C. Champlin and D. Morrell, “Target Tracking Using Arbitrarily Located Detectors and a Continuous-State Viterbi Algorithm,” *Conference Record of the 34th Asilomar Conference on Signals, Systems, and Computers,* October 2000, pp. 1100–1104.

Target Tracking using Irregularly Spaced Detectors and a Continuous-State Viterbi Algorithm

Cary Champlin
Independent Consultant
cary_champlin@ieee.org

Darryl Morrell
Department of Electrical Engineering
Arizona State University
morrell@asu.edu

Abstract

The primary contribution of this paper is the application of the CSVA, a recently developed algorithm for MAP state sequence estimation, to the problem of tracking a target constrained to two dimensional movement using detections from a field of stationary, arbitrarily located, identical sensors. Models for target motion and detector performance are developed for this problem, providing a system model to which the MAP estimator is applied to create a target trajectory estimator. The performance of the MAP target trajectory estimator is illustrated through simulation.

1. Introduction

In this paper, we address the following scenario: a target that emits a signal moves through a field of stationary, arbitrarily located sensors. The information provided by each sensor to the tracking algorithm is a binary decision–target detected or no target detected. The probability of a sensor detecting the target at a given time is a decreasing function of the distance from the sensor to the target; additionally, the sensor position is known. Thus, detection of the target by a sensor provides information about the target position at a particular time. The target trajectory is estimated from the time sequence of target detections.

The primary contribution of this paper is the application of a recently developed Maximum *A Posteriori* (MAP) state sequence estimator to this tracking problem[1]. This estimator, which we call the continuous-state Viterbi Algorithm (CSVA), is conceptually similar to the Viterbi Algorithm (VA), in that state metrics and survivor sequences are computed. Unlike the VA, this estimator is applied to a system with a continuous state. In the CSVA, recursive computation of the metric and survivor functions is implemented using polyhedral approximations. Simulation studies have shown that the CSVA has much lower computational complexity and much better performance than a VA applied to a continuous-state system with appropriate discretization of the state space.

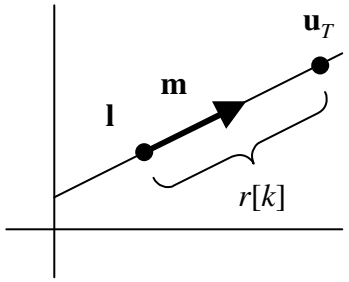

Figure 1. Line parameters for 1-D motion.

2. Target and Sensor Models

To estimate target position and velocity, we require probabilistic models of the target motion and the sensor performance. The target motion is modeled by a discrete-time linear system driven by white noise. The sensor performance model is obtained by analysis of the probability of detection given a particular target location.

2.1. Target Motion Model

The target is constrained to move along a line in two-dimensional space. The line is defined in terms of two two-dimensional vector parameters: $\mathbf{l}$, a point on the line, and $\mathbf{m}$, a unit direction vector. (This model extends easily to motion along a line in three-dimensional space.) Figure 1 illustrates these parameters. The target's position along the line at time k is denoted by $r[k]$. The target's speed along the line at time k is denoted by $c[k]$.

The system state is the target position and velocity along the line. We define the state by arranging $r[k]$ and $c[k]$ into

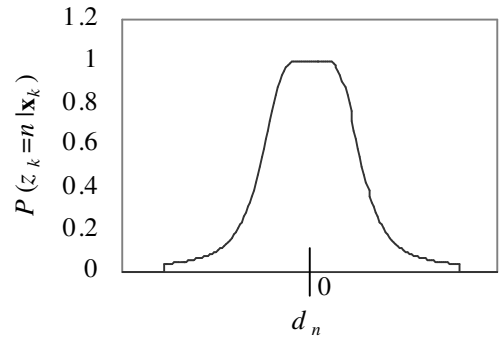

Figure 2. Observation probability for non-zero observation as a function of target-sensor distance.

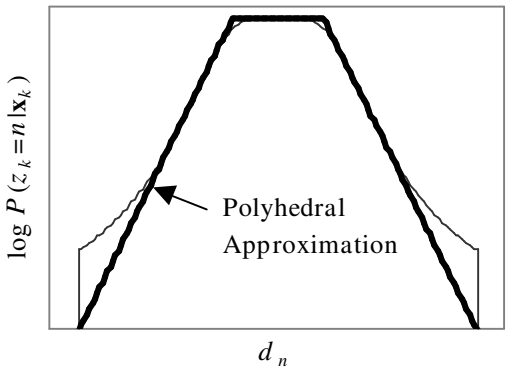

Figure 3. Logarithm of the observation probability.

a vector denoted $\mathbf{x}_k$:

$$\mathbf{x}_k = \begin{bmatrix} r[k] \\ c[k] \end{bmatrix}$$

The discrete-time system represents snapshots of the target position and velocity (which evolve continuously) at evenly spaced times t_0, t_1, etc.; the difference between times is Δ. The system dynamics are given by the following difference equation:

$$\begin{aligned} \mathbf{x}_{k+1} &= \begin{bmatrix} 1 & \Delta \\ 0 & 1 \end{bmatrix} \mathbf{x}_k + \mathbf{w}_k \\ &= A\mathbf{x}_k + \mathbf{w}_k \end{aligned} \quad (1)$$

where $\{\mathbf{w}_k\}$ is a vector Gaussian white noise process with constant covariance matrix Q.

2.2. Sensor Model

In our scenario, N sensors are at known locations. At each time k, a sensor makes a decision about whether the target is present on the basis of M samples of a received signal. We assume that the time interval in which these samples are gathered is short relative to Δ, the time between estimates of the target state. The received signal at each sensor is the sum of noise and the signal emitted by the target (if present). The target signal is modeled as a discrete-time white Gaussian process whose variance is inversely proportional to the square of the distance between the target and the sensor. The noise is modeled as a discrete-time white Gaussian process that is independent of the target signal; the noise at each sensor is independent.

The sensor finds the energy in its received signal and compares this energy to a threshold to determine the presence or absence of the signal from the target. We omit a detailed derivation of this decision rule and of the sensor performance, providing only those expressions necessary for the development of the target tracking algorithm. A detailed derivation is given in [2].

We denote the energy per sample of the signal received from the target as $\sigma_T^2(d)$. This energy is inversely proportional to the square of the distance d between the target and the sensor: $\sigma_T^2(d) = \sigma_{T_0}^2/d^2$, where $\sigma_{T_0}^2$ is the energy per sample of the target signal at a distance of 1 unit. The energy per sample of the noise at each sensor is σ_N^2.

The probability of false alarm and correct target detection can be computed as follows:

$$P_{FA} = \frac{\Gamma\left(\frac{M}{2}, \frac{\beta}{2}\right)}{\Gamma\left(\frac{M}{2}\right)},$$

where β is a parameter chosen to give a desired probability of false alarm and $\Gamma(\cdot,\cdot)$ is the incomplete Gamma function. For a given β (and corresponding probability of false alarm), the probability of detection can be expressed as

$$P_D(d) = \frac{\Gamma\left(\frac{M}{2}, \frac{\beta}{2}\frac{1}{1+\frac{SNR_0}{d^2}}\right)}{\Gamma\left(\frac{M}{2}\right)},$$

where $SNR_0 = \frac{\sigma_{T_0}^2}{\sigma_N^2}$ is the signal-to-noise ratio at a distance of one from the target.

The sensors may be located arbitrarily subject to the constraint that they are far enough apart that the probability of multiple sensors detecting the target at a given time is negligible. This constraint could be eliminated without significantly increasing the complexity of the tracking algorithm.

2.3. Obtaining the Observation

The observation used by the target tracking algorithm at time k is denoted z_k; z_k is zero if no sensor has detected the target or n if sensor n has detected the target. In this section, we describe the process by which the observation z_k is obtained and from this process determine an expression for $P(z_k|\mathbf{x}_k)$.

Let sensor n be the sensor to which the target is closest; we assume that this sensor is the only one that will have a chance to detect the target. We denote the distance between the target and this sensor as d_n. The probability that sensor n detects the target is $P_D(d_n)$. If sensor n detects the target, then $z_k = n$. Otherwise, $z_k = 0$.

This process is a significant over-simplification of the process that one would expect in real life. In particular, it does not address the issues of false detections and multiple detections. However, when the probability of false alarm is very small and the sensors are far enough apart that the probability of multiple detections is also small, it is a reasonable approximation.

This process leads to the following expression for $P(z_k = n|\mathbf{x}_k)$ in terms of the distances d_1 through d_N, where d_j is the distance between the target and sensor j; clearly, d_1 through d_N depend on the target state $\mathbf{x}_k$. When $n \neq 0$,

$$P(z_k = n|\mathbf{x}_k) = \begin{cases} P_D\left(d_n\right), & n = \arg\min_j d_j \\ 0, & n \neq \arg\min_j d_j \end{cases} \quad (2)$$

When $n = 0$,

$$P(z_k = 0|\mathbf{x}_k) = 1 - P_D\left(\min_j d_j\right)$$

Figure 2 shows $P(z_k = n|\mathbf{x}_k)$ for $n \neq 0$ as a function of the distance d_n.

2.4. Approximation of the Observation Probability

The implementation of the CSVA estimator requires that the logarithm of the observation probability be approximated by a polyhedral function; a polyhedral function is a concave function whose hypograph is a polyhedron. As Figure 3 shows, a suitable approximation of $\log P(z_k = n|\mathbf{x}_k)$ can be obtained when $n > 0$. However, any reasonably accurate polyhedral approximation of $\log P(z_k = 0|\mathbf{x}_k)$ will not be concave. Thus, we cannot apply the CSVA algorithm directly to this problem.

The model may be simplified to facilitate the CSVA. Whenever the observation is $z_k = 0$, the CSVA applies the state dynamics model to the estimate but does not update the estimate using the observation. Thus, when there is not a detection at a given time k, the position estimate is updated using "dead reckoning". A tracking algorithm that includes this simplification will provide poorer performance than one that uses $P(z_k = 0|\mathbf{x}_k)$, since "dead reckoning" ignores the position information provided by a no-detection observation.

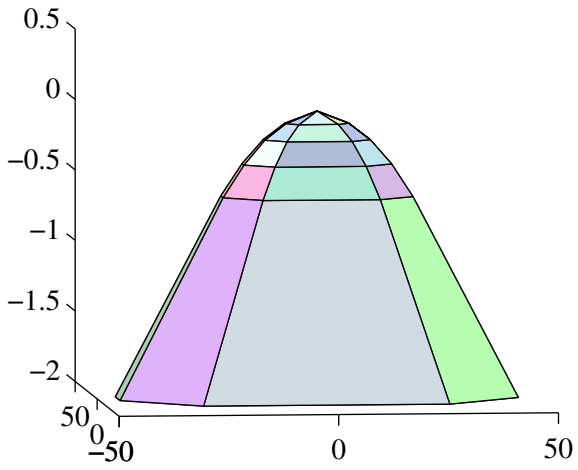

Figure 4. Polyhedral approximation to the state transition density.

3. The CSVA

In the CSVA, recursive computation of the metric and survivor functions is implemented using polyhedral approximations of conditional probability distributions. Thus, the computational structure of the estimator is based on algorithms associated with concave polyhedra; vertex enumeration, addition of polyhedral functions, and solution of linear programs play key roles.

3.1. Estimation Algorithm

The CSVA uses a probabilistic system model to estimate the state sequence from an observation sequence of length K. The state dynamics are described by the conditional state transition density $p(\mathbf{x}_{k+1}|\mathbf{x}_k)$, and the relationship between the observation and the state is described by the conditional observation probability $P(z_{k+1}|\mathbf{x}_{k+1})$. The CSVA computes a metric function $L_{k+1}\left(\mathbf{x}_{k+1}\right)$ according to the following equation[3, 1]:

$$\begin{aligned} L_{k+1}\left(\mathbf{x}_{k+1}\right) &= \max_{\mathbf{x}_k}\left[\log p(\mathbf{x}_{k+1}|\mathbf{x}_k) + L_k\left(\mathbf{x}_k\right)\right] \\ &\quad + \log P(z_{k+1}|\mathbf{x}_{k+1}) \end{aligned} \quad (3)$$

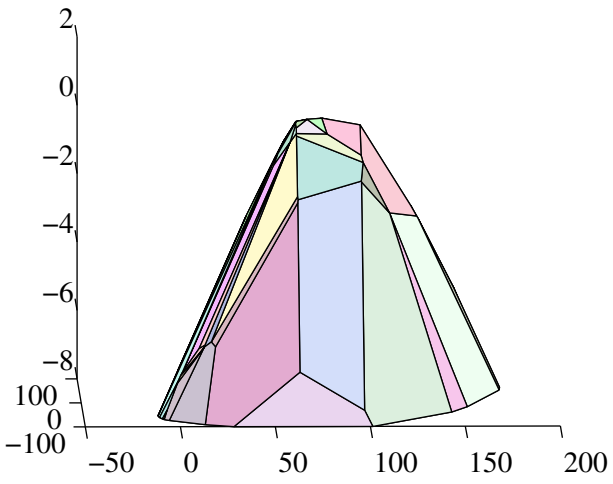

Figure 5. A typical metric function.

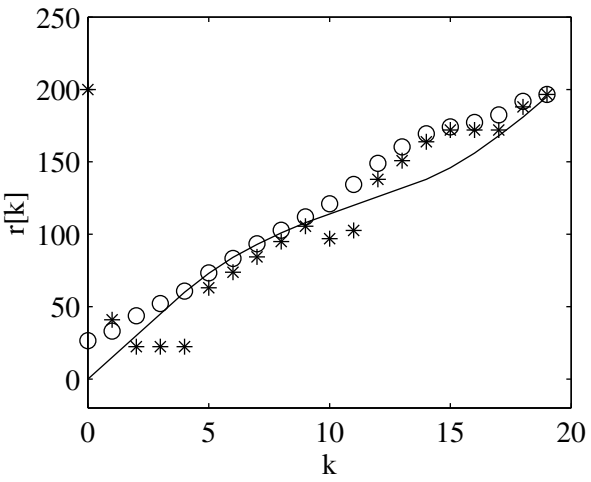

Figure 6. Target position: line-true position, *-forward estimate, O-backward estimate.

An estimate of $\mathbf{x}_k$ based on z_1 through z_k is

$$\hat{\mathbf{x}}_{k|k} = \arg\max_{\mathbf{x}_k} L_k(\mathbf{x}_k) \tag{4}$$

An estimate of $\mathbf{x}_k$ based on the entire observation sequence z_1 through z_K is

$$\hat{\mathbf{x}}_{k|K} = \arg\max_{\mathbf{x}_k} \left[\log p(\hat{\mathbf{x}}_{k+1|K}|\mathbf{x}_k) + L_k\left(\mathbf{x}_k\right)\right] \tag{5}$$

Note that (3) is a recursion that proceeds forward in time, while (5) proceeds backwards in time. Recursion (3) is initialized with $L_0(\mathbf{x}_0) = \log p(\mathbf{x}_0)$. To compute the entire MAP sequence estimate $\{\hat{\mathbf{x}}_{0|K}, \hat{\mathbf{x}}_{1|K}, \ldots, \hat{\mathbf{x}}_{K|K}\}$, a forward pass implements (3) and then a backward pass implements (5). We call $\hat{x}_{k|k}$ the forward estimate and $\hat{x}_{k|K}$ the smoothed (or backward) estimate.

Polyhedral functions provide the framework within which (3), (4), and (5) are implemented by the CSVA. In the CSVA, a polyhedral function is represented as the intersection of a set of half spaces defined by hyperplanes or, equivalently, as a collection of vertices. The maximum transform defined by [4] provides a straight-forward method of performing the maximization in (3); computationally, implementing the maximum transform requires vertex enumeration and addition of polyhedral functions.

3.2. Approximation of Distributions

In order to apply the CSVA to the target tracking problem, we compute polyhedral approximations to $\log p\left(\mathbf{x}_{k+1}|\mathbf{x}_k\right)$ and $\log P\left(z_k|\mathbf{x}_k\right)$. For the system model in (1),

$$\log p(\mathbf{x}_{k+1}|\mathbf{x}_k) = \log p_w\left(\mathbf{x}_{k+1} - A\mathbf{x}_k\right)$$

where p_w is the density function of the system noise $\mathbf{w}_k$. Since $\mathbf{w}_k$ is Gaussian, $\log p_w$ is a parabola; its approximation is shown in Figure 4.

The conditional probability of the observation z_k given the system state $\mathbf{x}_k$ is given by (2). As shown in Figure 3, a simple polyhedral function of three hyperplanes provides a good approximation to this probability. The computation of this approximation is detailed in [2].

An important component of the CSVA implementation, both in terms of reducing the computational load and improving the numerical stability of the algorithm, is the elimination of redundant hyperplanes. One hyperplane in a pair of (nearly) parallel hyperplanes can be eliminated with minimal effect on the accuracy of a polyhedral approximation. Elimination of a hyperplane speeds the algorithm; elimination of nearly parallel hyperplanes reduces the likelihood of numerical problems in the vertex enumeration process. In our implementation, a single parameter controls how close to parallel a pair of hyperplanes must be before one is eliminated.

4. Simulation Results

The performance of the CSVA estimator for a simple scenario is demonstrated through simulation. The target and sensors are located in a plane. The Cartesian coordinates of the sensors are (50,0), (100,0), (175,0), and (225,0). The target trajectory is constrained to the line coincident with the sensor locations. The target moves from left to right with varying velocity. The true target position and velocity along this line are plotted with the solid lines in Fig-

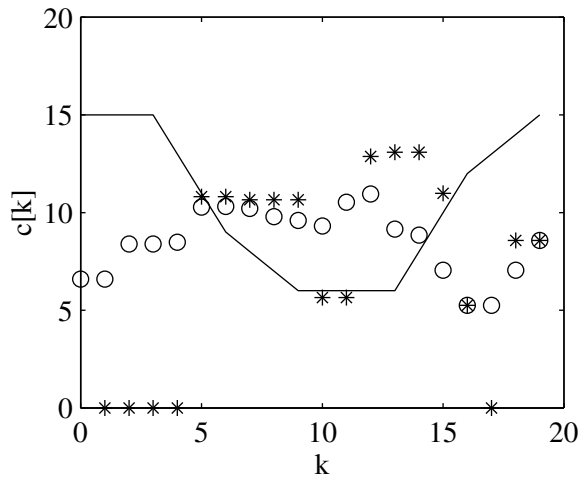

Figure 7. Target velocity: line-true velocity, *-forward estimate, O-backward estimate.

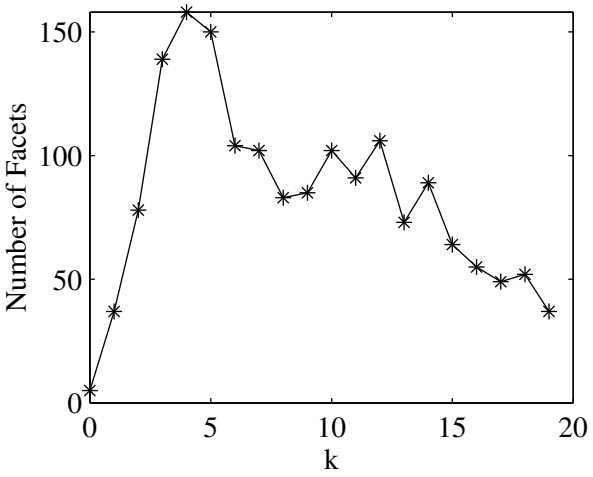

Figure 8. Number of facets in metric

ures 6 and 7. The sensor parameters for this simulation are $M = 10$, $SNR_0 = 1000$, and $\beta = 23.2$ (giving a P_{FA} of 10^{-2}).

The CSVA is initialized with a uniform density on the region $0 \leq r[0] \leq 200$ and $-10 \leq c[0] \leq 10$. The metric function is computed recursively using (3). A typical metric function is shown in Figure 5; the location of the peak of the metric function provides the estimate $\hat{x}_{k|k}$.

The position and velocity estimates obtained by the CSVA estimator are also shown in Figures 6 and 7. The true position and velocity are shown with the solid lines. The estimates track the position; the coarse position information provided by the sensors is insufficient to estimate the velocity accurately. Figure 8 shows the number of facets in the polyhedral approximations of the metric functions as a function of time. The number of facets does not grow beyond a maximum value; this value depends on the number of hyperplanes used in the approximation of $\log p_w$ and on the parameters used to determine when to combine nearly parallel hyperplanes.

5. Conclusions

A CSVA target tracker has been implemented and a preliminary evaluation of its performance has been conducted. This implementation has brought several important issues:

- As currently implemented, the algorithm requires that vertex enumeration be performed twice to compute the maximum transform. Vertex enumeration is very sensitive to numerical precision issues, which led to significant difficulty in implementing the CSVA. It may be possible to avoid these numerical problems and increase the computational speed by computing the maximum in (3) directly.
- As currently implemented, the CSVA computational complexity is very high, particularly in computing the sums in (3). This computational complexity could be significantly reduced.

Both of these issues are areas of current research. If these issues can be resolved, a more realistic model for $P(z_k|\mathbf{x}_k)$ should be investigated.

References

[1] C. Champlin, *Maximum A Posteriori State Sequence Estimation using Polyhedral Parameterization*, PhD Dissertation, Arizona State Univerity, May 2000.

[2] C. Champlin and D. Morrell, *Target Tracking using Irregularly Spaced Detectors and a Continuous-State Viterbi Algorithm*, Technical Report, Arizona State University, Nov. 2000.

[3] D. E. Larson and J. Peschon, "A dynamic programming approach to trajectory estimation," *IEEE Trans. Automatic Control*, vol. 11, pp. 537–540, July 1966.

[4] R. E. Bellman and W. Karush, "Mathematical programming and the maximum transform," *J. Soc. Indust. Appl. Math.*, vol. 10, pp. 551–567, Sep. 1962.