Digital and Hybrid Computational Aspects of the Discrete Representation Theorem of Nonlinear Estimation

EDGAR C. TACKER AND THOMAS D. LINTON

Abstract—A nonlinear filtering problem is investigated, and the discrete representation theorem is used to synthesize the optimal filter. An optimal filtering algorithm is described, and hybrid techniques are employed to effect several methods of implementing the algorithm on a hybrid computer. These implementations are then compared (via Monte Carlo analysis) on the bases of speed and accuracy. It is concluded that hybrid computation techniques can be used to advantage when studying proposed nonlinear optimal estimation formulations.

Index Terms—Digital and hybrid simulation, hybrid computer techniques, Monte Carlo analysis, optimal nonlinear filtering, representation theorem.

I. INTRODUCTION

In the past several years there has been a considerable amount of research in the area of optimal nonlinear filtering. Much of this work has been of a theoretical nature, while implementation efforts have focused on various methods of obtaining suboptimal nonlinear filters. References [2]–[5] (also see the references mentioned therein) consider the problem of digital implementation of optimal nonlinear filtering algorithms, and three recent papers [10]–[12] discuss hybrid synthesis of optimal discrete nonlinear filters. These latter three papers should be of interest to anyone desiring to obtain an estimate of the range of applicability of hybrid methods to problems in nonlinear filtering.

In this paper, a particular nonlinear filtering problem is investigated. The discrete representation theorem [1] is used to synthesize the optimal filter. The plant, measurement process, and noise signals are simulated with the aid of a digital computer. The optimal filtering algorithm is implemented both digitally and via hybrid techniques. The hybrid system used in these studies includes an EAI 680 analog computer and an XDS-SIGMA 5 digital computer. These various filtering implementations are subjected to a Monte Carlo analysis, and are then compared on the bases of speed and accuracy.

The work described herein was performed during the summer of 1970, and is a revised and expanded version of [6], [7], the initial published works in the area of applying hybrid computer techniques to nonlinear estimation problems wherein the discrete representation theorem is employed.

II. PROBLEM FORMULATION

Consider the dynamic system modeled by the following nonlinear stochastic difference equations

\[ x_n = f_n(x_{n-1}) + k_n(x_{n-1}) \cdot u_{n-1} \]  \hspace{1cm} (1)

\[ x_0 = c \]  \hspace{1cm} (plant equation)

\[ z_n = k_n(x_n) + v_n \]  \hspace{1cm} (observation equation) \hspace{1cm} (2)

where all variables are scalar-valued, and \( \{u_n\} \) and \( \{v_n\} \) are independent zero-mean Gaussian white sequences with variances \( Q_n \) and \( R_n \), respectively. The prior density \( j_0(\cdot) \) of the initial state \( c \) is assumed given, and is independent of \( \{u_n\} \) and \( \{v_n\} \). References [1]–[5] and Section X of this paper discuss vector formulations.

It is now desired to consider the optimal filtering problem for this system. From the point of view of an estimation theorist, the two most important objectives to be kept in mind when planning a simulation study of a nonlinear filtering algorithm are: 1) to obtain an intuitive feeling for how the filter is performing, and 2) to obtain statistical measures that quantify this performance. An important necessary condition in satisfying both of these objectives is that one be able to evolve the functions \( J_n(\cdot)(n = 1, 2, \cdots) \) with acceptable speed and accuracy, and thus this will be the main concern in this paper.

Filter performance will be measured quantitatively via computing, for each time step \( n \):

\[ \gamma_{n,S} \triangleq \frac{1}{S} \sum_{s=1}^{S} (x_n^{(s)} - \hat{x}_n^{(s)})^2 \]  \hspace{1cm} (3)

and

\[ \gamma_{n,S} \triangleq \frac{1}{S} \sum_{s=1}^{S} (x_n^{(s)} - \hat{x}_n^{(s)})^2 - v_n,S \]  \hspace{1cm} (4)

1 E. C. Tacker profited from discussions with Dr. R. S. Bucy and Dr. D. S. Miller, who commenced their research on hybrid synthesis of optimal nonlinear filters some months earlier. The specific set of hybrid computer techniques described in this paper is essentially disjoint with respect to the set given in [10]–[12]. These latter references discuss more general techniques than those considered herein.

2 \( J_n(\cdot) \) is the probability distribution of the state \( x_n \) of (1), conditioned on the noisy observations \( \{z_0, \cdots, z_{n-1}\} \).
the estimated mean error and estimated error covariance, respectively. The number $S$ of Monte Carlo runs to be employed depends, of course, on how fast these estimates converge.

The value $S=100$ proved to be acceptable for our purposes. The term $(x_n^{(1)} - \hat{x}_n^{(1)})$ is the estimation error for the $s$th Monte Carlo realization of the stochastic system (1)–(2), and

$$\hat{x}_n(s) \triangleq \sum_{i=1}^{N} y_i J_n^{(1)}(y_i)$$

where $J_n^{(1)}(\cdot)$ is the distribution resulting from the observation sequence $z_n(s)$ generated via (2).

III. The Bayes-Optimal One-Step Predictor

Based on results obtained thus far [2]–[5], it appears that the representation theorem (e.g., see [1, eq. (4.29)]) may be effectively employed in the design of optimal nonlinear filters. The representation theorem can be written in the form

$$J_{n+1}(y) = C_{n+1} \int_{R} N[x_n - h_n(L), R_n] \cdot N[y - f_n(L), Q_n] \cdot J_n(L) \, dL$$

where

$$N[a, b] \triangleq \exp \left[-\frac{1}{2}a^2B^{-1}\right].$$

$C_{n+1}$ is a normalization factor introduced to insure that $J_{n+1}$ has unit probability mass. The remaining portions of this paper will deal with discussing methods of using (6) to proceed computationally from $J_0$ to $J_1$ to $J_2$, etc.

IV. Discretization

Following [2], (6) is conveniently discretized by representing $J_n$ by an equally spaced set of $N = 2M + 1$ mass points. Thus, grid point $G_n(I)$ “supports” the probability mass $J_n(I) \triangleq J_n(G_n(I))$, where $\sum_{I=1}^{N} J_n(I) = 1$. Given a fixed number $N$ of grid points, the problem becomes one of centering and spacing these points, possibly as a function of the time step $n$. Senne and Bucy [4] consider positioning the grid for $J_{n+1}$ according to a best prediction of the mean $m_{J_{n+1}}$ and variance $\sigma_{J_{n+1}}^2$ of this (as yet unknown) $J_{n+1}$ distribution. This is accomplished by using the state equation (1) to extrapolate to time step $n+1$ from the information available in $J_n$. For instance, $m_{J_{n+1}}$ is simply the mean value of this extrapolated state. In this way, one obtains the following transformation from the “old” grid $G_n(\cdot)$ to the “new” grid $G_{n+1}(\cdot)$ defined by

$$G_{n+1}(I) = m_{J_{n+1}} - N_s \cdot \sigma_{J_{n+1}} \cdot \left(1 - \frac{I - 1}{M}\right),$$

$$I = 1, \ldots, 2M + 1$$

where

$$m_{J_n} = \sum_{I=1}^{2M+1} f_n(G_n(I)) \cdot J_{n-1}(I)$$

$$\sigma_{J_n}^2 = \sum_{I=1}^{2M+1} \left[f_n^2(G_n(I)) + Q_{n-1}\right] \cdot J_{n-1}(I) - m_{J_n}^2.$$  

Thus the grid for $J_{n+1}$ is centered at $m_{J_{n+1}}$ and the $N$ grid points are equally spaced over $\pm N_s$ standard deviations $\sigma_{J_{n+1}}$ about $m_{J_{n+1}}$.

Using the above discretization, one obtains

$$J_{n+1}(I) = C_{n+1} \sum_{K=1}^{2M+1} T(n, G_n(K), G_{n+1}(K)) \cdot J_n(K)$$

where

$$T(n, G_n(K), G_{n+1}(K)) \triangleq N[z_n - h_n(G_n(K)), R_n] \cdot N[G_{n+1}(I) - f_n(G_n(K)), Q_n].$$

V. An Example

Denoting the “new” grid points $G_{n+1}(I)$ by $y_i$ and the “old” grid points $G_n(\cdot)$ by $L_i$, (9) can be written as

$$J_{n+1}(y_i) = C_{n+1} \sum_{j=1}^{N} J_n(L_j) \exp\left[-\frac{(z_n - L_j)^2}{2R}\right] \cdot \exp\left[-\frac{(y_i - f_n(L_j))^2}{2Q}\right].$$

VI. Preliminary Computational Considerations

Because of the simple form of (11), it is very easy to obtain some insight into the statistical behavior of the state process, i.e., clearly $E[\{x_n\}] = 0$ for every $n$, and for $Q=1$ (the only value that shall be considered) $\text{var} x_n$ increases monotonically from 1.25 to 1.33. This suggests that there would be very little to be gained by varying the grid from one time step to the next. From a digital simulation point of view, it takes only an additional 1 to 2 percent increase in execution time to vary the grids. However, from a hybrid simulation point of view there are several advantages that can be effected if a stationary grid is employed.

In the next section, methods will be discussed for com-
putting \( J_n+1 \) from (13). The emphasis will be placed on decreasing execution times while maintaining a reasonably high level of accuracy. To obtain a standard for accuracy determinations, a straightforward digital implementation of (13) was effected. It seemed that a good check on this benchmark program would consist of comparing results with those in [2]. This was done by using the same observation sequence, the same method of positioning the grids, and the same values for \( Q, R, \var J_0, \text{and} N \); these values were, respectively, 1, 1, 1, and 57. Our results agreed very closely (4–5 figures of significance) with those in [2].

VII. Computational Techniques

To fix ideas, we shall once and for all set \( Q = 1, R = 1, \var J_0 = 1, N = 41, \) and keep a stationary grid centered at zero and spaced according to \( N_x = 3.5. \)

As mentioned earlier, the objective is to construct algorithms that are faster than the benchmark algorithm. The most fundamental way to speed the solution rate is to decrease the number of summands in (13), i.e., if it can be reliably predicted that a particular summand in (13) will be negligibly small, then this term should not be computed. Two methods ("exponential tracking" and "mode tracking") designed to eliminate unneeded summands will now be discussed.

A. Exponential Tracking

The idea [7] of exponential tracking is to employ only those summands of (13) for which the inequality

\[
\exp \left\{ -\left( z_n - L_j \right)^2 / 2R \right\} \geq \epsilon \left[ \max_j \exp \left\{ -\left( z_n - L_j \right)^2 / 2R \right\} \right]
\]

(14) holds, where \( \epsilon \) is a small positive number (\( \epsilon = 0.001 \) was found to be a good choice). Inequality (14) can be shown easily to be equivalent to

\[
\left| z_n - L_j \right| \leq a_n
\]

(15)

where \( a_n = \sqrt{z_n - \eta^2} - 2C \ln \epsilon \) and \( \eta \) is the \( L_j \) that minimizes \((z_n - L_j)^2\). The fact that the nonlinearity in (12) is monotone insures that \( \Omega_n = \{ j; \left| z_n - L_j \right| \leq a_n \} \) is a contiguous set, and is therefore completely defined whenever its endpoints are specified—a fact that considerably simplifies computation. \( \Omega_n \) need only be computed once for each time step, since it does not depend on \( \gamma \). Exponential tracking turns out to be very effective for this example problem since \( Q/R = 10 \) is relatively large.

B. Mode Tracking

The idea [3], [4] of "ellipsoidal mode tracking" (hereafter abbreviated to "mode tracking") is motivated by the form of (9) and properties of Gaussian density functions.

**Theorem.** If in (9) \( J_n(L) \) is Gaussian with mean \( \mu_n \) and variance \( P_n \), and if \( h_n(\cdot) \) is linearized about \( x = \mu_n \), then the summand of (9) is Gaussian with variance

\[
K_n = \left( Q_n^{-1} + P_n^{-1} + \left( \frac{\partial h_n(\mu_n)}{\partial x} \right)^2 R_n^{-1} \right)^{-1}
\]

and mean

\[
\lambda_n = K_n \left( Q_n^{-1} + P_n^{-1} \mu_n + \left( \frac{\partial h_n(\mu_n)}{\partial x} \right) R_n^{-1} \right)
\]

(16)

The proof, while relatively tedious, is straightforward and will be omitted.

Based on the above theorem, it is reasonable to expect that one would lose very little accuracy in calculating the sum in (9) if the range of summation were restricted to the "interval" \( \{ L_j; \left| \lambda_n - L_j \right| \leq n_{\varepsilon} \sqrt{K_n} \} \), assuming that the number \( n_{\varepsilon} \) of standard deviations is sufficiently large. The choice of \( n_{\varepsilon} \) is clearly a function of the nonlinearity in \( h_n \). We found \( n_{\varepsilon} = 6 \) to be sufficient for our purposes.

VIII. Computer Implementation

A. Digital Simulation—Exponential Tracking

Since straightforward determination of \( \eta \) would be fairly time-consuming, a more efficient algorithm is used to compute it. First, the index \( r = 1.0 + (\sqrt{z_n - L_1} / L_2 - L_1) \) is computed and truncated to integer format. Then \( \eta \) is easily computed as

\[
\eta = \min \{ (z_n - L_j)^2; (z_n - L_{r+1})^2 \}
\]

In a similar way, the left and right endpoints \( k \) and \( L \), respectively, of \( \Omega_n \) can be computed from the equations

\[
k = 2 + \frac{\sqrt{z_n - a_n - L_1}}{L_2 - L_1}
\]

\[
L = 1 + \frac{\sqrt{z_n + a_n - L_1}}{L_2 - L_1}
\]

(18)

(19)

The shortcuts used in computing \( \eta, k, \text{and} L \) also take advantage of the fact that, for this example, the nonlinearity in (2) is monotone. Such efficient algorithms will not be available for the general case.

B. Digital Simulation—Mode Tracking

In order to implement mode tracking, it is sufficient to compute \( K_n \) once for each time step, but \( \lambda_n(\gamma) \) and

A similar theorem holds if \( J_n \) can be represented by a linear combination of Gaussian densities and if both (1) and (2) are suitably linearized. See [4] for details.

1 In [2], Bucy used the formulation (8) where \( \mu_{L_n} \) and \( \sigma_{L_n}^2 \) were interpreted, respectively, as the mean and variance of the distribution \( J_n+1 \).

2 In the multidimensional case [3], [4], the region will be the interior of an "ellipsoid" centered at \( \lambda_n \). If \( J_n \) is a linear combination of Gaussian densities, then the region will be the interior of a union of such "ellipsoids"—see [4] for details.
\[ \Omega_n(y) = \{ j : |y_j - L_i| \leq N_{a_n} \sqrt{K_n} \} \]

must be computed for each \( y_j \). As with exponential tracking, it is sufficient to compute the endpoints of \( \Omega_n(y) \), and these can be computed from

\[
k(y) = 2 + \frac{\lambda_n(y) - N_{a_n} \sqrt{K_n} - L_1}{L_1 - L_1}
\]

(20)

\[
L(y) = 1 + \frac{\lambda_n(y) + N_{a_n} \sqrt{K_n} - L_1}{L_1 - L_1}.
\]

(21)

C. Hybrid Simulation—Exponential Tracking

A straightforward implementation of (13) on the digital computer is very slow because of the large number of exponentials that must be computed. For convenience, and to decrease the number of exponential function generations required, (13) is rewritten as

\[
J_{n+1}(y_i) = C_{n+1} \cdot \sum_{k=1}^{N} J_n(L_k)
\]

\[
\cdot \exp \left[ \frac{-(\varepsilon_n - L_k)^2}{2R} - \frac{(y_i - \frac{1}{2} L_k)^2}{2Q} \right].
\]

(22)

An additional reduction in the number of exponential terms can be obtained by employing either mode tracking or exponential tracking. If exponential tracking (mode tracking) is used, then \( \sum_{k=1}^{N} \) in (22) is replaced by \( \sum_{k \in \Omega_n} \) (\( \sum_{k \in \Omega_n(y)} \)), respectively. In a digital implementation, however, the exponential computation still requires a large percentage of the total computation time.

One motivation for using a hybrid computer is the availability of analog function generators. Thus, the exponential terms such as those in (22) can be computed almost instantaneously on the analog computer and summed by using an analog accumulator. Fig. 1 illustrates how (22) can be implemented with a hybrid computer. For the \( n \)th time step, the following sequence of events occurs.

1) Set \( i = 1 \) (this selects the first “new” grid point \( y_i \)).
2) Initialize the analog accumulator to zero.
3) Set \( k = 1 \) (this selects the first “old” grid point \( L_1 \)).
4) Send \( g(z_n, L_k, y_i) \) and \( J_n(L_k) \) to the analog computer.
5) Update the analog accumulator: \( \sum g \rightarrow \sum_{g+1} \).
6) Set \( k = k+1 \) (select the next “old” grid point).
7) If \( k \leq N \), go to 4).
8) Read \( J_{n+1}(y_i) \) (via ADC) into the digital computer.
9) Set \( i = i + 1 \) (select the next “new” grid point).
10) If \( i \leq N \), go to 2).
11) Normalize \( J_{n+1}(y) \).
12) Go to the next time step: \( n \rightarrow n + 1 \).

If an attempt is made to implement (22) as described, there will be a scaling problem on the analog computer. This is due to the fact that \( g(z_n, L_k, y_i) \) varies over a very wide range, and the domain of the analog function generator is limited in voltage to \([-10, +10]\). If \( g(z_n, L_k, y_i) \) is scaled down directly to fit the domain of the analog function generator, the loss of accuracy could be prohibitive. However, because of the properties of the exponential function, \( g(z_n, L_k, y_i) \) can be scaled by the following method.

Defining \( \alpha_n = \min_k \{(z_n - L_k)^2/2R\} \), \( \beta_n = \min_k \{(y_i - \frac{1}{2} L_k)^2/2Q\} \), (22) can be rewritten as

\[
J_{n+1}(y_i) = C_{n+1} \cdot \sum_{k=1}^{N} J_n(L_k) \exp \left[ -\alpha_n - \beta_n \right]
\]

\[
\cdot \exp \left[ \frac{-(\varepsilon_n - L_k)^2}{2R} + \alpha_n - \frac{(y_i - \frac{1}{2} L_k)^2}{2Q} + \beta_n \right],
\]

and since \( \alpha_n \) depends only on \( n \), this becomes

\[
J_{n+1}(y_i) = C_{n+1} \cdot \exp \left[ -\beta_n \right] \sum_{k=1}^{N} J_n(L_k)
\]

\[
\cdot \exp \left[ \frac{-(\varepsilon_n - L_k)^2}{2R} + \alpha_n - \frac{(y_i - \frac{1}{2} L_k)^2}{2Q} + \beta_n \right].
\]

(23)

Defining \( g' \) to be the argument of the second exponential of (23), the previously described algorithm may be employed, wherein \( g' \) replaces \( g \). Of course, \( J_{n+1}(y_i) \) must be multiplied by \( \exp \left[ \beta_n \right] \) in the digital computer before normalization. For each time step, the following data must be computed: \((\varepsilon_n - L_k)^2/2R\) for each \( k \), \( \alpha_n \) once, \( \beta_n \), and \( \exp(\beta_n) \) for each \( i \), and \((y_i - \frac{1}{2} L_k)^2/2Q\) for each \( i \) and each \( k \). Since there are
only a small number of possible \((y_i - \frac{1}{2} L_i)^2 / 2Q\) and exp \(\beta_n\) terms, they are stored in tables to avoid having to compute them each time they are needed. Appendix II contains a discussion of the structure and use of these tables.

IX. DISCUSSION AND PRESENTATION OF RESULTS

Four computational algorithms were implemented (based on the example problem) and subjected to a Monte Carlo analysis. For each of these algorithms, the statistics (3) and (4) were computed for the first 20 time steps, using \(S = 100\) Monte Carlo runs. Tables I and II of Appendix I present these data.

Considering the data in Tables I and II and taking the benchmark results as a standard of comparison, it can be seen that the other three algorithms provided essentially the same accuracy. This, together with other Monte Carlo data that were generated but not included herein, indicates that for the purposes of estimating ensemble averages, the errors in the hybrid loop tend to “average out.” We hasten to point out, however, that systematic errors can enter into the simulation, and the success of such simulation studies depends upon careful application of hybrid simulation methodology.

In the example, \(N = 41\) and exponential tracking reduced the number of \(L_j\) used by approximately 80 percent on the average. In those cases for which both mode tracking and exponential tracking are feasible, exponential tracking was found to be somewhat faster. This is partially due to the fact that, for exponential tracking, it is only necessary to determine \(\Omega_n\) once per time step, but the primary reason that exponential tracking is faster is the fact that, on the average, more \(L_j\) could be eliminated while retaining sufficient accuracy. Table III of Appendix I compares the various algorithms on the basis of execution time.

Another method of implementation was also considered. It involves using parallel analog circuits, such as the one in Fig. 1. This would allow one to generate several or perhaps all of the mass points \(J_n(y_i)\) simultaneously, according to the number of parallel circuits used. One such parallel-hybrid technique was analyzed. The hybrid algorithm described in Section VIII requires \((2L + 1)N + 1\) data transfer per time step in implementing (22). If the fully parallel hybrid algorithm were to be employed, only \(2L + N + 1\) data transfers per time step would be required. Taking into account the time taken by the other portion of the digital program, it appears that the maximum attainable speedup ratio is approximately \(N/2\). The final entry of Table III was determined by employing the foregoing analysis, and indicates that for the problem at hand, it is reasonable to expect a two-order-of-magnitude increase in computation speed by employing the hybrid techniques suggested in this paper.

X. CONCLUDING REMARKS

1) The hybrid implementation could be further speeded up by employing adaptive scaling methods. For instance, some of the scaling could be transferred to the analog computer—a wider dynamic signal range could be accommodated by making appropriate use of comparators, switches, limiters, etc. Another technique that could be used deals with updating the scale factors \(\beta_n\) less than \(N\) times per time step.

2) Although the discussion in this paper has been restricted to a scalar problem, the techniques described herein can be extended to handle higher dimensional problems. In particular, for the \(d\)-dimensional case, the representation theorem (6) can be written as

\[
J_{n+1}(y) = C_{n+1} \int_{\mathbb{R}^d} J_n(L) \cdot \exp \left[ \langle z_n, R_n^{-1} h_n(L) \rangle - \frac{1}{2} (h_n(L), R_n^{-1} h_n(L)) \right] \cdot \exp \left[ -\frac{1}{2} (y - f_{n+1}(L), A_{n+1}^{-1}(y - f_{n+1}(L))) \right] dL
\]

where \(C_{n+1}\) is a normalization factor, \(y\) is a \(d\)-vector, \(h_n(L)\) is an \(r\)-vector, \(A_{n+1} = h_n(L)Q(n-1)h_n^T(L)\), and \(\langle \cdot, \cdot \rangle\) denotes “scalar product.” The argument of the first exponential in the integrand does not depend on \(y\), so exponential tracking is still applicable.

For this case there are \(N^d\) exponentials to be evaluated per time step, where \(N\) is the number of grid points in each coordinate, so there is an even greater need to have an efficient method for computing the exponentials. By fixing all but one of the coordinates of \(y\), the hybrid method given earlier may be used to compute the corresponding sum of exponentials. This process is continued until \(y\) has varied over the whole grid.

3) The problem discussed in this paper may be considered from points of view differing from that described in Section II. For instance, the problem may be considered to be, primarily, one of

a) computing a particular type of exponential-convolution integral [(6) or (13)], or

b) determining the response of a dynamic system (deterministic and infinite dimensional) wherein (6) or (13) describes the system model, \(J_n(\cdot)\) serves as the system state, and \(z_n\) serves as the input, or

c) computationally analyzing a difference equation model [(6) or (13)] of a partial differential equation—the Fokker–Planck equation [8], [9], corresponding to a continuous-time model of the stochastic system (1), (2).

4) This simulation study is only a first step in applying hybrid computer techniques to the nonlinear filtering problem, and, obviously, much more can be done via a more complete utilization of the capabilities of the hybrid computing system. In any case, however, the data in Tables I, II, and III do indicate that a hybrid computer (general or special purpose) can be used to advantage in studying and evaluating nonlinear filtering algorithms, or, as mentioned in remark 3), in any other problem having a similar formulation.

* Also refer to [10]–[12].
APPENDIX I

TABLE I

Estimated Mean Error Based on 100 Monte Carlo Runs

<table>
<thead>
<tr>
<th>Time Step</th>
<th>Digital</th>
<th>Digital</th>
<th>Digital</th>
<th>Hybrid</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Benchmark</td>
<td>Node</td>
<td>Tracking</td>
<td>Exp.</td>
</tr>
<tr>
<td>1</td>
<td>-0.0293279</td>
<td>-0.2945272</td>
<td>-0.2967330</td>
<td>-0.0248297</td>
</tr>
<tr>
<td>2</td>
<td>-0.0282799</td>
<td>-0.0200165</td>
<td>-0.0206192</td>
<td>-0.0296239</td>
</tr>
<tr>
<td>3</td>
<td>-0.02160386</td>
<td>-0.0065194</td>
<td>-0.01599702</td>
<td>-0.01041749</td>
</tr>
<tr>
<td>4</td>
<td>-0.0294272</td>
<td>-0.0153599</td>
<td>0.0030212</td>
<td>-0.0311741</td>
</tr>
<tr>
<td>5</td>
<td>0.0682530</td>
<td>-0.0038042</td>
<td>-0.01042806</td>
<td>0.01547562</td>
</tr>
<tr>
<td>6</td>
<td>0.10488200</td>
<td>-0.1079007</td>
<td>-0.1073537</td>
<td>-0.1118033</td>
</tr>
<tr>
<td>7</td>
<td>-0.04981279</td>
<td>-0.476747</td>
<td>-0.04935234</td>
<td>-0.0506464</td>
</tr>
<tr>
<td>8</td>
<td>0.06330687</td>
<td>-0.6843641</td>
<td>-0.06689042</td>
<td>-0.0678519</td>
</tr>
<tr>
<td>9</td>
<td>-0.01674395</td>
<td>-0.0563199</td>
<td>-0.0633993</td>
<td>-0.0663978</td>
</tr>
<tr>
<td>10</td>
<td>0.00314933</td>
<td>-0.00712155</td>
<td>-0.00400430</td>
<td>-0.00592528</td>
</tr>
<tr>
<td>11</td>
<td>-1.117172</td>
<td>-1.1931980</td>
<td>-1.1037026</td>
<td>-1.1040057</td>
</tr>
<tr>
<td>12</td>
<td>-1.1492606</td>
<td>-1.2106222</td>
<td>-1.1153793</td>
<td>-1.1154260</td>
</tr>
<tr>
<td>13</td>
<td>-0.78654055</td>
<td>-0.7867038</td>
<td>-0.8117801</td>
<td>-0.8037287</td>
</tr>
<tr>
<td>14</td>
<td>-0.04320401</td>
<td>-0.03909944</td>
<td>-0.03909573</td>
<td>-0.03945251</td>
</tr>
<tr>
<td>15</td>
<td>-0.02726303</td>
<td>-0.07190707</td>
<td>-0.06823151</td>
<td>-0.06583518</td>
</tr>
<tr>
<td>16</td>
<td>-0.01714955</td>
<td>-0.00302623</td>
<td>-0.02064396</td>
<td>-0.02095320</td>
</tr>
<tr>
<td>17</td>
<td>-0.1298007</td>
<td>-0.0121056</td>
<td>-0.00030526</td>
<td>-0.00453265</td>
</tr>
<tr>
<td>18</td>
<td>0.3245559</td>
<td>0.3828425</td>
<td>0.3268587</td>
<td>0.3309426</td>
</tr>
<tr>
<td>19</td>
<td>-0.1075922</td>
<td>-0.1081749</td>
<td>-0.11086941</td>
<td>-0.1103733</td>
</tr>
<tr>
<td>20</td>
<td>-0.07036436</td>
<td>-0.07115357</td>
<td>-0.06475406</td>
<td>-0.05156005</td>
</tr>
</tbody>
</table>

APPENDIX II

Since the grid is fixed, both the old and new grid points can be denoted by $y_i$. It is convenient to define

$W_i = -N_n \cdot \sigma_n \left( \frac{1 - i - \Delta}{M} \right), \quad i = 1, \ldots, 123.$

Clearly, $W_i = y_{i-s}$ for $i = 83, 84, \ldots, 123$ so that $W_i$ is just an extension of the $y_i$. Now, construct the tabular array $\beta_1, \beta_2, \ldots, \beta_m$ where $\beta_i$ is defined by

$\beta_j = \frac{(y_1 - \frac{1}{2} W_i)^2}{2}$

Note that a straightforward construction of such a table could require as many as $N^2$ (here, 1681) storage locations. The technique given herein reduces the table size to 123 storage locations.

Consider any term of the form $\frac{1}{2}(y_i - \frac{1}{2} y_k)^2$. It can be written as

$\frac{1}{2}(y_i + (i - 1)\Delta - \frac{1}{2} y_k)^2 = (y_1 - \frac{1}{2} y_k - 2(i - 1)\Delta)^2$

$= \frac{1}{2}(y_1 - \frac{1}{2} y_k - (i - 1)\Delta)^2$

$= \frac{1}{2}(y_1 - \frac{1}{2} W_{i+44-2i})^2$

where $\Delta = W_{i+1} - W_i$, i.e., it is one of the $\beta$'s. Thus the $\beta$ array contains all terms of this form.

To illustrate the use of this table, suppose that exponential tracking is being used and that $\Omega$ has been determined to be $\{k_1, \ldots, k_L \}$. Then for $y_i$, the terms

$\frac{(y_1 - \frac{1}{2} y_k)^2}{2}$

are represented by

$\beta_{2k+4k-2i}, \ldots, \beta_{2k+4k-L-2i}$.

It remains to find the minimum of this set of $\beta$’s and subtract it from each element of the set. This can be done efficiently by recognizing the following property of the $\beta$’s:

If $k > 63$, then $\beta_k = \min_{L-k, \ldots, 123} \beta_L$.

If $k = 63$, then $\beta_k = 0 = \min_{L-1, \ldots, 123} \beta_L$.

If $k < 63$, then $\beta_k = \min_{L-1, \ldots, k} \beta_L$.

Thus, if $k_F > 63$, then $\beta_{2k_F+4k_F-2i}$ is the desired minimum; if $k_L < 63$, then $\beta_{2k_L+k_L-2i}$ is the minimum; otherwise the minimum is zero.

Since the unnormalized $J_{i+1}(y_i)$ must be multiplied by the exponential of this minimum $\beta$, the exponentials of the $\beta$’s are stored in another table, i.e., $a_i = \exp (-\beta_i)$. Since only integer arithmetic is required to compute the indices of the tables, this method is significantly faster than straightforward computation of each term.

REFERENCES


The Representation and Matching of Pictorial Structures

MARTIN A. FISCHLER AND ROBERT A. ELschLAGER

Abstract—The primary problem dealt with in this paper is the following. Given some description of a visual object, find that object in an actual photograph. Part of the solution to this problem is the specification of a descriptive scheme, and a metric on which to base the decision of "goodness" of matching or detection.

We offer a combined descriptive scheme and decision metric which is general, intuitively satisfying, and which has led to promising experimental results. We also present an algorithm which takes the above descriptions, together with a matrix representing the intensities of the actual photograph, and then finds the described object in the matrix. The algorithm uses a procedure similar to dynamic programming in order to cut down on the vast amount of computation otherwise necessary.

One desirable feature of the approach is its generality. A new programming system does not need to be written for every new description; instead, one just specifies descriptions in terms of a certain set of primitives and parameters.

There are many areas of application: scene analysis and description, map matching for navigation and guidance, optical tracking, stereo compilation, and image change detection. In fact, the ability to describe, match, and register scenes is basic for almost any image processing task.

Index Terms—Dynamic programming, heuristic optimization, picture description, picture matching, picture processing, representation.

Introduction

The primary problem dealt with in this paper is the following. Given some description of a visual object, find that object in an actual photograph. The object might be simple, such as a line, or complicated, such as an ocean wave, and the description can be linguistic, pictorial, procedural, etc. The actual photograph will be called the "sensed scene," a two-dimensional array of gray-level values, while the object being sought is called the "reference."

This ability to find a reference in a sensed scene, or, equivalently, to match or register the images of two scenes, is basic for almost any image processing task. Application to such areas as scene analysis and description, map matching for navigation and guidance, optical