A novel approach to target tracking using tree search techniques is presented. The tracking problem is framed as a generalized sequential detection problem in which every possible sequence of target states is mapped to a path through the search tree. The stack algorithm for depth-first tree search is then employed to navigate the tree and identify the most likely path, or equivalently the most likely sequence of target states, by extending a single promising path in each iteration. The tree-search tracking technique can be viewed as approximating the full Bayesian inference approach by computing the posterior distribution only in regions in which it has significant mass. Unlike approaches that build on Kalman filtering techniques, the proposed stack-based tracker suffers no performance loss in the presence of nonlinear and/or non-Gaussian motion and measurement models. Simulation results show that the stack-based tracker can achieve significant performance gains over the extended Kalman filter for both linear and nonlinear motion models.

Keywords: Target tracking, sonar tracking, tree search, Bayesian inference

1 Introduction

In this work, we consider the challenge of tracking a single target in the presence of clutter. The problem of target tracking in clutter appears in a variety of contexts, including radar, sonar, and video analysis. Given a sequence of scans or images, the objective of the target tracker is to estimate the evolution of the target state over time. In one class of tracking scenarios, both the motion and measurement of the target follow linear Gaussian models. In this case, minimum mean-square error (MMSE) state estimates at each time index can be generated efficiently using a Kalman filter [1]. In many scenarios, however, assumptions of linearity and Gaussianity do not hold, and application of the Kalman filter results in performance degradation.

Nonlinear motion models may be required to accurately represent a maneuvering target; maneuvering may be employed to avoid obstacles, for example, or perhaps to thwart detection and tracking efforts. Nonlinear measurement models are present in radar and sonar tracking applications, since the target location is typically represented in rectangular coordinates within the state, but target observations are received in polar coordinates (bearing and range). There are two common approaches this problem. One is to convert the measurements to Cartesian coordinates, which generates correlated errors. The second approach is to use linearization through the Extended Kalman Filter (EKF), but the performance is highly dependent on the accuracy of the linearization [2].

In the Bayesian inference approach to tracking, the posterior distribution on the target state is derived and propagated as measurements are obtained [3]. Unlike Kalman filter-based approaches, Bayesian inference provides a solution to target tracking that works well under nonlinear and/or non-Gaussian motion and measurement models. One of the advantages of the Bayesian inference approach is the ability of the Bayesian tracker to retain previous target state and measurement information for use in estimating the current target path. A closed-form expression for the posterior distribution exists only for certain classes of models, however, and numerical computation of the full state posterior at each time index is deemed prohibitively complex in some applications. Hence, the Kalman filter or variants thereof are often used even when performance degradation due to model mismatch results [4, 5].

We propose a novel tree-search based approach to target tracking in the presence of clutter. Stack-based tree search methods were developed to provide nearly optimal maximum likelihood decoding of convolutional and tree codes while maintaining low computational complexity [6]. Complexity reduction is achieved by exploring only the most promising path in the tree at each iteration and avoiding the computation required to explore others. By mapping each possible sequence of target states to a path through the tree, we frame target tracking in clutter as a generalized sequential detection problem that can be solved through stack-based tree search. This approach eliminates the need to make inaccurate model assumptions and hence provides a framework for elegantly incorporating more realistic models that may reflect nonlinearities in both target motion and state measurement. The proposed stack-based tracking algorithm can be viewed as an approximation to the Bayesian
The inference solution that allows for complexity reduction by first exploring promising regions of the posterior distribution.

The remainder of the paper is organized as follows. The system model under which we consider the target tracking problem is described in Section 2. Section 3 provides an overview of the stack algorithm for depth-first tree search. The proposed algorithm for target tracking via stack-based tree search is described in Section 4. Simulation models and results are presented in Section 5, and Section 6 concludes the paper and considers future directions.

2 System Model

We consider tracking environments in which one target is present and moving in a two-dimensional region of interest. The target moves according to a general first order Markov model given by

\[ X_{k+1} = f_t(X_k) + G_k V_k, \]

where \( X_k \) denotes the target state vector at index \( k \), \( V_k \) denotes the target state transition noise, and \( f_t \) denotes the function governing the deterministic state transition. The (possibly nonlinear) function \( f_k \) and matrix \( G_k \) are assumed to be known. The index \( k \) denotes the arrival of the \( k \)th set of contact data, or the \( k \)th scan. Scans are assumed to be separated by \( \Delta_t \) time units, and hence the time at which the \( k \)th scan is received is given by \( k \Delta_t \).

We consider a single sensor in the region of interest; the target observation model is given by

\[ Y_k = h_k(X_k) + W_k, \]

where \( Y_k \) denotes the target observation vector at scan index \( k \), and \( W_k \) denotes the measurement noise vector, which is assumed to be independent of the state transition noise \( V_k \).

The function \( h_k \) defines the transformation from the target state to the observed quantity. The tracker receives sets of contacts (measurements that exceed a specified threshold) in the region of interest at regular time intervals \( \Delta_t \). The target is assumed to generate exactly one contact in each scan. Hence, each set of contacts includes one noisy observation of the target position, and all other contacts in the scan are caused by clutter. Contacts generated by clutter are assumed to be uniformly distributed in the two-dimensional region in which the target travels. The number of contacts generated by clutter in scan \( k \) follows a Poisson distribution with parameter \( \lambda_k \).

3 Stack-Based Tree Search

The stack algorithm was originally developed as a sequential method for decoding error-correcting tree and convolutional codes in digital communications [7]. It has more recently been suggested as a lower-complexity alternative to trellis-based techniques (e.g. the Viterbi algorithm) for maximum likelihood sequence detection of data transmitted over a dispersive channel [8, 9]. In these applications, the stack algorithm navigates a tree in search of the path, or equivalently the data sequence, with the largest likelihood, or metric. A set of possible paths and their associated metrics are stored in a list (or stack), and at each iteration, the algorithm extends the path with the largest metric.

As a simple case, consider a data sequence drawn from a binary alphabet. Starting from a known initial state, the stack algorithm considers both possible path extensions from the initial node, i.e., the single-element path segments \( b_1 = 1 \) and \( b_1 = 0 \). The metrics of these two paths are computed, and the paths are stored in the stack along with their metrics. The algorithm then selects from the stack the path with the largest metric and extends it to both children. The two extended paths are placed in the stack, and the parent path is removed. This process continues; in each iteration, the algorithm finds the most likely path in the stack, extends it to all possible children, and places the extended paths in the stack. The stack algorithm terminates when the most likely path in the stack reaches a leaf of the tree, i.e., when a full-length path has the largest metric of any path in the stack. In order to impose a fixed limit on complexity and on memory requirements, a maximum stack size \( L \) is typically specified. If \( L \) paths have been explored and are stored in the stack, the paths with the lowest metrics are purged to allow for new path extensions. A simple example of the stack algorithm is given in Figure 1.

![Figure 1](image-url)

Figure 1: An example of the first three iterations of the stack algorithm for a rate-\( \frac{1}{2} \) code. Iteration 1: The first two paths are placed in the stack along with their associated metrics. Iteration 2: Path \([0]\) is extended to \([0 0]\) and \([0 1]\), which are placed in the stack. Iteration 3: Path \([0 1]\) is extended to \([0 1 0]\) and \([0 1 1]\), which are placed in the stack. The sequence \([0 1 1]\) has the largest metric at the end of the third iteration and will be extended in iteration 4.

In this paper, we frame target tracking in clutter as a tree-search problem in which we aim to find the most likely path through the tree. The search tree is constructed such that each path represents a possible sequence of target states, and the stack algorithm navigates the tree to identify likely paths, or equivalently likely target tracks. The stack algorithm for tree search bears some resemblance to the particle filtering approaches used in implementation of the probability hypothesis density (PHD) filter. The PHD filter provides
an alternative approach to target tracking in which the first moment of the target state posterior distribution is propagated as observations are obtained, and target state estimate is drawn from peaks of the posterior [10]. Sequential Monte Carlo methods (e.g. particle filters) are used to approximate the posterior in practice, since a closed-form solution generally does not exist [11]. Particle filtering algorithms can be viewed as a modified stack algorithm in which the maximum stack size \( L \) is equal to the number of particles. Rather than extending a single path, all paths in the stack are extended at each iteration, and only the \( L \) best resulting paths are retained. In contrast to the stack-based tree search used in this work, the particle filtering approach involves significant pruning of the tree and hence possible elimination of likely paths, since the tree is unable to grow with increasing path length. Additionally, the proposed stack-based tracker avoids the problems of degeneracy and resampling (which can cause both performance degradation and increased complexity) inherent in particle filtering [12].

4 Tree Search Approach to Tracking

The idea behind tree search-based tracking is to approximate the Bayesian inference solution by using a tree search algorithm to explore and evaluate only likely regions (e.g. regions with higher amplitude) of the posterior distribution on the target state. The full Bayesian inference approach propagates the posterior distribution on the current state,

\[
P(X_k|Y^k_1 = y^k_1),
\]

where \( Y^k_1 \) denotes the observations (contacts) received from index (scan) 1 to \( k \), e.g. \([Y_1, Y_2, \ldots, Y_k] \) [3]. Since the posterior distribution can be calculated analytically for only a small number of special cases (linear Gaussian motion and measurement models, for example), discretization of the state space is often required for the Bayesian inference approach.

Application of the stack-based tracker always requires discretization of the state space, since the number of possible states must be finite in order to map each possible target state value to a branch of the search tree. The continuous posterior distribution on the state is replaced by a probability mass function (PMF). Rather than computing the full PMF each time new contact data is received, the stack-based tree search reduces complexity by computing the discretized posterior distribution only at a subset of the mass points, namely those that are predicted to be likely, or equivalently to have significant mass. This may be considered an approximate version of maximum likelihood estimation given that only the most likely states are pursued. Since nothing is thrown away in the tree structure, however, it is still possible to compute the full posterior distribution within the tree, and the search algorithm can be dynamically adapted to compute as many of the posterior mass values as desired.

To illustrate the idea of target tracking as a tree search problem, consider a toy example in which the target is moving in one dimension (along a line). Let the state consist only of position and velocity, and assume both state variables are limited to one of three possible values. The possible values of the state at any time can be represented as a grid, for this example \( 3 \times 3 \), and hence the posterior PMF has nine discrete mass points. The value of the posterior for a particular state realization \( s^* \) at index \( k \), e.g. \( P(X_k = s^*|Y^k_1 = y^k_1) \), can be associated with the grid square that represents \( s^* \). Each branch of the search tree represents a particular state value at that index (scan) \( k \), and a path through the tree represents a sequence of state values over time. Figure 2 conveys this idea by superimposing the state grid on one possible realization of the explored tree through the first three stages, \( k = 1, 2, 3 \). Even for this simple example, the tree quickly becomes very large. The advantage of a stack-based tree search technique is that the exploration of the tree is guided by the likelihood of the paths explored so far. Much of the tree is left unexplored, and hence exponential growth of the tree does not impose exponential growth in computational complexity.

4.1 Calculation of Stack Metric

In order to implement the stack algorithm, each path in the tree must be assigned a metric that is related to the likelihood of that path. In the target tracking application, each path of length \( k \) corresponds to a sequence of possible target states from index 1 to \( k \). The path metric is related to the value of the discretized posterior distribution for the particular target state and index \( k \) represented. Because the stack algorithm retains the full path (or sequence of states) from index 1 to \( k \), we may introduce a form of smoothing by designing the path metric to incorporate any number of past
states, rather than considering only the current target state. When the last $M$ states are incorporated in the path metric, the metric at index $k$ takes the form

$$
    \mathbb{P}(\mathbf{X}_{k-M+1}^k|\mathbf{Y}_{k-M+1}^k),
$$

where $1 \leq M \leq k$. We refer to (4) as the “$M$-state posterior distribution.”

We first derive the general form of the path metric in the absence of clutter. To compute the $M$-state posterior distribution, we use Bayes’ Theorem the independence of measurement noise across scans to obtain

$$
    \mathbb{P}(\mathbf{X}_{k-M+1}^n|\mathbf{Y}_{k-M+1}^n) = \frac{\mathbb{P}(\mathbf{Y}_{k-M+1}^n|\mathbf{X}_{k-M+1}^n)}{\mathbb{P}(\mathbf{Y}_{k-M+1}^n)} \approx \prod_{j=k-M+1}^k \mathbb{P}(\mathbf{Y}_j|\mathbf{X}_j) \times \mathbb{P}(\mathbf{X}_{k-M+1}^k),
$$

where we assume $\mathbb{P}(\mathbf{Y}_{k-M+1}^n|\mathbf{X}_{k-M+1}^n)$ is constant across $k$. Using the Markov state transition model given in (1), we have

$$
    \mathbb{P}(\mathbf{X}_{k-M+1}^k) = \int \prod_{j=k-M+2}^k \mathbb{q}(\mathbf{Y}_j|\mathbf{X}_{j-1}) \mathbb{P}(\mathbf{X}_{k-M+1}^j) d\mathbf{X}_{k-M+1}^j.
$$

where $\mathbb{q}(\mathbf{X}_j|\mathbf{X}_{j-1})$ denotes the state transition probability function, and the integral is over all possible $\mathbf{X}_{k-M+1}^j$. Since the path metric is dependent only upon information from time $k-M+1$ to time $k$, we assume all target states are equally likely at time $k-M+1$ and hence $\mathbb{P}(\mathbf{X}_{k-M+1}^k)$ is constant. Using this assumption and substituting (6) into (5), we obtain

$$
    \mathbb{P}(\mathbf{X}_{k-M+1}^k|\mathbf{Y}_{k-M+1}^k) \propto \prod_{j=k-M+1}^k \mathbb{P}(\mathbf{Y}_j|\mathbf{X}_j) \times \prod_{l=k-M+1}^m \mathbb{q}(\mathbf{X}_l|\mathbf{X}_{l-1}).
$$

The stack-based tracker works with any class of observation noise and requires only that the distribution of the noise be known. For the simulation scenarios we consider in this paper, the observation noise is assumed to be Gaussian. Under this assumption, we have

$$
    \mathbb{P}(\mathbf{Y}_j|\mathbf{X}_j) = \mathcal{N}(\mathbf{h}_j(\mathbf{X}_j), \Sigma_w),
$$

where $\Sigma_w$ denotes the covariance matrix of the observation noise vector. Similarly, when the state transition noise vector $\mathbf{v}_k$ is drawn from a Gaussian distribution, the conditional state probability mass function is given by

$$
    \mathbb{q}(\mathbf{X}_k|\mathbf{X}_{k-1}) = \mathcal{N}(f(\mathbf{X}_{k-1}), G\Sigma_v G^t),
$$

where $\Sigma_v$ denotes the covariance matrix of the state transition noise vector.

We now consider computation of the path metric when contacts are generated by both the target and clutter. Let $m_k$ denote the number of contacts at index $k$. Since the target is assumed to generate exactly one contact in each scan, we let $\theta_i(k), k = 1, \ldots, m_k$, denote the event that, in scan $k$, the $i$th measurement is generated by the target. The path metric can then be calculated as

$$
    \mathbb{P}(\mathbf{X}_{k-M+1}^k|\mathbf{Y}_{k-M+1}^k) \propto \mathbb{P}(\mathbf{Y}_{k-M+1}^k|\mathbf{X}_{k-M+1}^k) \mathbb{P}(\mathbf{X}_{k-M+1}^k)
$$

$$
    \propto \prod_{l=k-M+1}^k \mathbb{P}(\mathbf{Y}_l|\mathbf{X}_l) \times \int \prod_{j=k-M+2}^n \mathbb{q}(\mathbf{X}_j|\mathbf{X}_{j-1}) \mathbb{P}(\mathbf{X}_{k-M+1}^j) d\mathbf{X}_{k-M+1}^j
$$

$$
    \times \prod_{l=k-M+1}^k \sum_{i=0}^{m_k} \mathbb{P}(\mathbf{Y}_l|\mathbf{X}_l, \theta_i(l)) \mathbb{P}(\theta_i(l)|\mathbf{X}_l) \times \int \prod_{j=k-M+2}^k \mathbb{q}(\mathbf{X}_j|\mathbf{X}_{j-1}) \mathbb{P}(\mathbf{X}_{k-M+1}^j) d\mathbf{X}_{k-M+1}^j
$$

The new parameter $\mathbb{P}(\theta_i(l)|\mathbf{X}_l)$ denotes the probability that, conditioned on the current target state $\mathbf{X}_l$, the $i$th contact was generated by the target. Because we assume the target generates exactly one contact, this probability is dependent upon $\mathbf{Y}_l$, the set of all contacts at time $l$. The target-contact association probability is computed as [13]

$$
    \mathbb{P}(\theta_i(l)|\mathbf{X}_l) = \frac{\mathcal{N}(\mathbf{Y}_l(h(\mathbf{X}_l), \Sigma_w))}{\sum_{j=1}^n \mathcal{N}(\mathbf{Y}_l(h(\mathbf{X}_l), \Sigma_w))}.
$$

In this work, we have discretized the state space by discretizing the state transition noise vector $\mathbf{v}_k$. In order to best represent the assumed Gaussian state transition noise, we have chosen a discretization scheme that minimizes the distortion of a discretized Gaussian random variable as described in [14]. Let $p_i, i = 1, \ldots, N + 1$, approximate the continuous distribution $p(x)$ in the interval $[u_i, u_{i+1}]$. The intervals and the associated weights $p_i$ are calculated to minimize the mean squared error (MSE), defined as

$$
    \sum_{i=1}^N \int_{u_i}^{u_{i+1}} (x - p_i)^2 p(x) dx.
$$

Because the chosen path metric incorporates information about the state over the last $M$ scans, the stack algorithm is initialized by extending all paths from the root of the tree to length $M$. In order to reduce complexity in the initialization stage, we employ coarse discretization of the state transition noise for the first $M$ stages of the tree and finer quantization as selective path extension is performed beyond the $M$th stage.
5 Simulations

To evaluate the performance of the proposed stack-based tracking algorithm, we perform simulations for two target scenarios. In the first scenario, the target follows a linear motion model, and in the second, the target follows a nonlinear motion model that reflects the presence of obstacles. For both scenarios, the target state vector is given by $X_k = [x_k, y_k, \dot{x}_k, \dot{y}_k]^T$, where $(x_k, y_k)$ denotes the position of the target in Cartesian coordinates, and $(\dot{x}_k, \dot{y}_k)$ denotes the target velocity in Cartesian coordinates.

The same observation model is employed in both scenarios. A single sensor is placed at $(0,0)$ in Cartesian coordinates. The target measurement model is given by (2), where

$$h(X_k) = \left[ \frac{\tan^{-1}(y_k/x_k)}{\sqrt{x_k^2 + y_k^2}} \right].$$

Note that the target observation vector contains noisy measurements of the bearing and range of the target, which are nonlinear transformations of the Cartesian target position coordinates.

The measurement noise covariance matrix is given by

$$\Sigma_w = \begin{bmatrix} \left( \frac{\pi}{180} \right)^2 & 0 \\ 0 & 0.02^2 \end{bmatrix}.$$  

For evaluation purposes, the performance of the proposed stack-based tracking algorithm is compared to that of the EKF. Linearization within the EKF is based on the first derivative of the nonlinear measurement function $h(\cdot)$. To reflect the fact that the initial target state is assumed to be known (e.g. $X_0 = X_0$), the initial covariance matrix of the EKF algorithm is given by $P_0 = 5 \times 10^{-5}I_4$, where $I_4$ denotes the $4 \times 4$ identity matrix.

The probabilistic data association filter (PDAF) is employed in conjunction with the EKF to combat the effects of clutter [13]. In order to avoid numerical instability that results from the inclusion of highly unlikely clutter measurements, gating is used to define a region over which clutter contacts are generated. Our simulations use a “g-sigma” ellipsoid gate with gate volume $\lambda G$ [15]. Let the noisy target measurement in polar coordinates be written as $(\theta_0 + \theta, r_0 + r)$, where $(\theta_0, r_0)$ is the true target position, and $\theta$ and $r$ are independent zero-mean Gaussian random variables with variance $\sigma_\theta^2$ and $\sigma_r^2$, respectively. Following a similar development found in [16], we approximate the elements of the covariance matrix $S$ using the following assumptions:

- The bearing perturbation $\theta$ is sufficiently small to allow the approximations $\sin(\theta) \approx \theta$ and $\cos(\theta) \approx 1$.
- The product of bearing and range perturbations $r\theta$ is negligible in magnitude with respect to the true range $r_0$.

Under these assumptions, we have correlated Gaussian noise in Cartesian coordinates, and the elements of $S$ are given by

$$\sigma_x^2 = r_0^2 \sin^2(\theta_0) \sigma_\theta^2 + \cos^2(\theta_0) \sigma_r^2,$$

$$\sigma_y^2 = r_0^2 \cos^2(\theta_0) \sigma_\theta^2 + \sin^2(\theta_0) \sigma_r^2,$$

$$\sigma_{xy}^2 = (\sigma_\theta^2 - r_0^2 \sigma_\theta^2) \sin(\theta_0) \cos(\theta_0).$$

The number of clutter contacts generated in scan $k$ is drawn from a Poisson distribution with parameter $\lambda_k$. In the simulation results presented here, the average number of clutter contacts in the gate is held constant over time, and hence the Poisson parameter is given by

$$\lambda_k = \frac{A}{a_k},$$

where $N_c$ denotes the average number of clutter contacts in the gate, $A$ denotes the total area of the space in which the target travels, and $a_k$ denotes the area of the gate at index $k$. Because the size of the gate varies with received data, $\lambda_k$ varies with time in order to maintain a constant $N_c$.

For all simulations, the path metric considers the posterior state likelihood over the past $M = 3$ indices. In initialization of the stack algorithm, all paths are extended to length $M = 3$; each element of the state transition noise vector $V_k$ is discretized to three possible values in the first three stages of the tree. Beyond initialization, when only one path is extended at each iteration, each element of $V_k$ is discretized to seven possible values. The stack size is limited to $L = 128$ entries.

5.1 Linear Motion Model

In the first simulation scenario, the target moves according to a linear motion model given by

$$X_{k+1} = FX_k + G V_k,$$

where the covariance matrix of $V_k$ is given by $\Sigma_v = 5 \times 10^{-4}I_2$. Matrices $F$ and $G$ are given by

$$F = \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad G = \begin{bmatrix} \frac{\Delta t^2}{2} & 0 & 0 & \frac{\Delta t^2}{2} \\ 0 & \Delta t & 0 & \Delta t \\ 0 & 0 & \Delta t \end{bmatrix},$$

where $\Delta t = 10$ is the time between scans. In this model, the state transition noise vector $V_k$ determines the acceleration (in the $x$ and $y$ directions) over the time interval from $k\Delta t$ to $(k+1)\Delta t$. (In this paper, we leave time and distance parameters unitless. Realistic units are highly dependent upon the application. The results presented here apply for any choice of time and distance units as long as these choices are carried into the definition of state transition noise variance, observation noise variance, etc.) Each target path begins at point $(10,10)$ in Cartesian coordinates. The initial velocity
is drawn randomly from a zero-mean Gaussian vector with covariance matrix $\Sigma_v$.

A sample target path drawn from the motion model described above is shown in Figure 3. The observations (both from the target and from clutter) are plotted, as are the target position estimates generated by the stack-based tracker and the EKF. An average of $N_c = 1$ clutter contact was present in each gate. In this sample realization, both the stack-based tracker and the EKF successfully track the target through all 50 scans. However, the stack approach is able to much more closely follow the target path.

To evaluate the average MSE performance of the stack-based tracker relative to the EKF, both techniques have been simulated over 50 scans for 500 independent realizations following the motion and measurement models given above. The average MSE in location estimation for each algorithm is plotted as a function of index $k$ in Figure 4. Simulated performance is shown for $N_c = 0$ (no clutter), $N_c = 1$ (average of one clutter contact per gate), and $N_c = 2$ (average of two clutter contacts per gate) for both algorithms.

In the absence of clutter, the performance of the two algorithms is essentially the same, though the stack approach performs slightly better. Both algorithms maintain nearly constant MSE across time. In the presence of clutter, however, the stack-based tracker performs significantly better than the EKF. For both $N_c = 1$ and $N_c = 2$, the MSE of the stack tracker is less than half that of the EKF. In fact, the performance of the stack tracker when $N_c = 2$ is better than that of the EKF when $N_c = 1$. Since both algorithms know the exact state of the target in initialization, both are able to track the target well through the first ten scans. The significant increase in MSE for the EKF beyond $k = 10$ is a result of scenarios in which the EKF loses the target and never regains an accurate estimate of its location.

### 5.2 Nonlinear Motion Model

We also evaluate the performance of the stack-based tracker and the EKF in tracking a maneuvering target whose motion model is nonlinear. The motion model used in this evaluation is designed to simulate a maneuvering ship that travels in an area bounded by shoreline [17]. The target moves in a circular region of radius $r = 1$. The coastline is modeled by the perimeter of the circle, and a negative force moves the target back toward the center of the circle when it moves beyond the circle’s perimeter. The target motion model is given by (1), where the covariance matrix of the state transition noise is given by $\Sigma_v = 10^{-2}I_2$, and

$$G = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}.$$ (20)

The nonlinear function $f(\cdot)$ is given by

$$f(X_t) = \begin{bmatrix} x_k + \Delta t \dot{x}_k \\ y_k + \Delta t \dot{y}_k \\ \dot{x}_k + f_1(x,y) \\ \dot{y}_k + f_2(x,y) \end{bmatrix},$$ (21)

where

$$f_1(x,y) = \frac{-10x}{\sqrt{x^2 + y^2}}$$ (22)

and

$$f_2(x,y) = \frac{-10y}{\sqrt{x^2 + y^2}}$$ (23)

when the following conditions hold:
1. \( \sqrt{x^2 + y^2} > r \)
2. \( x\dot{x} + y\dot{y} \geq 0. \)

When conditions (1) and (2) do not hold, \( f_1(x, y) = f_2(x, y) = 0 \). The nonlinear functions governing velocity changes serve the purpose of keeping the target within the circle of radius \( r \). When the target is within the circle, its velocity is perturbed only by zero-mean Gaussian noise. When the target lies outside the perimeter of the circle (condition 1) and is moving away from the center of the circle (condition 2), the target velocity is modified to direct the target back into the circle.

For simulation, a single sensor is present at \((0,0)\) (the center of the circle), and the sensor observes noisy measurements of bearing and range as described by (13). The time interval between scans is given by \( \Delta t = 0.15 \), and the measurement noise covariance matrix is given by

\[
\Sigma_w = \begin{bmatrix}
(\frac{\pi}{180})^2 & 0 \\
0 & 0.01^2
\end{bmatrix}
\]  \(, \)(24)

The initial position for each simulated target path is drawn from a uniform distribution within the circle, and the initial velocity for each axis (x and y) is drawn from a zero-mean Gaussian distribution with variance 0.5.

Two sample paths drawn from the nonlinear motion model, along with the associated realizations of the nonlinear motion model with noisy observation of target bearing and range, are shown in Figures 5 and 6. For both plots, \( N_c = 1 \). In order to make the plots more readable, observations generated by clutter are not included. In Figure 5, both the stack-based tracker and the EKF are able to follow the target through all 50 scans, though the EKF position estimates show notably larger error when severe changes in target direction take place. In Figure 6, the EKF loses the target track at the first change in direction (e.g., the first time the target travels outside the perimeter of the circle) and never regains an accurate estimate of the target location. This example highlights the advantage of the stack-based tracker over a Kalman filter-based approach in the presence of severe non-linearities.

As we did for the linear motion model, we simulate the MSE performance of the stack-based tracker and the EKF over 50 scans for 500 independent realizations of the nonlinear motion model with noisy observations of target bearing and range. The average MSE in target position estimation is shown in Figure 7 for no clutter \( (N_c = 0) \), as well as for \( N_c = 1 \) and \( N_c = 2 \). The dramatic performance improvement achieved by the stack-based tracker is immediately apparent. For all values of \( N_c \), the stack-based tracker maintains a small, nearly constant MSE across all 50 scans. In contrast, the MSE of the EKF grows as \( k \) increases, even in the absence of clutter. When clutter is present, the MSE of the EKF estimates grows dramatically, reflecting the frequent realizations in which the EKF is unable to follow the target track through severe non-linearities.

6 Conclusion

We have presented a novel approach to target tracking in which each possible sequence of target states is mapped to a path through a tree, and the stack algorithm for depth-first tree search is employed to identify the most likely sequence of states. The stack-based tracker functions as an approximation to Bayesian tracking by exploring only likely regions of the state posterior distribution at each scan. Simulation results reveal that the stack-based tracker can achieve significant performance gains over the EKF and can follow the target in scenarios where the EKF frequently loses track, especially for maneuvering targets in the presence of clutter.

The work presented here is preliminary in nature, but the promising results warrant further exploration of the tree-search approach to tracking. Future work will consider track initiation, evaluation of the performance and complexity of the stack-based tracker relative to techniques such as the unscented Kalman filter and the PHD filter, and extension of the tree-search based approach to multiple target tracking.

References


Figure 6: A sample target path with target contacts, stack-based tracker estimates, and EKF estimates for the nonlinear motion model with noisy observation of target bearing and range, $N_c = 1$. (Clutter measurements are omitted to improve readability.) The stack-based tracker maintains track through 50 scans, but the EKF is unable to follow the target’s nonlinear motion in clutter and diverges.


Figure 7: Average MSE of target position estimates for the stack-based tracker and the EKF under the nonlinear motion (maneuvering target) model with noisy observation of target bearing and range. Results are plotted for $N_c = 0, 1, 2$.


