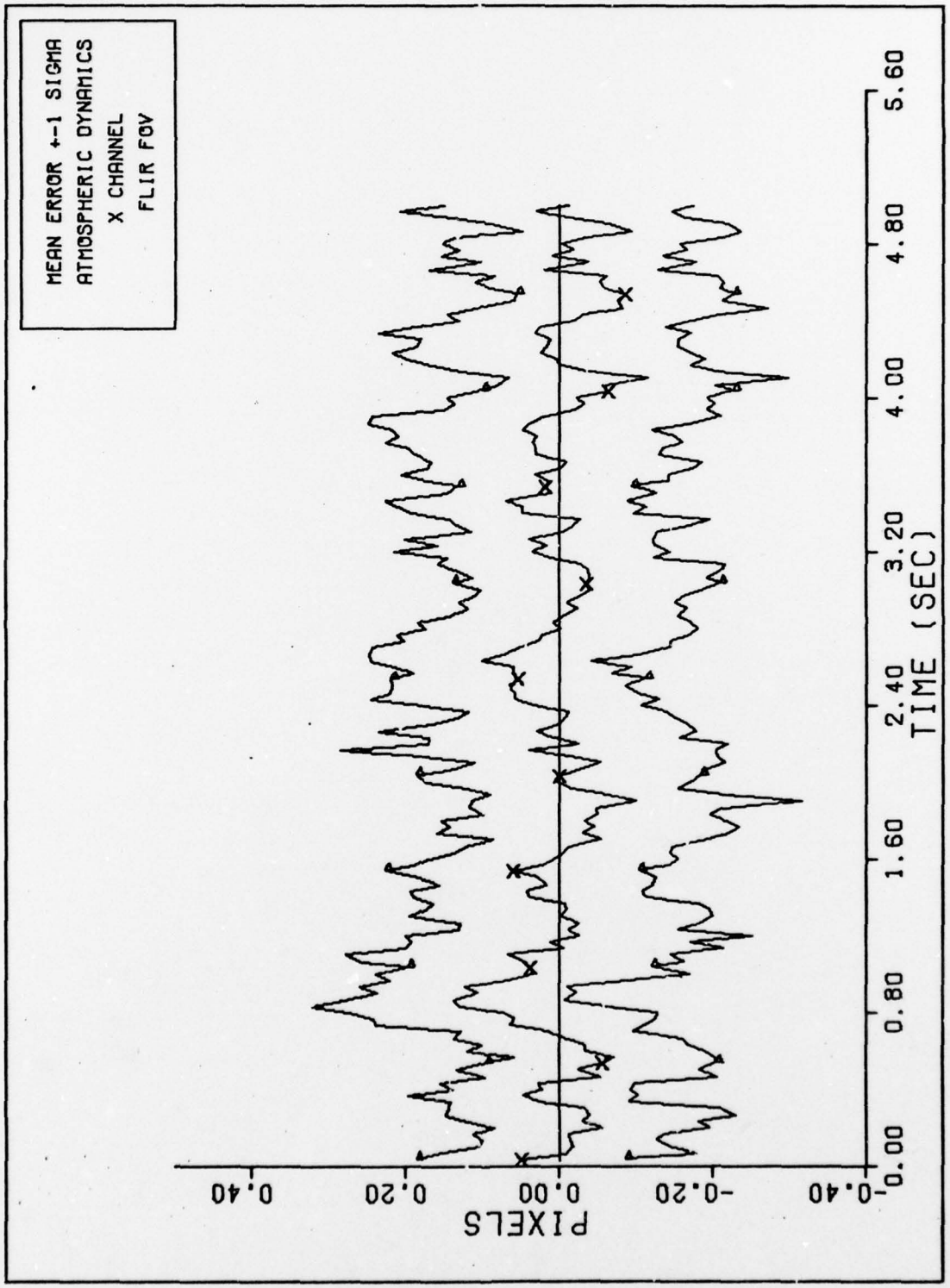


Figure 20c. X CHANNEL ATMOSPHERICS ERROR (S/N=20)

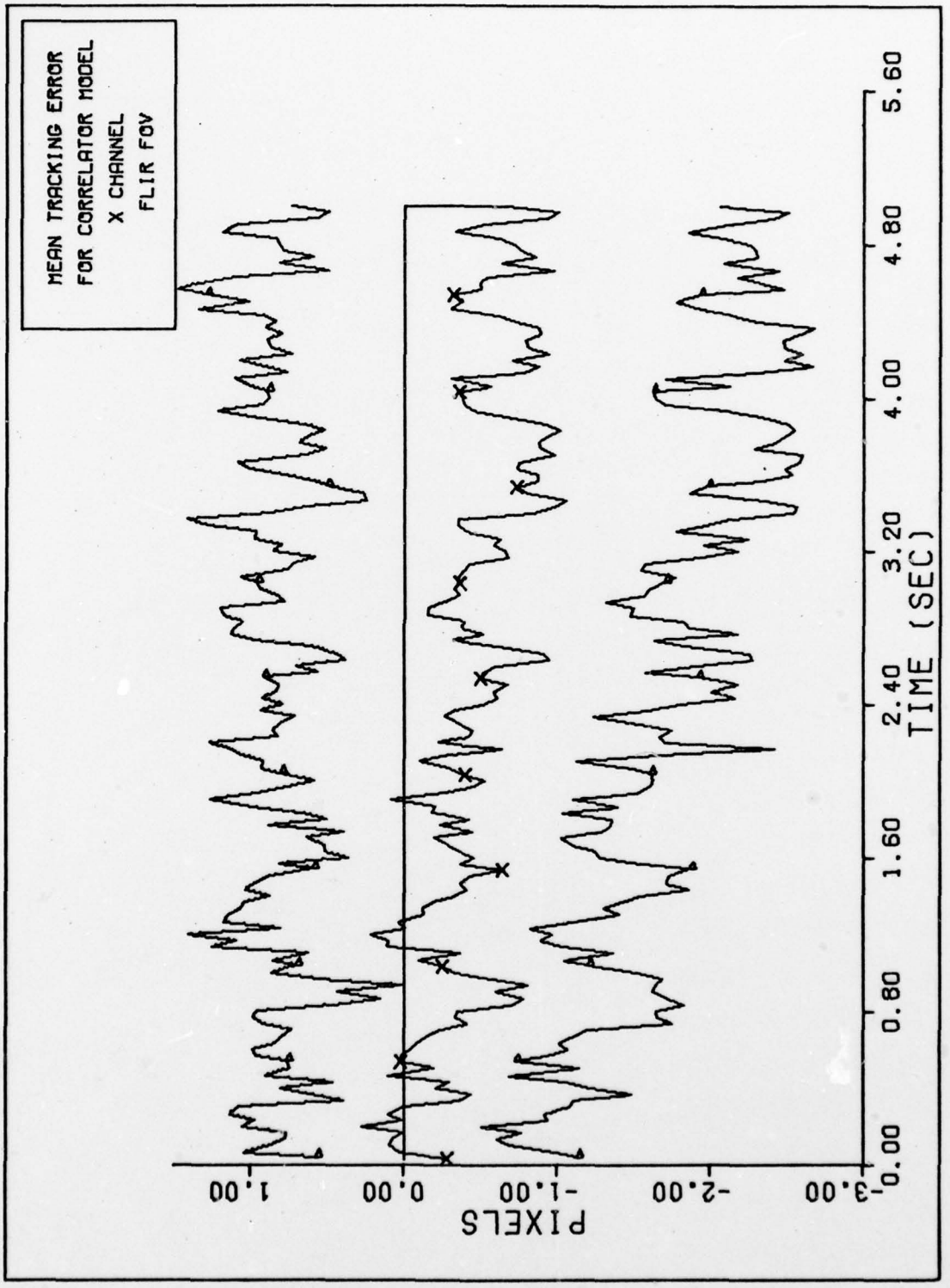


Figure 21a X CHANNEL CORRELATOR TRACKING ERROR (S/N = 20)

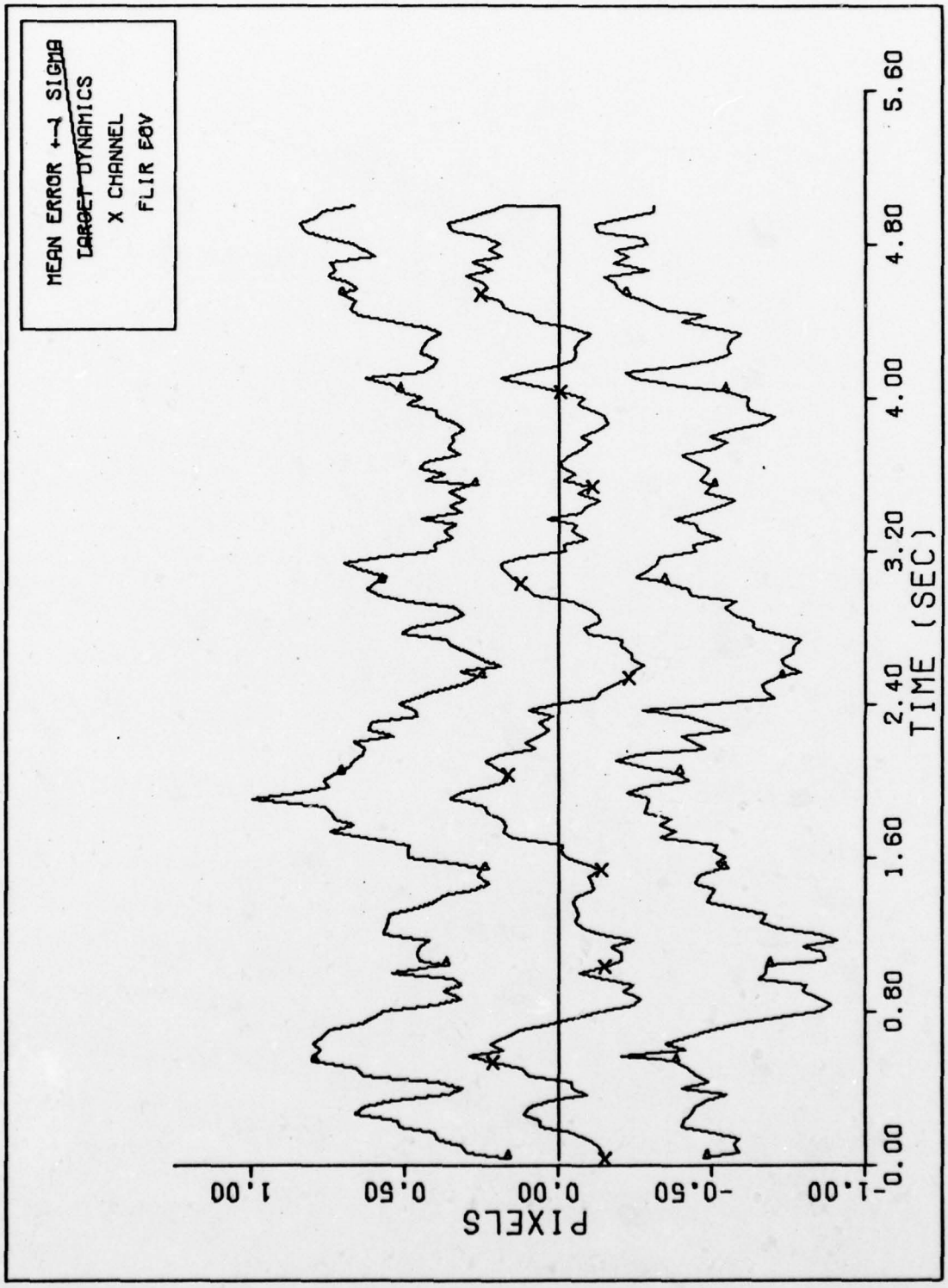


Figure 21b. X CHANNEL DYNAMICS ERROR (S/N=20)

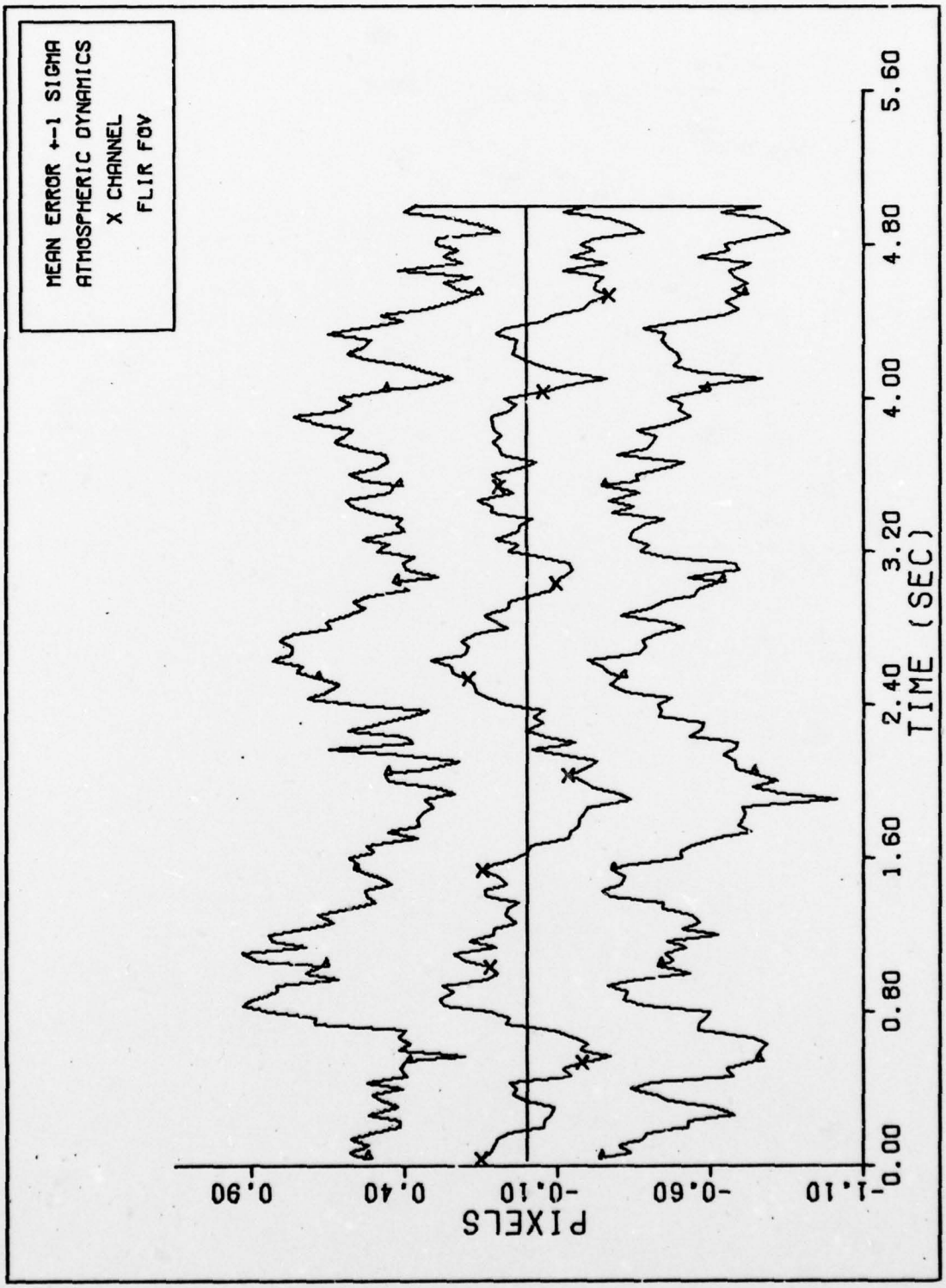


Figure 21c. X CHANNEL ATMOSPHERICS ERROR (S/N=20)

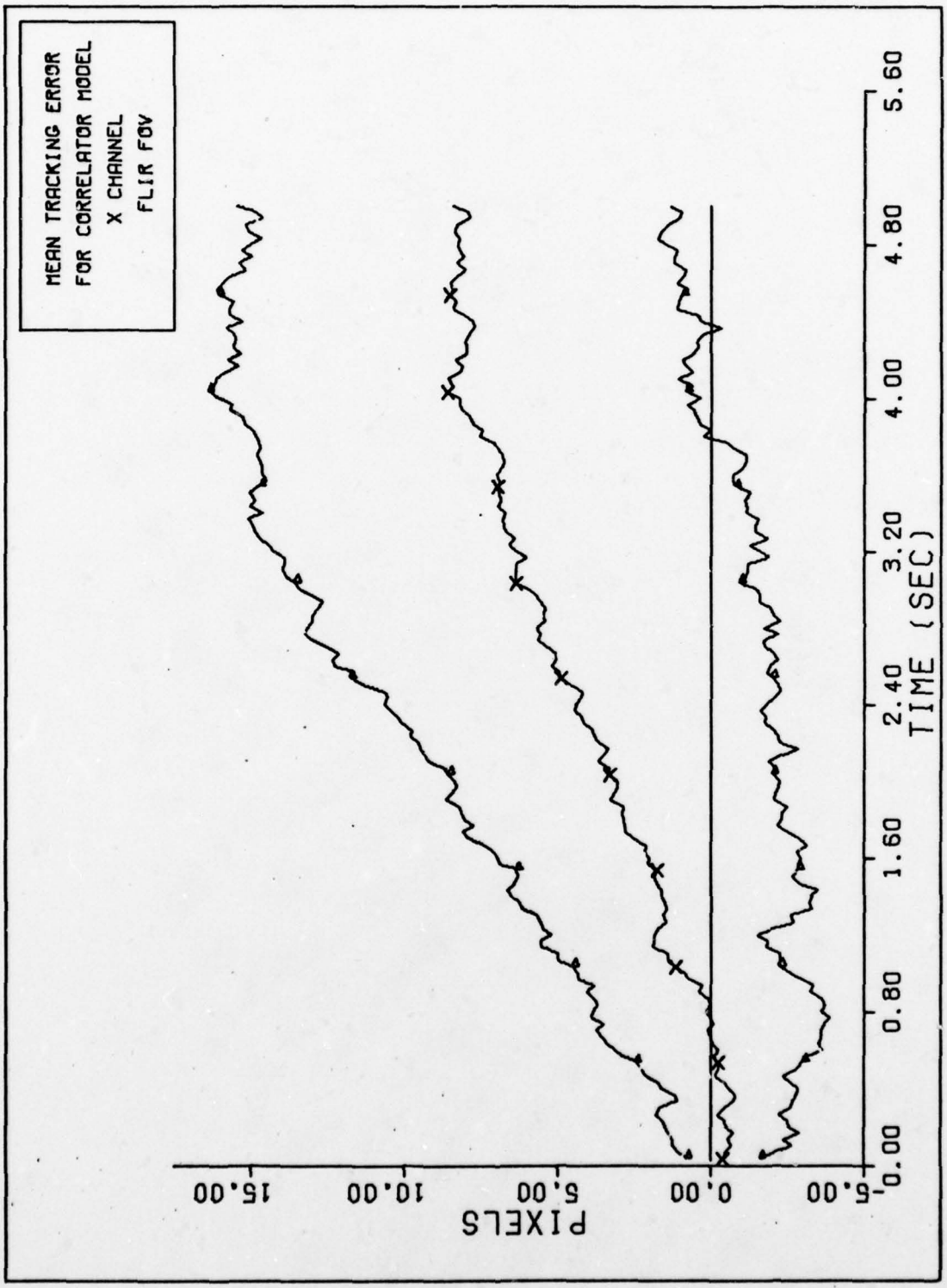


Figure 22a. X CHANNEL CORRELATOR TRACKING ERROR (S/N = 20)

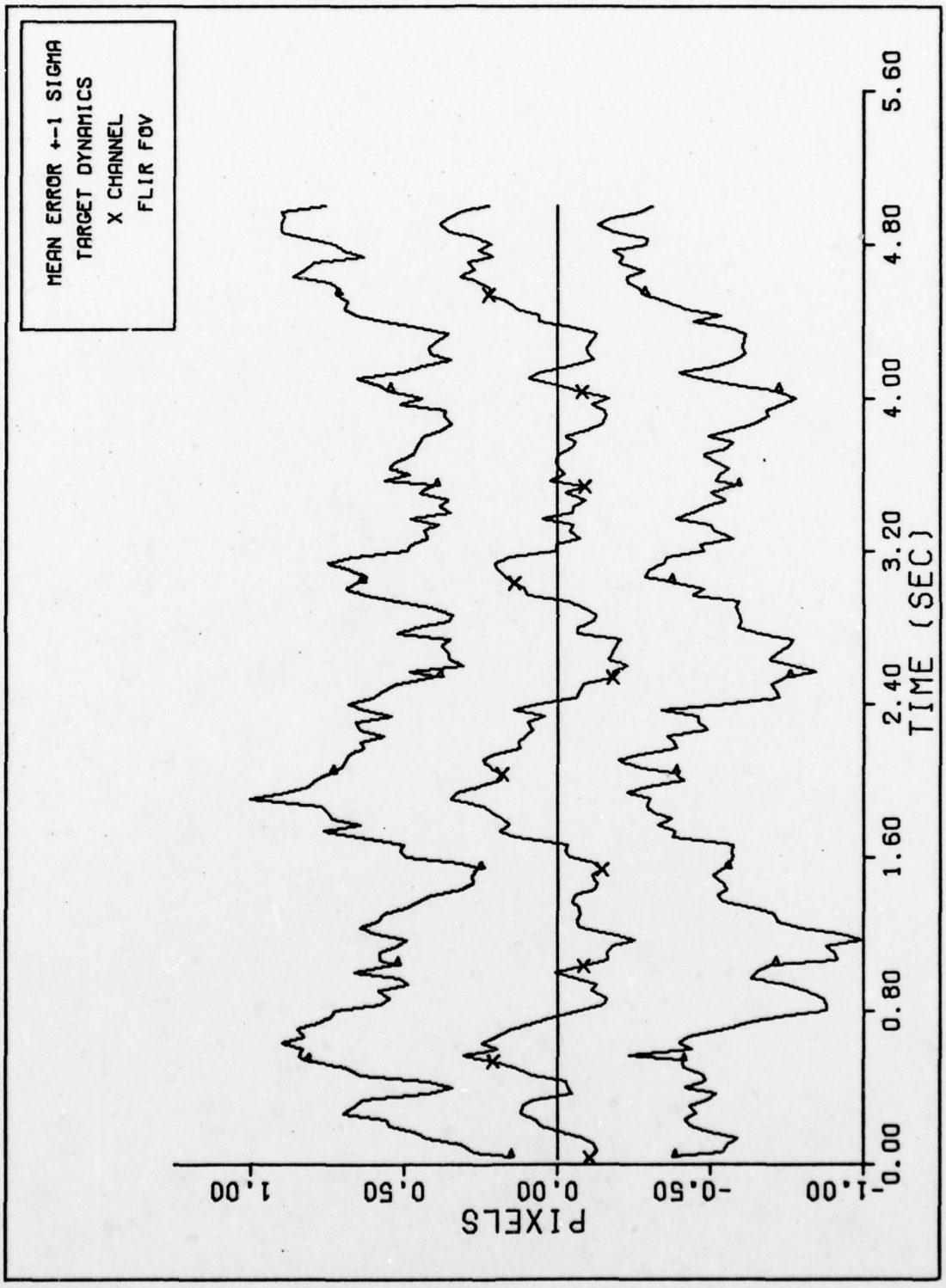


Figure 22b. X CHANNEL DYNAMICS ERROR (S/N=20)

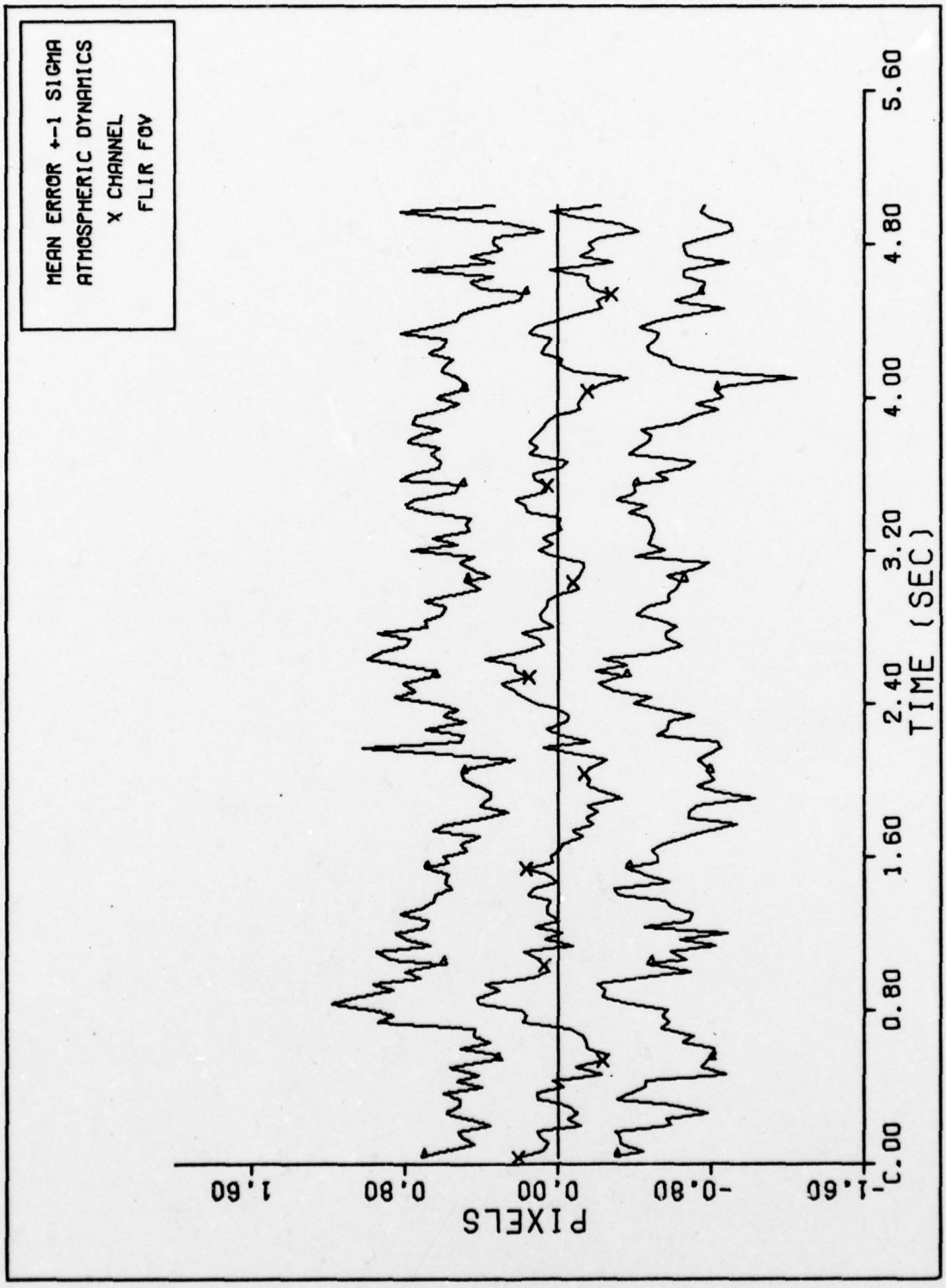


Figure 22c. X CHANNEL ATMOSPHERICS ERROR (S/N=20)

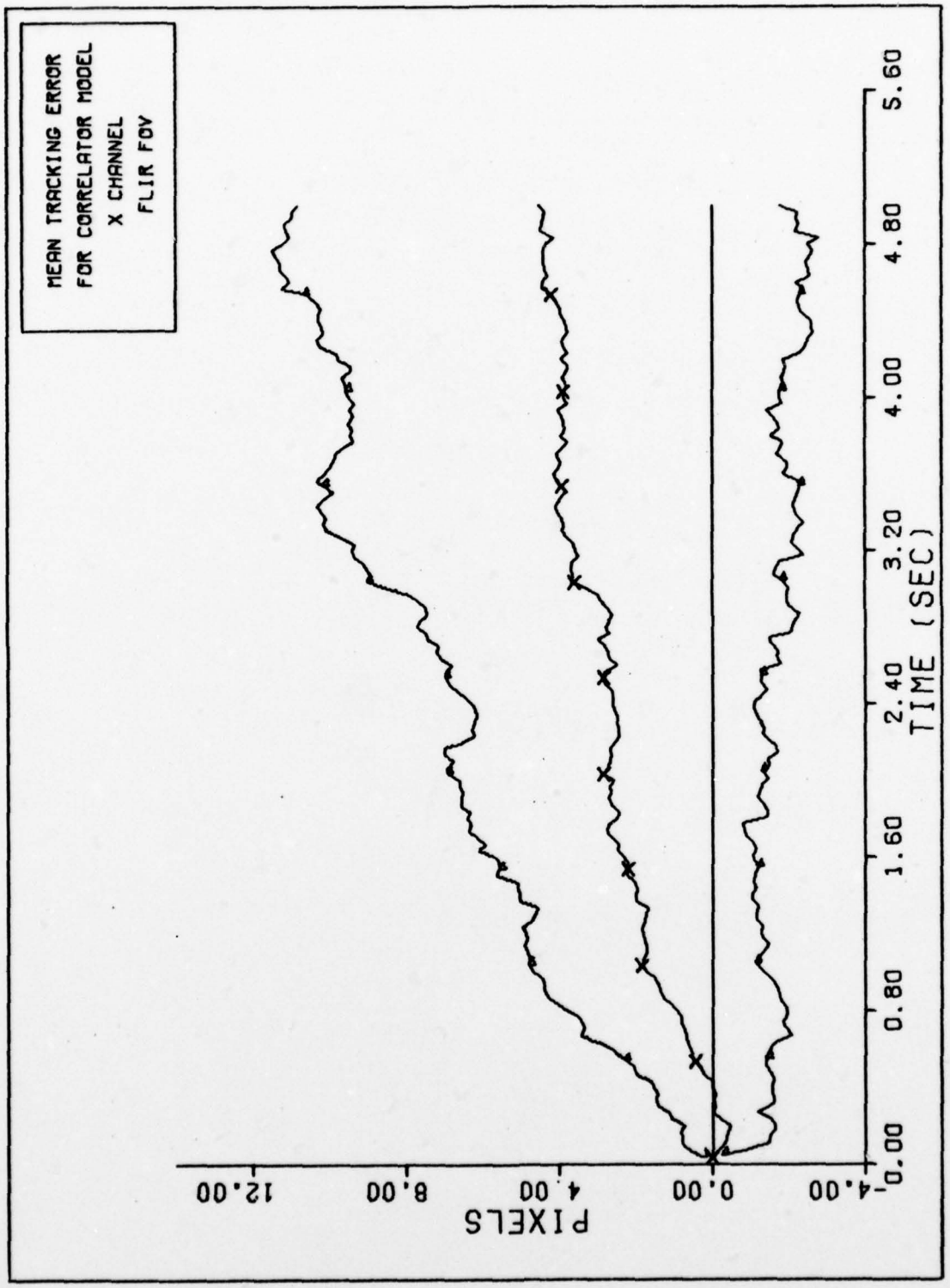


Figure 23a. X CHANNEL CORRELATOR TRACKING ERROR (S/N = 20)

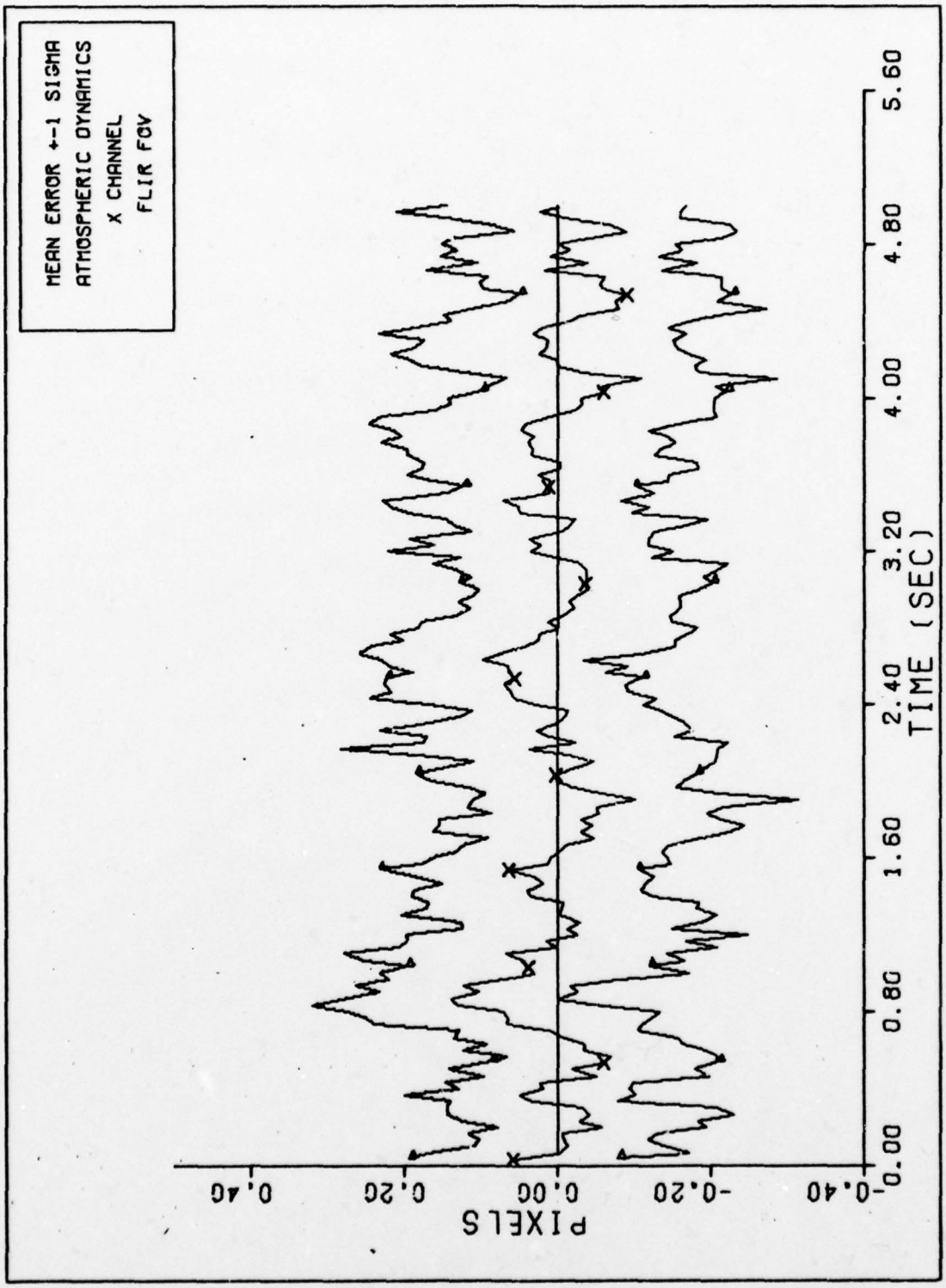


Figure 23b. X CHANNEL ATMOSPHERICS ERROR (S/N=20)

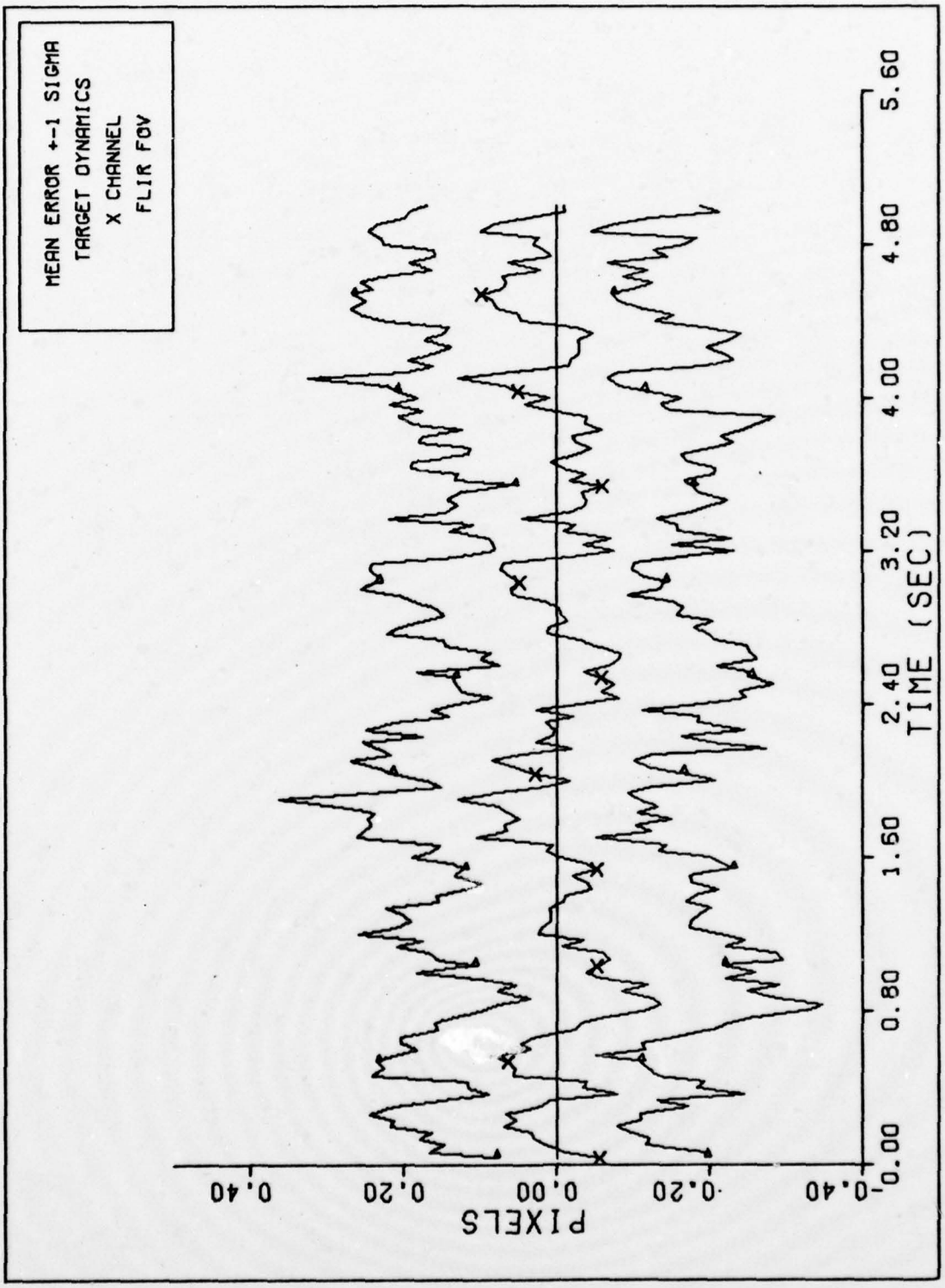


Figure 23c. X CHANNEL DYNAMICS ERROR (S/N=20)

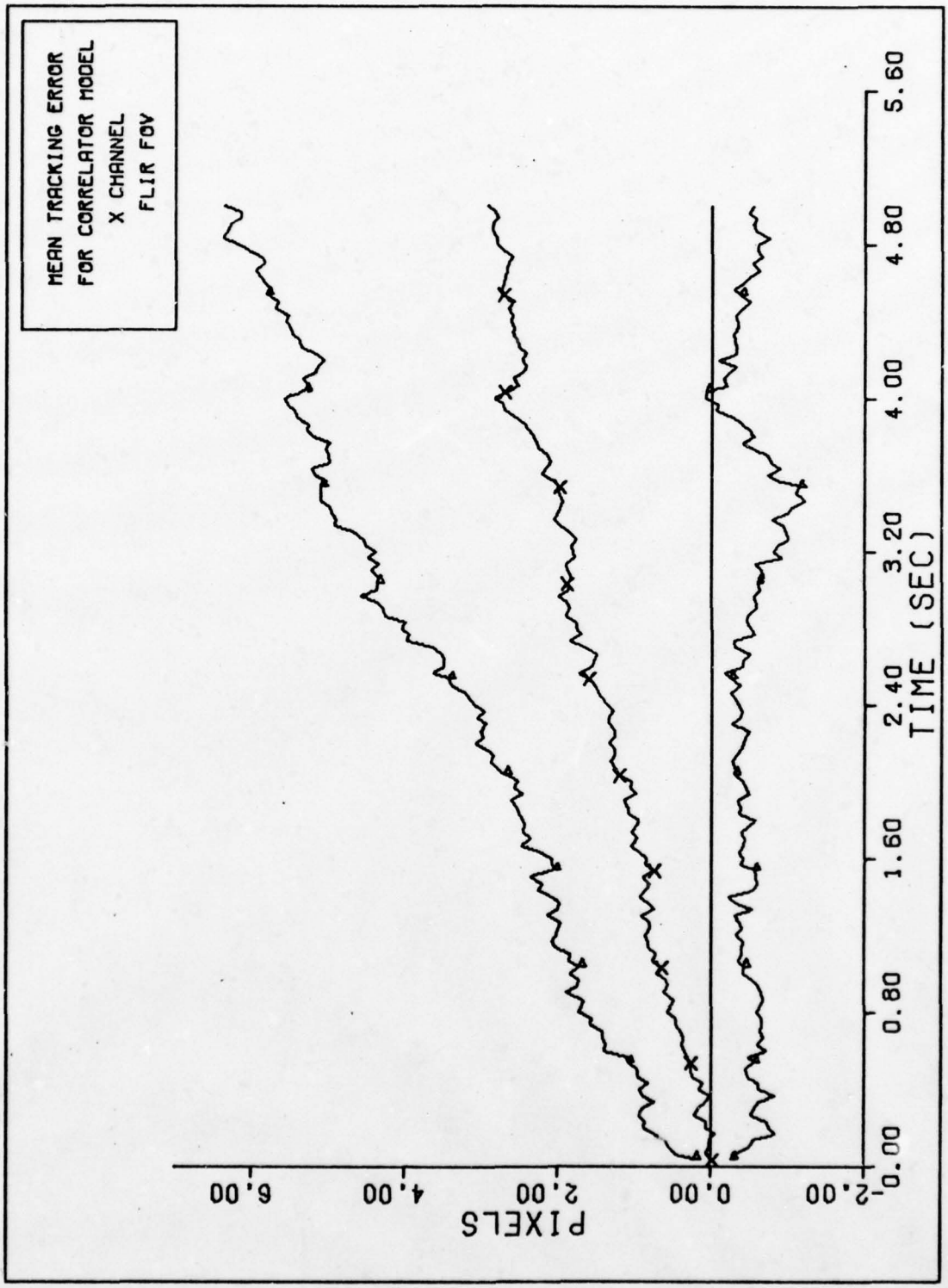


Figure 24a. X CHANNEL CORRELATOR TRACKING ERROR (S/N = 10)

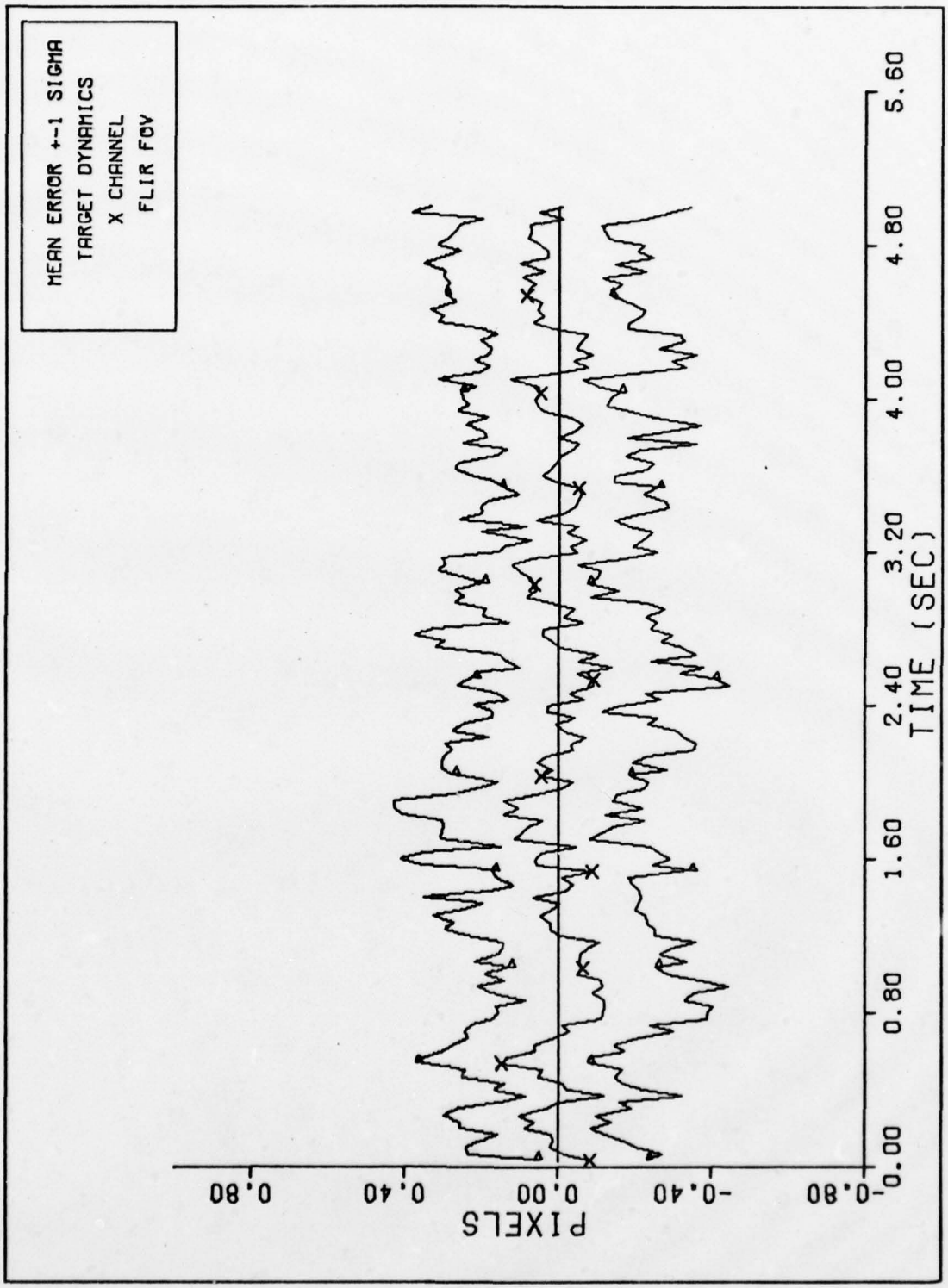


Figure 24b. X CHANNEL DYNAMICS ERROR (S/N=10)

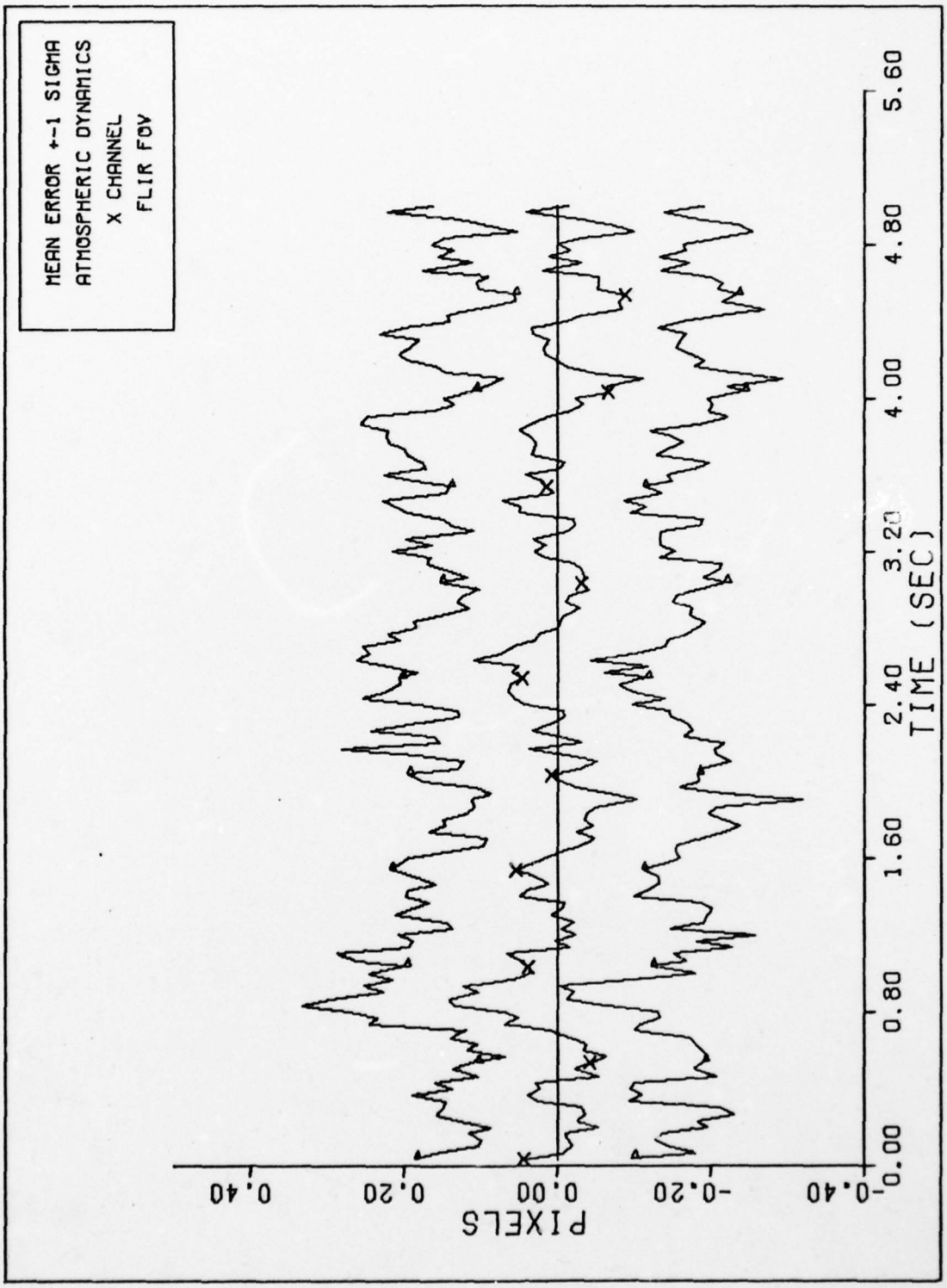


Figure 24c. X CHANNEL ATMOSPHERICS ERROR (S/N=10)

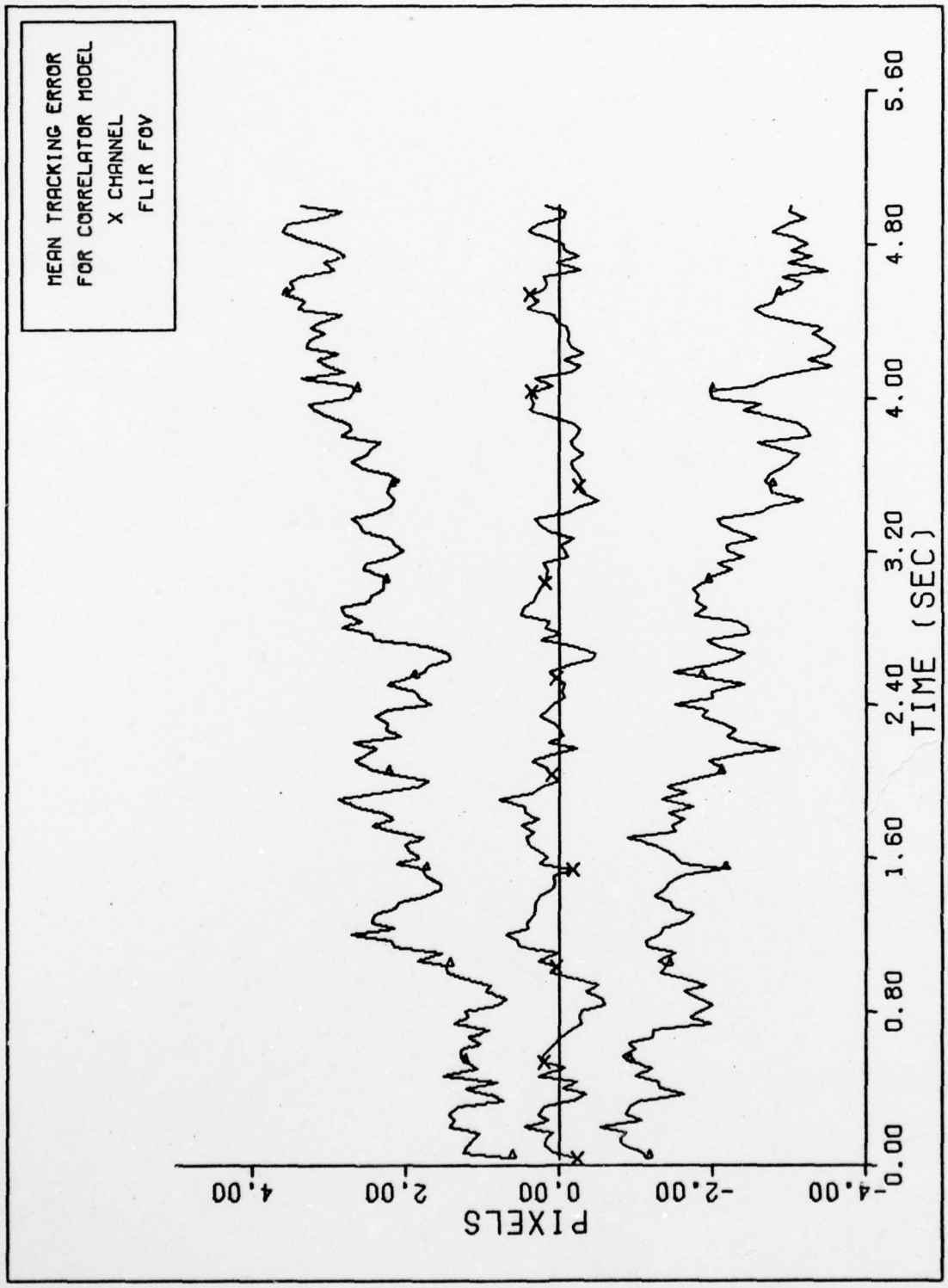


Figure 25a. X CHANNEL CORRELATOR TRACKING ERROR (S/N = 10)

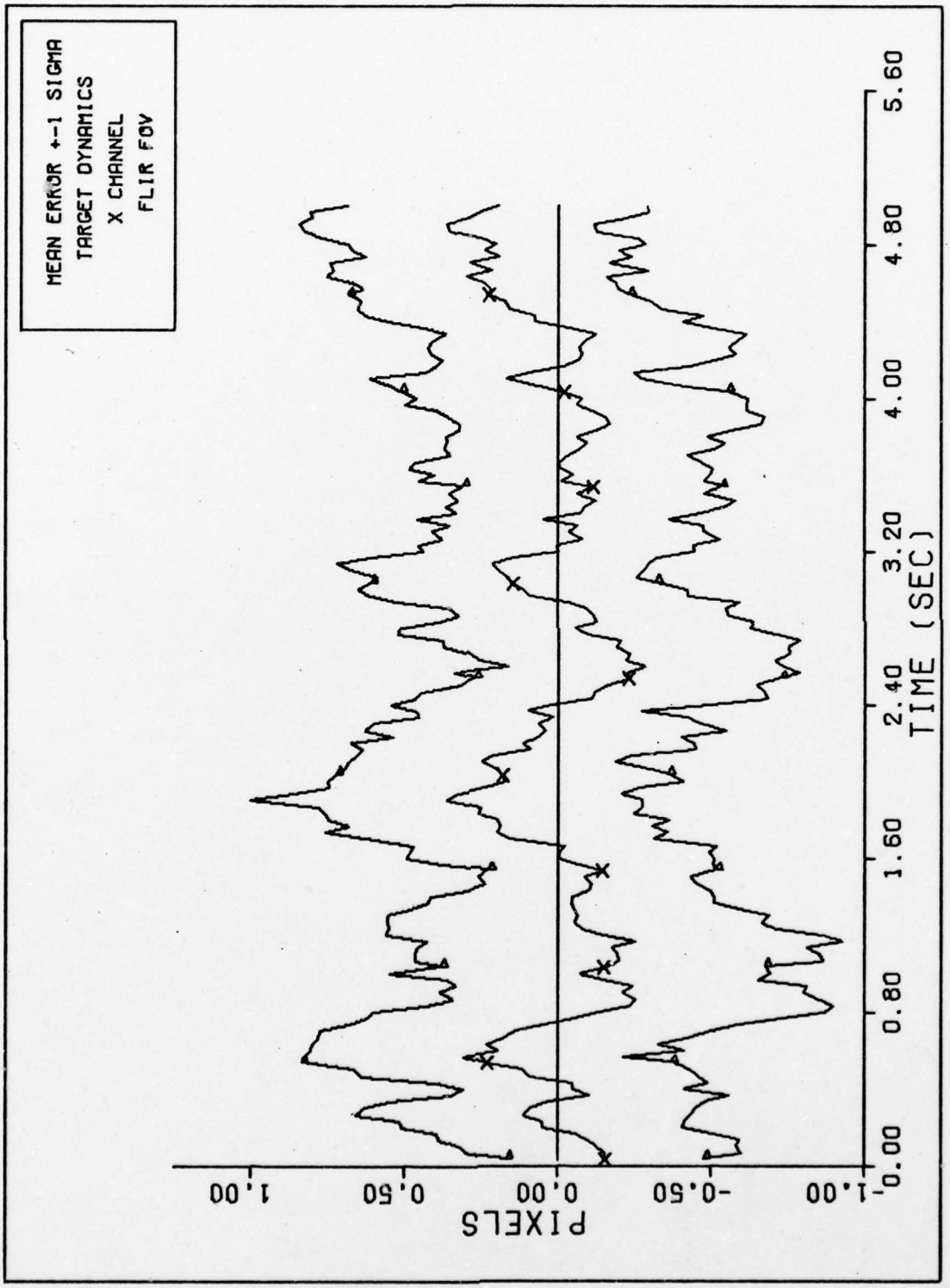


Figure 25b. X CHANNEL DYNAMICS ERROR (S/N=10)

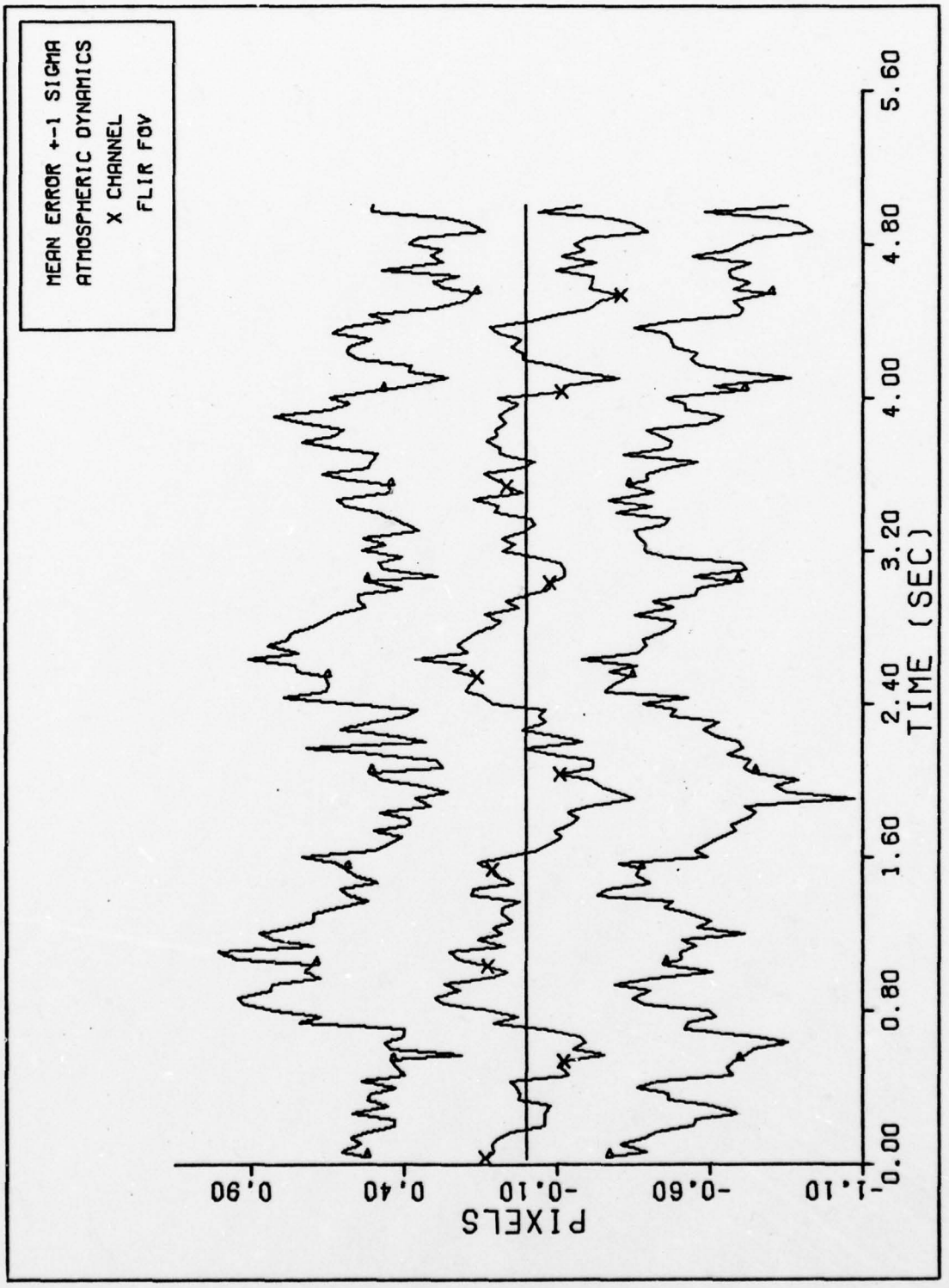


Figure 25c. X CHANNEL ATMOSPHERICS ERROR (S/N=10)

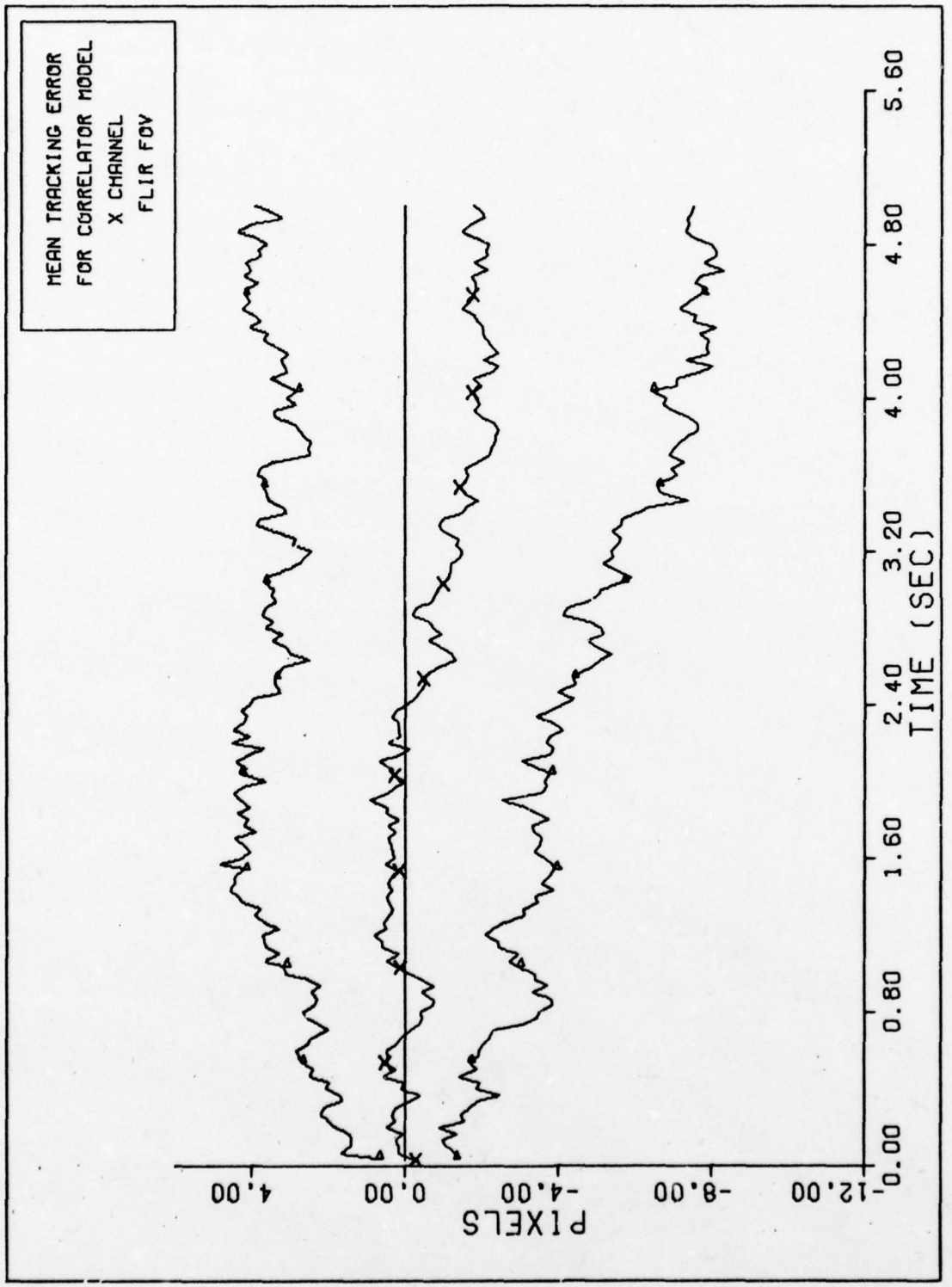


Figure 26a. X CHANNEL CORRELATOR TRACKING ERROR (S/N = 10)

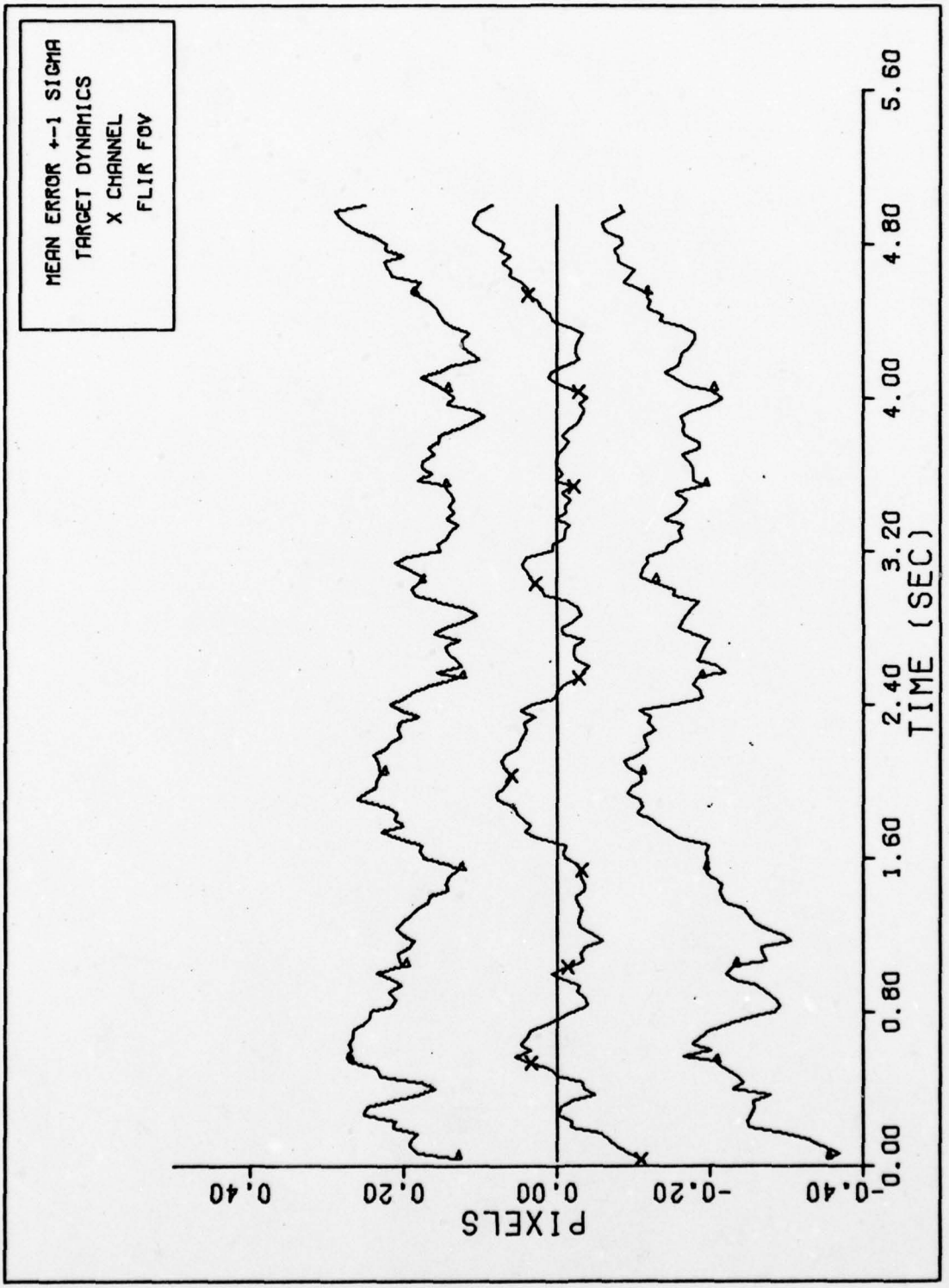


Figure 26b. X CHANNEL DYNAMICS ERROR (S/N=10)

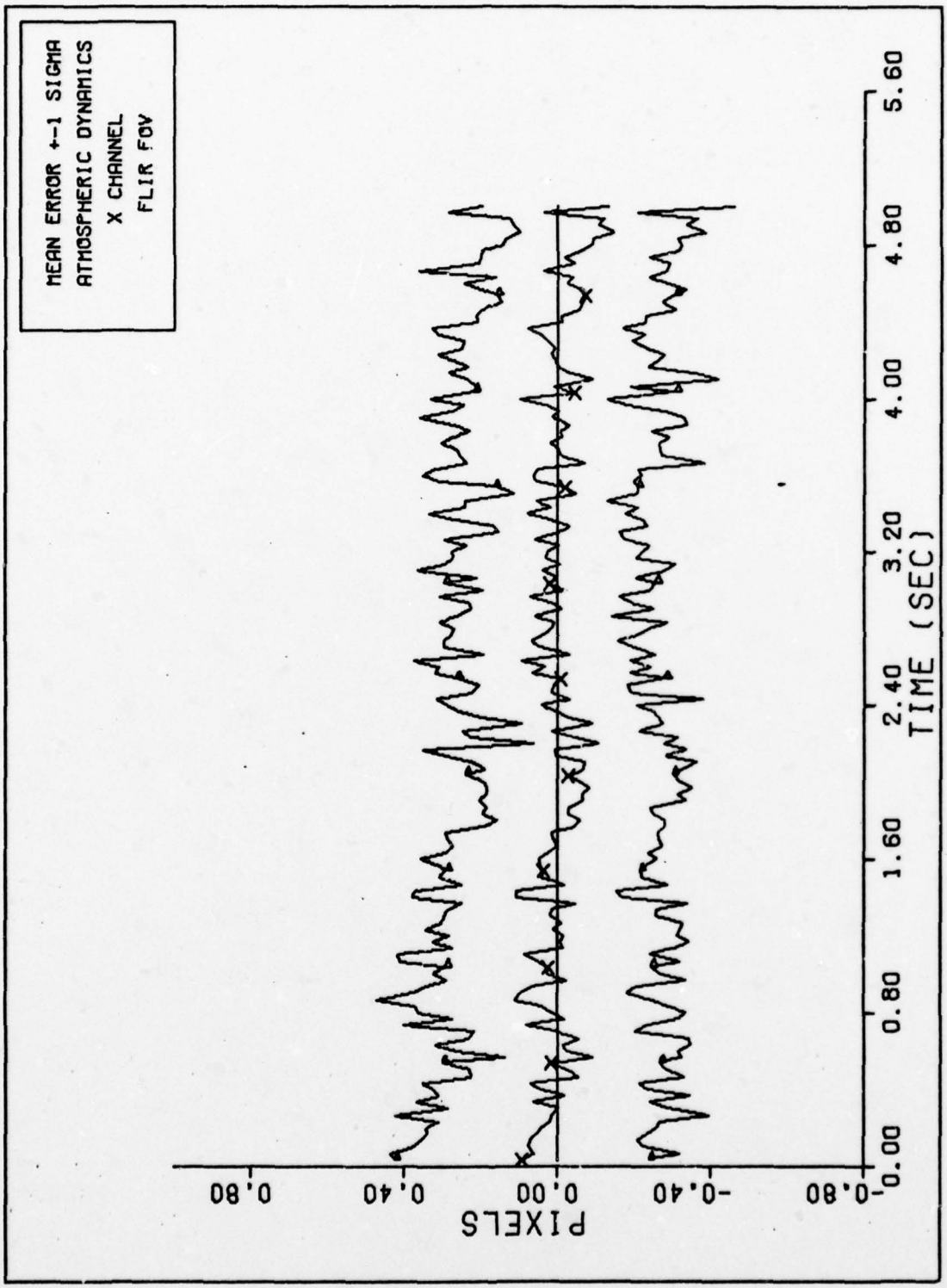


Figure 26c. X CHANNEL ATMOSPHERICS ERROR (S/N=10)

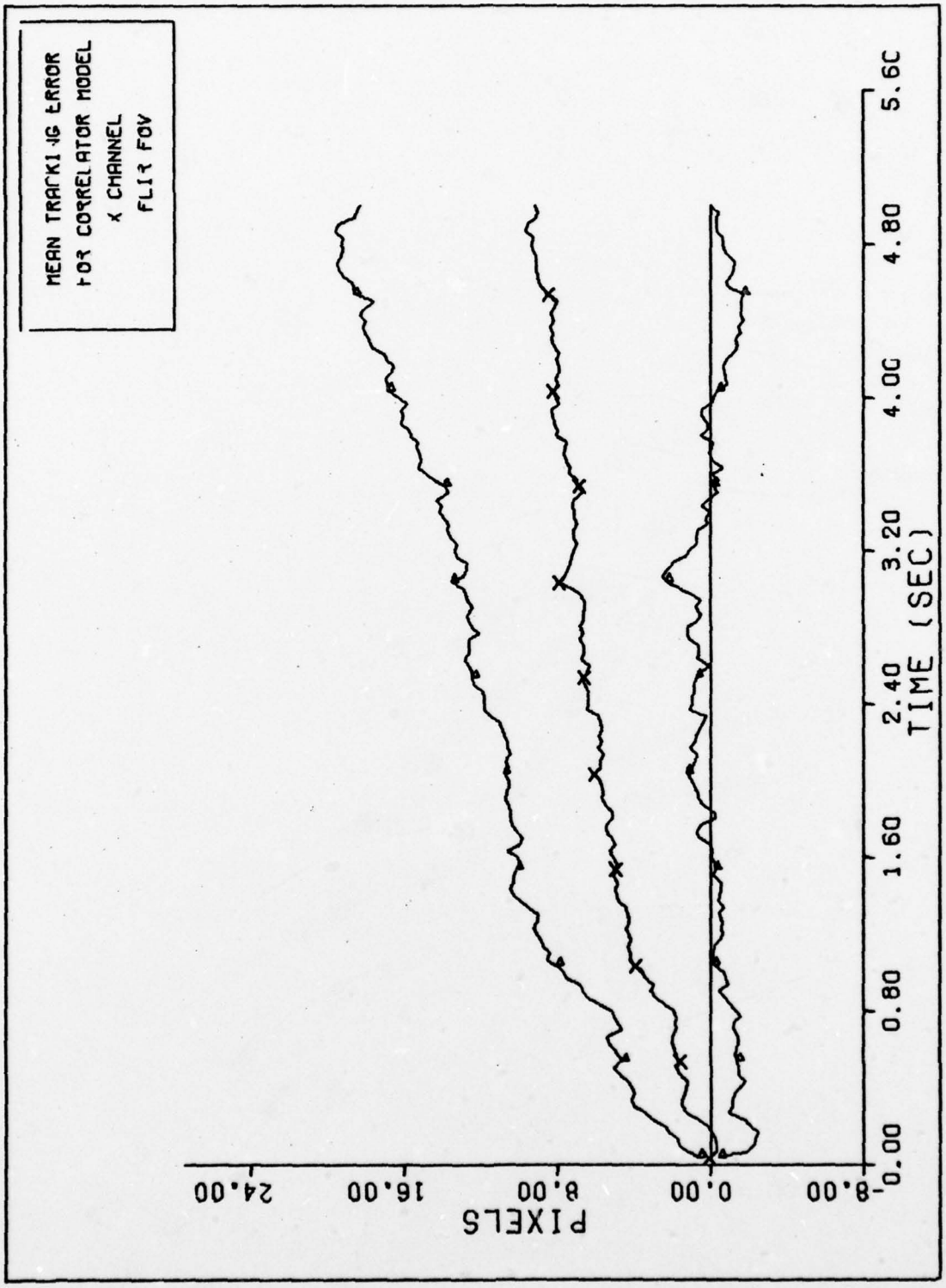


Figure 27a. X CHANNEL CORRELATOR TRACKING ERROR (S/N = 10)

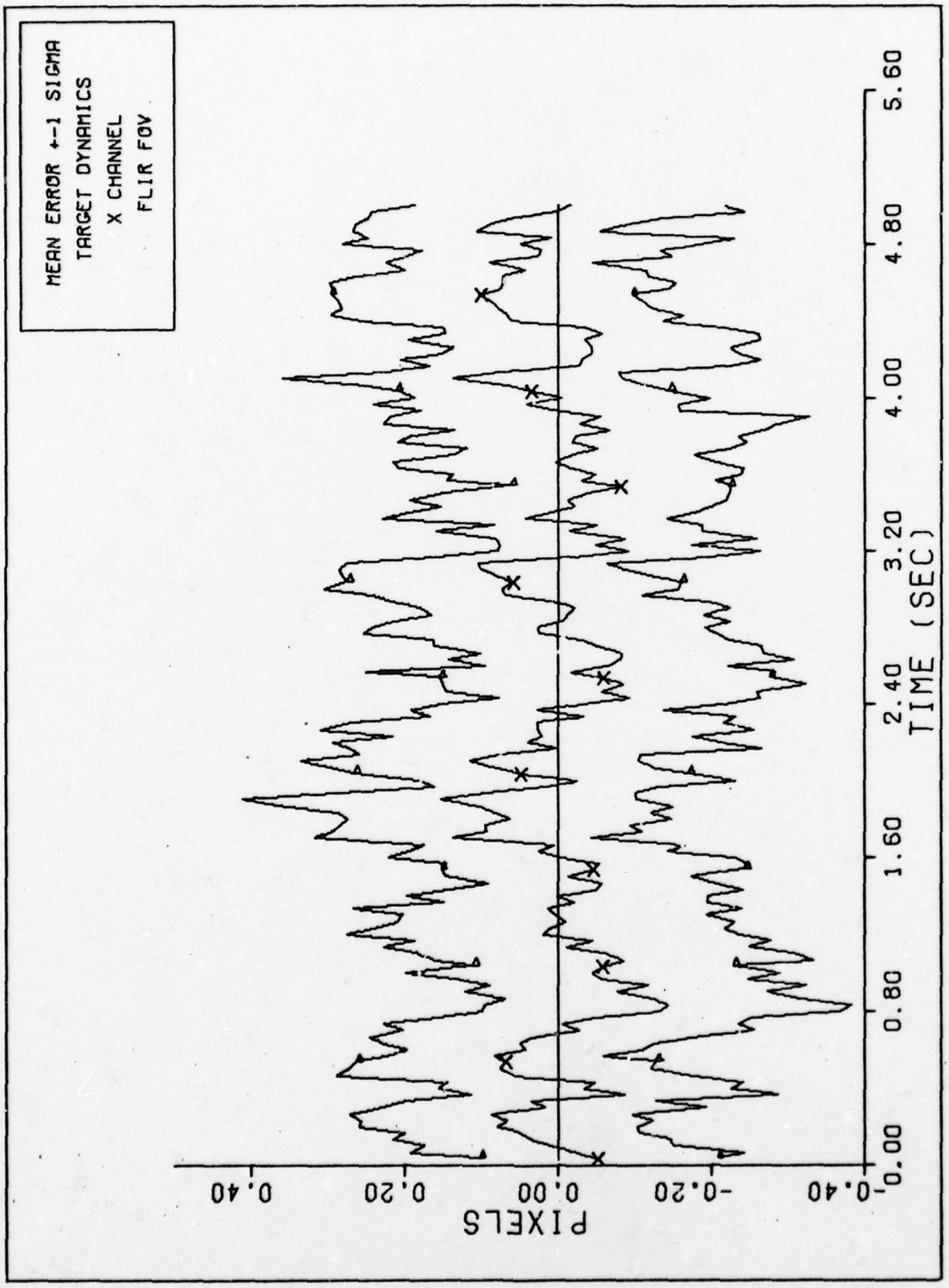


Figure 27b. X CHANNEL DYNAMICS ERROR (S/N=10)

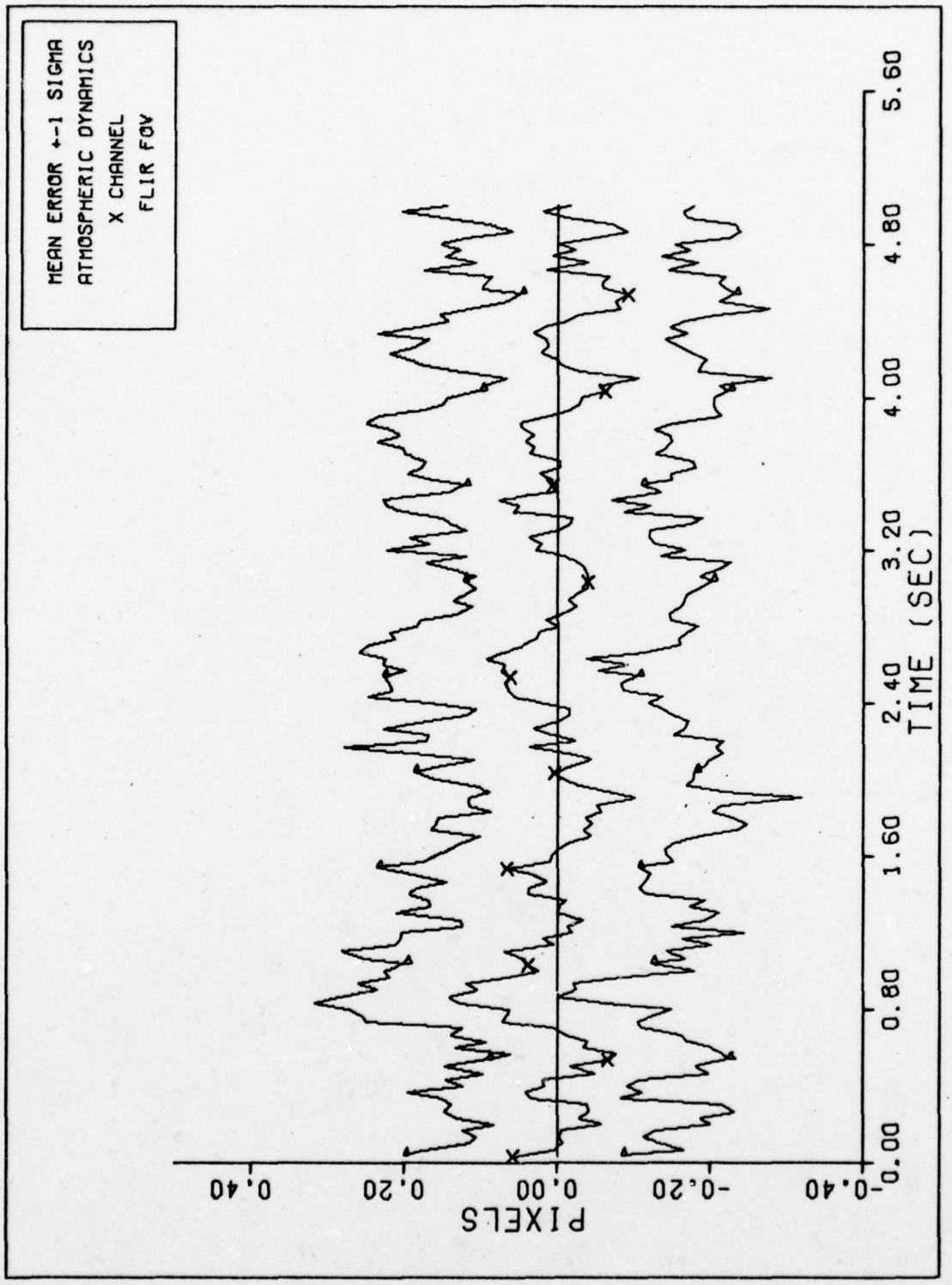


Figure 27c. X CHANNEL ATMOSPHERICS ERROR (S/N=10)

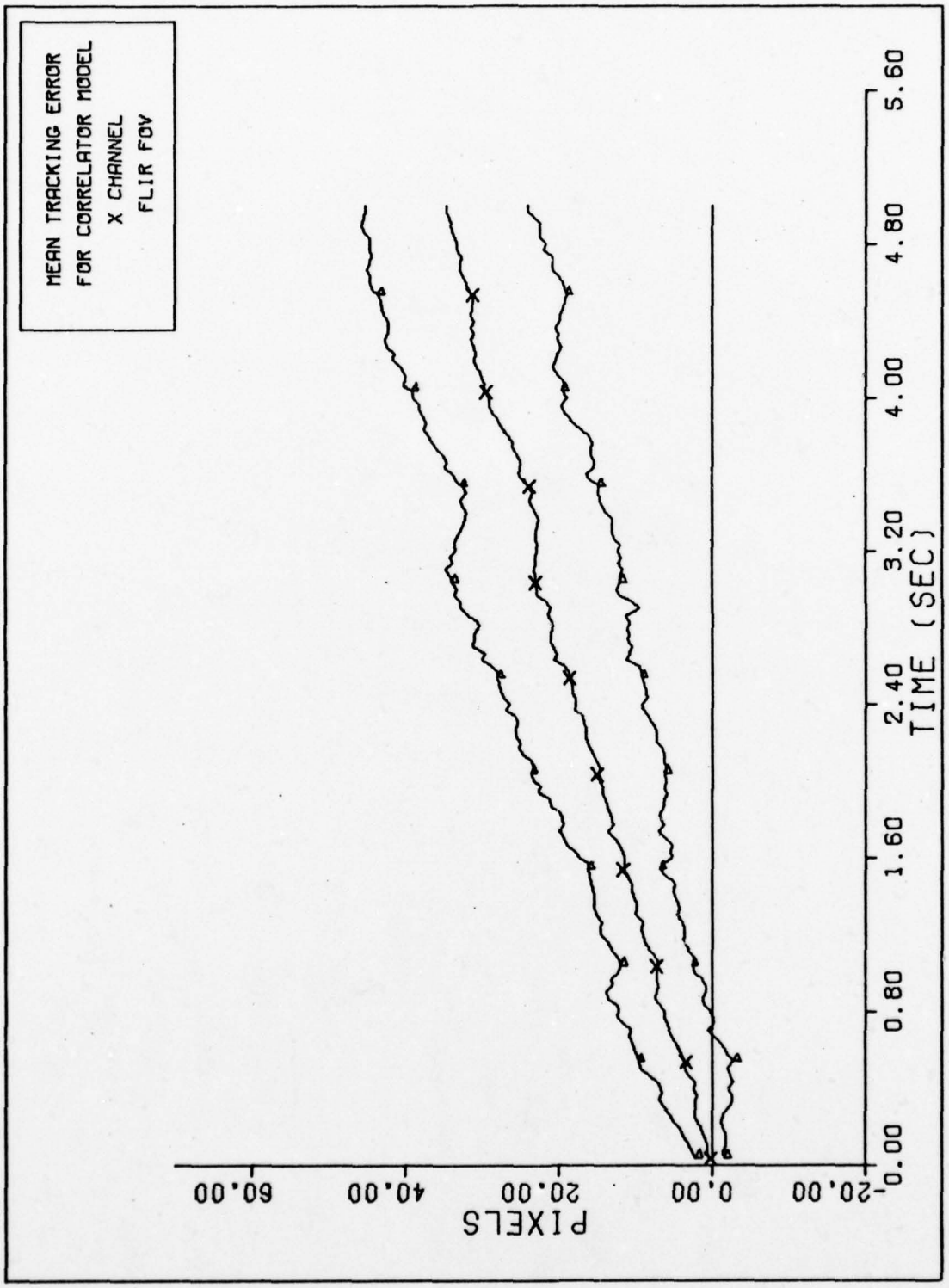


Figure 28a. X CHANNEL CORRELATOR TRACKING ERROR (S/N = 1)

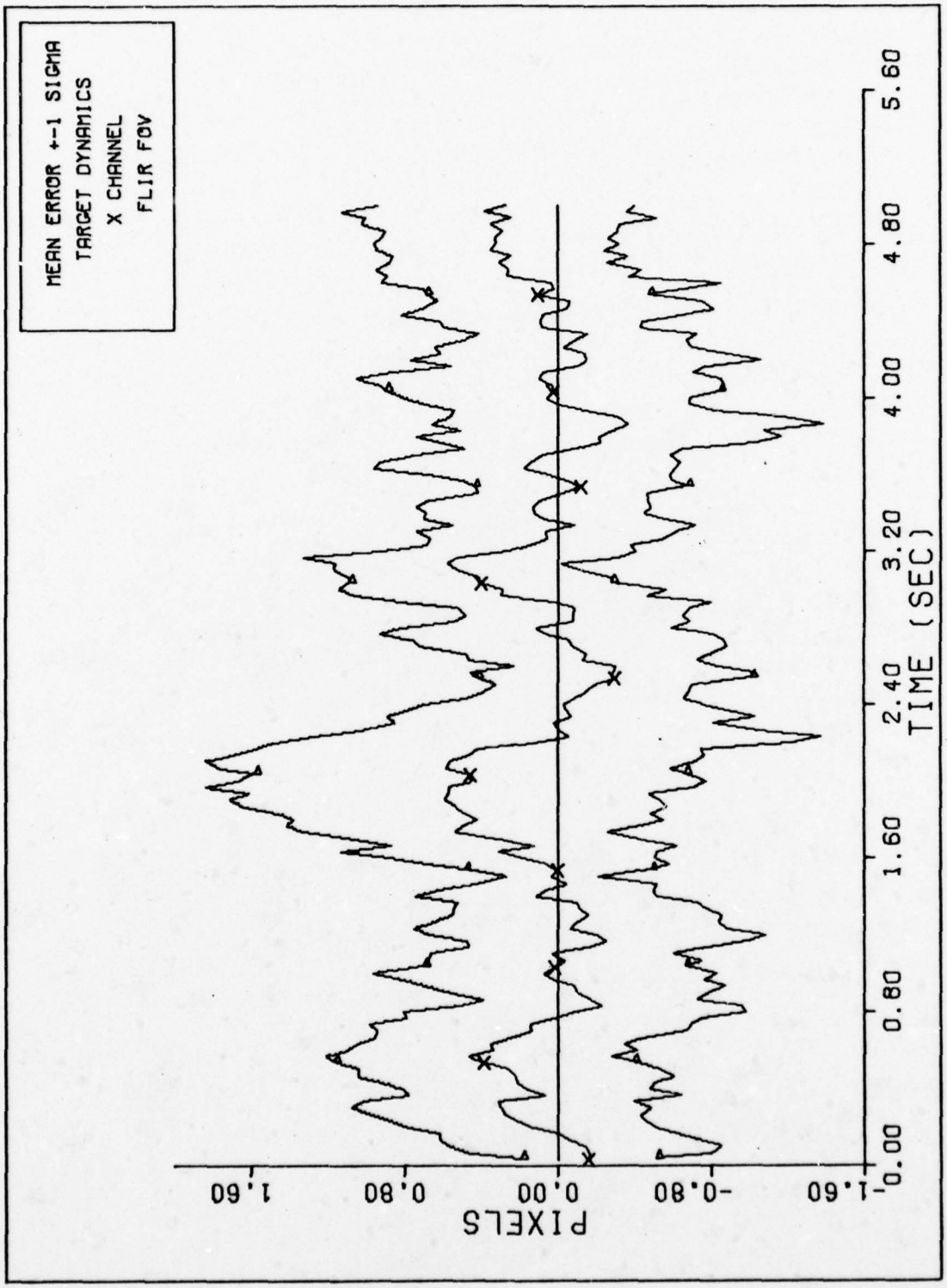


Figure 28b. X CHANNEL DYNAMICS ERROR (S/N= 1)

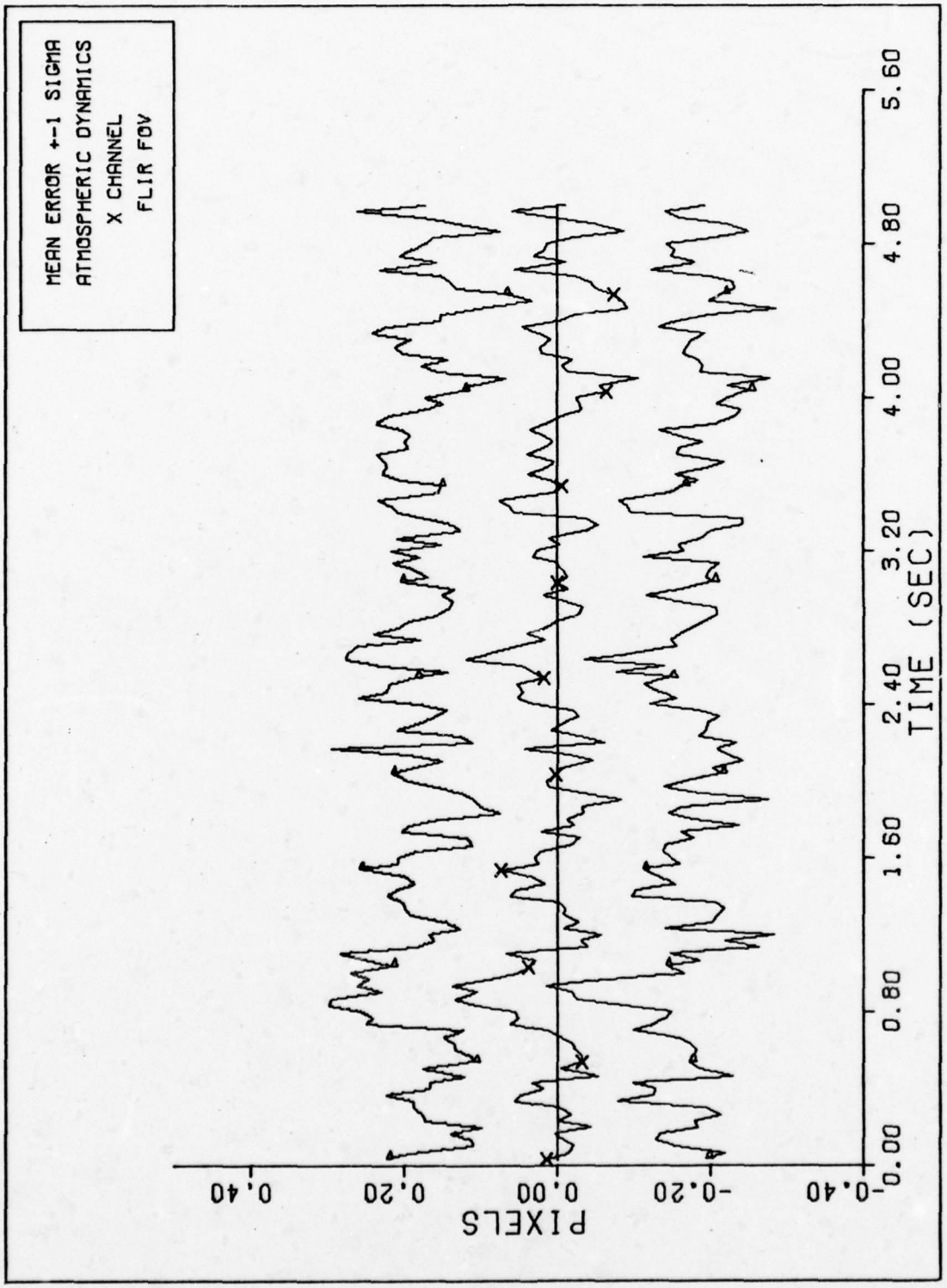


Figure 28c. X CHANNEL ATMOSPHERICS ERROR (S/N= 1)

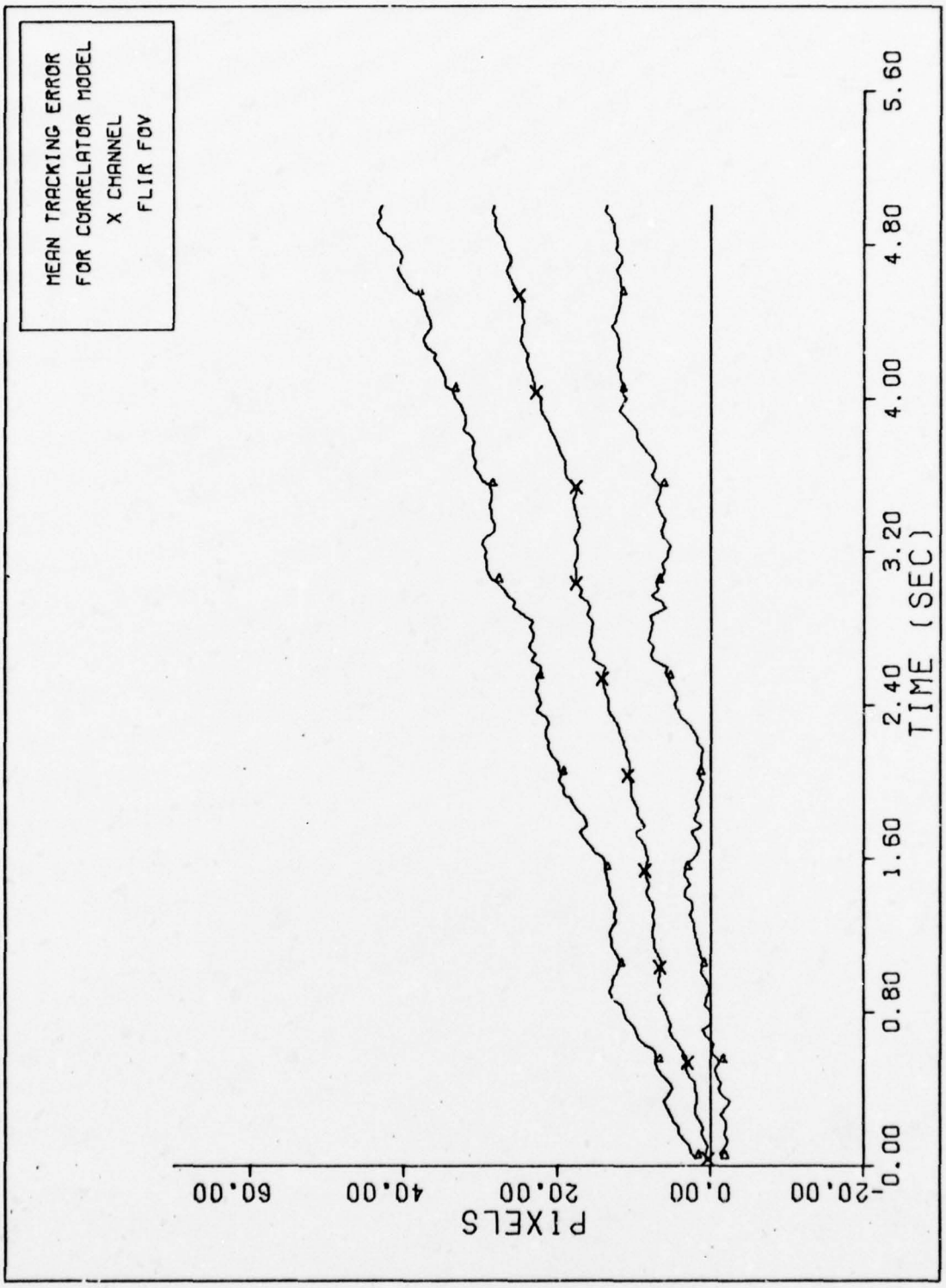


Figure 29a. X CHANNEL CORRELATOR TRACKING ERROR (S/N = 1)

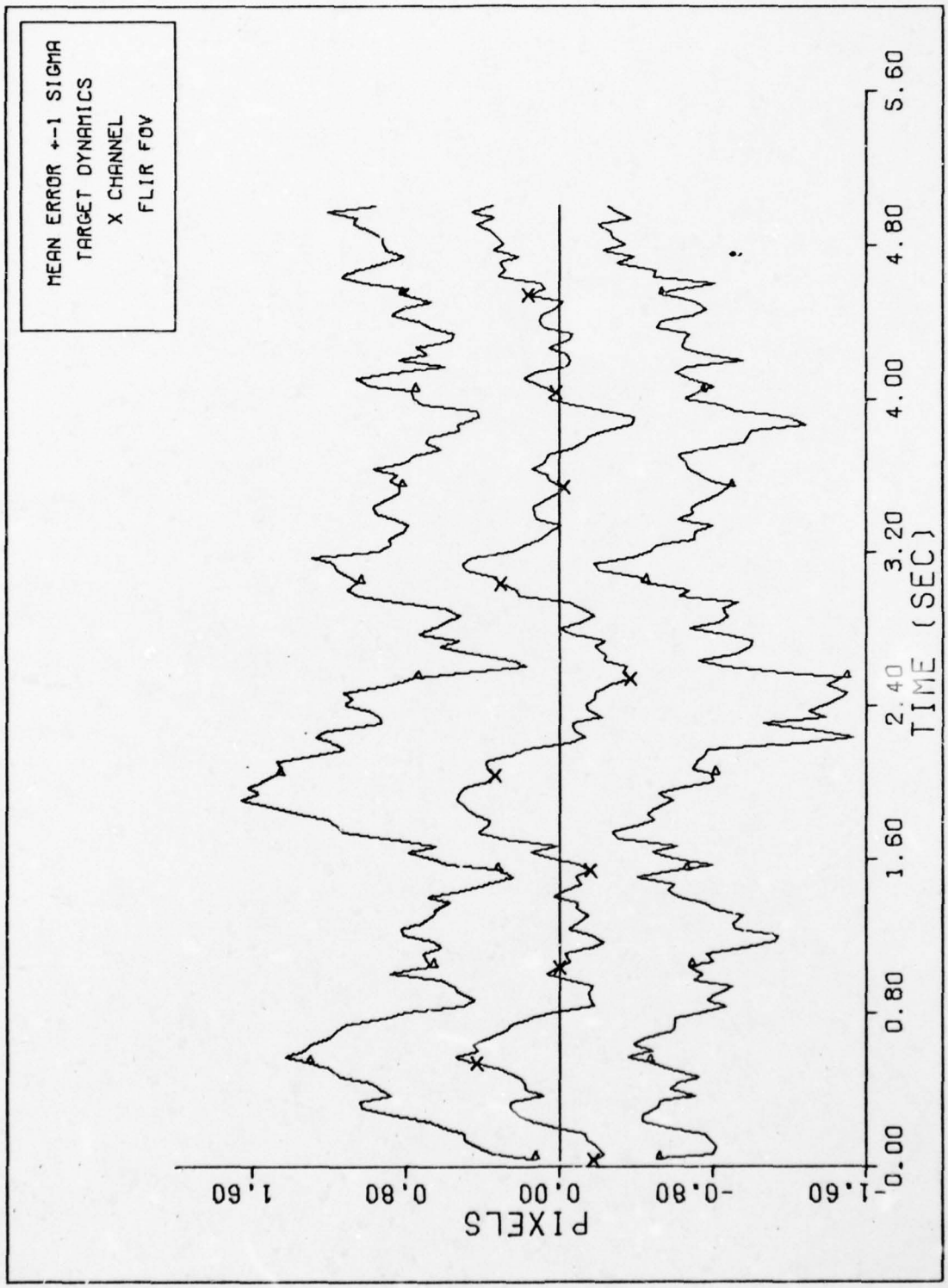


Figure 29b. X CHANNEL DYNAMICS ERROR (S/N= 1)

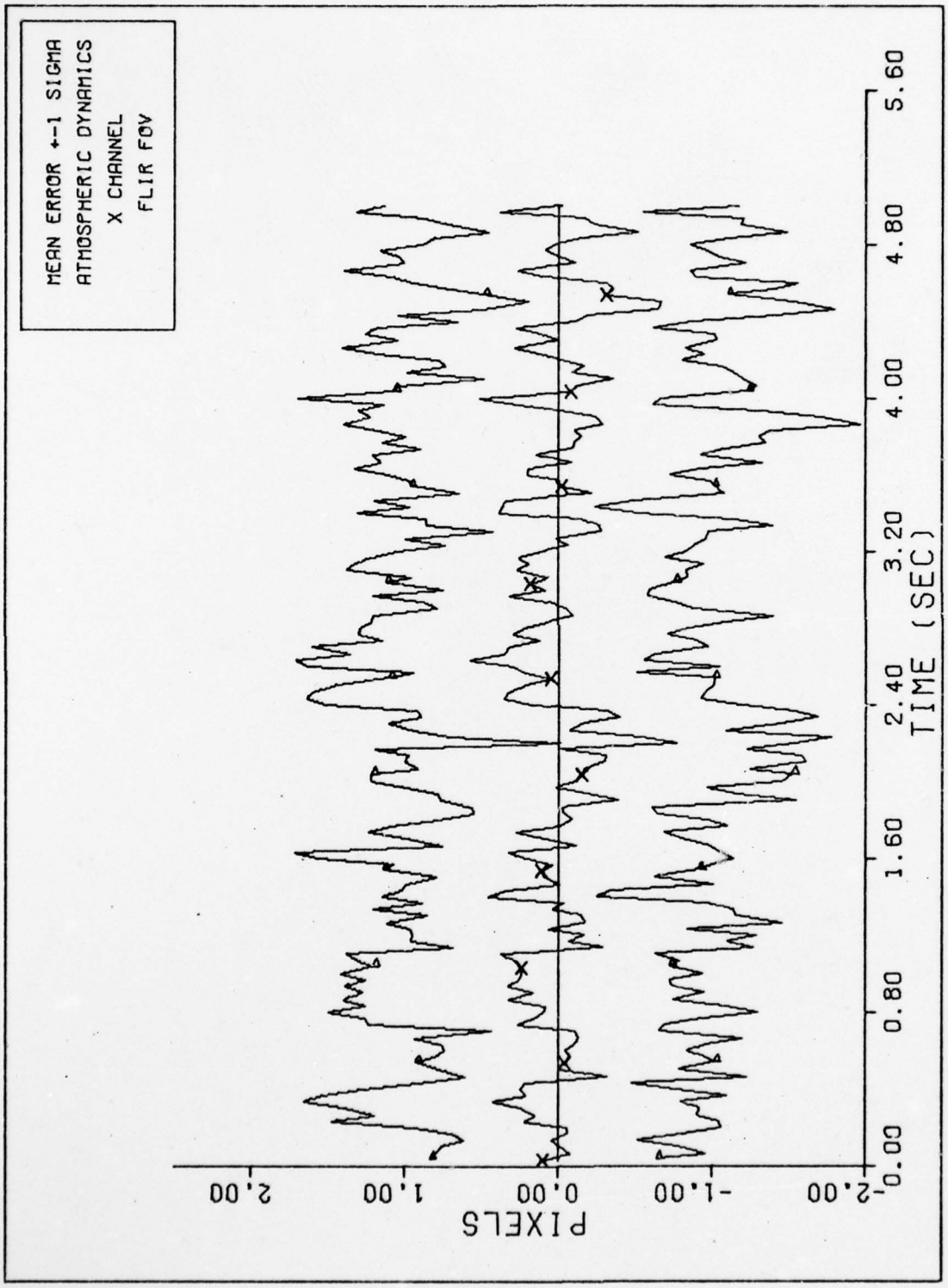


Figure 29c. X CHANNEL ATMOSPHERICS ERROR (S/N= 1)

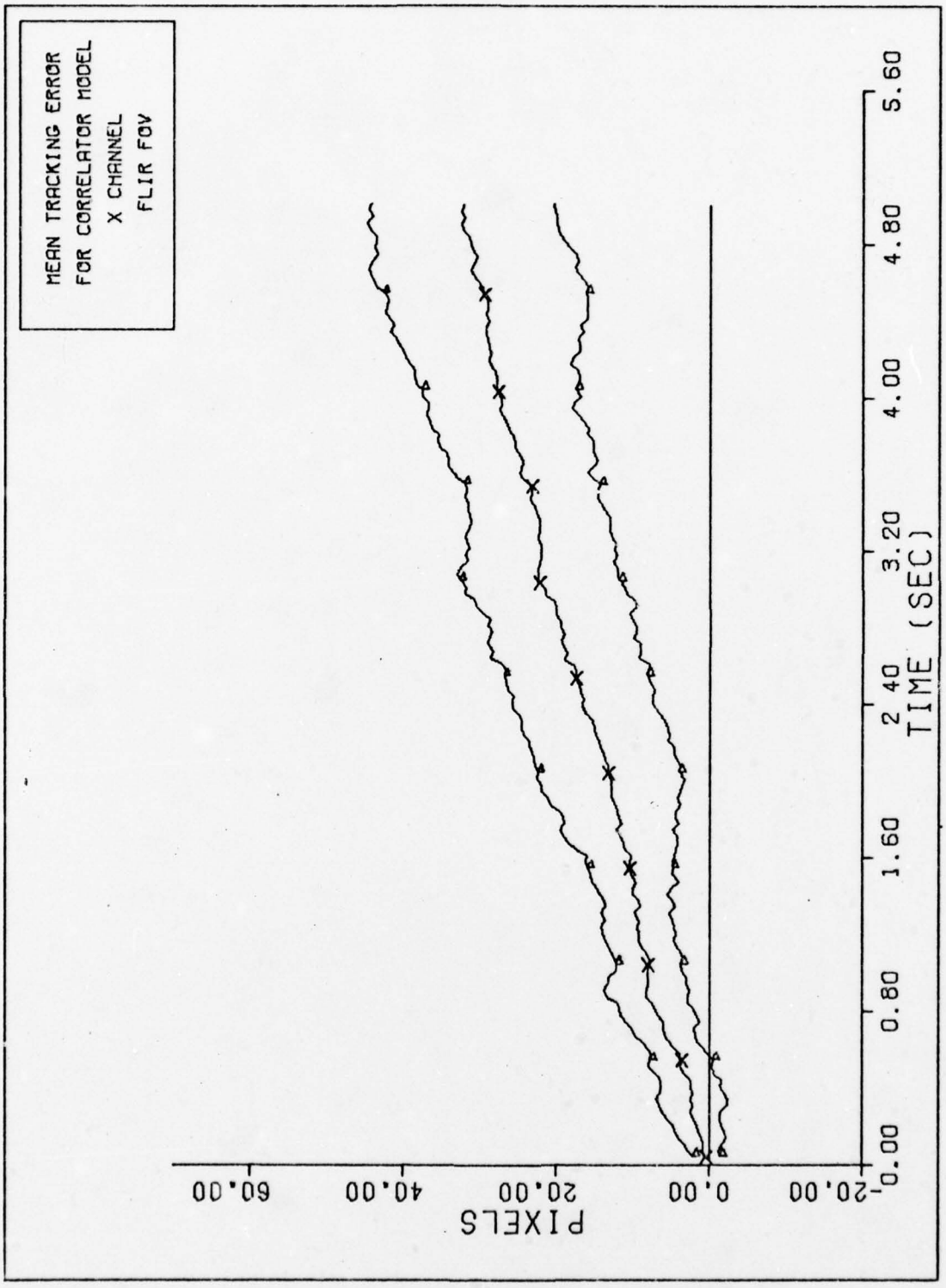


Figure 30a. X CHANNEL CORRELATOR TRACKING ERROR (S/N = 1)

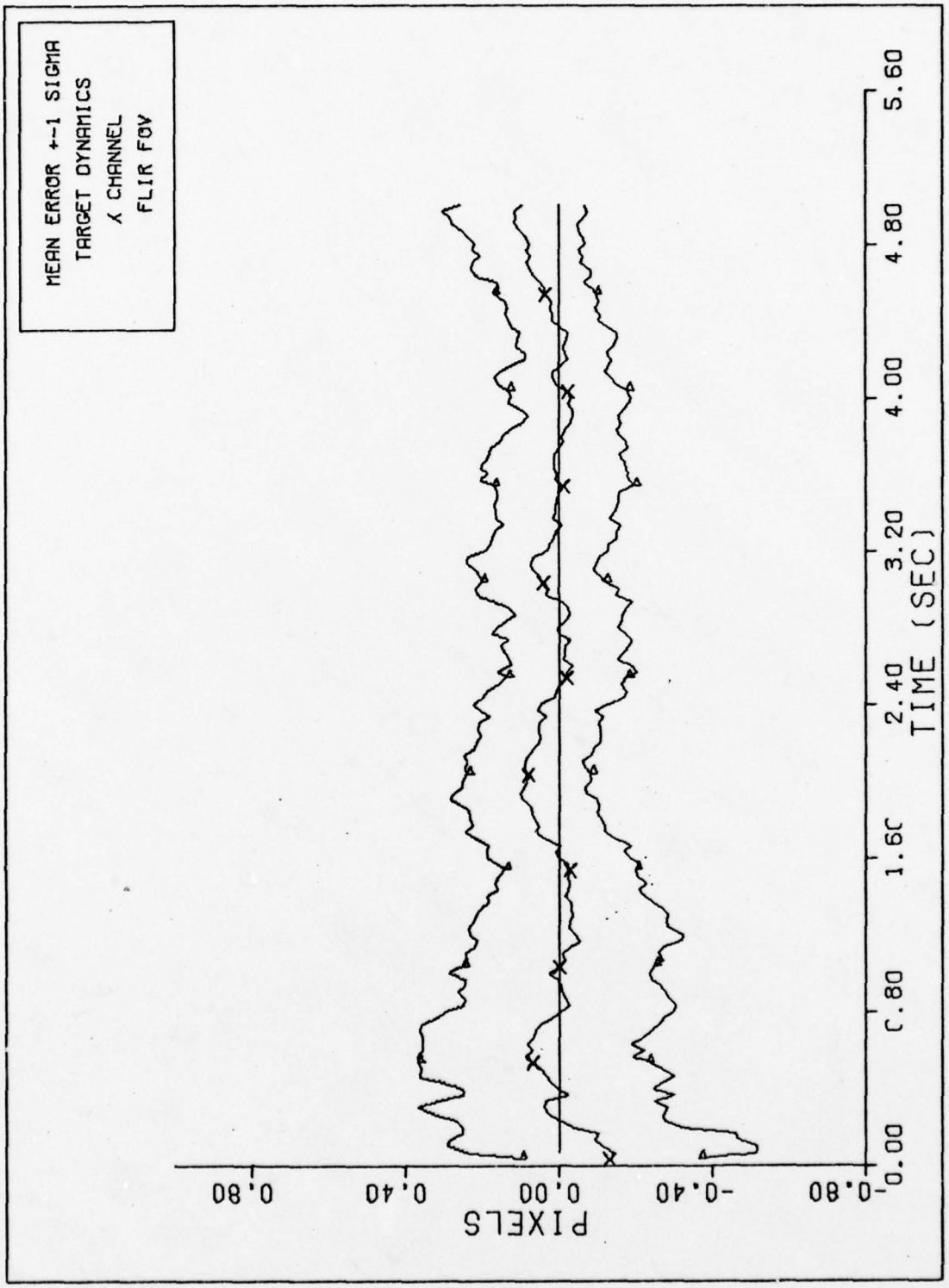


Figure 30b. X CHANNEL DYNAMICS ERROR (S/N= 1)

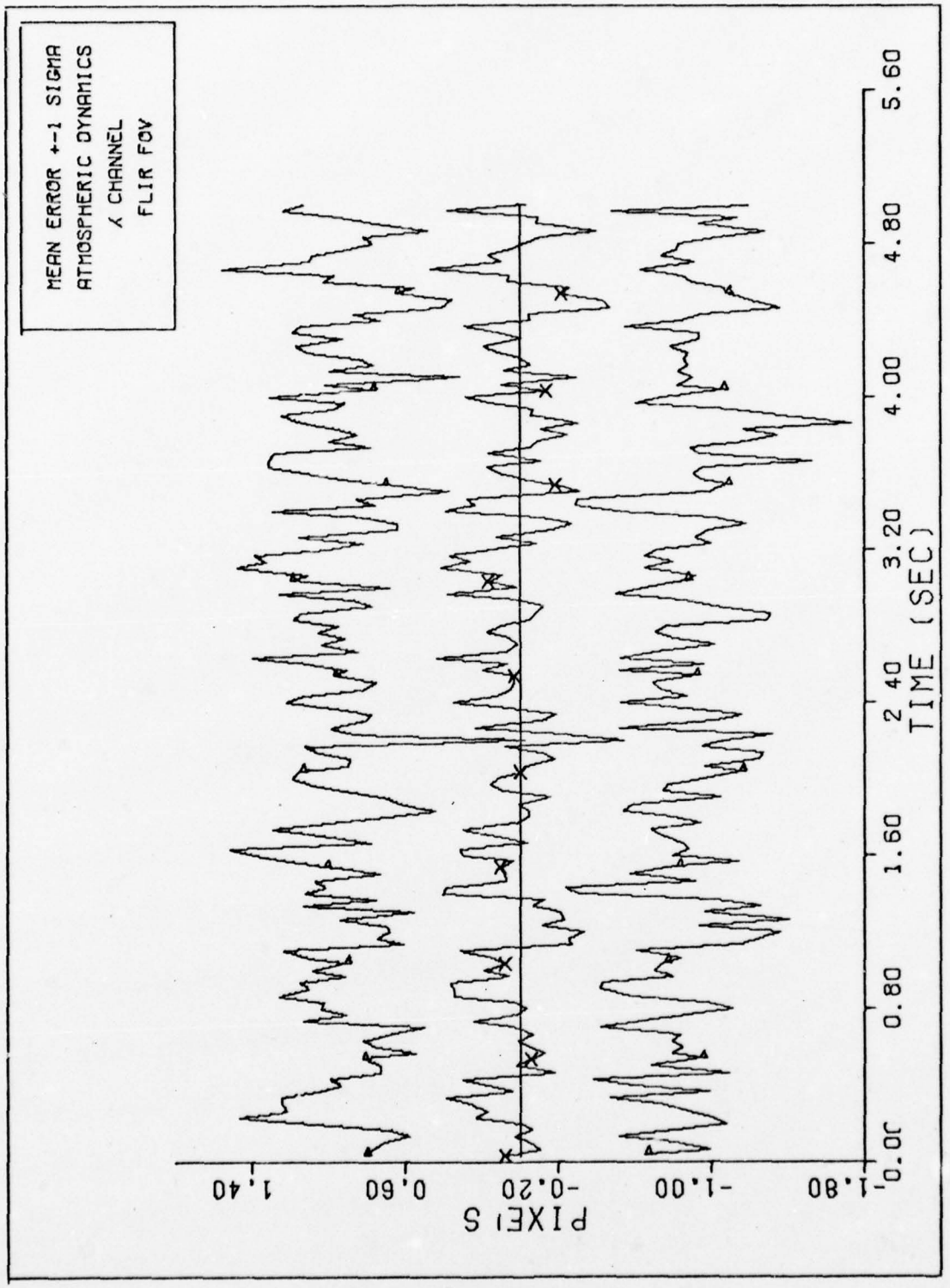


Figure 30c. X CHANNEL ATMOSPHERICS ERROR (S/N= 1)

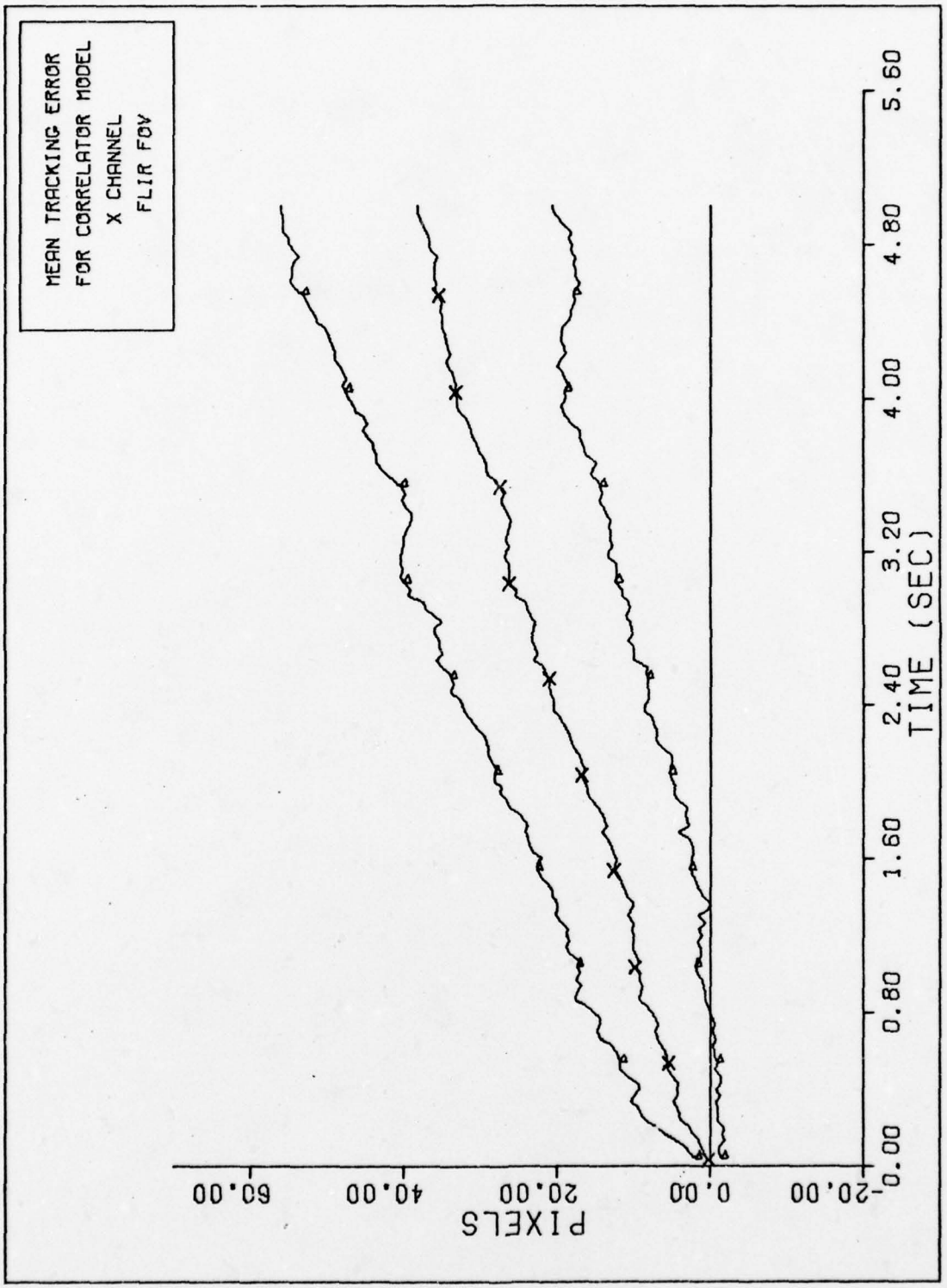


Figure 31a. X CHANNEL CORRELATOR TRACKING ERROR (S/N = 1)

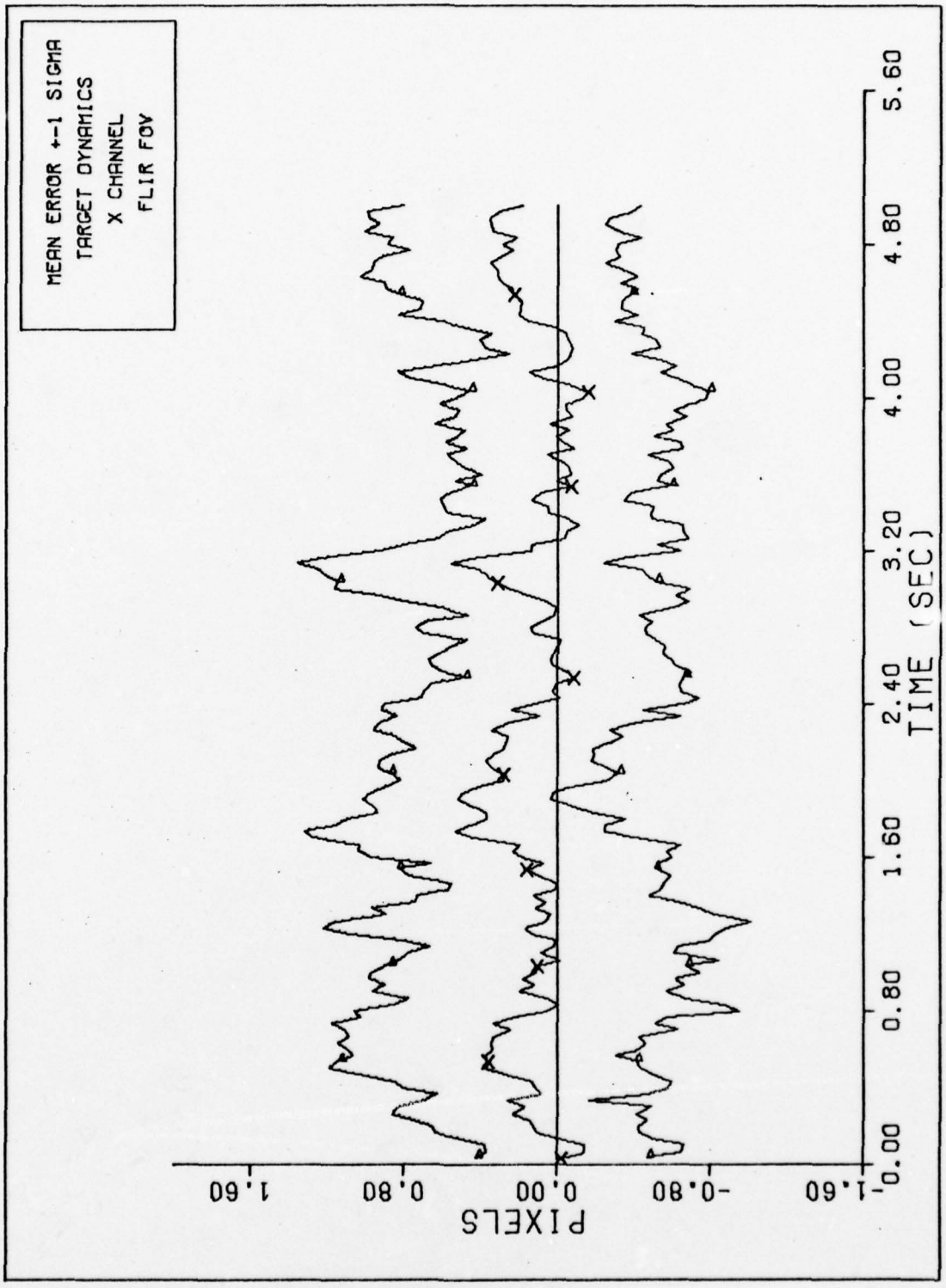


Figure 31b. X CHANNEL DYNAMICS ERROR (S/N= 1)

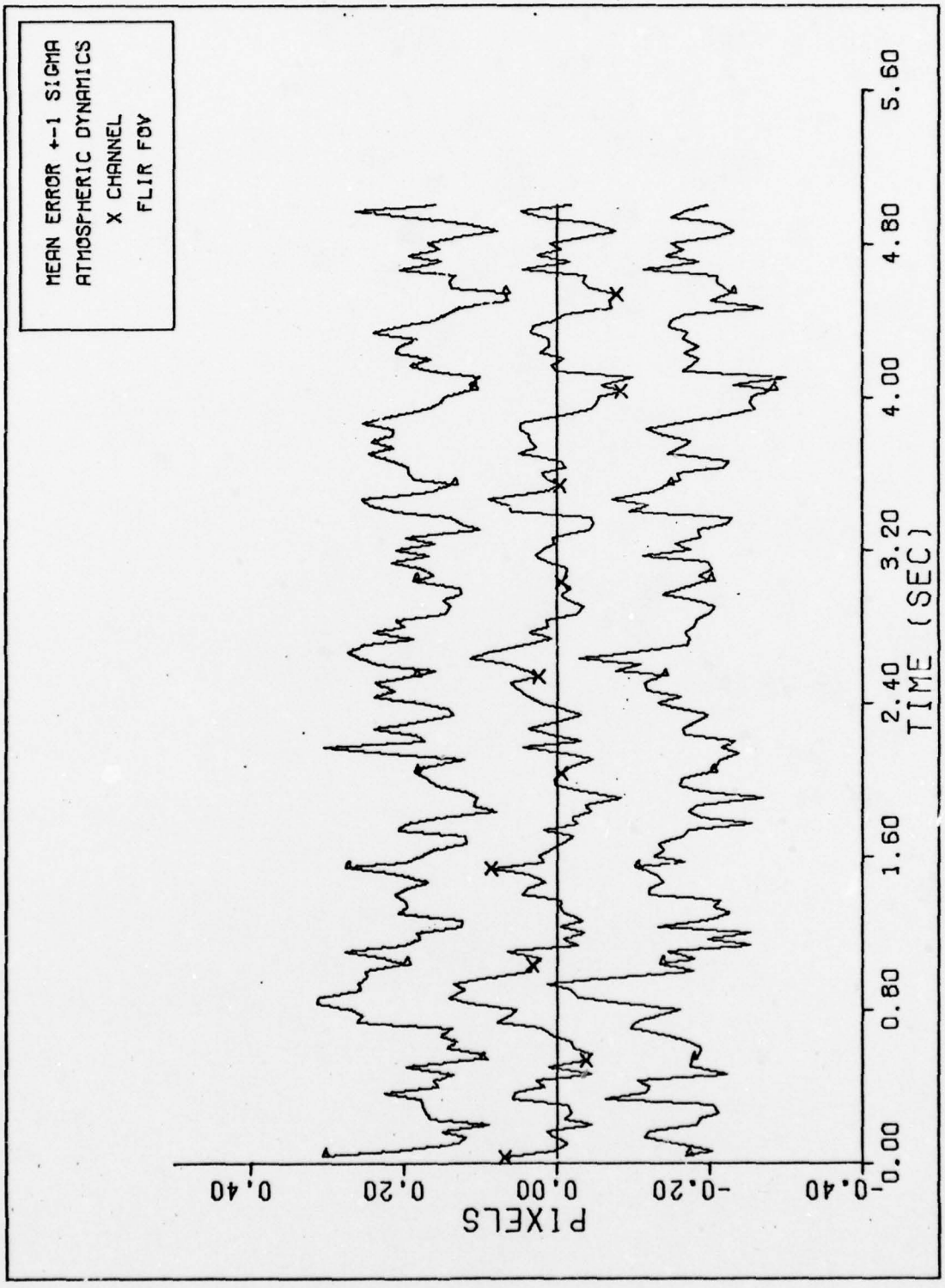


Figure 31c. X CHANNEL ATMOSPHERICS ERROR (S/N= 1)

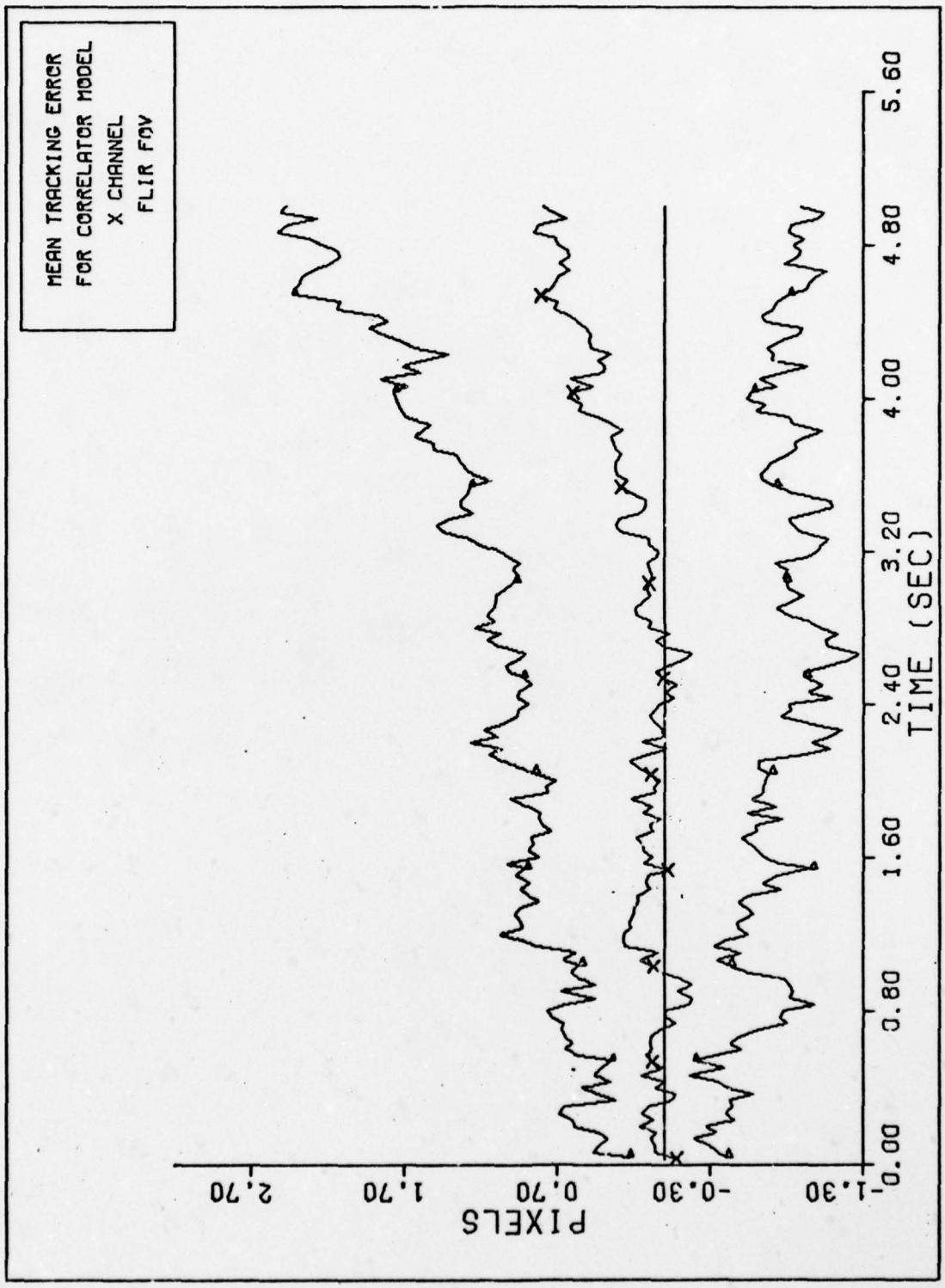


Figure 32a. X CHANNEL CORRELATOR TRACKING ERROR (S/N = 20)

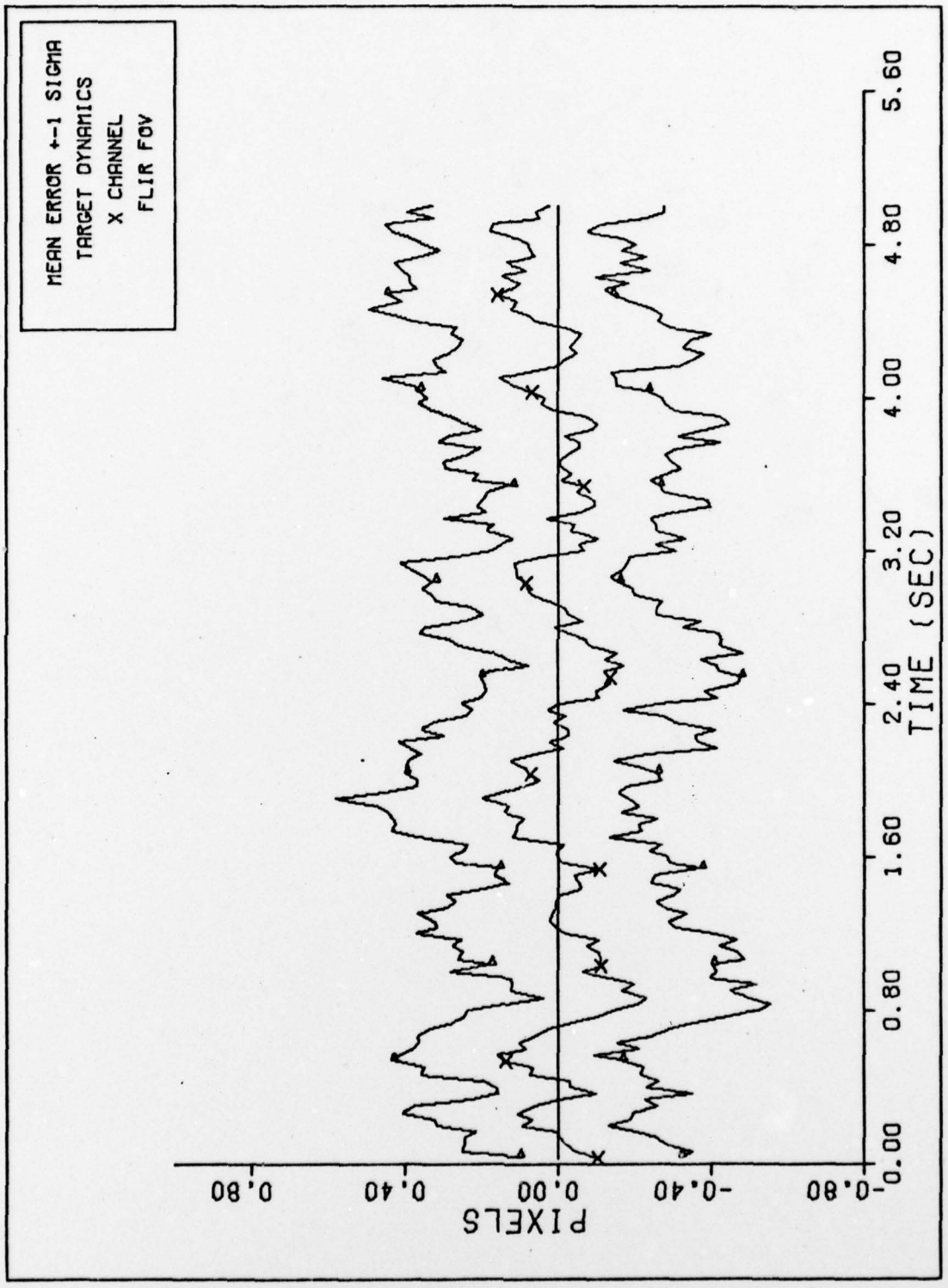


Figure 32b. X CHANNEL DYNAMICS ERROR (S/N=20)

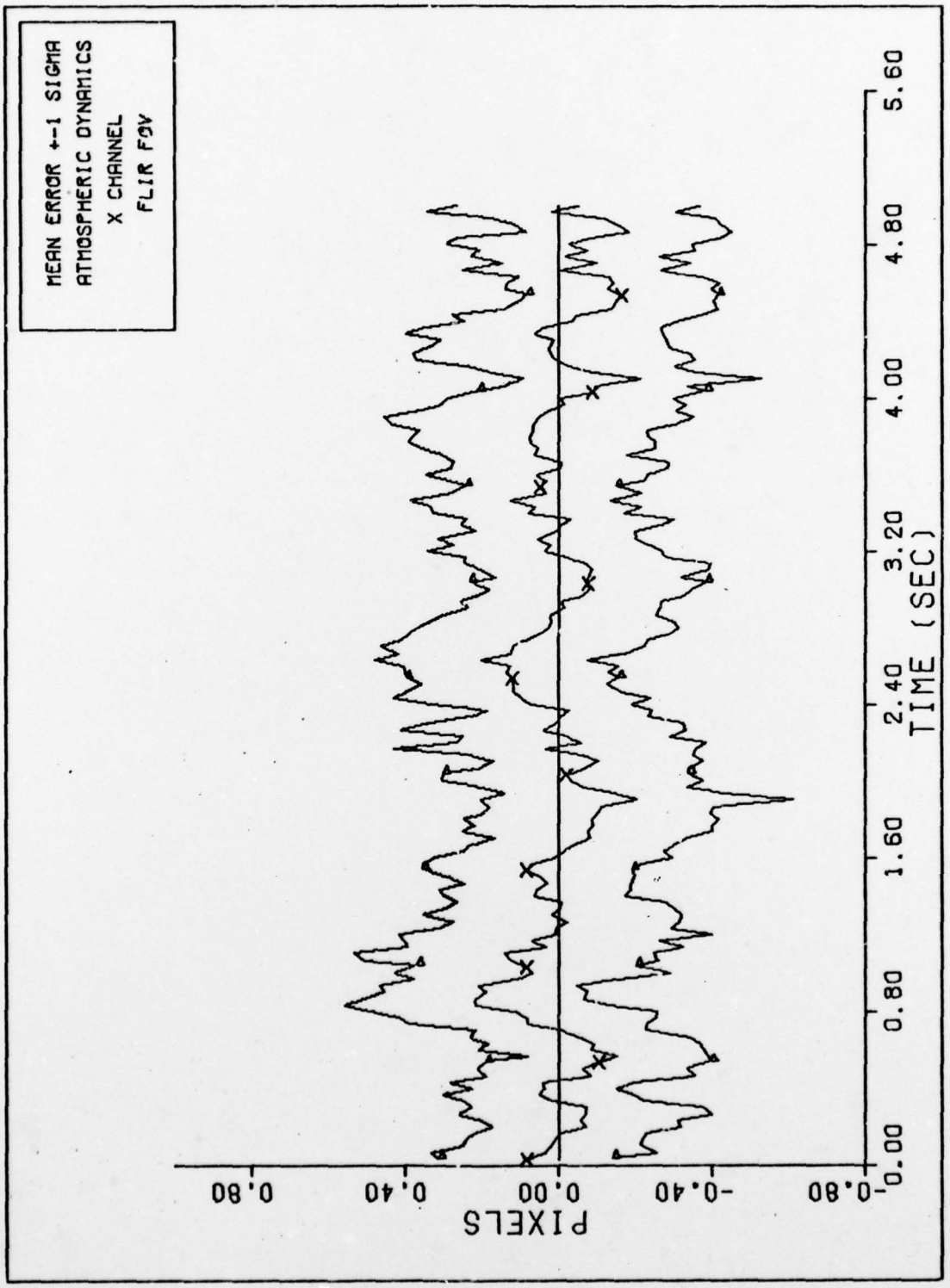


Figure 32c. X CHANNEL ATMOSPHERICS ERROR (S/N=20)

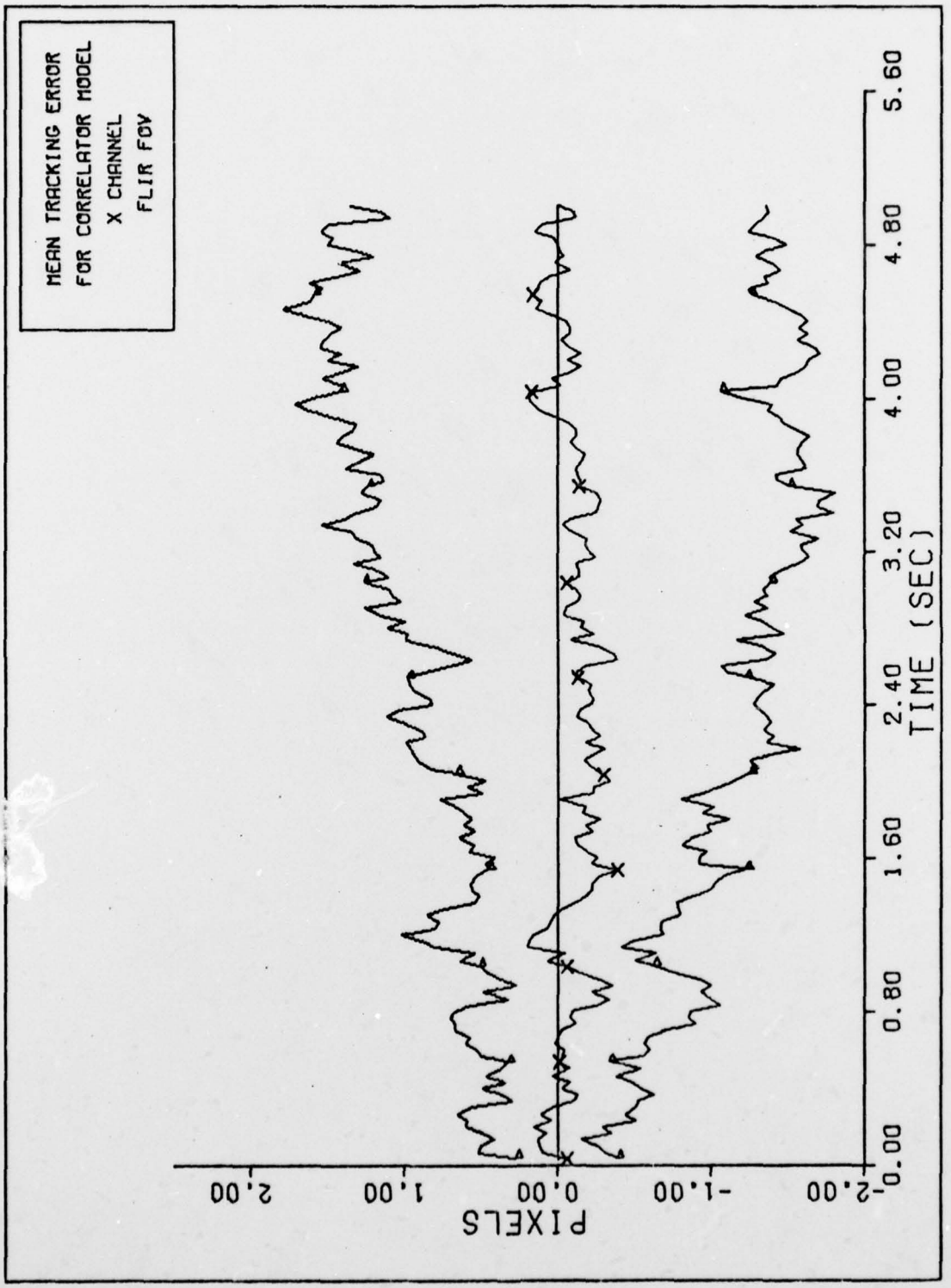


Figure 33a. X CHANNEL CORRELATOR TRACKING ERROR (S/N = 20)

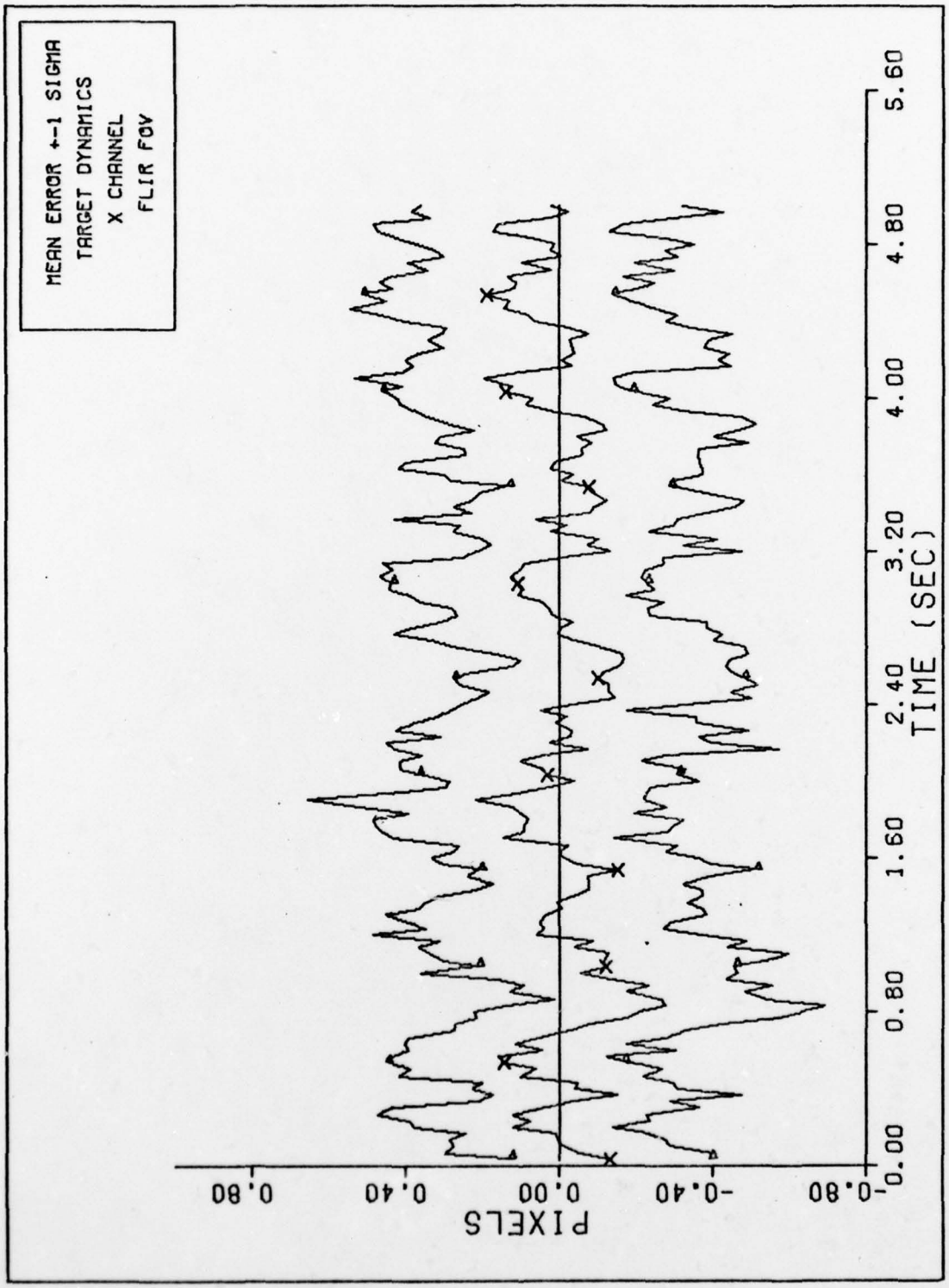


Figure 33b. X CHANNEL DYNAMICS ERROR (S/N=20)

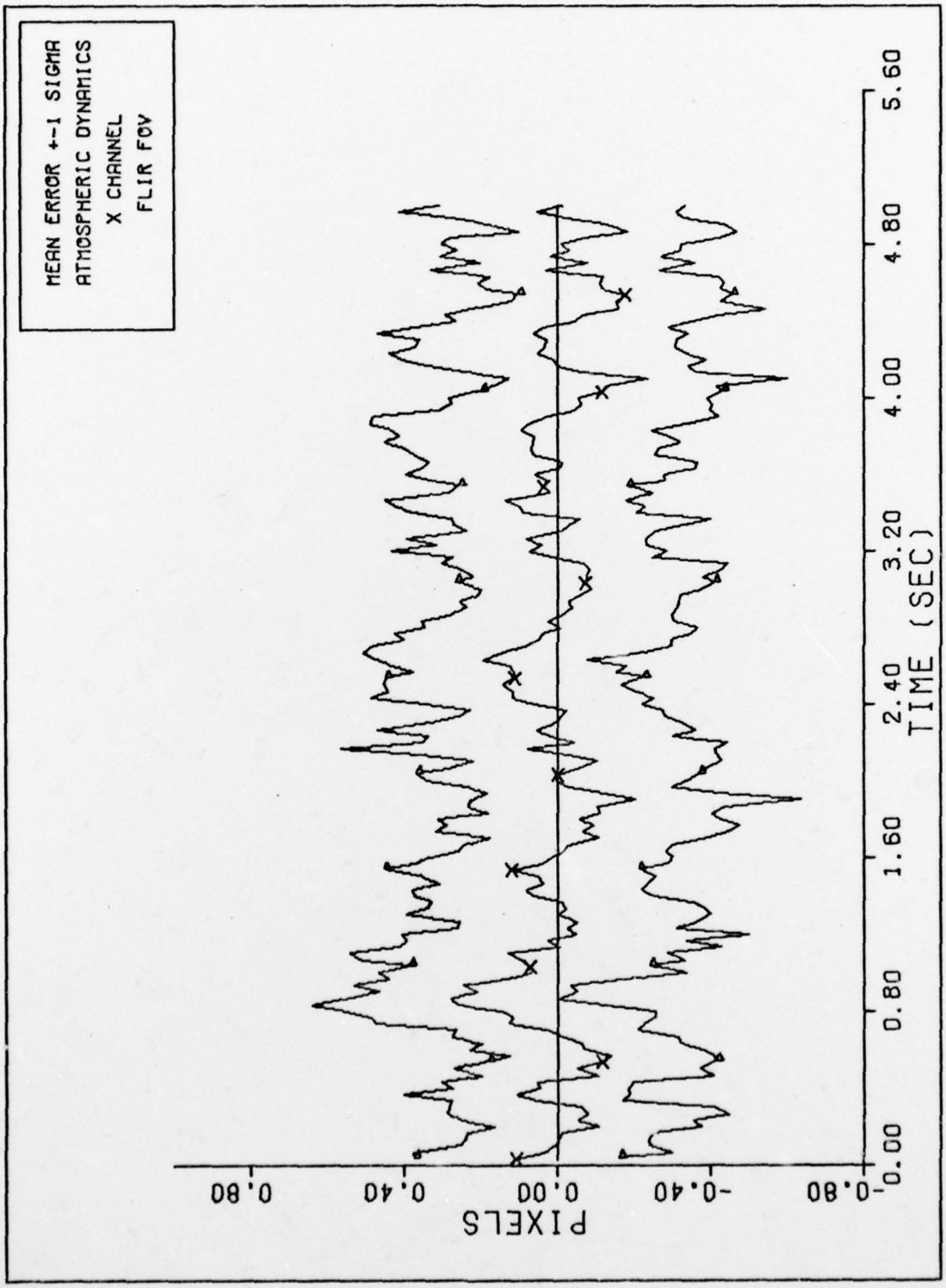


Figure 33c. X CHANNEL ATMOSPHERICS ERROR (S/N=20)

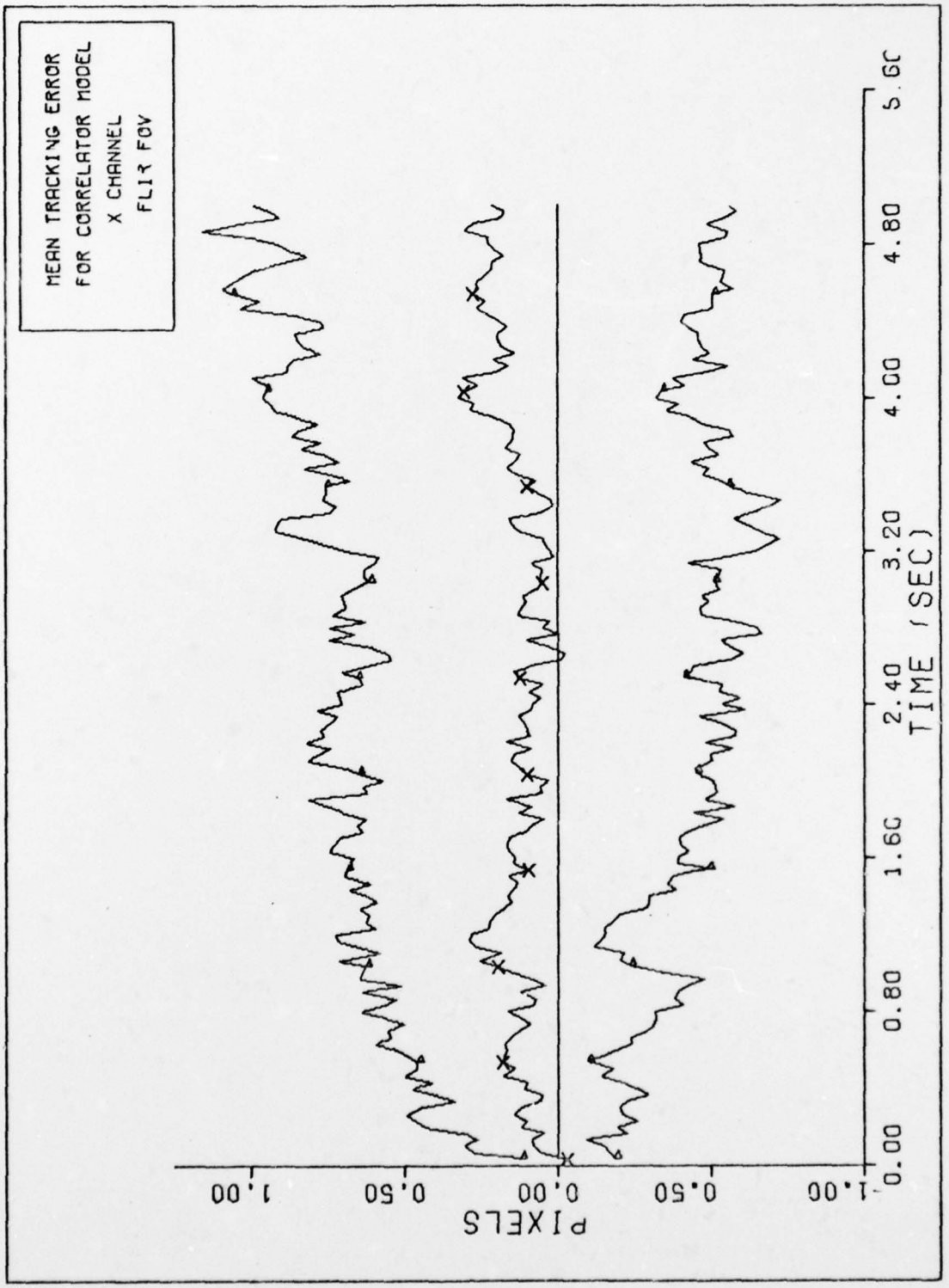


Figure 34a. X CHANNEL CORRELATOR TRACKING ERROR (S/N = 20)

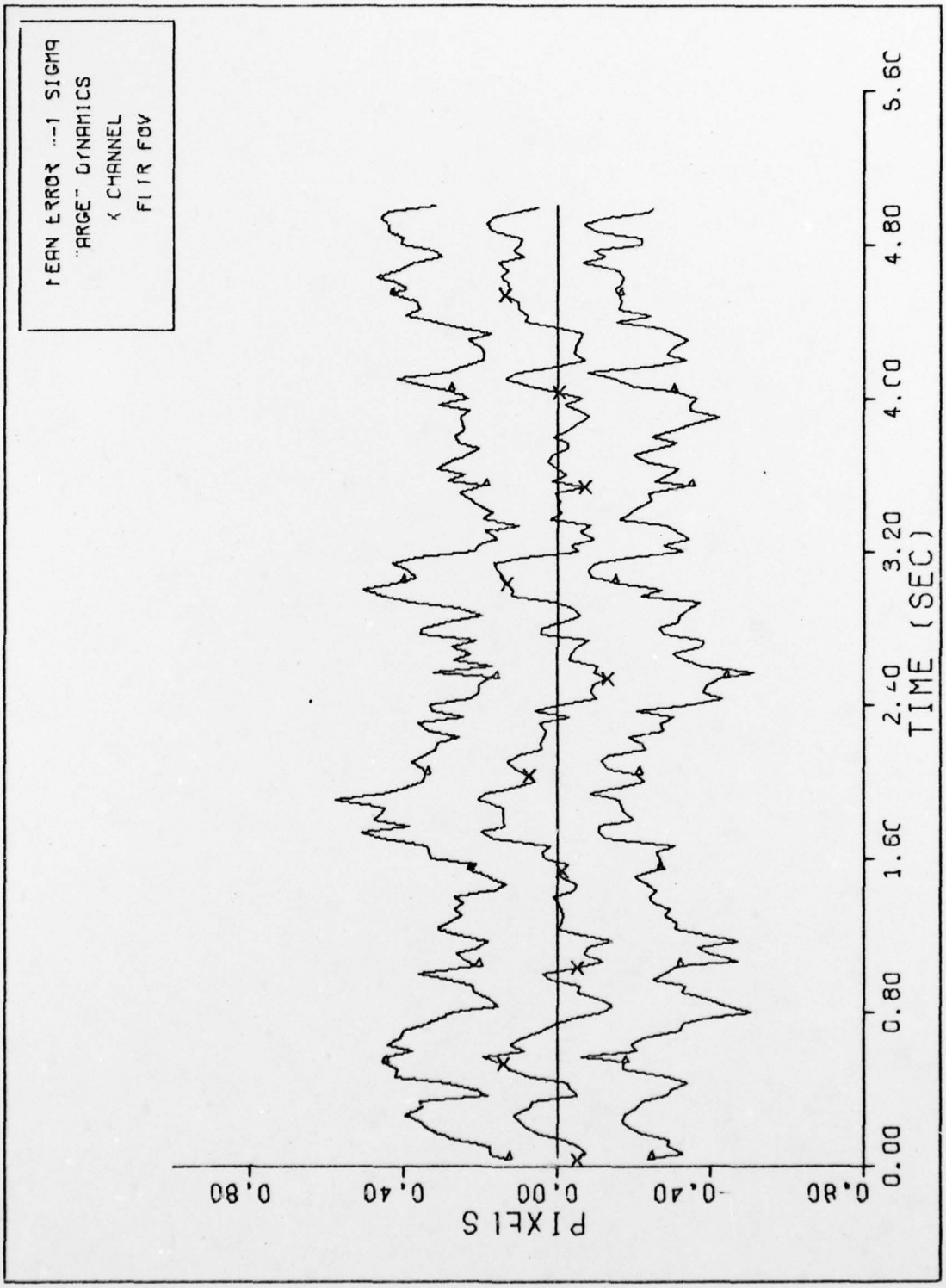


Figure 34b. X CHANNEL DYNAMICS ERROR (S/N-20)

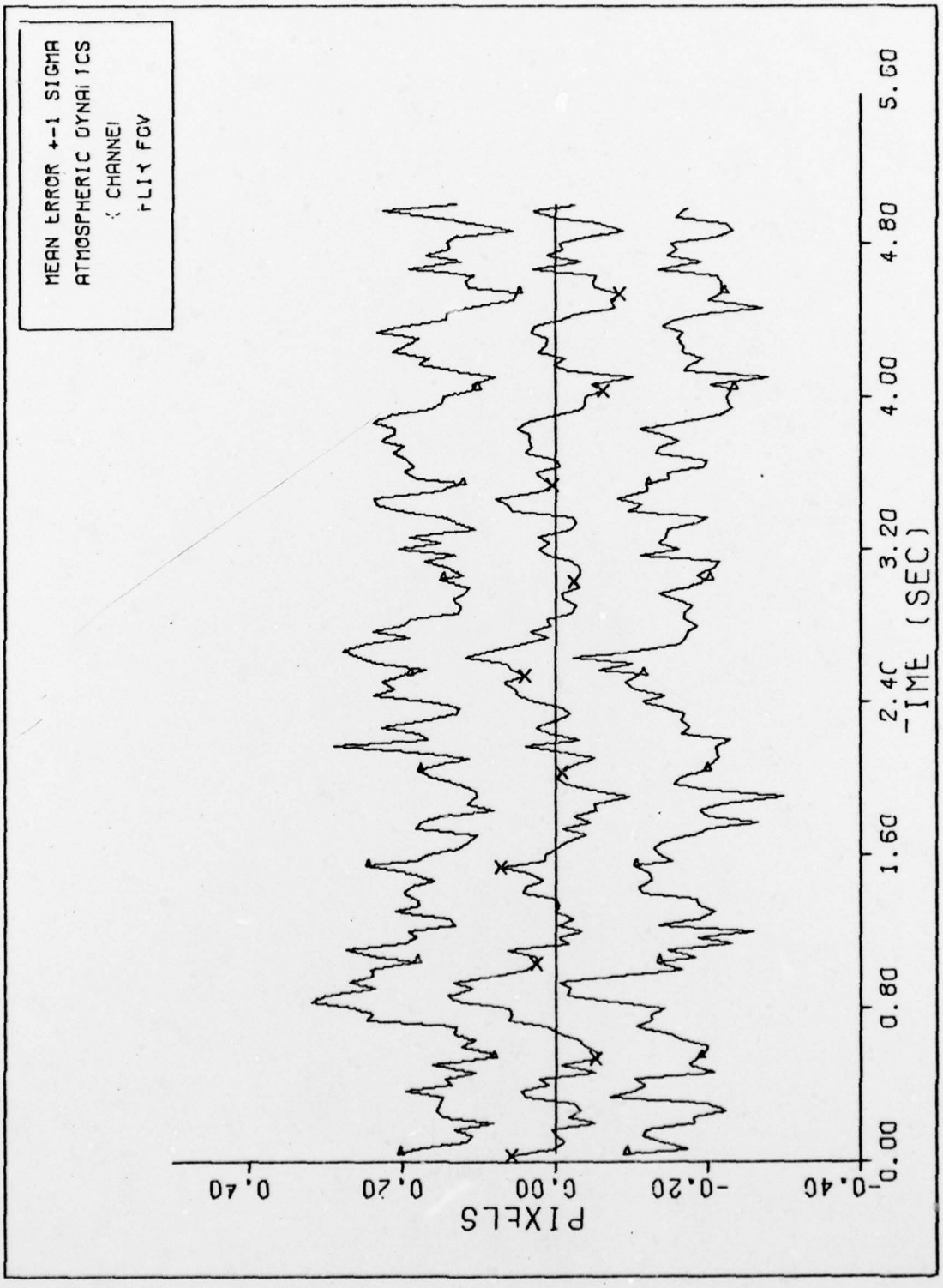


Figure 34c. X CHANNEL ATMOSPHERICS ERROR (S/N=20)

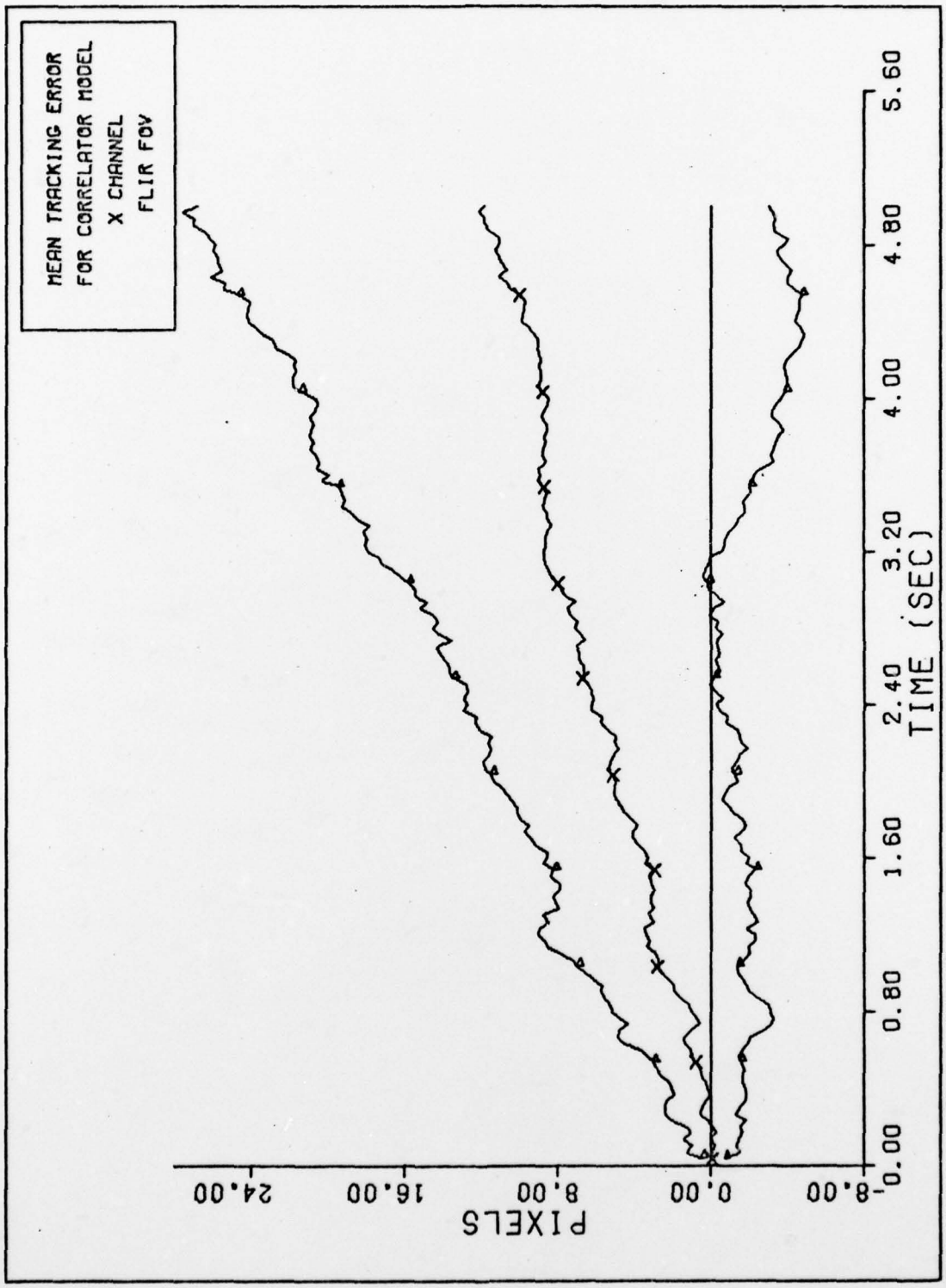


Figure 35a. X CHANNEL CORRELATOR TRACKING ERROR (S/N = 20)

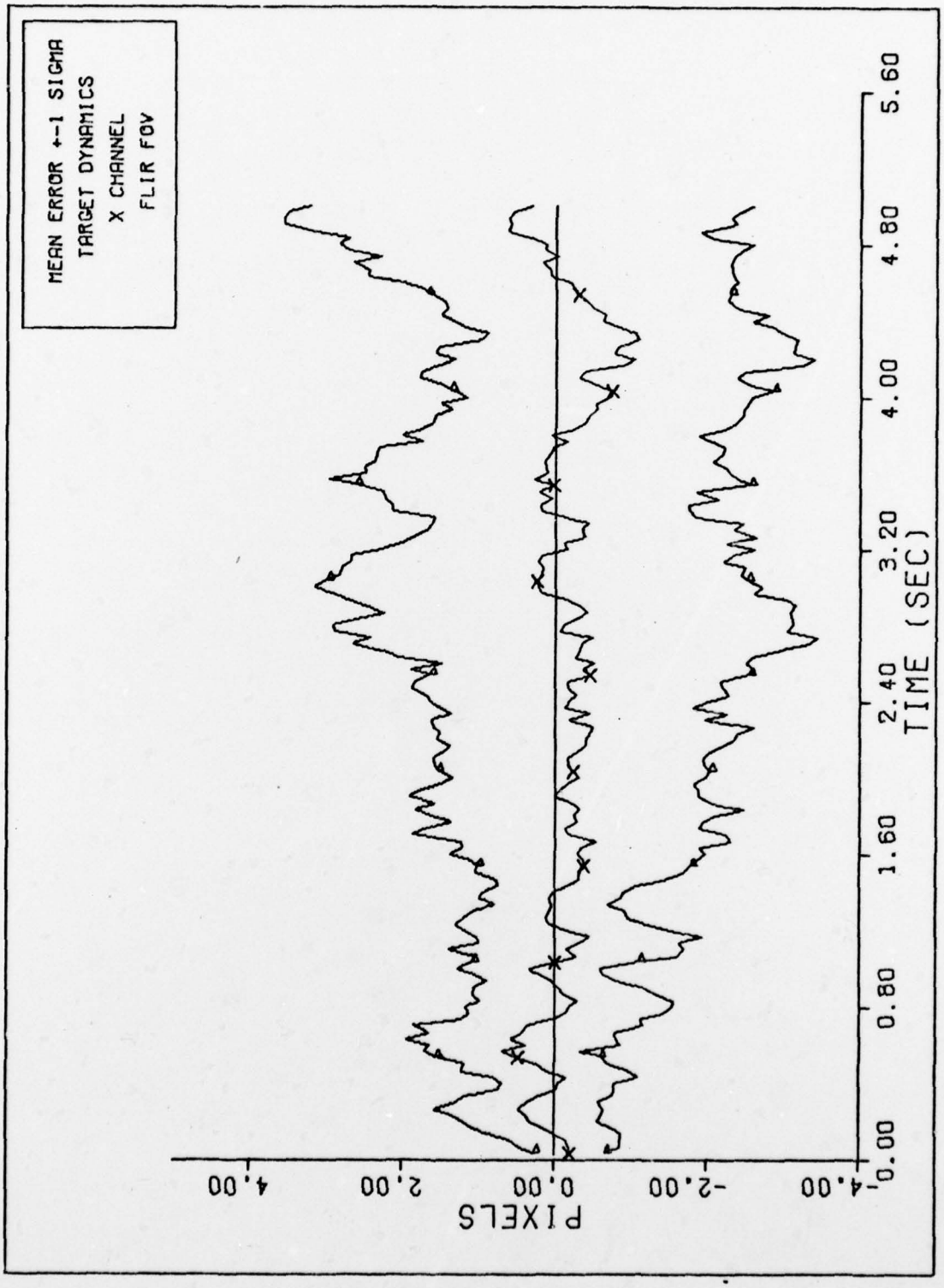


Figure 35b. X CHANNEL DYNAMICS ERROR (S/N=20)

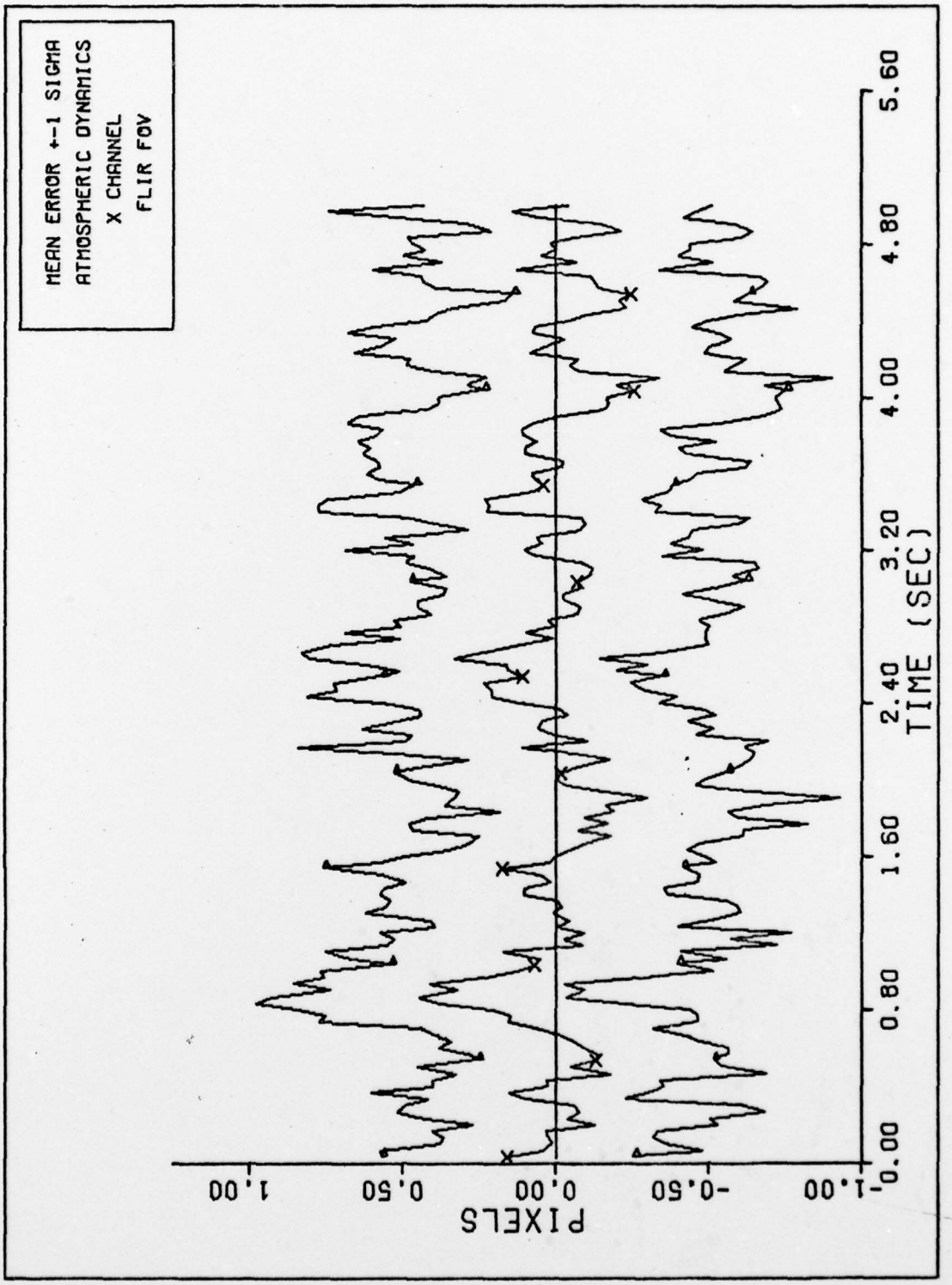


Figure 35c. X CHANNEL ATMOSPHERICS ERROR (S/N=20)

Vita

Daniel Edmond Mercier was born on December 7, 1950 in Biloxi, Mississippi. He graduated from Assumption Preparatory High School in Worcester, Massachusetts in June 1968 and entered the Air Force Academy that same month. In 1972, he graduated from the Air Force Academy with a Bachelor of Science degree in Engineering Sciences and a commission in the United States Air Force. From 1972 to 1977, he served as a trajectory engineer for the Strategic Air Command at Offutt Air Force Base, Nebraska. In 1977, Captain Mercier was assigned to the Air Force Institute of Technology in pursuit of a Master's Degree in Astronautical Engineering.

Permanent address: 1107 Border Avenue
Corona, California 91720

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER AFIT/GA/EE/78D-3	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) AN EXTENDED KALMAN FILTER FOR USE IN A SHARED APERTURE MEDIUM RANGE TRACKER		5. TYPE OF REPORT & PERIOD COVERED MS Thesis
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Daniel E. Mercier Captain, USAF		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS Air Force Institute of Technology (AFIT/EN) Wright-Patterson AFB, Ohio 45433		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Air Force Weapons Laboratory/ALO Kirtland AFB NM 87117		12. REPORT DATE December 1978
		13. NUMBER OF PAGES 156
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) Unclassified
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)
Approved for public release; distribution unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES
Approved for public release; IAW AFR 190-17

Joseph P. Hipps, Major, USAF
Director of Information 1-23-79

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)
Kalman filter
forward looking infrared (FLIR) sensor
atmospheric jitter

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

An extended Kalman filter algorithm, using outputs from a forward looking infrared (FLIR) sensor as measurements, is used to track a point source target in an open loop tracking problem. The filter estimates the translational position changes of the target in the FLIR field of view due to two effects: actual target motion, and apparent motion caused by atmospheric turbulence. Sixteen cases are examined to determine the performance of the filter as a function of signal-to-noise ratio, Gaussian beam, size, the ratio of RMS target motion to RMS atmospheric jitter, target correlation times, and →next page

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

20. mismatches between the true shape and the shape assumed by the filter. The performance of the extended Kalman filter is compared to the performance of an existing correlation tracker under identical initial conditions. A one sigma tracking error of .2 and .8 picture elements is obtained with signal-to-noise ratios of 20 and 1 respectively. No degradation in performance is observed when the beam size is decreased or when the target correlation time is increased over a limited range, when filter parameters are adjusted to reflect this knowledge. Sensitivity analysis shows that the filter is robust to minor changes in target intensity size.