The Jump Tracker

Nonlinear Bayesian Tracking with Adaptive Meshes and a Markov Jump Process Model

Steven T. Smith*

*MIT Lincoln Laboratory, Lexington, MA 02420; stsmith@ll.mit.edu.

This work was sponsored by the United States Air Force under Air Force contract FA8721-05-C-0002. Opinions, interpretations, conclusions, and recommendations are those of the author and are not necessarily endorsed by the United States Government.
Outline

• Introduction
  – Parametric vs nonparametric Bayesian filters
  – Direct (numerical) vs Monte Carlo approaches
• The Jump Tracker concept
• Implementation and complexity
• Jump Tracker example scenario
  – Bistatic radar tracking
• Summary and conclusions
New Contributions

• Finite-state Markov jump process motion model
  – Motivation for “Jump Tracker” name
  – Computational advantages
  – Relative realism of motion model vs random walk

• Adaptive moving mesh to describe target state
  – Used for tracker solution
  – Computational advantages

• New method for direct numerical solution of “forward equation”
  – Compare to Kalman filter equations (closed-form)
  – Compare to particle filtering (Monte Carlo)
Parametric vs Nonparametric Bayesian Filters

Kalman filter

- Parametric tracker
  - Gaussian mean and covariance
- Bayes’ rule update for linear, Gaussian motion and measurement models
  - \( d + d(d+1)/2 \) nonlinear ordinary differential equations

General Bayesian filter

- Non-Parametric tracker
  - Probability density vs state
- Bayes’ rule update for nonlinear, non-Gaussian models
  - Computationally intensive

Jump Tracker filter

- Parametric tracker
  - Mesh position and probability
  - \( Nd \) nonlinear partial differential equations
- Bayes’ rule update for nonlinear, non-Gaussian models
  - Computationally tractable

See also Daum-Beneš exact filters

\( d = \) dimension
Bayesian Tracker Implementation

\[ \rho(t, x, v) = \text{target state density at time } t \]

Fokker-Planck equation
\[ \frac{\partial \rho}{\partial t} = \frac{1}{2}a^2 \frac{\partial^2 \rho}{\partial (x,v)^2} + \ldots \text{ solves Kalman filter exactly} \]

This talk

Motion Model

<table>
<thead>
<tr>
<th>Measurement Model</th>
<th>General</th>
<th>Markov</th>
<th>Gaussian</th>
<th>Linear</th>
</tr>
</thead>
<tbody>
<tr>
<td>General</td>
<td>General</td>
<td>Bayesian</td>
<td>Bayesian</td>
<td>Bayesian</td>
</tr>
<tr>
<td>Gaussian</td>
<td>Bayesian: Jump Tracker, Particle Filter</td>
<td>Extended Kalman Filter (EKF)</td>
<td>EKF</td>
<td>Kalman Filter</td>
</tr>
<tr>
<td>Linear</td>
<td>Particle Filter</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Target measurements → Tracker → Probability of position and speed
Tracker Motion and Measurement Models

**Motion model**
- Specifies target movement through state space

**Kalman filter model**
- Random walk
  \[ dx = (Ax + Bv) \, dt + Edw_1 \]
  \[ dv = (Cx + Dv) \, dt + Fdw_2 \]

**Measurement model**
- Relationship between target measurements and state space

**Kalman filter model**
- Linear and Gaussian
  \[ dy = (Gx + Hv) \, dt + Kdw_3 \]
- Continuous contact

**This talk**
Markov jump process
- Straight-line motion
  \[ dx = v_k \, dt \]
  \[ v_k = \text{general speed model} \]
  \[ k = \text{finite state from \{1, 2, ..., K\}} \]

**General model**
- Nonlinear, Non-Gaussian, Ambiguous
- Irregular contacts
  Terrain masking
  Interference
  Fluctuating radar cross section
  Jazwinski’s “continuous-discrete” model
Outline

• Introduction

• The Jump Tracker concept
  – State space and motion models
  – Forward equation

• Implementation and complexity

• Jump Tracker example scenario

• Summary and conclusions
Jump Tracker State Space: $X, Y, v_X, v_Y$

- $X, Y, v_X, v_Y$ pdf (dBkm$^{-2}$)

- Target path
- 86% contour*

*2-sigma error ellipse in Gaussian case

MIT Lincoln Laboratory
Jump Tracker Probability Law
Bayesian/Optimal Tracking

\[ \frac{\partial \rho}{\partial t} = -\text{diag}(\frac{\partial \rho}{\partial x} \cdot V) + \lambda(11^T/K - I)\rho + \frac{1}{2}a^2 \frac{\partial^2 \rho}{\partial x^2} \]

- Forward equation for finite-state Markov jump process
  - This is not the Fokker-Planck equation (no Wiener process)
- The density \( \rho(x,t) \) is a vector formed from velocity hypotheses
  - State-space partitioned into physically distinct parts
    - Moving through and space and velocity takes very different time scales
- Coupled system of deterministic linear partial differential equations (PDEs)
  - Given initial \( \rho(x,0) \), integrate by operator splitting to easy parts
    \[ \rho(x, t + \Delta t) = e^{\Delta t \lambda(11^T/K - I)} \cdot e^{\Delta t/2a^2} \frac{\partial^2}{\partial x^2} \cdot e^{\Delta t \partial \partial x \cdot V} \cdot \rho(x,0) \]
Tracker Update Methods

- **Kalman (exact) filter**
  - Easy and fast $O(d^3)$ finite-difference implementation
  - Linear/Gaussian measurement and motion models required

- **Monte Carlo (particle filtering)**
  - Easy but expensive $O(d!)$ Monte Carlo sampling
    Daum and Huang (2003) show how the “curse of dimensionality” may be avoided in some important cases
  - Greatest flexibility, but “you should never trust a Monte Carlo simulation without some method to verify that it is correct” (Daum 2004)

- **Direct numerical method: conditional density equation**
  - In general, complicated and expensive $O(d!)$ PDE solution
  - Numerical solution for given measurement and motion model
  - Recent algorithm developments may make this attractive at lower dimensions

$d = \text{dimension}$
Outline

• Introduction
• The Jump Tracker concept
• Implementation and complexity
  – Moving mesh finite element method
  – Comparison to particle filtering
• Jump Tracker example scenario
• Summary and conclusions
Nonuniform Sampling: 1-D vs N-D

**One dimension**

Solution: Interpolate using CDF

\[ \Delta \xi = \text{pdf}(x) \cdot \Delta x \]

**N dimensions**

No closed-form sampling solution!

\[ \min \frac{1}{2} \int_{\Omega} \text{tr} (\partial \xi / \partial x) G^{-1} (\partial \xi / \partial x)^T dA \]
Moving Mesh and Finite Element Methods

Adaptive meshes based on PDEs

- Mesh points adapt to where they’re needed most (r-method)
- Use of FEM methods possible
- Straightforward solution of “moving mesh partial differential equations” (Huang, Cao, and Russell)

\[
\min 1/2 \int_{\Omega} \text{tr} \left( \frac{\partial \xi}{\partial \mathbf{x}} G^{-1} \left( \frac{\partial \xi}{\partial \mathbf{x}} \right)^T \right) dA
\]

Finite element method

- Choose sample points
- Delaunay triangulation
- Apply Green’s first identity to everything in sight

\[
\int_{\Omega} \psi \nabla^2 \phi \, dA = -\int_{\Omega} \langle \nabla \psi, \nabla \phi \rangle \, dA + \int_{\partial \Omega} \psi \nabla \phi \cdot ds
\]
- Turns PDEs into linear system of equations

Advection equation

Combustion-diffusion equation

Airfoil simulation

Computational Nonlinear Filtering

How many samples to represent a \( d \)-dimensional density?

Daum and Huang’s analysis† of Monte Carlo approaches

- \( \text{accuracy}^2 = \text{Var}[X]/N \)
- \( N = \text{Var}[X]/\text{accuracy}^2 \)
- \( \text{Cost} \propto N \text{ or } N^2 \)
- \( \text{Var}[X] = d \times \sigma^2 \text{ for } d\text{-Gaussian} \)
- \( \text{Var}[X] = 2^d \times \sigma^2 \text{ for } d\text{-uniform} \)

Conclusions

- Monte Carlo filtering can avoid the curse of dimensionality with Gaussian-like densities, smart sampling, and moment-only computation
  - N.b. Monte Carlo provides mean and variance estimates, not pdfs
- In general, curse of dimensionality

Analysis of direct approaches

- \( \text{accuracy} \propto 1/N \)
- \( \text{Cost} \propto T = \#\text{Delaunay triangles}(N) \)
- \( \text{Volume}(T^d) = 1/d! \)
- \( \text{Volume}(B^d) \sim (2\pi e)^{d/2} d^{-(d+1)/2} \)
- \( T_{\text{min}} \sim (2\pi d/e)^{d/2} \text{ (exponential growth)} \)
- \( N_{\text{min}} \sim (\pi/2)^{1/d} d^{2+1/d} \)

Conclusions

- Relatively small cost for low dimensions (10s of points)
- Curse of dimensionality appears to be unavoidable

Tracker Accuracy and Sample Size

- Tracker accuracy is measured by two parameters
  - The tracker error (mean of position)
  - The area of uncertainty (AOU; variance of position)

![Graphs showing the relationship between sample size, mean error, cost, AOU error, and numerical dissipation for Jump Tracker and Particle Filters.](https://via.placeholder.com/150)

- Jump Tracker:
  - Mean error vs. sample size: $O(N^{-1})$
  - AOU error vs. sample size: numerical dissipation

- Particle Filters:
  - Mean error vs. sample size: $O(N^{-1/2})$
  - AOU error vs. sample size: $O(N^{-1/2})$
Outline

• Introduction
• The Jump Tracker concept
• Implementation and complexity
• Jump Tracker example scenario
  – Bistatic radar tracking
• Summary and conclusions
Example Scenario

Bistatic Radar from Ristic et al.

Transmitter

Receiver

100 MHz
100 kHz BW

1-\lambda aperture
200 ms CPI

Localization via:
• Bistatic delay
• Bearing (rough)
• Doppler (speed)
Bistatic Radar Measurement Model

- Measurements
  - Delay
  - Bearing
  - Doppler

- Position and velocity estimates
  - Bistatic range = $c \times$ delay
  - Range error = $c$/bandwidth/SNR$^{1/2}$
  - Bearing error = $\lambda$/aperture/SNR$^{1/2}$
  - Doppler = \[-(\langle v_T, p_T - p_X \rangle/R_{TX} + \langle v_T, p_T - p_R \rangle/R_{TR})/\lambda\]
  - Doppler error = CPI$^{-1}$/SNR$^{1/2}$
Jump Tracker Performance

- Target path
- Target track
- 86% contour

*2-sigma error ellipse in Gaussian case

- Xmtr
- Rcvr

1:00 hr:mn
86% area = 191 km²

pdf (dBkm⁻²)
Jump Tracker Performance

- **Target path**
- **Target track**
- **86% contour**

*2-sigma error ellipse in Gaussian case

- **Xmtr**
- **Rcvr**

1:00 hr:mn
86% area = 191 km²

pdf (dBkm⁻²)

-40 0 40
-40 0 40
Jump Tracker Performance
Jump Tracker Performance (mesh)
Jump Tracker Error

\[ \text{error} = || \text{mean}(86\%) - \text{truth} ||^2 \]

- **Track error (km)**
- **86\% area (km\(^2\))**

*AOU; 2-sigma error ellipse in Gaussian case*
Summary and Conclusions

- New direct method nonlinear tracker proposed ("jump tracker")
  - Straight-line motion model
    Finite-state Markov jump process velocity state
    Yields coupled system of linear PDEs for density equation
  - Moving mesh PDE and finite-element method numerical solution
  - Efficient computation of optimum nonlinear Bayesian filter
- Computational complexity roughly quantified and compared to Monte Carlo (particle filtering) for arbitrary samples, dimensions
  - Direct numerical approach may be competitive at lower dimensions
  - Expect Monte Carlo to win for large dimensions, in spite of "curse of dimensionality" for both
    Open research area / Performance cross-over unknown
- Jump Tracker applied to textbook bistatic radar tracking example
  - Good tracking performance demonstrated with a single target
  - False alarm and multiple target methods applicable as well