Roadmap Enhanced Improvement to the VSIMM Tracker via a Constrained Stochastic Context Free Grammar

Sijia Gao and Vikram Krishnamurthy*, Fellow, IEEE

Abstract—The aim of syntactic tracking is to classify spatio-temporal patterns of a target’s motion using natural language processing models. In this paper, we generalize earlier work by considering a constrained stochastic context free grammar (CSCFG) for modeling patterns confined to a roadmap. The constrained grammar facilitates modeling specific directions and road names in a roadmap. We present a novel particle filtering algorithm that exploits the CSCFG model for estimating the target’s patterns. This meta-level algorithm operates in conjunction with a base-level tracking algorithm. Extensive numerical results using simulated ground moving target indicator (GMTI) radar measurements show substantial improvement in target tracking accuracy.

I. INTRODUCTION

Consider a moving target confined to the road network illustrated in Fig. 1. Assume that the target is being tracked by a ground moving target indicator (GMTI) radar system. At each discrete time $k$, let $z_k$ denote the noisy GMTI measurement and $x_k$ denote the state vector comprising position and velocity of the target as it moves in two dimensional space. Also, $d_k$ and $l_k$ denote, respectively, the direction of motion and the location (road or intersection names) of the target.

Classical (base-level) tracking algorithms have been well studied in the literature [1] [2]. These include the variable structure interacting multiple model (VSIMM) tracker. In a VSIMM tracker, the direction sequence $d_{1:k} = (d_1, \ldots, d_k)$ is modeled as a Markov chain with state space dependent on the location sequence $l_{1:k} = (l_1, \ldots, l_k)$. These direction and location sequences are chosen so that the target is confined to roads and intersections in a roadmap.

At a higher level of abstraction (lower degree of spatial resolution and slower time scale), a moving target confined to a roadmap can be characterized by an ordered sequence of intersection names it traverses. For example, $v_0v_2v_7$ in Fig. 1 indicates a target which starts at intersection $v_0$, traverses through $v_2$ and ends at $v_7$. For convenience, we call an ordered sequence of intersection names as a pattern. In this paper, we consider a “syntactic” enhancement to the basic VSIMM setup. This syntactic enhancement operates at a higher (meta) level and models the pattern of a target. Our aim is to estimate the target’s pattern $r^*$ such that

$$r^* = \arg\max_{r \in M} p_\theta(r|z_{1:k})$$

S. Gao is with the Department of ECE, University of British Columbia, Vancouver, Canada. Vikram Krishnamurthy is with the Department of Electrical and Computer Engineering, Cornell University, N.Y., USA. vikramk@cornell.edu and is the corresponding author.

Fig. 1. Example of a roadmap. $s_0, \ldots, s_{10}$ denote roads and $v_0, \ldots, v_9$ denote intersections. Arrows depict the direction of a road (one way or two way road).

Here, $z_{1:k} = (z_1, z_2, \ldots, z_k)$ is the noisy observation sequence recorded by a GMTI radar. $r$ is a pattern, $M$ denotes a set of patterns that a radar operator is interested in and $\theta$ denotes the roadmap statistic obtained from traffic data. Once the pattern $r^*$ is estimated, it can be used as additional information to estimate the state vector $x_k$ of the target by computing the posterior $p(x_k|z_{1:k}, r^*)$.

The syntactic tracker we propose in this paper consists of two parts: a meta-level tracker and a base-level tracker. The meta-level tracker uses tracklets generated by the base-level tracker to model higher level patterns. The architecture of a syntactic tracker is illustrated in Fig. 2. Our key idea is to model the pattern of a target moving on a roadmap using a constrained stochastic context free grammar (CSCFG) on a weighted, directed graph. We then propose a novel particle filtering algorithm that combines the functionalities of CSCFG and VSIMM. Detailed numerical simulations show that the resulting tracker yields substantially improved estimates compared to a baseline tracker.

A. Syntactic Tracker Architectures

To give insight into the main ideas of this paper, we briefly describe two architectures of the syntactic trackers, namely (i) the mode based syntactic tracker proposed in this paper and
(ii) the direction based syntactic tracker proposed in earlier works [3] [4] [5].

The direction based syntactic tracker in previous works [3] [4] [5] was used to classify shapes (lines, arcs or m-rectangles) of trajectories. In such a tracker, the direction sequence \( d_{1:k} \) is modeled via a stochastic context free grammar (SCFG). A CSCFG constitutes a model for the target’s directions and locations. This is modeled by the mode

\[
q_k = \{d_k, l_k\} \quad \text{for} \quad d_k \in \theta(l_k) \quad (1)
\]

where \( d_k \) and \( l_k \) denote the direction and location of the target. In (1), \( \theta(l_k) \) is the set of possible directions of motion if a target is at \( l_k \) (See Fig. 3). The direction sequence \( d_{1:k} \) is modeled by a SCFG with state space dependent on the location sequence \( l_{1:k} \) which is a Markov chain. Hence, the mode sequence \( q_{1:k} = (q_1, q_2, \ldots, q_k) \) is a combination of a SCFG and a Markov chain which is equivalent to a CSCFG; see [6]. The architecture of the mode based syntactic tracker is illustrated in Fig. 4(b). To give further insight, we also present the classical VSIMM tracker [1] [2] which is widely used for baseline target tracking. In VSIMM, the mode sequence \( q_{1:k} \) is modeled as a Markov chain dependent on the whole roadmap instead of patterns in a mode based syntactic tracker and there is no feedback from the base-level tracker to the upper model. The architecture of the VSIMM tracker is illustrated in Fig. 4(c).

**B. Organization and Main Results**

To put this paper into context, we first recall the Chomsky hierarchy of natural languages [7]. Let \( \subset \) denote a strict subset and \( \equiv \) denote equivalent. Then

- In the deterministic case:

\[
\text{Regular Grammar} \subset \text{Context Free Grammar}
\]
In the stochastic case\textsuperscript{[6]}:

\[
\text{Markov chain} \subseteq \text{Hidden Markov chain} \equiv \\
\text{Stochastic Regular Grammar} \subseteq \text{SCFG} \subseteq \text{CSCFG} \\
\text{(2)}
\]

From a signal processing point of view, this paper deals with Bayesian signal processing algorithms for CSCFGs; the key application being syntactic target tracking. Put simply, at a meta-level, we view the spatial trajectory of a target as a sequence of noisy alphabets generated by a CSCFG language\textsuperscript{[7]}. It turns out that Bayesian estimation algorithms (Earley Stolcke parser) for CSCFGs have polynomial computational cost (in the data length). Therefore, in the context of target tracking, we are able to derive a Rao-Blackwellized particle filter which makes such models practical from an engineering point of view.

Why syntactic models for target tracking? As described in\textsuperscript{[4, 5]}, syntactic models arising in natural language processing such as SCFG and CSCFG are suitable for meta-level tracking since they form generative models for complex spatial trajectories of a target. For example, a SCFG is a generative model for a trajectory sequence \(a^n b^n\) for an integer valued random variable \(n\); this trajectory models a target moving \(n\) steps in direction \(a\) followed by \(n\) steps in direction \(b\). A Markov chain cannot exclusively generate such trajectories and is therefore not a generative model. Also there are polynomial computational cost (in the data length) Bayesian signal processing parsing algorithms for SCFGs and CSCFGs which make such models practical from an engineering point of view.

The main results of this paper are as follows: Sec. \[II\] and Sec. \[III\] discuss a 3-level model for the roadmap syntactic tracking problem. In Sec. \[IV\] a novel CSCFG-driven particle filtering algorithm is given for the mode based syntactic tracker. Sec. \[V\] describes two CSCFG based models on a square grid. In Sec. \[VI\] we compare numerical results between the mode based syntactic tracker (this paper) and the baseline VSIMM tracker. For the CSCFG based round trip model on the square grid, we compare performance between the CSCFG Viterbi tracker and the hidden Markov model (HMM) Viterbi tracker.

II. ROADMAP CONSTRAINED SYNTACTIC TRACKING: A 3-LEVEL MODEL

In this section, we construct a model for the roadmap constrained syntactic tracking problem. The model we propose operates at three levels of abstraction. At the highest level, we have the roadmap which is modeled as a directed, weighted graph. At the second level, we model the pattern (an ordered sequence of intersection names) of a target on the roadmap as a CSCFG. Finally at the lowest level, the mode sequence \(q_{1:k}\) drives the base-level VSIMM state space model which has measurements from a GMTI radar.

\footnote{In simple terms, a CSCFG permits both tree dependencies and serial dependencies. For an illustrative example that shows the difference between a Markov chain, SCFG and CSCFG, please see Sec. \[III-A\] and Fig. \[8\].}

A. Level 1: Roadmap as a Directed Weighted Graph

Our aim is to perform meta-level tracking of a target moving confined to a roadmap. We model the roadmap as a directed, weighted graph \(G = \{V, E, L\}\) with vertices \(V\), edges \(E\) and weights \(L\).

\[
G = \{V, E, L\}
\]

The set of vertices \(V = \{v_1, v_2, \ldots, v_n\}\) denotes the \(n\) road intersections. The set of edges \(E = \{e_{v_i,v_j} \mid v_i, v_j \in V\}\) denotes the roads on the roadmap. The set of weights \(L = \{|e_{v_i,v_j}| \mid e_{v_i,v_j} \in E\}\) denotes lengths of the roads. Define a function \(\theta\) on both vertices \(V\) and edges \(E\). \(\theta(e_{v_i,v_j}), \forall e_{v_i,v_j} \in E\) denotes the angle of road \(e_{v_i,v_j}\) with respect to a reference coordinate and \(\theta(v_m), \forall v_m \in V\) denotes the set of angles of roads that intersect at the vertex \(v_m\). The directed, weighted graph for the roadmap in Fig. \[1\] is presented in Fig. \[5\].

B. Level 2: CSCFG Model on Directed Weighted Graph for Targets Constrained to a Roadmap

The second level of the syntactic tracking model is a CSCFG model that ensures the target is confined to the roadmap \(G = \{V, E, L\}\). The model determines the target’s mode sequence \(q_{1:k}\) where

\[
q_k = \{d_k, l_k\} \quad \text{with} \\
d_k \in \theta(l_k), l_k \in V \cup E \\
\text{(3)}
\]

Here, \(d_k\) is the direction of motion of the target at time \(k\). \(l_k\) denotes the edge (road) or vertex (intersection) where the target is located on \(G\). \(\theta(\cdot)\) is defined in \[1\]. A CSCFG is a 5-tuple of the form

\[
\text{CSCFG} = \{N, \mathcal{T}, S, R, P\}
\]

where \(N\) is a finite set of nonterminals and \(T\) is a finite set of terminals (also called the alphabet) such that \(N \cap T = \emptyset\). \(S \in N\) is chosen to be the start symbol. \(R\) is a set of production rules of the form

\[
\begin{align*}
X & \rightarrow \lambda a, \quad X \in N, X \neq S, a \in T, \lambda \in \{N \cup T\}^* \\
S & \rightarrow \lambda
\end{align*}
\]

which indicates the nonterminal \(X\) can be replaced with \(\lambda\) if the previous terminal is \(a\). \(P : R \rightarrow \{0, 1\}\) is a probability

\begin{figure}[h]
\centering
\includegraphics[width=0.4\textwidth]{fig5.png}
\caption{Formulation of the roadmap in Fig. \[1\] as a directed, weighted graph. \(v_0, \ldots, v_9\) denote road intersections.}
\end{figure}
function over production rules in $R$ such that
\[ \sum_{n \in N_X} P(X \rightarrow \lambda | a) = 1 \]
\[ \sum_{n \in N_S} P(S \rightarrow \lambda) = 1 \]

Here, $n_{X,a}$ is the number of rules in $R$ associated with the nonterminal $X$ and the terminal $a$. $n_S$ is the number of rules that have the start symbol $S$ on the left side of the arrow. Starting from $S$, replace the leftmost nonterminal (such deviations can be represented as a parse tree [3]) according to the production rules in $R$ and probabilities in $P$, the output of a CSCFG is a string of terminals. A parsing algorithm for the CSCFG is illustrated in Appendix [A].

Types of Target and Traffic Models: For illustrative purposes, we consider two types of target and traffic models for syntactic tracking. Let $e_k$ denote the speed of a target at time $k$.

1) Constant Traffic Flow or Constant Speed Model: Assume that the traffic flow $c$ throughout the road network is a positive random variable. Then the time taken to traverse a road segment of length $l(v_{i,v_j})$ with speed limit $\zeta_i$ is $l(v_{i,v_j})/\left(\zeta_i - c\right)$.

Alternatively, if the speed on a road or an intersection is an unknown constant modeled by a random variable $c$, then we have
\[ e_k = c \quad (4) \]

Constant speed targets can model (approximately) pedestrians and bicycles.

2) Average Speed Model: Here the speed on a road at time $k$ equals the speed averaged over all vehicular traffic moving on that road. Therefore, the speed on a road of vehicular traffic is location based (different roads) and time varying (peak or non-peak hours). Speed at an intersection is a known constant (denoted by $c_2$). Then we have
\[ e_k = \begin{cases} \pi_E(l_k,k) & l_k \in E \\ c_2 & l_k \in V \end{cases} \quad (5) \]

where $l_k$ is defined in [3]. $\pi_E(l_k,k)$ denotes the speed averaged over all vehicular traffic moving on $l_k$ at time $k$.

C. Level 3: GMTI Base Level Model

Here, we describe the third and final component of our 3-level tracking model. We construct a VSIMM model for the base-level target’s kinematics[2] which are measured by a GMTI radar system.

The target’s state evolves as
\[ x_k = f(x_{k-1}, d_{k-1}, e_{k-1}, l_{k-1}) + \tilde{w}_k(d_k) \]
\[ l_k = B(x_k), d_k \in \theta(l_k) \quad (6) \]

Here, $x_k = [x_k^T, y_k^T, \dot{x}_k^T, \dot{y}_k^T]^T$ is the 4-dimensional state vector of the target at time $k$ that comprises position and velocity components in the $x$ and $y$ directions. $d_k$ and $l_k$ are defined in [3]. $e_k$ is the speed of the target specified in [3] and [5]. $\theta(\cdot)$ is defined in [1]. $B$ is the function that maps $x_k$ to $l_k$. $f$ is a nonlinear function and models the target’s state process:
\[ f = \begin{cases} f_0 & \text{if } d_k = d_{k-1}, e_k = e_{k-1} \\ f_1 & \text{otherwise} \end{cases} \quad (7) \]

where
\[ f_0 = Fx_{k-1} \]
$F$ is the state matrix defined by
\[ F = \begin{bmatrix} 1 & 0 & T & 0 \\ 0 & 1 & 0 & T \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \]

where $T$ is the interval between GMTI measurements. $f_1$ in (7) denotes transforming the velocity components of $Fx_{k-1}$ to $e_k \cos d_k$ in the $x$ direction and $e_k \sin d_k$ in the $y$ direction.

The state noise $\tilde{w}_k(d_k)$ in (6) is a zero-mean white Gaussian process with covariance matrix $Q_k(d_k)$ computed as
\[ Q_k(d_k) = GQ_k(d_k)G' \]
\[ G = \begin{bmatrix} T^2/2 & 0 & 0 & 0 \\ 0 & T^2/2 & 0 & 0 \\ T & 0 & 1 & 0 \\ 0 & T & 0 & 1 \end{bmatrix}, \quad Q_k(d_k) = \rho_{d_k} \begin{bmatrix} \sigma^2_x & 0 \\ 0 & \sigma^2_y \end{bmatrix} \rho_{d_k}' \]

with
\[ \rho_{d_k} = \begin{bmatrix} \sin d_k & \cos d_k \\ -\cos d_k & \sin d_k \end{bmatrix} \]

where $'$ denotes transpose, $\sigma^2_a$ is the variance along the direction of motion indicated by $d_k$ and $\sigma^2_o$ is the variance along the direction of motion orthogonal to $d_k$.

The observation equation is specified by the GMTI radar:
\[ z_k = h(x_k, c_k) + v_k \text{ where} \]
\[ h(x_k, c_k) = \sqrt{(x_k - x^*_k)^2 + (y_k - y^*_k)^2 + (z^*_k)^2} \]
\[ \frac{(\dot{x}_k - \dot{x}^*_k)(x_k - x^*_k) + (\dot{y}_k - \dot{y}^*_k)(y_k - y^*_k)}{\sqrt{(x_k - x^*_k)^2 + (y_k - y^*_k)^2 + (z^*_k)^2}} \]
\[ \frac{180}{\pi} \tan^{-1}(y_k - y^*_k, x_k - x^*_k) \quad (8) \]

Here, $z_k = [r_k, \dot{r}_k, \theta_k]'$ denotes the 3-dimensional noisy observation vector recorded by a GMTI radar at time $k$. $r_k$, $\dot{r}_k$, $\theta_k$ denote, respectively, the range, range rate and azimuth (in degrees, $(-180^\circ, 180^\circ]$). $c_k = [x_k^T, y_k^T, \dot{x}_k^T, \dot{y}_k^T]'$ is the 4-dimensional state vector for the phase center of the GMTI radar’s antenna on the aircraft it is mounted on. It comprises position and velocity components in the $x$ and $y$ directions. $z^*_k$ is the (constant) altitude of the aircraft and the (constant) altitude of the target is assumed to be zero. $\tan^{-1}$ denotes the four-quadrant inverse tangent (in radians). The observation noise $v_k$ in (8) is assumed to be a zero-mean white Gaussian process with covariance matrix
\[ R = \begin{bmatrix} \sigma_{r_k}^2 & 0 & 0 \\ 0 & \sigma_{\dot{r}_k}^2 & 0 \\ 0 & 0 & \sigma_{\theta_k}^2 \end{bmatrix} \]

where $\sigma_{r_k}$, $\sigma_{\dot{r}_k}$ and $\sigma_{\theta_k}$ are standard deviations for range, range rate and azimuth, respectively. Note that $R$ is a diagonal
matrix reflecting the assumption that the errors in the range, range rate and azimuth are uncorrelated.

III. EXAMPLES OF PATTERNS

Given the three level model described in Sec. II.1 we now elaborate on the Level 2 CSCFG model described in Sec. II-B. In particular, we discuss the two examples of constant speed targets and vehicular traffic in more detail.

First we need to define what a pattern is; recall that a pattern was defined informally in Sec. II as an ordered sequence of road intersection names. More formally, given the roadmap graph \( G = \{ V, E, L \} \) formulated in Sec. II-A, a pattern is characterized by the sequence

\[
\rho_{\text{CN}} = (r_0, \ldots, r_n)
\]

where \( r_i \in V \), \( i = 0, 1, \ldots, n \),

\[
e_{r_ir_{i+1}} \in E \quad i = 0, 1, \ldots, n - 1
\]

(9)

We assume that apart from kinematic measurements obtained by the radar, the type of a target is also known. In Sec. II-B and Sec. III-C we build CSCFG based models for constant speed targets and vehicular traffic, respectively.

A. A Simple Illustrative Example

To give some intuition about the difference between a Markov chain, CSCFG used in earlier work, and a CSCFG used in this paper (recall (3)), consider the following example.

Markov Chain. Construct a first-order Markov chain with trajectory \( a_1, \ldots, a_n \) for some fixed time \( n \). The dependency structure, which is a chain graph, is shown in Fig. 5(a).

SCFG and Arc Trajectory. An arc is an example of a trajectory with a SCFG generative model.

1) Generate \( m, n \) as random positive integers with a pre-specified distribution.

2) Then generate the following three iid finite state sequences \( a_1, \ldots, a_m, b_1, \ldots, b_n \) and \( c_1, \ldots, c_m \) with specified probabilities. Concatenate these into a single string.

The dependency structure, which is a tree graph, is shown in Fig. 5(b). It can be verified via a pumping lemma that a Markov chain is not a generative model for an arc since the integer \( m \) has an arbitrary (random) value.

CSCFG. A CSCFG trajectory has more general dependencies as follows.

1) Generate \( m, n \) as random positive integers with some pre-specified distribution.

2) Then generate the following three Markovian finite state sequences \( a_1, \ldots, a_m, b_1, \ldots, b_n \) and \( c_1, \ldots, c_m \) with specified transition probabilities. Concatenate these into a single string.

The dependency structure, which is a tree-chain graph, is shown in Fig. 5(c). The main point is that a CSCFG model facilitates both serial and tree dependencies. We refer to [9] for a detailed discussion of CSCFGs. In the remainder of this section we describe more sophisticated examples involving constant traffic flow and average speed models.

B. Constant Traffic Flow or Constant Speed Model

Here, we provide more details of the constant traffic flow or constant speed model in Sec. II-B. Given a pattern \( \rho_{\text{CN}} \), let \( m_{r_ir_{i+1}}, i = 0, 1, \ldots, n - 1 \), denote the sequence of vectors comprising directions and locations of a target as it traverses from vertex \( (i, j) \) to \( (i, j+1) \).

To provide a concrete example, consider the setup in Fig. 7(a) where a target moves from \( r_i \) to \( r_{i+1} \). It moves in a constant direction \( \theta(e_{ri,r_{i+1}}) \) and via the locations \( r_i, e_{r_ir_{i+1}}, r_{i+1} \). Therefore,

\[
m_{r_ir_{i+1}} = \{ \theta(e_{r_ir_{i+1}}), r_i \}^{i_1} \{ \theta(e_{r_ir_{i+1}}), e_{r_ir_{i+1}} \}^{i_2} \{ \theta(e_{r_ir_{i+1}}), r_{i+1} \}^{i_3}
\]

(10)

where \( i_1, i_2, i_3 \) denote positive integers. In words: the target moving from \( r_i \) to \( r_{i+1} \) generates vectors \( \{ \theta(e_{r_ir_{i+1}}), r_i \} \) for \( i_1 \) time instants, followed by generating vectors \( \{ \theta(e_{r_ir_{i+1}}), e_{r_ir_{i+1}} \} \) for \( i_2 \) time instants, finally followed by generating vectors \( \{ \theta(e_{r_ir_{i+1}}), r_{i+1} \} \) for \( i_3 \) time instants. Therefore the total amount of time taken for traversing from \( r_i \) to \( r_{i+1} \) is

\[
| m_{r_ir_{i+1}} | = i_1 + i_2 + i_3
\]

\( L_{r_0,n} \), is denoted as the sequence of vectors comprising directions and locations of a constant speed target as it traverses vertex \( r_0, r_1, \ldots, r_n \). \( L_{r_0,n}^{cs} \) is the concatenation of \( m_{r_0r_1}, m_{r_1r_2}, \ldots, m_{r_{n-1}r_n} \), see Fig. 7(b). The key point is that since the target moves with constant speed, the time \( | m_{r_ir_{i+1}} |, i = 0, 1, \ldots, n - 1 \) taken to traverse from \( r_i \) to \( r_{i+1} \) is proportional to the length of the edge connecting \( r_i \) and \( r_{i+1} \), namely, \( l(e_{r_ir_{i+1}}) \). Therefore,

\[
L_{r_0,n}^{cs} = \bigoplus_{i=0}^{n-1} m_{r_ir_{i+1}} \quad \text{with}
\]

\[
| m_{r_0r_1} | : | m_{r_1r_2} | : \ldots | m_{r_{n-1}r_n} | \quad (11)
\]

Here, “\( \bigoplus \)” denotes concatenation. It can be shown that a generative model for \( L_{r_0,n}^{cs} \) and \( L_{r_0,n}^{cs} \) is a context sensitive grammar. In this paper, we approximate \( L_{r_0,n}^{cs} \) by generating two patterns with specified transition probabilities. Concatenate these into a single string.

\[
L_{r_0,n}^{cs} = \bigoplus_{i=0}^{n-1} m_{r_ir_{i+1}} \quad \text{and}
\]

\[
| m_{r_0r_1} | : | m_{r_1r_2} | : \ldots | m_{r_{n-1}r_n} | \quad (12)
\]

A CSCFG that generates \( L_{r_0,n}^{cs} \) \( (n = 1, 2, 3, 4) \) is given in Appendix C. The architecture of the mode based syntactic tracker for constant speed targets is given in Fig. 8(a).

C. Average Speed Model

Here, we describe the average speed model (denoted by \( L_{r_0,n}^{av} \)) described in Sec. II-B given a pattern \( \rho_{\text{CN}} \).

The average speed model \( L_{r_0,n}^{av} \) is the sequence of vectors comprising directions and locations of a vehicular traffic that starts from \( r_0 \) traverses through \( r_1, r_2, \ldots, r_{n-1} \) and ends at \( r_n \). From Fig. 7(b), we have

\[
L_{r_0,n}^{av} = \bigoplus_{i=0}^{n-1} m_{r_ir_{i+1}}
\]
where \( m_{r_i r_{i+1}} \), \( i = 0, 1, \ldots n-1 \) is defined in (10) and “⊕” is defined in (11). There is no additional constraint on \( |m_{r_i r_{i+1}}| \), \( i = 0, 1, \ldots n-1 \) because the speed of a vehicular traffic is location based and time varying (5). \( L_{cs r_0:n}^l \) can be modeled via a stochastic regular grammar which is a strict subset of CSCFGs (2). The architecture of the mode based syntactic tracker for vehicular traffic is illustrated in Fig. 8(b).

**IV. CSCFG-driven Particle Filter Tracker**

Thus far, we have discussed a 3-level model for a target confined to a roadmap. In this section, we describe a novel natural language based particle filtering algorithm for estimating the coordinates of the target.

For the 3-level model proposed in Sec. [II] and Sec. [III] direct computation of the posterior distribution is computationally intractable. Therefore, given the noisy observation sequence \( z_{1:k}, k = 1, 2, \ldots \), our aim is to compute the posterior distribution

\[
p(s_k|z_{1:k}, r_{0:n}^*) \quad \text{where} \quad s_k = (x_k, q_k, e_k), r_{0:n}^* = \arg \max_{r_{0:n} \in M} p(r_{0:n}|z_{1:k})
\]

Recall from [4], [5], that \( e_k \) denotes the speed of the target at time \( k, x_k, q_k \) are defined in [6], [3] and \( r_{0:n} \) denotes a pattern defined in [7]. In this section, a Rao-Blackwellized CSCFG-driven particle filtering algorithm is derived (see Algorithm [1]).
where \( N \) is the number of particles. The particle weights \( w_k^i \) are computed recursively as

\[
p(s_{1:k}^i | z_{1:k}, r_{0:n}) = \frac{p(s_{1:k}^i | z_{1:k}, r_{0:n})}{\sum_{i=1}^{N_p} w_k^i} \delta(s_{1:k}^i)
\]

where \( N_p \) is the number of particles. The particle filtering algorithm presented below exploits the fact that given the kinematic state \( x_k \), the modes can be estimated via a finite dimensional predictor in terms of the Earley-Stolcke parser.

For constant speed targets

\[
p(s_{1:k}^i | z_{1:k-1}, r_{0:n}) = p(s_k^i | z_{1:k-1}, \mathcal{L}_{r_{0:n}}^{cs})
\]

where \( \mathcal{L}_{r_{0:n}}^{cs} \) is the constant speed model discussed in Sec. III-B. For the vehicular traffic,

\[
p(s_{1:k}^i | z_{1:k-1}, r_{0:n}) = p(s_k^i | z_{1:k-1}, \mathcal{L}_{r_{0:n}}^{vt})
\]

where \( \mathcal{L}_{r_{0:n}}^{vt} \) is the average speed model discussed in Sec. III-C.

The particle approximation to the full posterior distribution is

\[
p(s_{1:k} | z_{1:k}, r_{0:n}) \approx \sum_{i=1}^{N_p} w_k^i \delta(s_{1:k}^i)
\]

Figure 8. Architectures of the mode based syntactic trackers for two types of targets. (a) is the architecture of the mode based syntactic tracker for constant speed targets. (b) is the architecture of the mode based syntactic tracker for vehicular traffic. \( r_{0:n} \) is a pattern defined in (9). \( \mathcal{L}_{r_{0:n}}^{cs} \) and \( \mathcal{L}_{r_{0:n}}^{vt} \) are, respectively, the constant speed model and the average speed model for pattern \( r_{0:n} \). \( k \) is the discrete time, \( z_{1:k} \) denotes the noisy observation sequence recorded by a GMTI radar and \( x_k \) is the state vector of the target. The mode \( q_k \) is defined in (9).

Choose the bootstrap proposal distribution

\[
\pi(s_k^i | z_{1:k-1}, z_{1:k}, r_{0:n}) = p(s_k^i | z_{1:k-1}, r_{0:n})
\]

(14)

The first term is the CSCFG one-step predictor in (25). The second term is computed in terms of (4) and (5). The third term in (14), according to (6) is

\[
p(s_k^i | z_{1:k-1}, d_{k-1:k}, e_{k-1:k}) = \mathcal{N}(f(z_{k-1}^i, d_{k-1:k}, e_{k-1:k}), \hat{Q}_k(d_k^i))
\]

(15)

where \( f(\cdot) \) and \( \hat{Q}_k(d_k^i) \) are defined in (6). Recursive computation of \( w_k^i \) before normalization is

\[
\hat{w}_k^i = p(z_k | s_k^i) w_{k-1}^i \quad \text{with} \quad p(z_k | s_k^i) = \begin{cases} 0 & l_k^i \neq B(x_k^i) \\ \mathcal{N}(h(x_k^i, c_k), R) & l_k^i = B(x_k^i) \end{cases}
\]

(16)

where \( h(\cdot), c_k, R \) are defined in (8) and \( B \) is defined in (6). The CSCFG-driven particle filtering algorithm is illustrated in Algorithm 1.

**Algorithm 1 CSCFG-Driven Particle Filter Tracker**

**Function** CSCFG-PF \( \{s_{1:k-1}^i, w_{k-1}^i, z_k, r_{0:n}\} \)

**for** \( i = 1 \) to \( N_p \) **do**

Sample \( q_k^i \) according to (25)

Sample \( e_k^i \) according to (4) or (5)

Sample \( x_k^i \) according to (15)

Compute \( \hat{w}_k^i \) using (16)

**end for**

\( W_k = \sum_{i=1}^{N_p} \hat{w}_k^i \)

Normalize \( w_k^i = \hat{w}_k^i / W_k \), \( \forall i = 1, 2, \ldots, N_p \)

\( N_{\text{effective}} = \sum_{i=1}^{N_p} \hat{w}_k^i \)

**if** \( N_{\text{effective}} < \text{threshold} \) **then**

RESAMPLE

**for** \( i = 1 \) to \( N_p \) **do**

\( w_k^i = \frac{1}{N_p} \)

**end for**

**end if**

**return** \( \{s_{1:k}^i, w_k^i\} \)
Decision Directed Scheme

Strictly speaking, we should assign each particle a CSCFG parser. Here, we apply a decision-directed scheme introduced in [9], namely, \( N_p \) particles drive a single parser. Denote \( \hat{q}_k \) the soft estimate at time \( k \) and is computed as

\[
p(\hat{q}_k = m) = \sum_{i=1}^{N_p} \delta(q^i_k - m)w_k^i
\]

(17)
The one step predictor in (14) is approximated by

\[
p(q^i_k | q^i_{k-1}, r_{0:n}) \approx \sum_{i=1}^{N_p} \delta(q^i_k - m)w_k^i
\]

(18)

where \( q_{1:k-2} = (q_1, q_2, \ldots, q_{k-2}) \). (18) is computed via (26) in Appendix A. A CSCFG-driven particle filtering algorithm with a decision directed scheme is presented in Algorithm 2. The posterior probability \( p(r_{0:n} | z_{1:k}) \) in (13) is computed using Bayes’ formula,

\[
p(r_{0:n} | z_{1:k}) = \sum_{r_{0:n} \in M} p(z_{1:k} | r_{0:n}) p(r_{0:n})
\]

Here, \( p(z_{1:k} | r_{0:n}) \) equals the prefix probability \( p(\hat{q}_{1:k} | r_{0:n}) \) computed by (27). \( M \) denotes a set of patterns that a radar operator is interested in. \( p(r_{0:n}) \) is the prior probability for pattern \( r_{0:n} \) and \( \sum_{\forall r_{0:n} \in M} p(r_{0:n}) = 1. \)

It is assumed that the target is moving confined to a square grid illustrated in Fig. 9 and is tracked by a GMTI radar. The aim is to compute the maximum a posterior sequence estimate

\[
x_{1:n}^* = \arg\max_{x_{1:n}} p(x_{1:n} | z_{1:n}, L)
\]

(19)

Here, \( x_{1:n} = (x_1, x_2, \ldots, x_n) \) denotes the target’s state sequence and \( z_{1:n} = (z_1, z_2, \ldots, z_n) \) denotes the noisy GMTI radar measurement sequence; \( x_k, \hat{q}_k, k = 1, 2, \ldots, n \) are defined later. \( L \) is the round trip or cost constrained palindrome trajectory modeled by a CSCFG.

Let \( o_{ij} \) denote the node with Cartesian coordinates \((i, j)\) and \( G_{grid} \) denote the set of all nodes of the square grid. Define the following relationships on nodes in \( G_{grid} \)

\[
o_{ij} \uparrow \downarrow o_{i'j'} \text{ if } i' = i, j' = j \text{ or } i' = i, j' = j
\]

(20)

At each discrete time, we assume a target can only move up, down, left, right to its adjacent nodes. \( x_k \) is specified by

\[
x_k = \{x_k, y_k\} \in G_{grid}
\]

(21)

\( z_k \) is the two dimensional (comprising range and azimuth components) noisy observation vector recorded by a GMTI radar and the observation function is described in [6]. Note that the target’s position in the \( (x, y) \) direction equals the coordinate in the \( (x, y) \) direction times the size of the unit grid.

A. Example 1. Round Trip Model

In the round trip model, denoted by \( L_{round} \), a target departs from some node \( A \), arrives at another node \( B \) and finally returns to \( A \). On the forward trip, it is assumed that the target can only move up and right; while on the return trip, the target can only move down and left. The resulting target’s trajectory is as follows:

Definition 1. \( L_{round} = x_{1:n} \) where

\[
\begin{align*}
\begin{cases}
X_k \uparrow \downarrow X_{k+1} & \forall k = 1, 2, \ldots, n-1 \\
X_k \downarrow \uparrow X_{k+1} & X_k \downarrow \left\{ \begin{array}{l}
\uparrow k = \frac{n+1}{2} \\
\downarrow k = \frac{n+3}{2}, \ldots, n-1
\end{array} \right.
\end{cases}
\end{align*}
\]

(20)

Here, \( \uparrow \), \( \downarrow \) are defined in (20) and \( x_k \) is defined in (21). An example of the round trip model is presented in Fig. 9. \( L_{round} \) indicates a target moves to some other node and returns to the start point with minimal distance and possibly minimal probability being detected (routine of return trip is probably different from that of forward trip). From Def. 1, \( L_{round} \) is the concatenation of two equal length (tree dependency) Markov chains (serial dependency): \( x_{1: \frac{n+1}{2}} \) and \( x_{\frac{n+3}{2}:n} \). The transition probability is dependent on the traffic data. Such a tree and serial combined dependency cannot be modeled via a SCFG as explained in Sec. III-A. \( L_{round} \) is modeled via a CSCFG and details are illustrated in Appendix B.
B. Example 2. Cost Constrained Palindrome

In a palindrome trajectory model, a target starts from some node \( A \), moves to some node \( B \) and then retraces its path to \( A \). Several examples in surveillance involve detecting palindrome trajectories of targets. An example of \( L_{\text{palindrome}} \) is shown in Fig. 9. It is well known that a SCFG is a generative model for a palindrome.

Here we consider a generalization called the cost constrained palindrome which has serial dependencies and therefore requires a CSCFG model. The serial dependencies are introduced by considering the process \( \Delta_k \) which denotes the cost (threat level or fuel consumption) incurred at each time \( k \). \( \Delta_k \) is modeled by a Markov chain with state space dependent on \( x_{k-1} \) and \( x_k \). Then \( \sum_{i=1}^{k} \Delta_i \) denotes the cost (fuel or funds) incurred up to time \( k \). We assume that the target can only continue moving if the total cost (threat) accrued lies within the bound: \( \sum_{i=1}^{k} \Delta_i \leq C \) for some pre-specified constant \( C \).

The cost constrained palindrome model, denoted by \( L_{\text{palindrome}} \) is formally specified as follows:

**Definition 2.** \( L_{\text{palindrome}} = \{ y_{1:n}, \ y_k = \{ x_k, \Delta_k \} \mid \) where

\[
\begin{align*}
    x_k &\xrightarrow{\text{adjacent}} x_{k+1}, \ k = 1, 2, \ldots, n - 1 \\
    x_n x_{n-1} \cdots x_{n+1} &\sim x_1 x_2 \cdots x_{n+1} \\
    x_k &\sim p(x_k | x_{k-1}), \ k = 1, 2, \ldots, n + 1 \\
    \Delta_k &\sim p(\Delta_k | \Delta_{k-1}, x_k, x_{k-1}), \ k = 1, 2, \ldots, n \\
    \sum_{i=1}^{k} \Delta_i &\leq C, \ k = 1, 2, \ldots, n
\end{align*}
\]

Here, \( x_k \) is defined in (21) and \( \xrightarrow{\text{adjacent}} \) is defined in (20). The third item in Def. 2 is dependent on the traffic data and the fourth item specifies the evolution of the cost process.

Note that the second, third and fourth items in Def. 2 exhibit a combined tree and serial dependency which cannot modeled by a SCFG; recall the discussion in Sec. III-A. Details of \( L_{\text{palindrome}} \) constructed as a CSCFG are given in Appendix B.

Finally, computing the MAP sequence estimate in (19) for a round trip or a cost constrained palindrome trajectory uses a CSCFG Viterbi parser which is illustrated in Fig. 10. The CSCFG Viterbi parsing algorithm is detailed in Appendix B.

VI. NUMERICAL EXAMPLES

This section presents three detailed numerical examples involving CSCFGs for meta-level tracking. For the first two examples, the setup is as follows. The mode based syntactic tracker proposed in this paper estimates the state moving of a target that is confined to a roadmap given noisy GMTI radar observations. The main algorithm (Algorithm 2) is a Rao-Blackwellized particle filter which combines the Earley Stolcke parser with a base-level particle filter. The empirical performance of the mode based syntactic tracker (denoted by \( M \) in the figures below) is evaluated by simulating measurements from a GMTI radar against a target moving on the road network described by Fig. 1. See Table I for the primary properties of the radar measurements; further detail is provided in Appendix D. Note we are assuming no missing measurements (i.e. the probability of detection is unity). The aircraft where a GMTI radar is mounted starts from (-30 km,-30 km) with (constant) altitude 3000m and moves with constant velocity (100 m/s in the x direction and 20 m/s in the y direction). We also evaluate the performance of the baseline VSIMM tracker (proposed in [1] [2]) for comparison. The performance are evaluated by the simulated sample path state root mean square error averaged over 50 independent trials. Root mean square error at time \( k \) is computed as

\[
RMSE(k) = \sqrt{\frac{1}{N} \sum_{n=1}^{N} (\hat{x}_{k,n}^t - x_k^t)'(\hat{x}_{k,n}^t - x_k^t)}
\]

Here, \( \hat{x}_{k,n}^t \) is the estimated target state vector at time \( k \) in the \( n_{th} \) trial and \( x_k^t \) is the target’s true state vector at time \( k \).

A. Example 1. Constant Speed Targets

Consider the model of Sec. III-B. The speed of a constant speed target is an unknown constant and we assume its initial

---

4Here are three examples: (i) For a smuggler, if the forward trip from \( A \) to \( B \) is safe (e.g., no guards), then retracing this path from \( B \) to \( A \) minimizes being captured. (ii) A vehicle (bus) transports passengers from \( A \) to \( B \) and on the return trip transports passengers from \( B \) to \( A \) via the same stops. (iii) Retracing the path also occurs when searching for a dropped or lost item.
distribution is evenly distributed on 3 states: 2 m/s, 3 m/s and 4 m/s. Performance of the mode based syntactic tracker and the baseline VSIMM tracker when tracking constant speed targets under different patterns are displayed in Table II. The “improvement” in the final column is computed as

\[
\text{improvement} = 1 - \frac{\text{average RMSE}_{M}}{\text{average RMSE}_{VSIMM}}
\]

where average (\(\text{RMSE}_{M}\)) is the average ratio between the RMSE for the mode based syntactic tracker and the RMSE for the VSIMM tracker at each discrete time. An example of RMSEs for the two trackers is illustrated in Fig. 11(a). We see that the mode based syntactic tracker performs better than the VSIMM tracker in terms of the positive “improvement” which is more than 5% under 3 specific patterns: \(v_3v_0v_2v_3\), \(v_3v_0v_3v_8\), \(v_0v_1v_1v_4\). The explanation is that the meta-level modeling in the mode based syntactic tracker is pattern based and therefore the tracker can definitely know which road to move on to after passing an intersection. In addition, CSCFG at the meta-level can model longer and more complicated dependencies compared with a Markov chain such that properties of a constant speed model discussed in Sec. III-B can be well captured.

Table III

The average speed model of Sec. III-C which is suitable for vehicular traffic. For vehicular traffic, the simulations assume that the speed on roads is location-based and time-varying and is a known constant at intersections (5 m/s); see Table VII in Appendix D for details. Performance of the mode based syntactic tracker and the baseline VSIMM tracker when tracking vehicular traffic are shown in Table III. An example of RMSEs for the two trackers is illustrated in Fig. 11(b). For specific patterns such as \(v_3v_0v_2\) and \(v_3v_0v_3v_8\), the mode based syntactic tracker performs much better than

| pattern   | speed | M       | VSIMM | improvement |
|-----------|-------|---------|-------|-------------|
| \(v_3v_0v_2\) | 2 m/s | average  | 4.0064 | 4.7274    | 13.87%     |
|           |       | peak    | 4.8093 | 7.6005    |            |
| \(v_3v_0v_3v_8\) | 2 m/s | average  | 2.8432 | 3.1155    | 8.32%      |
|           |       | peak    | 4.4128 | 4.7523    |            |
| \(v_3v_0v_1v_4\) | 3 m/s | average  | 2.8012 | 3.0669    | 6.60%      |
|           |       | peak    | 4.3233 | 5.1304    |            |
| \(v_3v_0v_1v_3\) | 3 m/s | average  | 3.3663 | 3.5585    | 4.49%      |
|           |       | peak    | 5.9035 | 8.8663    |            |
| \(v_0v_1v_0v_3\) | 4 m/s | average  | 3.1753 | 3.7875    | 15.51%     |
|           |       | peak    | 3.9942 | 5.6048    |            |
| \(v_0v_1v_0v_3v_8\) | 4 m/s | average  | 3.0721 | 3.1331    | 1.72%      |
|           |       | peak    | 4.7566 | 4.8202    |            |
the VSIMM tracker by over 8%. However, there is little difference between the two competing models for the remaining 4 patterns in Table III. Generally speaking, the mode based syntactic tracker for vehicular traffic does not show as drastic an improvement as for constant speed targets. The explanation is that the VSIMM tracker is able to infer which road the target is moving on after passing an intersection by distinct average speed which is location based [5]. An example is shown in Fig. 12 where a target moves from node \( r_0 \) to \( r_2 \) with those for constant speed targets. \( r_0 \), \( r_1 \) and \( r_2 \) are road names. 10 m/s and 5 m/s are the real time average speeds at some discrete time for \( r_1 \) and \( r_2 \), respectively.

![Figure 12](image)

Figure 12. VSIMM tracker decreases uncertainties on which road the target is moving on by distinct speeds on different roads. \( r_0, r_1 \) and \( r_2 \) are road names. 10 m/s and 5 m/s are the real time average speeds at some discrete time for \( r_1 \) and \( r_2 \), respectively.

An example is that the VSIMM tracker is able to infer which road the target moves on after the intersection, the VSIMM tracker propagates particles on both \( r_1 \) and \( r_2 \). However, particles on \( r_1 \) are assigned (real time average) speed of 10 m/s which is much larger than the (real time average) speed on \( r_2 \) (5 m/s) where the target is truly moving on. This may result in obvious differences on the range rate component in the observation vector in (8). Hence, particles on \( r_1 \) have much lower weights compared with those on \( r_2 \). In this way, (real time average) speeds for different roads give “hints” to the VSIMM tracker on which road the target is moving on after passing an intersection and thus VSIMM tracker can improve the tracking accuracy. In addition, notice that RMSEs for both trackers (when tracking vehicular traffic) decrease faster than those for constant speed targets. The reason is that vehicular traffic have higher speeds. An example is shown in Fig. 13 at time \( k \), particles (red arrows) moving on \( r_1 \) and \( r_2 \) have similar weights. However, particles (green arrow on \( r_1 \)) moving on \( r_1 \) immediately become “far away” from target’s true state vector (right black arrow) after time \( \Delta \) and therefore are assigned lower weights. Smaller \( \Delta \) due to higher speeds of vehicular traffic results in sharp decrease in the RMSEs as shown in Fig. 13.

![Figure 13](image)

Figure 13. RMSEs for the mode based syntactic tracker and the baseline VSIMM tracker when tracking vehicular traffic decrease faster compared with those for constant speed targets. \( r_0, r_1 \) and \( r_2 \) are road names. \( \Delta \) is a positive integer. Black arrows denote target’s true state vectors at time \( k \) (left) and \( k + \Delta \) (right), respectively. Red arrows denote particles at time \( k \) and green arrows denote particles at time \( k + \Delta \).

C. Example 3. Round Trip Model

The round trip model describes a target that departs from some node \( A \), arrives at another node \( B \) and then returns to \( A \); see Sec. V-A. The trajectory is constrained as follows: the target moves to its adjacent nodes at each discrete time; it moves up and right on the forward trip; on the return trip, it moves down and left. We present simulations on a 20×20 square grid roadmap with size of the unit grid as 2 meters (m). Targets are recorded by a GMTI radar with parameters listed in Table I. The GMTI radar is mounted on an aircraft. The aircraft starts from coordinates (-300 m,-300 m) with constant velocity (100 m/s in the \( x \) direction and 20 m/s in the \( y \) direction).

Given noisy GMTI radar measurements, our aim is to compute the most likely state sequence \( x^*_1:n \) defined in (19) and the architecture of the CSCFG Viterbi tracker is illustrated in Fig. 10. We also illustrate a hidden Markov model (HMM) Viterbi tracker for comparison and their performance are shown in Fig. 14 evaluated by averaging error over 40 independent trials. The error for each trial is computed as

\[
\text{error} = \frac{1}{n} \sum_{k=1}^{n} \|x^*_k - x_k^*\|^2 \quad (22)
\]

where \( n \) is the length of observations, \( x_k^* \) is the Viterbi estimate for time \( k \) and \( x_k^* \) is the true state vector.

![Figure 14](image)

Fig. 14 shows that the CSCFG Viterbi tracker has a significantly smaller tracking error (defined in (22)) compared to the HMM Viterbi tracker. The reason is that a CSCFG can model longer and more complicated spatial dependencies than a Markov chain so that all properties of a round trip model discussed in Sec. V-A are captured. Fig. 15 shows an example of state estimates by the CSCFG Viterbi tracker and the HMM Viterbi tracker. We can see for the HMM Viterbi tracker, there are lefts and downs in the forward trip (red) and rights and ups in the return trip (blue) which are impossible for the round trip model discussed in Sec. V-A.

VII. CONCLUSION

In this paper, we constructed a 3-level model for the syntactic tracking problem of targets whose movement is confined to a roadmap by using natural language models. At the first level, the roadmap was modeled as a weighted, directed graph; at the second level, the mode sequence was modeled via a CSCFG; finally the base level kinematics of the target were modeled by a VSIMM. A novel CSCFG-driven particle filtering algorithm was devised to track a target’s kinematics given GMTI measurements. Numerical studies show that compared with the classic VSIMM tracker, the mode based syntactic tracker proposed in this paper can substantially improve the tracking accuracy. We also discussed two CSCFG based models for targets moving on the square grid. A Viterbi algorithm was illustrated to compute the most likely hidden state sequence of the target given noisy GMTI radar measurements. Numerical results based on the round trip model show the CSCFG Viterbi tracker can substantially decrease the tracking error compared with a classical Viterbi tracker.

Acknowledgement.: We gratefully acknowledge Dr. Martie Goulding of MacDonald Dettwiler Associates (MDA) for several useful technical discussions and careful comments on the paper. Parts of this research were funded by MDA via an NSERC CRD Grant.
Figure 14. Performance of the CSCFG Viterbi tracker (red line) and the HMM Viterbi tracker (green line). $\sigma_r$ and $\sigma_\theta$ are defined in (8). (a) illustrates the tracking errors for the two Viterbi trackers under a range of $\sigma_r$ (step=0.5m) and we set $\sigma_\theta = 0.5^\circ$. (b) illustrates the tracking errors for the two Viterbi trackers under a range of $\sigma_\theta$ (step=0.05°) and we set $\sigma_r = 5$m.

REFERENCES

[1] T. Kirubarajan, Y. Bar-Shalom, K. R. Pattipati, and I. Kadar, “Ground target tracking with variable structure imm estimator,” IEEE Transactions on Aerospace and Electronic Systems, vol. 36, pp. 26–46, Jan 2000.
[2] H. A. P. Blom and Y. Bar-Shalom, “The interacting multiple model algorithm for systems with markovian switching coefficients,” IEEE Transactions on Automatic Control, vol. 33, pp. 780–783, Aug 1988.
[3] A. Wang, V. Krishnamurthy, and B. Balaji, “Intent inference and syntactic tracking with GMTI measurements,” IEEE Transactions on Aerospace and Electronic Systems, vol. 47, pp. 2824–2843, Oct 2011.
[4] M. Fanaswala and V. Krishnamurthy, “Detection of anomalous trajectory patterns in target tracking via stochastic context-free grammars and reciprocal process models,” IEEE Journal of Selected Topics in Signal Processing, vol. 7, pp. 76–90, Feb 2013.
[5] M. Fanaswala and V. Krishnamurthy, “Spatiotemporal trajectory models for metalevel target tracking,” IEEE Aerospace and Electronic Systems Magazine, vol. 30, pp. 16–31, Jan 2015.
[6] K. E. Mark, M. I. Miller, and U. Grenander, “Constrained stochastic language models,” in Image Models (and Their Speech Model cousins), pp. 131–140, Springer, 1996.

Figure 15. Simulated performance of state estimates by the CSCFG Viterbi tracker and the HMM Viterbi tracker ($\sigma_r = 5$m and $\sigma_\theta = 0.5^\circ$). The horizontal and the vertical axis denote the $x$ and $y$ coordinate of the target. (a) is the true state sequence of the target. (b) displays the state estimates by the CSCFG Viterbi tracker. (c) displays the state estimates by the HMM Viterbi tracker. Red line is for the forward trip and blue for the return trip.
A parsing algorithm for the CSCFG is shown in Algorithm 3.

Algorithm 3 Parsing Algorithm for the Constrained Stochastic Context Free Grammar (CSCFG)

\[
X, Y, \Gamma \in \mathcal{N}, \lambda, \mu, \beta, \eta \in (\mathcal{N} \cup \mathcal{T})^*, \eta \notin \mathcal{N}, a \in \mathcal{T}. n is a general denotation for an Earley state and \( u_k \) is the set of all Earley states at epoch \( k \). \( \hat{q}_k \) is the hard or soft estimate at time \( k \).

1. Scanning

\[
\text{for } k \rightarrow k' X \rightarrow \lambda a \mu[s, f][\alpha, \gamma] \in u_{k-1} \text{ do}
\]

Add \( k' X \rightarrow \lambda a \mu[s', f'][\alpha', \gamma'] \) to \( u_k \) if \( p(\hat{q}_k|a) > 0 \)

\[
\alpha' = \alpha p(\hat{q}_k|a)
\]

\[
\gamma' = \gamma p(\hat{q}_k|a)
\]

\[
s' = s
\]

\[
f' = a
\]

end for

\[
\zeta_k = \sum_{n \in u_k} \alpha(k_n X \rightarrow \lambda a \mu)
\]

\[
\forall n \in u_k, \text{ normalize } \alpha, \gamma \text{ using } \zeta_k
\]

2. Completion

\[
\text{for } k \rightarrow \eta[s, f][\alpha, \gamma] \in u_k \text{ do}
\]

\[
\text{for } k' \rightarrow k X \rightarrow \lambda Y \mu[s', \eta][\alpha'', \gamma''] \in u_{k'} \text{ do}
\]

if \( R_u(Y, \Gamma|s) \neq 0 \)

Add \( k' X \rightarrow \lambda Y \mu[s', \eta][\alpha', \gamma'] \) to \( u_k \)

\[
\alpha' = \alpha'' R_u(Y, \Gamma|s)
\]

\[
\gamma' = \gamma'' R_u(Y, \Gamma|s)
\]

\[
s' = s''
\]

\[
f' = f
\]

end if

end for

end for

3. Prediction

\[
\text{for } k \rightarrow \lambda Y \mu[s, f][\alpha, \gamma] \in u_k \text{ do}
\]

Add \( k \rightarrow \lambda Y \mu[\eta'] [\alpha', \gamma'] \) to \( u_k \) if \( R_t(Y, \Gamma|f) \neq 0 \)

\[
\alpha' = \alpha R_t(Y, \Gamma|f) p(\Gamma \rightarrow \beta|f)
\]

\[
\gamma' = p(\Gamma \rightarrow \beta|f)
\]

\[
s' = f
\]

\[
f' = f
\]

end for

The one step prediction is computed as

\[
p(\hat{q}_k|q_{k-1}, \hat{q}_{1:k-2}, \mathcal{L}_{\text{CSCFG}}) = \frac{\sum_{n \in u_{k-1}} \alpha(k_{k-1} X \rightarrow \lambda a \mu[s, f])}{\sum_{n \in u_{k-1}} \alpha(k_{k-1} X \rightarrow \lambda a \mu[s, f])}
\]

where \( q_k, a \in \mathcal{T}, X \in \mathcal{N}, \lambda, \mu \in (\mathcal{N} \cup \mathcal{T})^* \). \( \mathcal{L}_{\text{CSCFG}} \) is the language generated by the CSCFG. \( \hat{q}_{1:k-1} \) is a hard or soft partial observation sequence. \( n \) is a general denotation for an Earley state and \( u_{k-1} \) is the set of all Earley states at epoch \( k-1 \). The modified one step predictor is computed as

\[
p(\hat{q}_k|q_{k-1}, \hat{q}_{1:k-2}, \mathcal{L}_{\text{CSCFG}}) = \frac{\sum_{n \in u_{k-1}} \alpha(k_{k-1} X \rightarrow \lambda a \mu[s, f])}{\sum_{n \in u_{k-1}} \alpha(k_{k-1} X \rightarrow \lambda a \mu[s, f])}
\]

The prefix probability is computed as

\[
p(\hat{q}_1|\mathcal{L}_{\text{CSCFG}}) = \sum_{n \in u_k} \alpha(k_1 X \rightarrow \lambda a \mu[s, a])
\]
APPENDIX B

MODEL $\mathcal{L}_{\text{round}}$ AND $\mathcal{L}_{\text{palindrome}}$ VIA A CSCFG AND A VITERBI ALGORITHM FOR THE CSCFG

Here, we give constrained stochastic context free grammar models that generate $\mathcal{L}_{\text{round}}$ and $\mathcal{L}_{\text{palindrome}}$ discussed in Sec. V and introduce a Viterbi algorithm for the CSCFG.

Denote

$$\text{CSCFG}_{\text{round}} = \{N, T, S, R, P\}$$

(28)

the constrained stochastic context free grammar that generates $\mathcal{L}_{\text{round}}$ defined in Def. 1. $N = \{S, X, A, C, D\}$ is a finite set of nonterminals and $S$ is the start symbol. $T = G_{\text{grid}}$ is a finite set of terminals where $G_{\text{grid}}$ is the set of all nodes on the square grid discussed in Sec. V. $R$ is a finite set of rules illustrated in Fig. 16 and $P$ is a probability function on rules in $R$. Note that the production rules in Fig. 16 include a $\epsilon$-type production rule

$$X \rightarrow \epsilon | a, X \in N, a \in T$$

which indicates the nonterminal $X$ cannot be rewritten if the previous terminal is $a$. The derivation stops if a $\epsilon$-type production rule is applied. In other words, CSCFG$_{\text{round}}$ cannot generate terminal strings if $\epsilon$-type production rules are applied in one derivation.

Denote CSCFG$_{\text{palindrome}}$ the constrained stochastic context free grammar that generates $\mathcal{L}_{\text{palindrome}}$ defined in Def. 2. The production rules are illustrated in Fig. 17.

The Viterbi algorithm is an offline algorithm to find the most likely terminal string (denoted by $a_{1:n}$) generated by a CSCFG given a noisy observation sequence (hard or soft) $q_{1:n}$. The aim is to compute

$$a^*_{1:n} = \arg\max_{a_{1:n}} p(a_{1:n}|q_{1:n}, \text{CSCFG})$$

where $\text{L}_{\text{CSCFG}}$ is the language generated by the CSCFG. In the Viterbi algorithm, we extend the Earley state to be

$$k^{-1} X \rightarrow \lambda, \beta|s, f|\gamma||\text{str}$$

(29)

Definitions except $[\text{str}]$ are introduced in [1]. $[\text{str}]$ is a string of clean terminals $x_{k+1} \ldots x_k$ directly or indirectly generated by the Earley state $k^{-1} X \rightarrow \lambda, \beta|s, f|\gamma$. \forall $X, Y \in N (X \neq Y), \forall a \in T$, define

$$R_{\text{max}}^{\text{CS}}(X, Y|a) = \max p(X \rightarrow Z_1|a)p(Z_n \rightarrow Y|a) \prod_{i=1}^{n-1} p(Z_i \rightarrow Z_{i+1}|a)$$

and

$$R_{\text{max}}^{\text{CS}}(X, a) = 1$$

Here, $X, Y, Z_1, Z_2 \ldots Z_n$ are different nonterminals. $R_{\text{max}}^{\text{CS}}$ is computed to avoid completion loops. A Viterbi algorithm for the CSCFG is presented in Algorithm 4.

APPENDIX C

MODEL $\mathcal{L}_{\text{str}}^{CS}$ VIA A CSCFG

Here, we give production rules for the CSCFG that generates $\mathcal{L}_{\text{str}}^{CS}$ ($n = 1, 2, 3, 4$) defined in (11); see Fig. 18.

Algorithm 4 Viterbi Algorithm for the Constrained Stochastic Context Free Grammar (CSCFG)

$X, Y, \Gamma, \lambda, \beta \in (N) \cup T^*$, $\beta \notin N$, $\alpha \in T$. $n$ is a general denotation for an Earley state. $u_k$ is the set of all Earley states at epoch $k$. $\psi_k$ is the set of Earley states generated by the completion operation at epoch $k$. $\hat{q}_k$ is the hard or soft estimate at time $k$. Set $p(\hat{q}_k | \psi) = 0$.

1. Scanning

for $k^{-1} X \rightarrow \lambda, \mu|s, f|\gamma||\text{str} \in u_{k-1}$ do

Add $k^{-1} X \rightarrow \lambda, \mu|s', f'|\gamma'||\text{str}'$ to $u_k$ if $p(\hat{q}_k | \alpha) > 0$

$\gamma' = \gamma p(\hat{q}_k | \alpha)$

$s' = s$

$f' = a$

$\text{str}' = [\text{str}; a]$

end for

for $k^{-1} \Gamma \rightarrow \beta, \gamma|s, f||\text{str} \in u_k$ do

for $k^{-1} \Gamma \rightarrow \lambda, \mu|s', f'|\gamma'||\text{str}' \in u_k$ do

Add $k^{-1} \Gamma \rightarrow \lambda, \mu|s', f'|\gamma'||\text{str}'$ to $\psi_k$ if $R_{\text{max}}^{\text{CS}}(Y, \Gamma|s) > 0$

$\gamma' = \gamma p(\hat{q}_k | \alpha)$

$s' = s''$

$f' = f$

$\text{str}' = [\text{str}''; \text{str}]$

end for

end for

Add $k^{-1} X \rightarrow \lambda, \mu|s', f'|\gamma'||\text{str}'$ to $u_k$

$$x^*_{1:n} = \text{arg\max}_{\gamma \in \psi_k} p(\hat{q}_{1:n} | \psi_k, \text{CSCFG}) = \gamma(0)$$

where

$$\gamma(0) = \text{arg\max}_{\gamma \in \psi_k} p(\hat{q}_{1:n} | \psi_k, \text{CSCFG})$$

and

$$\gamma(0) = \text{arg\max}_{\gamma \in \psi_k} p(\hat{q}_{1:n} | \psi_k, \text{CSCFG})$$

for $x^*_{1:n}$
Figure 18. Production rules in the CSCFG that generates $L^\circ_{r_{0,n}}$ ($n = 1, 2, 3, 4$). $m_0, m_1, \ldots, m_{11}$ are defined in Table V. $n_0$ and $n_1$ are positive integers. In (b), $n_0/n_1 = l(e_{r_{0,1}})/l(e_{r_{1,2}})$ and is irreducible. In (c), $n_0/n_1 = l(e_{r_{0,1}})/l(e_{r_{2,3}})$ and is irreducible. In (d), $n_0/n_1 = l(e_{r_{0,1}})/l(e_{r_{2,3}})$ and is irreducible.

### Appendix D

**Parameters used in the simulations**

Here, we give additional parameters used in our simulations. Lengths and angles of roads on roadmap in Fig. 1 are listed in Table V. Real time speeds of vehicular traffic moving on the roadmap in Fig. 1 are listed in Table VI.
Table VI
REAL TIME AVERAGE SPEEDS (m/s) OF VEHICULAR TRAFFIC ON ROADMAP IN FIG. 1

| Road | 0-100 | 101-200 | 201-300 | 301-400 | 401-500 | 501-600 | 601-700 | 701-800 | 801-900 | 901-1000 |
|------|-------|---------|---------|---------|---------|---------|---------|---------|---------|-----------|
| s_0  | 10    | 14      | 8       | 10      | 5       | 12      | 14      | 10      | 7       | 10        |
| s_1  | 8     | 10      | 12      | 5       | 14      | 7       | 10      | 7       | 13      | 10        |
| s_2  | 10    | 7       | 10      | 14      | 10      | 5       | 13      | 14      | 10      | 10        |
| s_3  | 12    | 7       | 8       | 10      | 5       | 8       | 12      | 10      | 7       | 10        |
| s_4  | 6     | 13      | 8       | 10      | 10      | 12      | 10      | 7       | 10      | 10        |
| s_5  | 8     | 13      | 9       | 9       | 12      | 7       | 10      | 10      | 8       | 10        |
| s_6  | 10    | 13      | 9       | 12      | 7       | 7       | 10      | 8       | 10      | 10        |
| s_7  | 10    | 13      | 13      | 12      | 7       | 7       | 10      | 8       | 10      | 10        |
| s_8  | 8     | 10      | 12      | 7       | 12      | 12      | 7       | 12      | 8       | 12        |
| s_9  | 11    | 10      | 12      | 7       | 12      | 12      | 7       | 12      | 11      | 12        |
| s_{10}| 11    | 12      | 10      | 7       | 10      | 11      | 8       | 10      | 10      | 12        |

Figure 16. Production rules in the CSCFG_{round}. $T$ is defined in [23]. $O_{\text{northeast}}$ and $O_{\text{southwest}}$ are the most northeast and southwest nodes on the square grid, respectively. $\underset{\text{up right}}{\rightarrow}$ and $\underset{\text{down left}}{\rightarrow}$ are defined in [20].

Table IV
SPECIFICATIONS OF $L_{\text{path}}^{\text{alpha}}$

| parameters | direction | location |
|------------|-----------|----------|
| $m_0$      | $\theta(r_0,r_1)$ | $r_0$    |
| $m_1$      | $\theta(r_1,r_2)$ | $r_1$    |
| $m_2$      | $\theta(r_2,r_3)$ | $r_2$    |
| $m_3$      | $\theta(r_3,r_4)$ | $r_3$    |
| $m_4$      | $\theta(r_4,r_5)$ | $r_4$    |
| $m_5$      | $\theta(r_5,r_6)$ | $r_5$    |
| $m_6$      | $\theta(r_6,r_7)$ | $r_6$    |
| $m_7$      | $\theta(r_7,r_8)$ | $r_7$    |
| $m_8$      | $\theta(r_8,r_9)$ | $r_8$    |
| $m_9$      | $\theta(r_9,r_{10})$ | $r_9$   |
| $m_{10}$   | $\theta(r_{10},r_{11})$ | $r_{10}$|
| $m_{11}$   | $\theta(r_{11})$ | $r_{11}$ |

$m_i$, $\forall i = 0, 1, \ldots, 11$ is a vector comprising directions and locations, e.g., $m_0 = \{\theta(r_0,r_1), r_0\}$.

Figure 17. Production rules in the CSCFG_{palindrome}. $\underset{\text{adjacent}}{\rightarrow}$ is defined in [20]. $\Delta_{\text{initial}}$ is the set of initial cost states of the target. $G_{\text{grid}}$ is the set of all nodes in the square grid. $\text{ST}(x_k,x_{k'})$ is the state space of Markov chain that models cost distribution and is dependent on $x_k$ and $x_{k'}$.

Table V
LENGTHS AND ANGLES OF ROADS ON ROADMAP IN FIG. 1

| road name | length/m | angle/radian |
|-----------|----------|--------------|
| s_0       | 100      | $3\pi/4$     |
| s_1       | 100      | $\pi/2$      |
| s_2       | 200      | 0            |
| s_3       | 100      | $\pi$        |
| s_4       | 100      | $\pi/2$      |
| s_5       | 200      | $5\pi/4$     |
| s_6       | 200      | 0            |
| s_7       | 100      | $-\pi/2$     |
| s_8       | 100      | $-\pi/2$     |
| s_9       | 200      | $\pi$        |
| s_{10}    | 100      | $\pi/2$      |