Parallel Implementation of the Extended Square-Root Covariance Filter for Tracking Applications

Edward K. B. Lee, *Member, IEEE*, and Simon Haykin, *Fellow, IEEE*

Abstract— **Parallel implementations of the extended square-root covariance filter (ESRCF) for tracking applications are developed in this paper. The decoupling technique and special properties in the tracking Kalman filter (KF) are explored to reduce computational requirements and to increase parallelism. The application of the decoupling technique to the ESRCF results in the time and measurement updates of m decoupled (n/m)-dimensional matrices instead of 1 coupled n-dimensional matrix, where m denotes the tracking dimension and n denotes the number of state elements. The updates of m decoupled matrices are found to require approximately m times less processing elements and clock cycles than the updates of 1 coupled matrix. The transformation of the Kalman gain which accounts for the decoupling technique is found straightforward to implement. The sparse nature of the measurement matrix and the sparse, band nature of the transition matrix are explored to simplify matrix multiplications.**

Index Terms—**Decoupling technique, extended square-root covariance filter, Kalman filter, parallel implementation, systolic array, tracking KF properties, VLSI.**

I. Introduction

THE Kalman filter (KF) has been widely used in signal processing applications such as adaptive control, air and ship navigation, and target tracking. However, its high computational demand has limited its use to some extent. Nevertheless, due to the growing availability and cost effectiveness of VLSI technology, the use of the Kalman filter is becoming increasingly attractive. There have been a number of papers on the topic of parallel implementation of the KF ([3], [4], [6], [8], [9], [13], [16], [18], [19], [22], [24]). Most of these papers have focused on the implementation of either the standard KF or the extended covariance KF (ECKF), but not the extended square-root covariance filter (ESRCF), except for [3], [8], and [16].

Of the three papers discussing implementations of the ESRCF, [8] does not discuss the implementation of the nonlinear coordinate transformation from polar to Cartesian coordinates, or the linearization of the measurement matrix. References [3] and [16] present techniques for mapping the ESRCF onto linear arrays of programmable cells. These linear arrays are general-purpose architectures which are not specialized for the ESRCF. A general-purpose architecture is more flexible than a special-purpose architecture, but for a particular application, the former is not as efficient as the latter. In this paper, we develop parallel architectures specialized for the ESRCF for tracking applications.

The extended Kalman filter (EKF) is usually required in tracking systems to deal with a nonlinear coordinate transformation. This occurs when tracking is performed in Cartesian coordinates and measurements are made in polar coordinates in terms of range, azimuth angle, and elevation angle. The ESRCF has better numerical properties than the ECKF. However, the ESRCF requires more computation than the ECKF.

In the development of parallel architectures, a decoupling technique is used to reduce computational requirements and to increase parallelism ([2], [7], [20]). The use of the decoupling technique results in the propagation of m decoupled (n/m)-dimensional covariance matrices instead of 1 coupled n-dimensional covariance matrix, where m denotes the tracking dimension and n denotes the number of state elements. Furthermore, we exploit properties of the tracking KF to simplify its implementation.

In Section II, we review the standard square-root covariance filter, followed by the extended square-root covarince filter for tracking applications. In Section III, we develop a simplification of the extended square-root covariance filter using a decoupling technique and special properties of the tracking KF. In Section IV, we present parallel architectures for the extended square-root covariance filter. Finally, we present conclusions in Section V.

II. The Square-Root Covariance Filter

A. The Standard Square-Root Covariance Filter

The square-root covariance filter is a recursive linear state estimator, based on the *a priori* representation of system dynamics and measurement equations ([1], [12], [14]). The system dynamics and measurement equations are:

a) System dynamics equation:

$$X(k+1) = \phi(k)X(k) + D(k) \tag{1}$$

b) Measurement equation:

$$Z(k) = H(k)X(k) + E(k). \tag{2}$$

The system dynamics noise $D(k)$ and the measurement noise $E(k)$ are assumed to be zero mean, white Gaussian random sequences. Covariance matrices of $D(k)$ and $E(k)$ are defined by $E[D(k)D^T(i)] = Q(k)\delta_{ki}$ and $E[E(k)E^T(i)] = R(k)\delta_{ki}$, respectively, where E, T, and δ_{ki} denote the expectation

Manuscript received July 24, 1990; revised January 21, 1991.

E. K. B. Lee is with Motorola Inc., Ft. Lauderdale, FL 33322.
S. Haykin is with Communication Applied Research Laboratory, McMaster University, Hamilton, Ont., Canada.

IEEE Log Number 9206273.

operator, transpose operation, and the Kronecker delta, respectively. The square-root covariance filter (SRCF) is summarized below ([5], [15]):

Measurement update

$$\begin{bmatrix} F(k) & G(k) \\ 0 & S^T(k) \end{bmatrix} = Q_1 \begin{bmatrix} V^T(k) & 0 \\ S^T(k|k-1)H^T(k) & S^T(k|k-1) \end{bmatrix} \tag{3}$$

$$K(k) = G^T(k)(F^T(k))^{-1} \tag{4}$$

$$\hat{X}(k) = \hat{X}(k|k-1) + K(k)(Z(k) - H(k)\hat{X}(k|k-1)) \tag{5}$$

Time update

$$\begin{bmatrix} S^T(k+1|k) \\ 0 \end{bmatrix} = Q_1 \begin{bmatrix} S^T(k) & \phi^T(k) \\ & U^T(k) \end{bmatrix} \tag{6}$$

$$\hat{X}(k+1|k) = \phi(k)\hat{X}(k) \tag{7}$$

where Q_1 denotes an orthogonal matrix that upper-triangularizes a matrix on the right hand side and

$\hat{X}(k)$ = state estimate vector
$\hat{X}(k|k-1)$ = predicted state vector
$K(k)$ = gain matrix
$Z(k)$ = measurement vector
$H(k)$ = measurement matrix
$\phi(k)$ = transition matrix
$Q(k)$ = system dynamics noise covariance matrix $= \boldsymbol{E}[D(k)D^T(k)]$
$R(k)$ = measurement noise covariance matrix $= \boldsymbol{E}[E(k)E^T(k)]$
$P(k)$ = filtered state estimate error covariance matrix
$P(k|k-1)$ = predicted state estimate error covariance matrix
$U(k)$ = lower triangular matrix that is the square root of $Q(k)$
$V(k)$ = lower triangular matrix that is the square root of $R(k)$
$S(k)$ = lower triangular matrix that is the square root of $P(k)$
$S(k|k-1)$ = lower triangular matrix that is the square root of $P(k|k-1)$

B. The Extended Square-Root Covariance Filter

The standard square-root covariance filter, described in Section II-A, assumes a linear measurement equation. However, the measurement equation for tracking applications is generally nonlinear. A typical tracking radar measures range r and azimuth angle θ_A in two-dimensional tracking, and range r, azimuth angle θ_A and elevation angle θ_E in three-dimensional tracking. Specifically, the measurement vector $Z(k)$ for three-dimensional tracking is given by

$$Z(k) = [r(k) \quad \theta_A(k) \quad \theta_E(k)]^T.$$

Since the motion of a target is linear in Cartesian coordinates, the state vector for typical three-dimensional tracking is defined in Cartesian coordinates as follows:

$$X(k) = [x(k)\ y(k)\ z(k)\ \dot{x}(k)\ \dot{y}(k)\ \dot{z}(k)\ \ddot{x}(k)\ \ddot{y}(k)\ \ddot{z}(k)]^T$$

where $x(k)$, $\dot{x}(k)$, and $\ddot{x}(k)$ denote position, velocity, and acceleration of the target in the Cartesian coordinate x, respectively. The other elements in the state vector $X(k)$ are similarly defined in the Cartesian coordinates y and z. The nonlinear relationship between the state vector $X(k)$ and the measurement vector $Z(k)$ is expressed by

$$Z(k) = h(X(k)) + E(k) \tag{8}$$

where the three components of $h(X(k))$ for three-dimensional tracking are

$$r(k) = ((x(k))^2 + (y(k))^2 + (z(k))^2)^{1/2} \tag{9}$$

$$\theta_A(k) = \tan^{-1}(y(k)/x(k)) \tag{10}$$

$$\theta_E(k) = \tan^{-1}(z(k)/x(k)^2 + y(k)^2)^{1/2}). \tag{11}$$

To account for the nonlinear relationship between the state vector $X(k)$ and the measurement vector $Z(k)$, the idea of an extended SRCF involving the linearization of $h(.)$ has been proposed ([1], [12]). In this filter, the linearization of $h(.)$ described by the equation

$$H(k) = \left. \frac{\partial h(u)}{\partial u} \right|_{u=\hat{X}(k|k-1)}$$

is performed at every filtering instant. For three-dimensional tracking, the linearization of $h(.)$ results in

$$H(k) = \begin{bmatrix} \frac{x}{r} & \frac{y}{r} & \frac{z}{r} & 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{-y}{(x^2+y^2)} & \frac{x}{(x^2+y^2)} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{-xy}{r^2(x^2+y^2)^{1/2}} & \frac{-yz}{r^2(x^2+y^2)^{1/2}} & \frac{x^2+y^2}{r^2} & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

where x, y, and z denote the elements $\hat{x}(k|k-1)$, $\hat{y}(k|k-1)$, and $\hat{z}(k|k-1)$ of the predicted state vector $\hat{X}(k|k-1)$, respectively, and r denotes $(\hat{x}(k|k-1)^2 + \hat{y}(k|k-1)^2 + \hat{z}(k|k-1)^2)^{1/2}$ for simplicity.

In addition to the linearization of $h(.)$, a coordinate transformation of $X(k|k-1)$ is required in the ESRCF to compare the predicted state estimates in Cartesian coordinates and measurements in polar coordinates.

III. Simplification of the Kalman Filter

A. Simplification by the Decoupling Technique

The extended square-root covariance filter requires at each filtering instant a number of matrix multiplications, two orthogonal upper-triangularizations, a coordinate transformation, and linearization of the measurement equation. Computation can be reduced significantly by updating the square-root $S(k)$ of the state estimate error covariance matrix $P(k)$ in a decoupled system and evaluating the state estimate and predicted state estimate vectors in a coupled system ([2], [7], [20]). In a line of sight (LOS) Cartesian frame with the LOS as the x-axis, the covariance matrix $P_o(k)$ is decoupled and can be updated separately for each axis if the LOS frame does not rotate very much [20]. Fortunately, this condition is met in many radar tracking applications. It was found in [7] that the decoupling technique reduces the effects of truncation and round-off errors due to finite arithmetic. In this paper, the subscript o indicates the LOS frame.

The relationship between the polar coordinates (r, θ_A, θ_E) and the reference Cartesian coordinates (x, y, z) for three-dimensional tracking is described by

$$x = r\cos\theta_A\cos\theta_E \quad (12)$$

$$y = r\sin\theta_A\cos\theta_E \quad (13)$$

$$z = r\sin\theta_E \quad (14)$$

where r, θ_A, and θ_E denote range, azimuth angle, and elevation angle of a target, respectively. Based on (12), (13), and (14), the estimation errors in the reference Cartesian coordinates are expressed in terms of the estimation errors in the polar coordinates as follows:

$$\begin{bmatrix}\sigma_x\\ \sigma_y\\ \sigma_z\end{bmatrix} = \begin{bmatrix}\cos\theta_A\cos\theta_E & -r\sin\theta_A\cos\theta_E & -r\cos\theta_A\sin\theta_E\\ \sin\theta_A\cos\theta_E & r\cos\theta_A\cos\theta_E & -r\sin\theta_A\sin\theta_E\\ \sin\theta_E & 0 & r\cos\theta_E\end{bmatrix} \cdot \begin{bmatrix}\sigma_r\\ \sigma_{\theta_A}\\ \sigma_{\theta_E}\end{bmatrix} \quad (15)$$

where σ_x, σ_y, and σ_z denote the estimation errors in the reference Cartesian coordinates x, y, and z, respectively. Similarly, σ_r, σ_{θ_A}, and σ_{θ_E} denote the estimation errors in the polar coordinates r, θ_A and θ_E, respectively.

Using (15) and the relationships $\sigma_{0-x} = \sigma_r, \sigma_{0-y} = r\sigma_{\theta_A}$, and $\sigma_{0-z} = \sigma_{\theta_E}$ (where σ_{0-x}, σ_{0-y}, and σ_{0-z} are estimation errors in the LOS Cartesian x, y, and z-axes) the estimation errors in the reference Cartesian coordinates may be expressed in terms of the estimation errors in the LOS Cartesian coordinates as follows:

$$\begin{bmatrix}\sigma_x\\ \sigma_y\\ \sigma_z\end{bmatrix} = F_1\begin{bmatrix}\sigma_{0-x}\\ \sigma_{0-y}\\ \sigma_{0-z}\end{bmatrix} \quad (16)$$

where

$$F_1(k) = \begin{bmatrix}\cos\theta_A\cos\theta_E & -\sin\theta_A\cos\theta_E & -\cos\theta_A\sin\theta_E\\ \sin\theta_A\cos\theta_E & \cos\theta_A\cos\theta_E & -\sin\theta_A\sin\theta_E\\ \sin\theta_E & 0 & \cos\theta_E\end{bmatrix}. \quad (17)$$

Since the state estimate vector $\hat{X}(k)$ and predicted state vector $\hat{X}(k+1|k)$ are usually expressed in the reference Cartesian frame, the Kalman gain matrix for the reference Cartesian frame $K(k)$ is needed. It can be evaluated from the Kalman gain matrix for the LOS Cartesian frame $K_o(k)$, using (16), as shown below:

$$K(k) = \begin{bmatrix}F_1(k) & 0 & 0\\ 0 & F_1(k) & 0\\ 0 & 0 & F_1(k)\end{bmatrix} K_o(k). \quad (18)$$

The simplified extended square-root covariance filter has the update of the decoupled covariance matrices in the LOS frame $S_o(k)$'s, and the transformation of the Kalman gain matrix from the LOS to reference Cartesian frames. The simplified ESRCF is referred to as the decoupled ESRCF, and it is summarized as follows:

1) $S_o(k), S_o(k+1|k)$, and $K_o(k)$ are processed in the LOS frame:

$$\begin{bmatrix}S_o^T(k|k-1)\\ 0\end{bmatrix} = Q_1\begin{bmatrix}S_o^T(k-1)\,\phi_o^T(k-1)\\ U_o^T(k)\end{bmatrix} \quad (19)$$

$$\begin{bmatrix}F_o(k) & G_o(k)\\ 0 & S_o^T(k)\end{bmatrix} = Q_1\begin{bmatrix}V_o^T(k) & 0\\ S_o^T(k|k-1)H_o^T(k) & S_o^T(k|k-1)\end{bmatrix} \quad (20)$$

$$K_o(k) = G_o^T(k)(F_o^T(k))^{-1}. \quad (21)$$

The matrix Q_1 denotes an orthogonal matrix that upper-triangularizes a matrix on its righthand side.

2) $K(k)$ is calculated from $K_o(k)$ using the Jacobian transformation:

$$K(k) = \begin{bmatrix}F_1 & 0 & 0\\ 0 & F_1 & 0\\ 0 & 0 & F_1\end{bmatrix} K_o(k) \quad (22)$$

where F_1 is defined in (17) for three-dimensional tracking system.

3) $\hat{X}(k)$ and $\hat{X}(k+1|k)$ are processed in the reference Cartesian coordinates frame:

$$\hat{X}(k) = \hat{X}(k|k-1) + K(k)(Z(k) - h(\hat{X}(k|k-1))) \quad (23)$$

$$\hat{X}(k|k+1) = \phi(k)\hat{X}(k). \quad (24)$$

In the decoupled ESRCF, m decoupled (n/m)-dimensional $S_o(k)$'s are updated separately for each axis, instead of one coupled n-dimensional $S(k)$ for all axes, where n is the number of state elements and m is the tracking dimension. It is assumed that n is a multiple of m. Fortunately, this assumption is generally valid for typical tracking applications. For a typical three-dimensional tracking, n and m are 9 and 3, respectively.

The number of required operations for the update of m decoupled square-roots, $S_o(k)$'s is less than that for the update of 1 coupled $S(k)$ by a factor of between m^2 and m. The reason for this reduction factor is that the required orthogonalization for the update of $S_o(k)$ is a combination of an order of n^3 and n^2 operations. The decoupling technique reduces an order of n^3 operations by a factor of m^2, and an order of n^2 operations by a factor of m. The update of m decoupled covariance matrices for each axis can be propagated simultaneously; the decoupling technique thus increases parallelism.

B. Simplification by the Use of Special Properties of the Tracking KF

A typical transition matrix $\phi(k)$ that relates the state vectors $X(k)$ and $X(k+1)$ is a sparse, band matrix [21], as shown below:

$$\phi = \begin{bmatrix} I & TI & 0 \\ 0 & I & I \\ 0 & 0 & \rho I \end{bmatrix} \tag{25}$$

where I is a (3×3) identity matrix, TI and ρI are (3×3) identiy matrices multiplied by constants T and ρ, respectively. T is the sampling period and ρ is a correlation coefficient for acceleration. The correlation coefficient for acceleration ρ is used to specify the acceleration characteristics of a target. The transition matrix may be separated for each Cartesian coordinate in the LOS frame, as follows:

$$\phi_{o-x} = \phi_{o-y} = \phi_{o-z} = \begin{bmatrix} 1 & T & 0 \\ 0 & 1 & 1 \\ 0 & 0 & \rho \end{bmatrix} \tag{26}$$

where the subscripts $o-x$, $o-y$, and $o-z$ denote axes in the LOS frame. In this paper, when an axis is not needed to be specified, ϕ_o is used in place of ϕ_{o-x}, ϕ_{o-y}, and ϕ_{o-z}.

The sparse, band matrix properties of $\phi_o(k)$ can be utilized to simplify matrix multiplication. The multiplication of $S_{o-x}^T(k)$ and $\phi_{o-x}^T(k)$ for x-axis in (19) is expanded below:

$$\begin{aligned} S_o^T(k)\phi^T(k) &= \begin{bmatrix} s_{11} & s_{21} & s_{31} \\ 0 & s_{22} & s_{32} \\ 0 & 0 & s_{33} \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ T & 1 & 0 \\ 0 & 1 & \rho \end{bmatrix} \\ &= \begin{bmatrix} s_{11} + Ts_{21} & s_{21} + s_{31} & \rho s_{31} \\ Ts_{22} & s_{22} + s_{32} & \rho s_{32} \\ 0 & s_{33} & \rho s_{33} \end{bmatrix}. \end{aligned}$$

This expansion indicates that a column of the output matrix is a linear combination of two columns of $S_o^T(k)$. This band matrix multiplication requires $2 * m^2$ multiplications and m^2 additions, where m is the dimension of the matrix. This requirement is an order of magnitude less than the conventional requirement of a matrix-matrix multiplication, which is m^3 multiplications and $m^2(m-1)$ additions. In addition to the reduction in computational requirements, the use of the sparse, band matrix properties of $\phi_o(k)$ simplifies implementation, as described in Section IV.

In the LOS frame, the measurement matrix $H(k)$ becomes a sparse matrix, as shown below:

$$H_o(k) = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & r^{-1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & r^{-1} & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{27}$$

where r denotes the target range. The matrix $H_o(k)$ can be divided for each axis, as follows:

$$H_{o-x} = [1 \quad 0 \quad 0] \tag{28}$$

$$H_{o-y} = [r^{-1} \quad 0 \quad 0] \tag{29}$$

$$H_{o-z} = [r^{-1} \quad 0 \quad 0]. \tag{30}$$

The sparse nature of the measurement matrix $H_o(k)$ for the LOS frame could be utilized to further simplify the ESRCF equations. In the multiplication of $S_o^T(k|k-1)$ and $H_o^T(k)$ in (20), the use of the sparse nature of the measurement matrix $H_{0-x}(k)$ for the x-axis turns the multiplication of $S_{o-x}^T(k|k-1)$ and $H_{o-x}^T(k)$ into the extraction of the first column vector from $S_{o-x}^T(k|k-1)$. The multiplication of $S_{o-x}^T(k|k-1)$ and $H_{o-x}^T(k)$ therefore does not require any multiplication or addition. Similarly, the multiplication of $S_{o-y}^T(k|k-1)$ and $H_{o-y}^T(k)$, and that of $S_{o-z}^T(k|k-1)$ and $H_{o-z}^T(k)$ are simplified.

The fact that the product of $S_o^T(k|k-1)$ and $H_o^T(k)$ for each axis has only one nonzero element may be used to simplify the orthogonal upper triangularization needed for the measurement update in (20). The orthogonal triangularization for the measurement update for the x-axis is expressed in detail below:

$$\begin{bmatrix} f_{11} & g_{11} & g_{12} & g_{13} \\ 0 & s_{11} & s_{21} & s_{31} \\ 0 & 0 & s_{22} & s_{32} \\ 0 & 0 & 0 & s_{33} \end{bmatrix} = Q_1 \begin{bmatrix} \sigma_r & 0 & 0 & 0 \\ s_{11}^- & s_{11}^- & s_{21}^- & s_{31}^- \\ 0 & 0 & s_{22}^- & s_{32}^- \\ 0 & 0 & 0 & s_{33}^- \end{bmatrix}.$$

This equation shows that only one element s_{11}^-, which is the first element on the second row, is to be nullified in this orthogonalization. Similarly, only one element needs to be nullified in the orthogonal triangularizations for measurement updates for the y- and z-axes. It is shown in Section IV that the utilization of the requirement of only one element to be nullified simplifies the implementation of (20).

IV. Parallel Implementation of the Extended SRCF

In this section, we present parallel architectures for the decoupled extended square-root covariance filter for tracking applications, described in (19) to (24). In Section III, these equations were broken down into 3 parts by the decoupling technique. Correspondingly, the architecture for the decoupled ESRCF may be split into 3 parts in the following way:

1) The processor for the LOS coordinates computes $S_o(k|k-1), S_o(k)$ and $K_o(k)$ [(19), (20), and (21)]
2) The processor for the coordinate transformation computes $K(k)$ from $K_o(k)$ by using the Jacobian transformation [(22)]
3) The processor for the reference Cartesian coordinates computes $\hat{X}(k)$ and $\hat{X}(k+1|k)$ [(23) and (24)]

The processor for the LOS frame, that for the coordinate transformation of $K_o(k)$, and that for the reference Cartesian coordinates will be hereafter referred to as Processor 1, Processor 2, and Processor 3, respectively. Fig. 1 shows a block digram of the tracking KF implementation. Processor 1 that implements (19), (20), and (21) needs to perform two QR-decompositions and one division of a vector by a scalar number. The QR-decomposition may be performed by using various algorithms such as Gram-Schmidt orthogonalization,

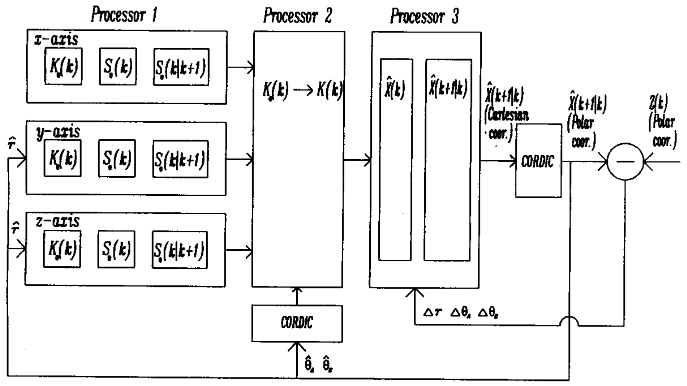


Fig. 1. Block diagram of the ESRCF.

Householder transformation, and Givens rotation [11]. Givens rotation using a QR systolic array [10] is chosen for an implementation of the QR-decomposition, because the QR systolic array has desirable characteristics for a parallel architecture such as modularity, regularity, local interconnection, and a high degree of pipelining and parallelism.

The two QR-decompositions can be implemented using one or two QR systolic arrays. However, sharing one systolic array for two QR-decompositions is difficult, because the delay elements and multipliers necessary for (19) have to be somehow bypassed in calculating (20). Hence, an architecture based on two QR systolic arrays is chosen for the implementation of the ESRCF.

A triangular systolic array for the QR-decomposition, described in [10], assumes that an input matrix to be triangularized by orthogonalization enters a systolic array from the top row to the bottom row. We find that this systolic array can be easily modified to allow an input matrix to enter the systolic array in reverse order, that is, from the bottom row to the top row. The use of these two different systolic arrays for the implementation of Processor 1 results in two different architectures: one with a unidirectional bus, and the other with a bidirectional bus.

An architecture for Processor 1, based on a unidirectional bus, requires longer interconnection than an architecture with a bidirectional bus. However, the unidirectional bus is usually easier to control than the bidirectional bus. It should be noted that an architecture based on a unidirectional bus requires a unidirectional interconnection among processing elements, and vice versa for an architecture based on a bidirectional bus. Hereafter, we will call a processor with a bidirectional bus Processor 1A, and a processor with a unidirectional bus Processor 1B.

In this section, for convenience of presentation, we assume that the number of tracking dimension, m, is 3 and that the number of state elements, n, is 9.

A. Processor 1A for the LOS Coordinates

The proposed architecture for Processor 1A is shown in Fig. 2. We use a systolic array on the top for (20), and one at the bottom for (19) in Fig. 2.

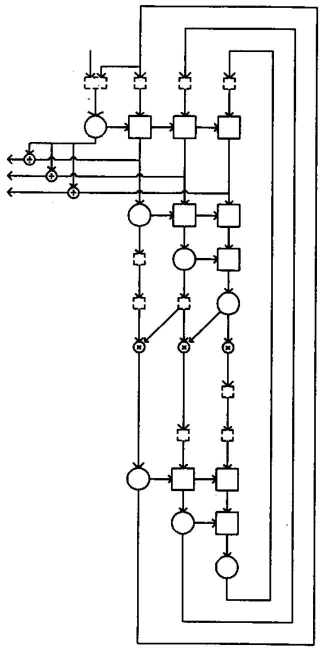


Fig. 2. Implementation of Processor 1A for the LOS frame.

The implementation of (20) for the x-axis and the corresponding input matrix flow are shown in Fig. 3. This implementation is based on a QR systolic array with 10 processing elements and the input data matrix entering the systolic array bottom row first in a skewed manner. Note that in the input data stream, nonzero elements are preceded by a number of zero elements, corresponding to the zero elements at the bottom left side of the input matrix. We may eliminate the first two rows of zeros in the input data stream by making internal cells generate 0 and 1 for $\cos\theta$ and $\sin\theta$, respectively, right before the input matrix enters the systolic array. This ensures that the elements s_{22}^{-} and s_{33}^{-} would be treated as if the two rows of zeros were not eliminated.

If we assume that at the end of the calculation of (19) the matrix $S_o(k|k-1)$ is stored in the systolic array for (19), then

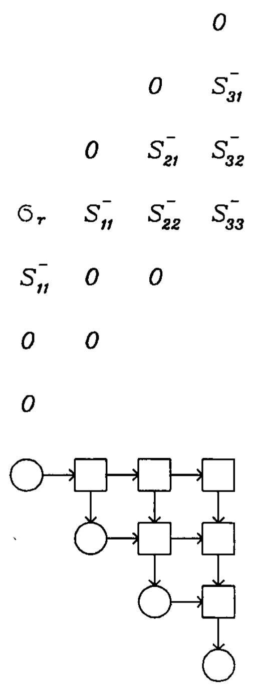


Fig. 3. Implementation of (20).

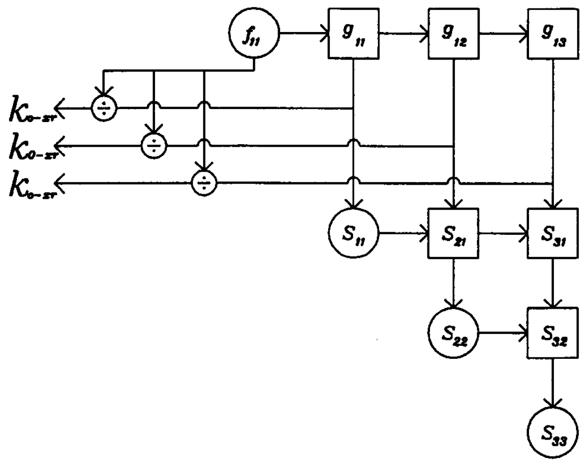


Fig. 4. Calculation of $K_o(k)$, (21).

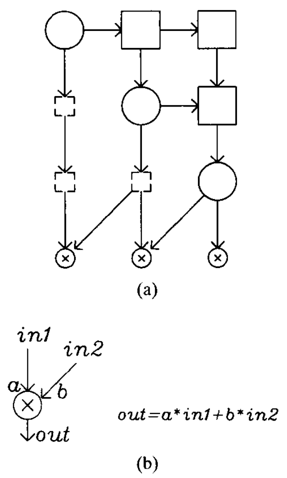


Fig. 5. Multiplication of S_o^T and ϕ_o^T. (a) Structure. (b) Function of a cell.

the data flow, shown in Fig. 3, can be realized by connecting two systolic arrays for (19) and (20), as shown in Fig. 2. The implementation of (20) requires 7 clock cycles from the completion of (19) to the completion of the QR-decomposition.

Since the result of (20) is stored in the systolic array at the end of the calculation of (20), we find that (21) may be performed by reading out $G_o^T(k)$ and $F_o^T(k)$ from the systolic array and dividing $G_o^T(k)$ by $F_o^T(k)$. This process, which requires 3 dividers, is shown in Fig. 4.

We now consider an implementation of (19). It consists of a matrix-matrix multiplication of $S_o^T(k)$ and $\phi_o^T(k)$, followed by the QR-decomposition. A matrix-matrix multiplication can be performed in various ways [17]. We exploit the sparse, band matrix property of $\phi^T(k)$ to simplify the multiplication of $S_o^T(k)$ and $\phi_o^T(k)$. In particular, each column of the multiplication result is a linear combination of two columns of $S_o^T(k)$. Hence, the multiplication of $S_o^T(k)$ and $\phi_o^T(k)$ is implemented by multiplying $S_o^T(k)$, which is stored in the systolic array at the end of (20), by $\phi_o^T(k)$, as shown in Fig. 5(a). In Fig. 5(a), the QR systolic array is connected to three delay elements represented by dotted squares, and three multipliers represented by circles with a cross inside. The delay elements synchronize the elements of $S_o^T(k)$ for a matrix-matrix multiplication. The functionality of a multiplier is presented in Fig. 5(b). This implementation requires an order of n multipliers and clock cycles, where n denotes the number of state elements in the state vector $\hat{X}(k)$.

The proposed implementation in Fig. 5 using the sparse, band matrix property of $\phi_o(k)$ is much more efficient than the conventional implementation for two reasons: 1) It does not require to read any matrix to be multiplied into the systolic array. 2) It takes only an order of n multipliers, compared to an order of n^2 multipliers for the conventional implementation.

After the multiplication of $S_o^T(k)$ and $\phi_o^T(k)$, the QR-decomposition is performed on a matrix shown in detail below:

$$\begin{bmatrix} S_o^T(k)\phi_o^T(k) \\ \\ U_o^T \end{bmatrix} = \begin{bmatrix} b_{11} & b_{21} & b_{31} \\ b_{12} & b_{22} & b_{32} \\ b_{13} & b_{23} & b_{33} \\ u_{11} & u_{21} & u_{31} \\ 0 & u_{22} & u_{32} \\ 0 & 0 & u_{33} \end{bmatrix}$$

where the elements of $S_o^T(k)\phi_o^T(k)$ are denoted by b_{ij}'s. Typically, for the QR-decomposition of a matrix, all the elements in the input matrix have to be fed into a systolic array. However, we can use the upper triangular shape of $U_o^T(k)$ to eliminate feeding $U_o^T(k)$ into the systolic array. An upper triangular matrix does not change, when the QR-decomposition is applied to it. Hence, after preloading $U_o^T(k)$

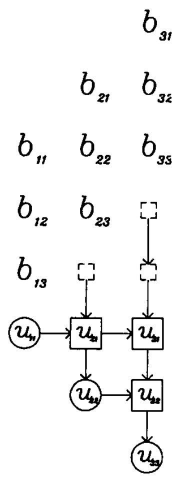


Fig. 6. Implementation of (19).

at the beginning of filtering, we need to feed only the product of $S_o^T(k)$ and $\phi_o^T(k)$, as shown in Fig. 6. It reduces the computational time by three clock cycles. Note that $U_o^T(k)$ is assumed constant, which is usually valid in typical tracking environments. This QR-decomposition requires 6 processing elements. The computation of (19) requires 8 clock cycles.

At the end of the calculation of (19), $S_o^T(k+1|k)$ is stored in the systolic array, as assumed earlier in describing the implementation of (20).

B. Processor 1B for the LOS Coordinates

The proposed architecture for Processor 1B is shown in Fig. 7. We use a systolic array on the top of Fig. 7 for (20), and one at the bottom for (19).

The implementation of (20) is first considered. Fig. 8 shows an implementation of (20) with the flow of an input data matrix. This implementation, consisting of 10 processing elements, shows that in contrast to the implementation in Fig. 3, the elimination of any row of the input matrix from the input data stream is impossible, because of the reversed direction of the input data flow. With the assumption that the systolic array for (19) stores $S_o(k+1|k)$ at the end of (19), the required data flow, illustrated in Fig. 8, is realized when systolic arrays for (19) and (20) are connected through 6 delay elements, as shown in Fig. 7. Equation (20) requires 10 clock cycles to compute.

As in Processor 1A, (21), the calculation of the Kalman gain $K_o(k)$ can be performed by reading out $F_o(k)$ and $G_o(k)$ stored in the systolic array for (20) at the end of the calculation of (20), and dividing $G_o(k)$ by $F_o(k)$. Its implementation, which consists of 3 dividers, is illustrated in Fig. 7.

We now consider the implementation of (19). This equation has two parts: a matrix-matrix multiplication, and the QR-decomposition. The matrix-matrix multiplication in (19) can be performed by combining two columns of $S_o^T(k)$ for each column of the output matrix, as described in Section IV-A. This implementation requires three arithmetic units, whose function is described in Fig. 5(b), but it does not require any delay element in contrast to the implementation in Fig. 5(a). The zero elements in $S_o^T(k)$, not stored in the systolic array, but needed in multiplying $S_o^T(k)$ and $\phi_o^T(k)$, may be generated by the boundary cells after the content of the boundary cells is passed out.

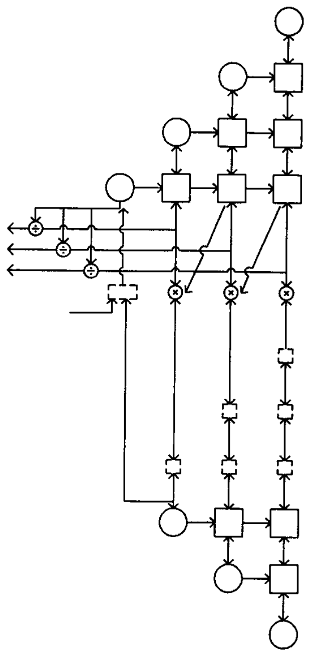


Fig. 7. Implementation of Processor 1B for the LOS frame.

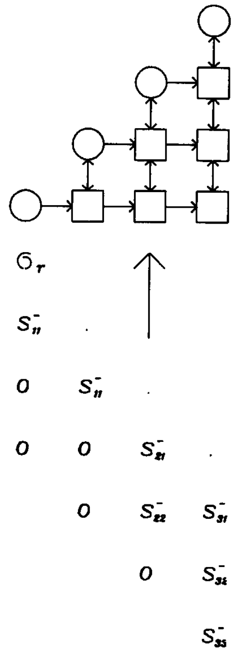


Fig. 8. Implementation of (20).

Fig. 9 shows an implementation of the second phase of (19), the QR-decomposition. In Fig. 9, the product of $S_o^T(k)$ and $\phi_o^T(k)$, represented by the b_{ij}'s, enters the systolic array before $U_o^T(k)$, because the systolic array is designed to receive first the bottom row of an input matrix to be triangularized. This means that $U_o^T(k)$ has to go through the QR-decomposition after the product of $S_o^T(k)$ and $\phi_o^T(k)$ goes through, and that $U_o^T(k)$ cannot be prestored in the systolic array for (19). However, $U_o^T(k)$ can be prestored in the systolic array for (20), and be fetched after $S_o^T(k)$ is read from the systolic array. The complete implementation of (19) requires 10 clock cycles. Note that $S_o^T(k+1|k)$ is stored in the systolic array at the end of (19). The assumption made earlier in describing the implementation of (20) is validated.

C. Processor for the Coordinate Transformation of the Kalman Gain

Equation (22) transforms the gain matrix from the LOS frame to the reference Cartesian coordinates frame. Equation (22) for two-dimensional tracking is expanded below:

$$K(k) = \begin{bmatrix} k_{xr} & k_{x\theta_A} \\ k_{yr} & k_{y\theta_A} \\ k_{\dot{x}r} & k_{\dot{y}\theta_A} \\ k_{\dot{y}r} & k_{\dot{y}\theta_A} \\ k_{\ddot{x}r} & k_{\ddot{y}\theta_A} \\ k_{\ddot{y}r} & k_{\ddot{y}\theta_A} \end{bmatrix}$$

$$= \left[\begin{array}{c|c|c} F_1 & 0 & 0 \\ \cdots & \cdots & \cdots \\ 0 & F_1 & 0 \\ \cdots & \cdots & \cdots \\ 0 & 0 & F_1 \\ & & \end{array}\right] \begin{bmatrix} k_{o-xr} & 0 \\ 0 & k_{o-y\theta_A} \\ k_{o-\dot{x}r} & 0 \\ 0 & k_{o-\dot{y}\theta_A} \\ k_{o-\ddot{x}r} & 0 \\ 0 & k_{o-\ddot{y}\theta_A} \end{bmatrix}$$

$$F_1(k) = \begin{bmatrix} \cos\theta_A & -\sin\theta_A \\ \sin\theta_A & \cos\theta_A \end{bmatrix}$$

where k_{xr} denotes an element of $K(k)$ that relates $r(k)$ element in the measurement vector $Z(k)$ to the $x(k)$ element in the state vector $X(k)$. Similarly, k_{o-xr} denotes an element of $K_o(k)$ that relates $r(k)$ element in the measurement vector $Z(k)$ to the $x_o(k)$ element in $X_o(k)$. The other elements in $K(k)$ and $K_o(k)$ are similarly defined. Note that elements in $K_o(k)$ that are always zero are indicated by zeros in corresponding positions.

This expanded form indicates that each component of the Kalman gain matrix $K(k)$ is a product of a component of matrix $K_o(k)$ and a component of matrix $F_1(k)$. The implementation of (22) is shown in Fig 10; it requires 12 multipliers to multiply a component of $K_o(k)$ and the corresponding component of $F_1(k)$. This is a small price to pay for the reduction in computational requirement realized by the use of decoupling. The use of the decoupling technique is found in Section III to reduce the computational requirements of the update of $S(k)$ by a factor of between m^2 and m, where m is the tracking dimension.

The implementation in Fig. 10 generates all the gain matrix elements at the same time. It may be modified to produce the gain matrix elements in a pipelining manner. However, a pipeline implementation requires more computational time and delay elements to synchronize incoming data at processing elements than a broadcasting implementation, even though the data bus in the former implementation is simpler than that in the latter implementation.

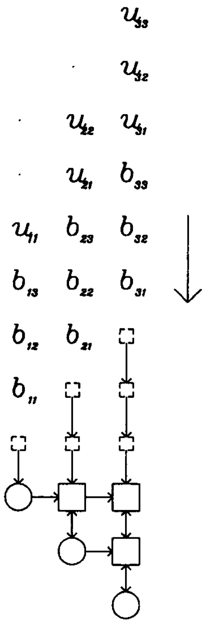


Fig. 9. QR decomposition for (19).

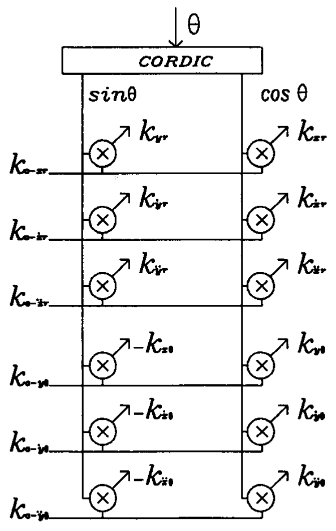


Fig. 10. Transformation of the Kalman gain in a broadcasting manner, (22).

D. Processor for the Reference Cartesian Coordinates

Equation (23) that estimates the state vector may be separated into the following three parts:

a) $Z(k|k-1) = h(X(k|k-1))$

b) $\Delta Z = Z(k) - Z(k|k-1)$

c) $\hat{X}(k) = \hat{X}(k|k-1) + K(k)\,\Delta Z.$

The first part a) is a coordinate transformation of $\hat{X}(k|k-1)$, which may be implemented using the Coordinate Rotation Digital Computer (CORDIC) [23]. The CORDIC is described in Section IV-E. The second part b) calculates the correction vector ΔZ. It requires 2 adders for two-dimensional tracking, and 3 adders for three-dimensional tracking. The third part c) of (23) for two-dimensional tracking may be expanded as follows:

$$\hat{X}(k) = \hat{X}(k|k-1) + K(k)\,\Delta Z$$

$$\begin{bmatrix} x(k) \\ \dot{x}(k) \\ \ddot{x}(k) \\ y(k) \\ \dot{y}(k) \\ \ddot{y}(k) \end{bmatrix} = \begin{bmatrix} x(k|k-1) \\ \dot{x}(k|k-1) \\ \ddot{x}(k|k-1) \\ y(k|k-1) \\ \dot{y}(k|k-1) \\ \ddot{y}(k|k-1) \end{bmatrix} + \Delta r \begin{bmatrix} k_{xr} \\ k_{\dot{x}r} \\ k_{\ddot{x}r} \\ k_{yr} \\ k_{\dot{y}r} \\ k_{\ddot{y}r} \end{bmatrix} + \Delta\theta \begin{bmatrix} k_{x\theta} \\ k_{x\theta} \\ k_{\ddot{x}\theta} \\ k_{y\theta} \\ k_{\dot{y}\theta} \\ k_{\ddot{y}\theta} \end{bmatrix}$$

where Z is defined as follows:

$$\Delta Z = \begin{bmatrix} \Delta r \\ \Delta\theta \end{bmatrix} = \begin{bmatrix} \text{correction factor in range} \\ \text{correction factor in azimuth angle} \end{bmatrix}.$$

The expansion shows that the state estimate vector $\hat{X}(k)$ is a sum of the state prediction vector $\hat{X}(k|k-1)$, the first column vector of $K(k)$ scaled by Δr, and the second column vector of $K(k)$ scaled by $\Delta\theta$.

Fig. 11 shows a broadcast implementation of part c) of (23) for 2-dimensional tracking. The architecture in Fig. 11 consists of two columns of six multipliers and one column of processing elements. In this architecture, Δr, $\Delta\theta$, and the elements of $K(k)$ from Processor for the coordinate transformation of the Kalman gain are broadcasted to necessary multipliers. Six multipliers on the left implement the multiplication of the first column of $K(k)$ by the broadcasted r, while six multipliers on the right implement the multiplication of the second column of $K(k)$ by $\Delta\theta$. Six processing elements in the middle add 3 following vectors: 1) the result of left hand column of multipliers, 2) the result of right hand column of multipliers, and (3) the prestored vector $\hat{X}(k|k-1)$. The implementation of part c) of (23) requires 2 clock cycles.

Equation (24), predicting the state vector, requires a matrix-vector multiplication. The implementation of (24) may be simplified using the property of the sparse, band transition matrix $\phi(k)$ in the same way as in the implementation of (19). In Fig. 11, processing elements for (23) are interconnected. This interconnection is used to move the elements of $\hat{X}(k)$ for the calculation of $\hat{X}(k+1|k)$ in (24). These interconnected processing elements alternatively store elements of the state vector $\hat{X}(k)$ and those of the predicted state vector $\hat{X}(k+1|k)$, and perform additions and multiplications for (23) and (24). The implementation of (24) in Fig. 11 requires 2 clock cycles. This requirement is very small compared to that of the conventional implementation of a matrix-vector multiplication. The conventional implementation requires an order of n clock cycles on a systolic array of an order of n processing elements, where n is the dimension of the array. Hence, the use of the sparse, band property of $\phi(k)$ reduces the computational time requirement of a matrix-vector multiplication by an order of magnitude.

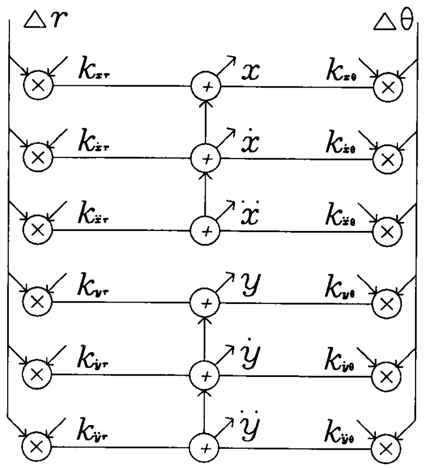
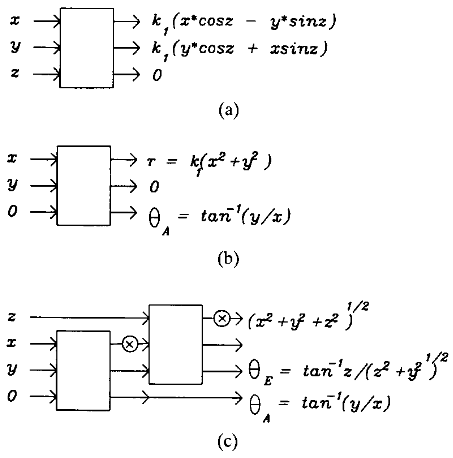


Fig. 11. Estimation of the state $X(k)$ and prediction of the state $X(k+1)$ in a broadcasting manner, (23) and (24).



$k_t(x{*}\cos z - y{*}\sin z)$
$k_t(y{*}\cos z + x\sin z)$
0
(a)
$r = k_t(x^2+y^2)$
0
$\theta_A = \tan^{-1}(y/x)$
(b)
$(x^2+y^2+z^2)^{1/2}$
$\theta_E = \tan^{-1} z/(x^2+y^2)^{1/2}$
$\theta_A = \tan^{-1}(y/x)$
(c)

Fig. 12. Coordinate digital computer (CORDIC). (a) CORDIC for the evaluation of trigonometric functions. (b) CORDIC for the two-dimensional coordinate transformation. (c) Series of two CORDIC's for the three-dimensional coordinate transformation.

E. Coordinate Rotation Digital Computer (CORDIC)

Equation (22) requires the evaluation of trigonometric functions and (23) requires a coordinate transformation from Cartesian to polar coordinates. Volder [23] proposed the coordinate rotation digital computer (CORDIC), which is suitable for two-dimensional coordinate transformation and the evaluation of trigonometric functions. These transformations are shown in Fig. 12(a) and (b). It is found that Volder's scheme can be extended to three-dimensional coordinate transformation by placing two CORDIC's in series and a scaler between the CORDIC's, as shown in Fig. 12(c).

F. Requirements of Hardware and Computational Time

We first examine the hardware and computational time requirements of the QR-decomposition, which is a major part

TABLE I
HARDWARE REQUIREMENTS FOR THE ESRCF (USING PROCESSOR 1A) FOR m-DIMENSIONAL TRACKING WITH n STATE ELEMENTS

Eq. number	19,20,21	22	23,24	Total
PE.	$\frac{n^2}{m}+2n+m$	0	n	$\frac{n^2}{m}+3n+m$
Mul.*	n	nm+4	nm−3	2nm+n+1
Div.	n+m−1	0	0	n+m−1
Add.	0	0	m	m
Cordic	0	m−1	m−1	2m−2
Mux.	m	0	0	m
Delay.	$\frac{n^2}{m}$	0	0	$\frac{n^2}{m}$

* For 2–dimensional tracking, Eqation (22), and Equations (23) and (24) both require nm multipliers; the total number of required multipliers is 2nm+n.

TABLE II
HARDWARE REQUIREMENTS FOR THE ESRCF (USING PROCESSOR 1B) FOR m-DIMENSIONAL TRACKING WITH n STATE ELEMENTS

Eq. number	19,20,21	22	23,24	Total
PE.	$\frac{n^2}{m}+2n+m$	0	n	$\frac{n^2}{m}+3n+m$
Mul.*	n	nm+4	nm−3	2nm+n+1
Div.	n+m−1	0	0	n+m−1
Add.	0	0	m	m
Cordic	0	m−1	m−1	2m−2
Mux.	m	0	0	m
Delay.	$\frac{n^2}{2m}+\frac{n}{2}$	0	0	$\frac{n^2}{2m}+\frac{n}{2}$

* For 2–dimensional tracking, Eqation (22), and Equations (23) and (24) both require nm multipliers; the total number of required multipliers is 2nm+n.

of (19) and (20). The QR-decomposition of the $(n \times n)$ matrix $S(k)$ in the coupled ESRCF requires $n(n+1)/2$ processing elements and $3n-2$ clock cycles, whereas the QR-decomposition of the $((n/m) \times (n/m))$ matrix $S_o(k)$ in the decoupled ESRCF requires $(n/m)(n/m+1)/2$ processing elements and $3n/m-2$ clock cycles. Hence, the QR-decomposition of m decoupled $((n/m) \times (n/m))$ matrices $S_o(k)$'s in the decoupled ESRCF requires approximately m times less processing elements and computational time than that of the $(n \times n)$ matrix $S(k)$ in the coupled ESRCF.

The product of the reduction ratio for the number of processing elements and that for the computational time is m^2, and it is greater than the overall computational reduction ratio by the decoupling technique, which is found to be between m^2 and m in Section III-A. This can be explained by the fact that the decoupling technique reduces only the number of internal processing elements, not that of computationally intensive boundary processing elements.

Tables I and II present the hardware requirements for each processor and the total hardware requirements for the parallel implementation of the extended SRCF using Processors 1A and 1B respectively. A comparison of Tables I and II shows that the implementations using Processors 1A and 1B require the same number of arithmetic units except for delay elements. The implementation based on Processor 1A requires more delay elements than that based on Processor 1B for the following two reasons:

1) The delay elements in Processor 1B are shared both in the time and measurement updates due to the bidirectional bus.
2) The delay elements that synchronize $S_o^T(k-1)$ for the multiplication $S_o^T(k-1)$ and $\phi_O^T(k-1)$ in Processor 1A are not necessary in Processor 1B.

Figs. 13 and 14 show the number of clock cycles needed to implement each KF equation of the ESRCF for three-dimensional tracking and how the equations can be performed in parallel according to their interdependence. Figs. 13 and 14 are based on Processors 1A and 1B, respectively. Here we assume that multiplication, division, CORDIC calculation, and addition are all performed within 1 cycle. In Figs. 13 and 14, (22) is divided into two parts and (23) is divided into three parts. Equation (22), which rotates the Kalman gain $K_o(k)$ for the LOS frame by the rotation matrix, may be performed in two steps:

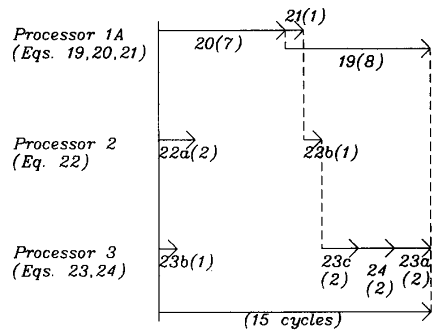


Fig. 13. Parallel computation of the simplified ESRCF using Processor 1A (numbers in brackets indicate the number of clock cycles required for each step).

1) The calculation of $\sin\theta$ and $\cos\theta$ in $F_1(k)$ immediately after $\theta(k|k-1)$ is determined.
2) The rotation of the Kalman gain $K_o(k)$ by the rotation matrix $F_1(k)$ is computed.

The precalculation of sin and cos allows the rotation of the Kalman gain $K_o(k)$ to be performed immediately after $K_o(k)$ is available. It eliminates having to wait for the calculation of $\sin\theta$ and $\cos\theta$, after $K_o(k)$ is available. Similarly, (23) may be separated into the following three parts to reduce the waiting period:

a) $Z(k|k-1) = h\Big(\hat{X}(k|k-1)\Big)$
b) $\Delta Z = Z(k) - Z(k|k-1)$
c) $\hat{X}(k) = \hat{X}(k|k-1) + K(k)\,\Delta Z.$

The transformation of $\hat{X}(k|k-1)$ in Step a) and the calculation of ΔZ in Step b) are precomputed before $K(k)$ is available.

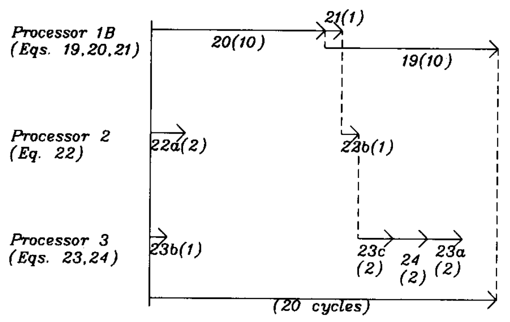


Fig. 14. Parallel computation of the simplified ESRCF using Processor 1B (numbers in brackets indicate the number of clock cycles required for each step).

In addition to the concurrent update of the state estimate error covariance matrices $S_o(k)$'s for each axis, parallelism can be further explored by calculating more than one equation simultaneously. After $K_o(k)$ is calculated, the processor for the LOS coordinates can continue computing $S_o(k)$ and $S_o(k+1|k)$ while the processor for the reference Cartesian coordinates calculates $\hat{X}(k)$ and $\hat{X}(k+1|k)$.

Figs. 13 and 14 show that for three-dimensional tracking one iteration of the ECKF requires 15 clock cycles using Processor 1A, and 20 clock cycles using Processor 1B. Equations (19) and (20) take longer to implement on Processor 1B than on Processor 1A. This is due to the fact that in the implementation of (19) and (20) using Processor 1A the first two rows of zeros are eliminated in an input data stream for the measurement update, and that $U(k)$, which takes 3 cycles to feed, is prestored in the systolic array for the time update.

Figs. 13 and 14 show that Processor 2 and Processor 3 are left idle after computing (22a) and (23b), respectively. Hence, if the implementation of (22a) and (23a) takes more than 2 cycles and less than 8 cycles for the architecture based on Processor 1A, or more than 2 cycles and less than 11 cycles for the architecture based on Processor 1B, then the total computational time would not increase. As a result, the required time for a CORDIC operation can be increased from an assumed 1 cycle to 4 cycles for the architecture based on Processor 1A without affecting the total computational time. Similarly, for the architecture based on Processor 1B, the required time for a CORDIC operation can be increased from 1 to 5 clock cycles.

G. Comparisons

The proposed specialized architecture for the ESRCF is difficult to compare with other architectures. The linear arrays of processors used in [3] and [16] are general-purpose architectures that are not specialized for the ESRCF. However, the decoupling technique and special properties of the tracking KF can be exploited in mapping the ESRCF onto linear arrays.

We have chosen to compare the proposed architecture with the Sung–Hu architecture that is a parallel architecture specialized for the standard SRCF [22]. Sung and Hu have explored parallelism by separating the KF equations into two loosely dependent groups. One of these two groups performs measurement and time updates on the square-root of the state estimate error covariance matrix, whereas the other group estimates state vectors $\hat{X}(k)$ and $\hat{X}(k|k-1)$. A comparison between the Sung–Hu architecture and the architecture developed in this paper on the hardware requirements for the time update confirms that the decoupling technique reduces the number of processing elements by a factor of approximately m. However, for the measurement update, the Sung–Hu architecture requires $nm + m(m+1)/2$ processing elements, and the architecture proposed in this paper requires $(n+m)(n/m+1)/2$ processing elements. These requirements are difficult to compare, for their relationship changes with m and n. Nevertheless, for three-dimensional tracking with 9 state elements, the Sung–Hu architecture is found to require 33 processing elements, whereas the architecture developed in this paper is found to require 30 processing elements. These requirements are comparable. This means that Sung and Hu's use of the sparse nature of an input matrix in the measurement update is as effective as the use of the decoupling technique in our paper.

For computational time requirements, the Sung–Hu architecture requires $4n + m - 1$ clock cycles, which corresponds to 38 clock cycles for three-dimensional tracking with 9 state elements. On the other hand, the architecture proposed in this paper requires 15 or 20 clock cycles, depending on the form of implementation adopted. The smaller computational time requirement for the architecture developed in this paper is due to the simplification that results from the use of the decoupling technique and the special properties of the tracking KF.

V. Conclusions

In this paper, we have developed parallel implementations of the extended square-root covariance filter for radar tracking applications. We have made extensive use of the decoupling technique and special properties of matrices in the tracking KF.

The use of the decoupling technique reduces the required number of processing elements and that of clock cycles for the update of $S(k)$ in the ESRCF by a factor of m, where m denotes the tracking dimension. The combined use of the sparse nature of the measurement matrix $H(k)$ and the sparse, band nature of the transition matrix $\phi(k)$ simplifies matrix-vector and matrix-matrix operations. The implementation of the multiplication of $S(k)$ and $\phi(k)$ in the ECKF requires n multipliers, which is an order of magnitude less than the conventional implementation. Similarly, the implementation of the multiplication of $\phi^T(k)$ and $X(k)$, using the sparse, band nature of $\phi(k)$, requires an order of magnitude less number of clock cycles than the conventional implementation.

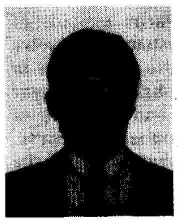
References

[1] B. D. O. Anderson and J. B. Moore, *Optimal Filtering.* Englewood Cliffs, NJ: Prentice-Hall, 1979.

[2] R. S. Baheti, "Efficient approximation of Kalman filter for target tracking," *IEEE Trans. AES,* vol. AES-22, no. 1, pp. 8–14, 1986.

[3] R. S. Baheti, O. R. Halloran, and H. R. Itekowitz, "Mapping extended Kalman filters onto linear arrays," *IEEE Trans. Automat. Contr.,* vol. AC-35, no. 12, pp. 1310–1319, 1990.

[4] R. S. Baheti and O. R. Halloran, "Efficient parallel implementation of target tracking Kalman filter," in *Proc. IEEE 27th Conf. Decision and Contr.*, 1988, pp. 376–381,.
[5] G. Bierman, *Factorization Methods for Discrete Sequential Estimation.* New York: Academic, 1977.
[6] M. J. Chen and K. Yao "On realization of least-squares estimation and Kalman filtering," in *Proc. 1st. Int. Workshop Systolic Arrays,* Oxford, 1986, pp. 161–170.
[7] F. E. Daum and J. Fitzgerald, "Decoupled Kalman filters for phased array radar tracking," *IEEE Trans. Automat. Contr.,* vol. AC-28, pp. 269–283, 1983.
[8] J. L. Fisher, D. P. Casasent, and C. P. Neuman, "A factorized extended Kalman filter," in *Proc. SPIE 85, Real-Time Signal Processing,* 1985, pp. 119–130.
[9] F. M. F. Gaston and G. W. Irwin, "VLSI architecture for square root covariance Kalman filtering," in *Proc. SPIE 89, Real-Time Signal Processing,* 1989, pp. 44–55.
[10] W. M. Gentleman and H. T. Kung, "Matrix triangularization by systolic arrays," in *Proc. SPIE 81, Real-Time Signal Processing,* 1981, pp. 19–26.
[11] G. H. Golub and C. F. Van Loan, *Matrix Computations.* Baltimore, MD: John Hopkins University Press, 1983.
[12] A. H. Jazwinski, *Stochastic Processes and Filtering Theory.* New York: Academic, 1970.
[13] J. M. Jover and T. Kailath, "A parallel architecture for Kalman filter measurement update," in *Proc. IFAC 84*, Hungary, 1984, pp. 1005–1009.
[14] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Trans. ASME,* (*J. Basic Eng.*), vol. 82, pp. 35–50, 1960.
[15] P. G. Kaminski, A. E. Bryson, and S. F. Schmidt, "Discrete square root filtering: A survey of current techniques," *IEEE Trans. Automat. Contr.,* vol. AC-16, pp. 727–735, 1971.
[16] R. T. Kee and G. W. Irwin, "Transputer implementation of tracking Kalman filters," in *Proc. IEEE Int. Conf. Contr.,* vol. 1, 1991, pp. 145–150.
[17] S. Y. Kung, *VLSI Array Processors.* Englewood Cliffs, NJ: Prentice-Hall, 1988.
[18] S. Y. Kung and J. N. Hwang, "An efficient triarray systolic design for real-time Kalman filtering," in *Proc. ICSASSP,* 1988, pp. 2045–2048.
[19] E. K. B. Lee and S. Haykin, "Parallel implementation of the tracking KF," in *Proc. ICASSP,* 1988, pp. 2092–2095.
[20] T. Ohmuro, "A decoupled Kalman tracker using LOS coordinates," in *Proc. 1984 Int. Symp. Noise and Clutter Rejection in Radars and Imaging Sensors,* 1984, pp. 451–455.
[21] R. A. Singer and K. W. Behnke, "Real-time tracking filter evaluation and selection for tracking applications," *IEEE Trans. AES,* vol. AES-7, no. 1, pp. 100–110, 1971.
[22] T. Sung and Y. Hu, "Parallel VLSI implementation of the Kalman filter," *IEEE Trans. AES,* vol. AES-23, no. 2, pp. 215–224, 1987.
[23] E. J. Volder, "The CORDIC trigonometric computing technique," *IRE Trans. Electron. Comput.,* pp. 330–334, 1959.
[24] H. Yeh, "Systolic implementation on Kalman filters," *IEEE Trans. Acoust., Speech, Signal Processing,* vol. ASSP-36, no. 9, pp. 1514–1517, 1988.



Edward K. B. Lee (S'86–M'90) received the B.A.Sc. and M.Eng. degrees from the University of Toronto, Toronto, Ont., Canada, in 1982 and 1985, respectively, and the Ph.D. degree in electrical engineering from McMaster University, Hamilton, Ont., Canada, in 1990.

From 1982 to 1985, he was employed with Motorola Information System Ltd., Brampton, Ont., Canada. After finishing his doctorate study, he was employed with Bell-Northern Research, Toronto, Ont., Canada, from 1989 to 1990. In 1990, he joined Motorola Communication Applied Research Laboratory, Ft. Lauderdale, FL, where he is currently employed as a staff engineer. His main research interests are in adaptive filtering, communication theories, and computer architectures.

Simon Haykin (SM'70–F'82) received the B.Sc. (First-Class Honours) in 1953, the Ph.D. degree in 1956, and the D.Sc. degree in 1967, all in electrical engineering from the University of Birmingham, England.

He is the founding Director of the Communications Research Laboratory and Professor of Electrical and Computer Engineering at McMaster University, Hamilton, Ont., Canada. His research interests include neural networks, adaptive filters, and multidimensional signal processing with applications to radar.

In 1980, Dr. Haykin was elected Fellow of the Royal Society of Canada. He was awarded the McNaughton Gold Medal, IEEE (Region 7), in 1986.