Digital Signal Processing 21 (2011) 648–659

Contents lists available at ScienceDirect

Digital Signal Processing

www.elsevier.com/locate/dsp

Intent inference via syntactic tracking

Vikram Krishnamurthy *, Mustafa H. Fanaswala

Dept. of Electrical and Computer Engineering, Kaiser Building, 2332 Main Mall, The University of British Columbia, Vancouver, BC, Canada V6T 1Z4

ARTICLE INFO

Article history:
Available online 27 May 2011

Keywords:
Intent inference
SCFG-modulated state space
Syntactic tracking
Stochastic context-free grammar
Video surveillance

ABSTRACT

This paper considers the intent inference problem in a video surveillance application involving person tracking. The video surveillance is carried out through a distributed network of cameras with large non-overlapping areas. The target dynamics are formulated using a novel grammar-modulated state space model. We identify and model trajectory shapes of interest using the modeling framework of stochastic context-free grammars. The inference of such grammar models is performed using a Bayesian estimation algorithm called the Earley–Stolcke parser. The intent inference procedure is proposed at a meta-level, which allows the use of conventional trackers at the base level. The use of a suitable fusion scheme shows that intent inference using stochastic context-free grammars in a distributed camera network performs suitably even in the presence of large observation gaps and noisy observations.

1. Introduction

Consider a person being tracked by a network of cameras distributed on the exterior walls of a building. A traditional tracking system would output the position of the person being tracked as "blips" on a map. A human operator is then required to decide whether the trajectory of the person is of a suspicious nature. A trajectory is defined as the path followed by the person. The aim of this paper is to develop useful models for target trajectories and automated algorithms for target intent inference. Intent inference [1–3] can be described as the analysis of the behavior of a target to identify the target's purpose. For example, in person surveillance, the movement patterns of a person being tracked can enable a human operator to infer whether the target is exhibiting any malicious intent. In radar tracking, the loitering behavior of a target boat is a strong indication of smuggling activity. The search terms used by a user in a search engine can also be used to infer the intent of the user. The problem of intent inference, thus, is of fundamental importance in many scenarios.

An automated intent inference system is desirable in surveillance tasks to assist human operators who make decisions regarding target intent based on information from various sensors. In order to develop an automated intent inference system, it is essential to devise appropriate models for target intent. In this paper, we consider intent inference when human targets are tracked by a network of distributed cameras. The key tool used to model interesting trajectories is that of *stochastic context-free grammars* (SCFGs). SCFGs are powerful modeling tools for time-series data that exhibit long-term dependency. In particular, they are useful for trajectory identification because they impart a notion of scale invariance to the trajectories considered. Our approach to tackle the inference problem comprises of a two-level procedure as shown in Fig. 1. The lower level tracking module is called a tracklet estimator whose output provides the input stream for a meta level inference engine. The higher-level inference of the structure or pattern within the observed input is performed by the meta level inference engine. When working in tandem, we refer to this two-level architecture as a *syntactic tracker*. We define a syntactic tracker as a system which tracks a target following a path with a

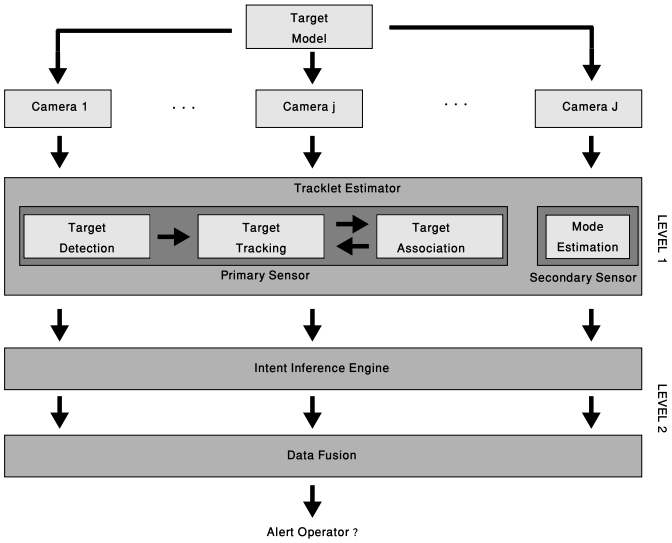

Fig. 1. The architecture of the proposed intent inference system.

* Corresponding author.
E-mail address: vikramk@ece.ubc.ca (V. Krishnamurthy).

1051-2004/$ – see front matter
doi:10.1016/j.dsp.2011.04.005

specific syntax, structure or shape. As a result, a syntactic tracker not only updates conventional position and velocity states but is also able to use the embedded structure within the tracked states for higher-level tasks. Such a hierarchical system allows the use of legacy trackers at the base level while offering the flexibility of different inference engines at the higher level.

1.1. Literature survey

A survey of recent literature in video surveillance shows that intent inference is mainly approached in two ways: (a) anomaly detection [4–6] or (b) model-based intent inference [7–9]. In the former, a specific intent or trajectory is not identified. Rather, all possible actions of a person are categorized as either normal or anomalous. For example, in [5], a support-vector machine approach is taken to classify aberrant trajectories from normal trajectories. The latter approach of model-based intent inference is taken in this paper. It involves specifically identifying trajectories or events of interest.

A key feature of model-based intent inference is to obtain a semantic interpretation of a complex event through the use of simpler sub-events. For example, in [8], a dynamic Bayesian network is used to identify scenarios where a shopper either enters a retail store, leaves a store or passes by the store. Our work can be considered to be related in spirit to the approach taken in [10] and [11]. A context-free grammar approach is taken in [10] to identify two-person interactions like hugs, hand-shakes, kicks and punches. A stochastic context-free grammar approach is taken in [11] to recognize cheating actions in card-games at casinos. Our work departs from them significantly as we consider a tracking situation and not an exclusive action recognition system. The work presented in this paper builds upon the work in [12,13] on target tracking using stochastic context-free grammars in radar tracking applications.

The use of SCFGs as a modeling tool for intent inference requires the ability to compute the model likelihoods. The inside–outside (IO) algorithm [14] is the conventional method used to compute the probability of an observed trajectory belonging to a given model. However, the IO algorithm requires the stochastic grammar to be in a restrictive form. The Earley–Stolcke parser [15] on the other hand is able to deal with arbitrarily structured grammars and is the algorithm used in this paper for the Bayesian estimation of the posterior probabilities.

1.2. Main results

In Section 2, we first provide a brief review of stochastic context-free grammars, which is the key modeling paradigm we will use to model complex trajectories. A detailed formulation of the target dynamics is also given which is based on an SCFG-modulated state space model that is novel to the person tracking application. Finally, we present the SCFG grammar models for various trajectories considered in this paper. The modeling of an approximate rectangular trajectory as an SCFG is another novel contribution made in the paper.

In Section 3, we present the standard form of the Earley–Stolcke parser which is an efficient computational algorithm used to deal with SCFGs. In particular, the Earley–Stolcke parser is used to obtain a Bayesian estimate of the posterior model probabilities (model classification). We also describe how the conventional Earley–Stolcke parser can be extended to deal with noisy tracklet estimates and missing observations. The treatment of missing observations in an SCFG framework is a novel contribution of this paper. Finally, we describe how observations at different sensors can be fused to provide an improved estimate of the target intent. The proof-of-concept in this paper is done numerically in Section 4. We examine two surveillance scenarios and demonstrate the performance of our proposed intent inference system over a distributed network of cameras. The results clearly indicate that while individual sensors can only make myopic decisions, the SCFG representation together with a suitable fusion scheme results in correct inference of the target trajectory.

2. Syntactic models for intent inference

We begin with a brief review of SCFGs in Section 2.1. In Section 2.2 the dynamics of the target are modeled using a switched state-space model that is modulated by a stochastic context-free grammar. The observation equations for a two-tier sensing module called a tracklet estimator are also detailed. Finally, Section 2.3 presents examples of particular geometric patterns of target trajectories like lines, arcs and rectangles.

2.1. Stochastic context-free grammar review

We begin with a short description of SCFGs (see [16] for a textbook treatment). In formal language theory, a grammar G is a four-tuple $\langle \mathcal{N}, \mathcal{T}, \mathcal{P}, S \rangle$. Here $\mathcal{N}$ is a finite set of non-terminals, $\mathcal{T}$ is a finite set of terminals; $\mathcal{N} \cap \mathcal{T} = \emptyset$. $\mathcal{P}$ is a finite set of probabilistic production rules, and $S \in \mathcal{N}$ is the starting symbol. In the sequel, lower-case letters are used to denote terminals, and upper-case letters represent non-terminals. A *symbol* can refer to either a terminal or a non-terminal. A sequence of such symbols is called a *string*. A *sentence* is used to refer to a string of only terminals. A production rule is defined as a symbol-replacement method which specifies the manner in which symbols can be re-written. These terms are relevant in the following discussion.

Definition 2.1 *(Stochastic regular grammar).* Stochastic regular grammars, denoted as G_{RG}, only have production rules of the form $A \to aA$ and $A \to a$ with probabilities $P(A \to aA)$ and $P(A \to a)$ specified, where $A \in \mathcal{N}$. The $\to$ notation means "can be rewritten as". The set of all complete strings (or sentences) generated by a regular grammar is called a regular language and it is denoted as $\mathcal{L}_{RG}$. The restricted form of the production rules for a regular grammar only allow one non-terminal on either side of a production rule. This constrains strings generated by a regular grammar to grow in a linear left–right fashion.

Definition 2.2 *(Stochastic context free grammar).* An SCFG, denoted as G_{CFG}, has production rules, $\mathcal{P}$, of the form $A \to \eta$ with probabilities $P(A \to \eta)$ specified, where $A \in \mathcal{N}$ and $\eta \in (\mathcal{N} \cup \mathcal{T})^+$. $(\mathcal{N} \cup \mathcal{T})^+$ denotes the set of all finite length strings of symbols in $(\mathcal{N} \cup \mathcal{T})$, excluding strings of length 0 (the case where strings of length 0 is included is indicated by $(\mathcal{N} \cup \mathcal{T})^*$). The set of all sentences generated by a particular SCFG is called a context-free language and it is denoted as $\mathcal{L}_{CFG}$.

In the person tracking application considered in this paper, the terminals of the grammar correspond to *modes* exhibited by the target. A target mode $\{a_k\}$, $k = 1, 2, \ldots$, is defined as the direction of acceleration or orientation of the target at any given time instant. The possible modes belong to the set $\Theta = \{0, \frac{\pi}{4}, \frac{\pi}{2}, \frac{3\pi}{4}, \pi, \frac{5\pi}{4}, \frac{3\pi}{2}, \frac{7\pi}{4}\}$, which represent the radial angular direction of acceleration of the target. These modes are shown in Fig. 2. Each mode is associated with a lower-case alphabet in Fig. 2 for labeling purposes. The terminal symbols correspond to modes of the target which are generated from non-terminal symbols. A non-terminal symbol can be thought of as an intermediary symbol which is eventually replaced by a single or a combination of terminal symbols. They can be abstracted as high-level components of the shape of the trajectory that a target might travel.

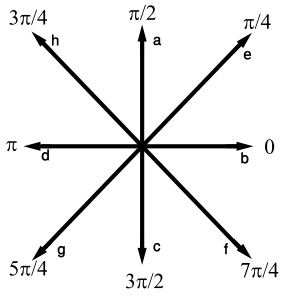

Fig. 2. The possible targets modes (radial angular acceleration directions). Each angle is associated with a lower-case alphabet for labeling purposes.

The trajectory generation process begins with a unique starting non-terminal symbol S. A set of production rules $\mathcal{P}$ with associated probabilities of selection $p : \mathcal{P} \to [0, 1]$ define the manner in which non-terminal symbols produce other non-terminals and terminals. The set of all strings that can thus be generated by a grammar is called a language. Hence, each of the different trajectory shapes considered in this paper (lines, arcs and m-rectangles) is a language with a specific grammar to model them. Let θ denote the set of trajectory shapes of interest, which in our case are, $\theta = \{\text{arc}, \text{line}, \text{rectangle}\}$. The syntactic tracking referred to in this paper decides which model $\theta^m \in \theta$, $m = \{\text{line}, \text{arc}, \text{rectangle}\}$ is able to best explain the observed trajectory. The languages corresponding to the trajectory shapes are denoted as $\mathcal{L}_{\text{arc}}$, $\mathcal{L}_{\text{line}}$, and $\mathcal{L}_{\text{rectangle}}$ respectively, and the precise description of SCFGs that generate them are described in Section 2.3.

2.2. SCFG-modulated state space model

In this section, we present a model for the target dynamics as well as the observation models for the primary and secondary sensors as shown in Fig. 1. At the base level, the tracklet estimator is the term used in this paper to describe the manner in which an input video sequence is processed to obtain estimates of the target modes which represent the directional maneuvers of a person in a particular environment. The tracklet estimator accepts an input sequence of images in the form of a video stream and outputs a sequence of estimated target modes from the set $\Theta = \{0, \frac{\pi}{4}, \frac{\pi}{2}, \frac{3\pi}{4}, \pi, \frac{5\pi}{4}, \frac{3\pi}{2}, \frac{7\pi}{4}\}$. Our usage of the term "mode" and "tracklet" is synonymous. A tracklet is an estimate of the target mode at a particular time instant while the term target mode is used to refer to the "true" value of the quantized acceleration direction.

2.2.1. Target dynamics

The dynamics of the person being tracked are modeled using a switched state-space model which amounts to a hybrid state estimation problem. The state variables considered are the x and y position of the target in world co-ordinates represented by $\mathbf{x}_k(1)$ and $\mathbf{x}_k(2)$ and the x and y components of the target velocity represented by $\mathbf{x}_k(3)$ and $\mathbf{x}_k(4)$. The state vector $\mathbf{x}_k$ is thus given by

$$\mathbf{x}_k = \begin{pmatrix} \mathbf{x}_k(1) \\ \mathbf{x}_k(2) \\ \mathbf{x}_k(3) \\ \mathbf{x}_k(4) \end{pmatrix}. \tag{1}$$

The state equation is given by

$$\mathbf{x}_k = F\mathbf{x}_{k-1} + G\mathbf{w}_{k-1}(a_{k-1}), \tag{2}$$

where $\mathbf{x}_k$ is the state vector, k represents a time index that corresponds to the frame index in the image sequence, F represents the state matrix and G is the noise gain matrix which can be constructed from the equations of motion. The state transition matrix F and the noise gain matrix G are represented by

$$F = \begin{pmatrix} 1 & 0 & T & 0 \\ 0 & 1 & 0 & T \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \qquad G = \begin{pmatrix} \frac{T^2}{2} & 0 \\ 0 & \frac{T^2}{2} \\ T & 0 \\ 0 & T \end{pmatrix}. \tag{3}$$

The sampling time is represented by T which corresponds to the frame rate of the video stream. It is worthwhile to point out that the x and y position co-ordinates are with reference to a world co-ordinate frame and not in the conventional image co-ordinate frame. More details about co-ordinate frames and multi-view geometry in multi-camera systems can be found in [17].

The process noise $\mathbf{w}_k(a_k)$ is assumed to be directionally dependent in nature with a component along the mode a_k (with variance σ_a^2) and another component orthogonal (with variance σ_o^2) to the mode a_k. The covariance of the process noise is given by

$$\mathcal{W} = \rho_{a_k} \begin{pmatrix} \sigma_o^2 & 0 \\ 0 & \sigma_a^2 \end{pmatrix} \rho_{a_k}^T, \quad \text{where } \rho_{a_k} = \begin{pmatrix} -\cos a_k & \sin a_k \\ \sin a_k & \cos a_k \end{pmatrix}.$$

The essence of such a switched state-space model lies in the fact that the dynamics of a maneuvering target cannot be modeled using a single model. Rather, multiple models describe the kinematics of the target when it is in different modes. The target is assumed to maneuver according to a finite set of modes $a_k \in \Theta$, each of which is associated with a possibly different kinematic model. The modes evolve as a stochastic process which is commonly assumed in the literature to be a Markov chain. However, a key difference in the modeling of this paper is that the modes of the target in this paper are considered to arise from a multi-type Galton–Watson branching process which is the underlying theory behind languages generated by SCFGs. The state-space equations are obtained by the corresponding equations of motion for a nearly constant velocity model. Such a model assumes that the target obeys a nearly constant velocity kinematic model in each of its possible modes. A nearly constant velocity model assumes that the change in velocity per unit time is equal to white noise with a small variance. This allows non-zero albeit small accelerations while maintaining a velocity that is "nearly" constant.

2.2.2. Observation model for sensors

As depicted in Fig. 1, the observations come from two channels which we refer to as a two-tier sensing module that is very similar to image-enhanced target tracking in [18]. We refer to the first-tier as the primary sensor whose observation equation is given by

$$\mathbf{z}_k = \mathbf{x}_k + \mathbf{v}_k, \tag{4}$$

where $\mathbf{z}_k$ is a noisy measurement of the target state and $\mathbf{v}_k$ is a zero-mean white Gaussian process. The covariance $\mathcal{M}$ of the noise $\mathbf{v}_k$ is a diagonal matrix with the diagonal elements equal to variances of the target position and velocity measurements in x and y directions. These are represented by $\sigma^2_{\mathbf{x}_k(1)}$, $\sigma^2_{\mathbf{x}_k(2)}$, $\sigma^2_{\mathbf{x}_k(3)}$ and $\sigma^2_{\mathbf{x}_k(4)}$ respectively. We noted above that the target position co-ordinates must be with reference to a world co-ordinate frame and not an image co-ordinate frame. This transformation is required in multi-camera systems to establish a common reference field of view. The transformation from image co-ordinates to world co-ordinates is performed through a perspective transformation that depends on the extrinsic parameters of each camera in the sensor network. These extrinsic parameters can either be obtained through off-line camera calibration techniques [19] or on-line self-calibration methods [20]. The camera calibration can also be used to estimate the intrinsic parameters of the camera which can help correct for

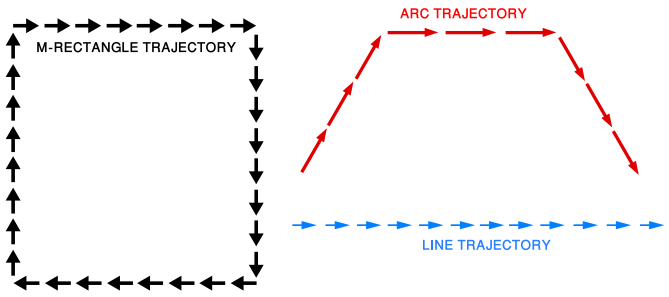

Fig. 3. The different trajectories considered in θ.

various photometric defects and eventually aid in more reliable detections. Using this observation model, a suitable base-level tracker can be designed to estimate the state of the target. We do not consider a particular tracking algorithm in this paper.

The second-tier tracker is responsible for estimating the mode of the target using inputs from the base-level tracker. This step is essential in the intent inference procedure of (5). The output of this secondary sensor is represented by the process $\{n_k\}$ where each $n_k \in \{a, b, c, d, e, f, g, h, x\}$ is an estimate of the actual target mode $a_k \in \{0, \frac{\pi}{4}, \frac{\pi}{2}, \frac{3\pi}{4}, \pi, \frac{5\pi}{4}, \frac{3\pi}{2}, \frac{7\pi}{4}\}$. The symbols $a, \ldots, h$ respectively represent unit acceleration vectors in the 8 radial directions in Θ. We reserve a special symbol x to represent the instance when the secondary sensor has no useful estimate for the target mode. This is reserved for the case of observation gaps in the camera network. So $n_k = x$ implies that the target mode cannot be estimated at time k and $n_k = i$, $i \in \{a, b, c, d, e, f, g, h\}$ implies that the secondary sensor believes that the target is in mode $a_k \in \{0, \frac{\pi}{4}, \frac{\pi}{2}, \frac{3\pi}{4}, \pi, \frac{5\pi}{4}, \frac{3\pi}{2}, \frac{7\pi}{4}\}$. The conditional probability of the sensor emitting any symbol is modeled as a probability mass function that is related to the parameters (conditional probabilities of production rules) of the stochastic grammars defined next in Section 2.3. The mode estimates $n_1, n_2, \ldots, n_k$ form the input to the meta level stochastic parser which infers the intent (trajectory shape) of the person being tracked. The intent inference task involves computing

$$\theta^* = \arg\max_{\theta^m \in \theta} P\big(\theta^m \big| n_1, n_2, \ldots, n_k\big), \tag{5}$$

where θ^* is the inferred intent (most likely grammar model). The details of how to compute the posterior model probability $P(\theta^m | n_1, n_2, \ldots, n_k)$ are discussed in Section 3.1.

2.3. Syntactic pattern modeling

In the following, with the aim of modeling target trajectory shapes, we present SCFG models for various geometric shapes of interest such as arcs and rectangles. A pictorial representation of such trajectories is shown in Fig. 3. Why do we consider these patterns as interesting? Target trajectories resembling such patterns can signify strong intent. For example, [13] describes a military maneuver called a pincer operation which comprises of two opposing arc patterns. An arc can also signify a target doubling back by making a U-turn. Recognizing such maneuvers can provide useful information in battle-field situational awareness. A rectangular trajectory is an intuitive model for targets that are circling back on themselves. Such a trajectory indicates a reconnoitering operation which is of specific interest to surveillance applications. We offer a proof of concept in this paper for inference of such trajectories, keeping in mind that more elaborate trajectories can be built up from such trajectories, thus leading to models for more sophisticated target intent.

$$\begin{aligned} S &\to AXC \\ X &\to AXC \mid ABC \mid B \\ A &\to a \\ B &\to bB \mid b \\ C &\to c \end{aligned}$$

(a)

$$\begin{aligned} S &\to AXCD \\ X &\to AXC \mid ABC \mid B \\ A &\to a \\ B &\to bB \mid b \\ C &\to c \\ D &\to dD \mid d \end{aligned}$$

(b)

Fig. 4. An arc grammar in (a) and an m-rectangle grammar in (b).

2.3.1. Line trajectory

A person following a line pattern creates a linear trajectory with local Markov dependency, and it is characterized by rules of the form $S \to aS|a$ with a representing the target mode. These production rules generate a language that is equivalent to that of a hidden Markov model formulation [21]. Since Markov models have been studied extensively, the line pattern will not be discussed any further. The two other geometric patterns of interest are arcs and rectangles, which possess long range and self-embedding dependencies that require production rules which Markov models cannot represent.

2.3.2. Arc trajectory

The language of an arc can be expressed as a language $\mathcal{L}_{\text{arc}} = \{x \in a^n b^+ c^n\}$, where there is an equal number of matching upward a and downward c tracklets and an arbitrary number of forward tracklets b. The $+$ symbol denotes an arbitrary number of b symbols. The symbol x represents any arbitrary string belonging to the language. The grammar can be constructed based on many techniques and a review of grammar construction methods can be found in [22]. For each a in the string, there must be a matching c, and the corresponding grammar rules are shown in Fig. 4(a). Such a trajectory is called an arc while keeping in mind that the central segment b^+ can be arbitrarily long. If b^+ is much longer than a^n and c^n, then the trajectory resembles a line. However, with a proper choice of the production rule probabilities, we can constrain the middle segment b^+ so that it is not much longer than the side segments of the arc.

As seen in Fig. 4(a), the rules needed to generate patterns such as arcs have a syntax that is more complex than a regular grammar (because there is more than one non-terminal on the right-hand side of the production rule). The rules can be shown to strictly contain regular grammars using the following property: A self-embedding context free grammar cannot be represented by a Markov chain. A context-free grammar is self-embedding if there exists a non-terminal A such that $A \overset{*}{\Rightarrow} \eta A \beta$ with $\eta, \beta \in (V_N \cup V_T)^+$. The relation $\overset{*}{\Rightarrow}$ denotes a transitive and reflexive closure of the production relation $\Rightarrow$. In simpler terms, such a notation is used to signify that even though there might not exist a production rule of the exact form $A \Rightarrow \eta A \beta$, existing production rules can be used on each other to derive such a rule. For the arc production rules presented in Fig. 4(a), the self-embedding property manifests in the second production rule $X \to A\ X\ C$ where X can repeatedly call itself to lengthen the output string.

2.3.3. Rectangular trajectory

We define m-rectangles (modified rectangles) as trajectories comprising of three left turns (or 3 right turns) each of ninety

degrees but are not necessarily closed trajectories (if they were closed, it would coincide with a rectangle). However, we constrain at least two opposite sides of the figure to be of equal length. Why do we consider m-rectangles instead of rectangles? Firstly, the language comprising of only rectangles is not context free. The language comprising of only rectangles can be generated by a more specific class of grammars called context-sensitive grammars. A proof of this can be found in [23]. As a result, algorithms of polynomial complexity for extracting such trajectories cannot be constructed. Secondly, m-rectangles provide a degree of robustness because they do not constrain the starting and end points of the trajectory to coincide.

The m-rectangle language is $\mathcal{L}_{\text{m-rectangle}} = \{a^m b^+ c^m d^+\}$ and it can model any trajectory comprising of four sides at right angles (not necessarily a closed curve) with at least two opposite sides being of equal length. Additionally, a regular grammar cannot be used to model an m-rectangle. We show, in Appendix A, using the pumping lemma for regular grammars that the m-rectangle grammar is not a regular grammar and hence an HMM (or equivalent regular grammar) cannot be used as a generative model for such patterns.

Lemma 1. *$L = \{a^m b^+ c^m d^+ \mid m \geqslant 1\}$ is not a regular language.*

Proof. Shown in Appendix A. □

We note that although the m-rectangle language does not represent rectangles in the conventional sense, the language still enforces the rectangular shape based on the rectangular template constraints. This is because the set of closed rectangular or square shapes is a subset of the m-rectangle language $\mathcal{L}_{\text{m-rectangle}}$. The m-rectangle production rules in Fig. 4(b) decompose a rectangular pattern using a template consisting of more primitive patterns such as four line patterns.

3. Syntactic signal processing algorithms

Until now, we have discussed how the target modes affect the target dynamics and the use of the tracklet estimator to provide us with estimates of the target mode. The Earley–Stolcke parser is described in Section 3.1 which is used to compute the posterior model probabilities in (5). We also describe the extensions to the Earley–Stolcke parser required to implement the syntactic pattern estimator for our specific application. Finally, in Section 3.3, we describe how to combine the information from the distributed cameras to infer the trajectory followed by the target.

3.1. The Earley–Stolcke parser

The Earley–Stolcke parser comprises the top-level of the syntactic tracking architecture shown in Fig. 1. It accepts a string of terminals as input and outputs a most likely production rule sequence that led to the generation of the observed symbols. In a formal language context this is called stochastic parsing, since we are parsing the language corresponding to the trajectory of the target. The Earley–Stolcke parser is a top down parser [15,24], and it builds a parse tree that best describes a sequence of terminals, namely, the modes followed by the target.

For the purposes of intent inference (which is a classification problem), we are mainly interested in evaluating the posterior probability $p(\theta^m | n_1, n_2, \ldots, n_k)$ of an SCFG grammar θ^m amongst $m = 1, \ldots, M$ models generating the estimated mode sequence $n_1, n_2, \ldots, n_k$. Here θ^m represents a grammar model describing a certain trajectory shape. In this paper, we consider lines, arcs and m-rectangles as described in Section 2.3. The Earley–Stolcke parser is able to compute the related likelihood probabilities $p(n_1, n_2, \ldots, n_k | \theta^m)$ in "real-time" by a single left–right pass over the symbol string. The likelihood $p(n_1, n_2, \ldots, n_k | \theta^m)$ is called the sentence probability which is intuitively defined as the probability of the string $n_1, n_2, \ldots, n_k$ being generated by the grammar represented by model θ^m.

In our specific application, we need to deal with noisy observations as well as large observations gaps. The extensions to the standard Earley–Stolcke parser include the following features: (1) model the uncertainties in the estimation of target modes within the grammar and (2) incorporate issues with missing information in observation gaps. The problem of noisy observations or perturbations in the estimated tracklets is incorporated into the grammar modeling by assigning production rules with low probability to the auxiliary directions. Thus each emitting non-terminal has an associated conditional probability mass function over all the possible terminal symbols (modes in the set Θ). This observation probability mass function (pmf) is chosen to be symmetric around the mode it is conditioned upon with an exponential fall-off. An emitting non-terminal produces only one terminal symbol corresponding to the target mode it is attached to in the grammar model. However, to deal with uncertainty in the mode estimation, each such emitting non-terminal has alternative production rules for each target mode chosen according to the observation pmf described above.

Any video surveillance task with a network of cameras also has to account for the possibility of observation gaps where a target cannot be tracked. Moreover, the problem of missed detections is also possible where the target detection algorithm fails to identify a target. In keeping within the SCFG framework, we deal with observation gaps by introducing a special symbol x which is used to indicate the lack of information about the maneuvers of the target in camera blind spots. Such a modification implicitly assumes that the cameras are synchronized in time. The special symbol x is defined as a terminal symbol of the generating grammar. Every emitting non-terminal is assigned an equal probability of emitting this special symbol. This amounts to a uniform distribution over emitting non-terminals. Such a distribution is intuitive because it expresses the fact that we have no information about which terminal symbol was generated. The modification to include the missing symbol x and dealing with noisy tracklets leads to a more robust grammar. Each of the grammar models in Section 2.3 has been made robust in such a manner.

The evaluation of the sentence likelihoods given a certain grammar model $p(n_1, n_2, \ldots, n_k | \theta^m)$ is computed by the Earley–Stolcke parser. The key quantity involved in the parser operation is the inside probability $p(X \to n_i, \ldots, n_k; \theta^m)$, where $X \in \mathcal{N}$ is a non-terminal, $n_i, \ldots, n_k$ are terminal symbols and θ^m is a given grammar model. This quantity can be described as the probability that a non-terminal X produces the observations $n_i, \ldots, n_k$ given the grammar model θ^m. If the non-terminal $X = S$ is the start symbol, then the inside probability $p(S \to n_1, \ldots, n_k; \theta^m)$ is the same as the sentence probability. The Earley–Stolcke parser computes the inside probability recursively using a particular control structure described below.

The control structure used by the parser to store the incomplete parse trees is defined as

$$i : X_k \to \lambda . Y u [\alpha, \gamma].$$

The upper-case letters X and Y are non-terminals, λ and u are substrings of non-terminals and terminals, "." is the marker that specifies the end position for the partially parsed input and that position is indexed by i, k is the starting index of the substring that is generated by the non-terminal X. This control structure is

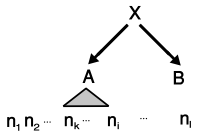

Fig. 5. The pictorial representation of the state $i : X_k \to A.B$, assuming the non-terminal A has a production rule of the form $A \to n_k \dots n_i$.

used by the parser to represent an incomplete parse tree. As the parser receives each subsequent tracklet n_k, it builds on the incomplete parse tree through a sequence of operations defined below. An illustration of an example state $i : X_k \to A.B$ is shown in Fig. 5, where A and B are non-terminals; the indices k and i specify the beginning and the end of the substring respectively which the non-terminal X can "explain" so far, and the index marker "." specifies the part of X's production rule that has been applied to explain the substring. With the marker in front of the non-terminal B, B is not yet applied, and the state is still incomplete. α is called the forward probability and it is the sum of probabilities of all incomplete parse trees yielding the terminals $n_1, \dots, n_i$. γ is called the inner probability and it is the sum of probabilities of all incomplete parse trees yielding the terminals $n_k, \dots, n_i$.

An illustration of the operation of both non-probabilistic and probabilistic versions of the Earley–Stolcke parser is provided in [15]. To initialize the parsing process, the dummy state $0 : {}_0 \to .S\ [1, 1]$ is inserted, where the forward and inside probabilities are initialized to 1. The dummy state simply says that at the index position 0, the start symbol is applicable to parse the input string. With the dummy state in place, the parser builds the parse tree by iteratively going through three operations: *prediction, scanning, and completion.*

The operations are applied sequentially, and each operation works on the set of states produced by the previous operation. Given a set of states (or just the initial dummy state at index 0), the prediction operation searches for states whose index marker has a non-terminal to the right of it, and those non-terminals, with their production rules, are used to generate a set of predicted states. From the predicted states, the scanning operator looks if there are states whose index marker has a terminal to the right of it. For those states whose terminal matches the input string at the indexed position, their index marker is advanced by one position and produced as a scanned state. Lastly, from the set of scanned states, the completion operation looks for states whose index marker is at the end of its production rule. If any are found, the states that generated those scanned states will have their index advanced by one position. After processing all the input symbols, the parser declares a successful parse by verifying whether the state $l : {}_0 \to S.\ [\alpha_l, \gamma_l]$ exists in the final state set. The total length of the string is denoted by l. The forward and inner probability of this final state are equal and represent the sentence probability, which is the likelihood $p(n_{1:k}|\theta^m)$. The details of the three operations mentioned are discussed in turn below.

3.1.1. Prediction

The prediction operator adds states that are applicable to explain the unparsed input string. For all states of the form

$$i : X_k \to \lambda.Yu\ [\alpha, \gamma],$$

where λ and u may be empty, Y is the non-terminal that could possibly generate the next terminal in the input string, the operator adds Y's production rule

$$i : Y_i \to .v\ \left[\alpha', \gamma'\right],$$

as a predicted state. The α' and γ' are updated according to

$$\alpha' = \sum_{\lambda, u} \alpha(i : X_k \to \lambda.Zu)\mathcal{R}_L(Z, Y)P(Y \to v)$$

and

$$\gamma' = P(Y \to v),$$

where $\mathcal{R}_L$ is a reflective transitive closure of a left corner relation and it computes the probability of indefinite left recursion in the productions (details of the relation are provided in Section 3.2 and further explanation can be found in [15]). A pruning capability can also be embedded in the parser by discarding the predicted states if its forward probability is lower than a threshold. This is especially useful to speed up parsing.

3.1.2. Scanning

The scanning operator matches the terminal in the input string to the states generated from the prediction operator. For all states of the form

$$i : X_k \to \lambda.au\ [\alpha, \gamma],$$

where λ and u can be empty, the state

$$i + 1 : X_k \to \lambda a.u\ \left[\alpha', \gamma'\right]$$

is added if the terminal at the index $i + 1$ of the input string is a. The α' and γ' are updated according to

$$\alpha' = \alpha(i : X_k \to \lambda.au)P(a)$$

and

$$\gamma' = \gamma(i : X_k \to \lambda.au)P(a),$$

where $P(a)$ is the probability of the input. It is noted that by including $P(a)$ in updating α and γ, the parsing process also takes the input uncertainty in account.

3.1.3. Completion

The completion operator advances the marker position of the pending predicted states if their derived states match the input string completely. The scanned states whose marker is at the end of their rule have the form

$$i : Y_j \to v.\left[\alpha'', \gamma''\right]$$

and each of them has a corresponding pending predicted state with the form

$$j : X_k \to \lambda.Yu\ [\alpha, \gamma].$$

The purpose of the completion operator is to find those pending prediction states and advance their marker. It is important to notice the relationship of the indices in the scanned state and the pending prediction state. The indices of the pending prediction state imply that the non-terminal Y is applied at the position j, and the indices of the scanned state imply that the non-terminal Y matches the substring from the index j to i. As a result, the two states generate the complete state

$$i : X_k \to \lambda Y.u\ \left[\alpha', \gamma'\right],$$

which means that the pending prediction state can now explain the substring from the index k to i. The associated α and γ probabilities are updated according to

$$\alpha' = \sum_{v} \alpha(i : X_k \to \lambda.Zu)\mathcal{R}_U(Z, Y)\gamma''(i : Y_j \to v.)$$

and

$$\gamma' = \sum_{v} \gamma(i : X_k \to \lambda.Yu)\mathcal{R}_U(Z, Y)\gamma''(i : Y_j \to v.)$$

respectively, where $\mathcal{R}_U$ is a reflective transitive closure of a unit production relation and it computes the probability of an infinite summation due to cyclic completions (explained in Section 3.2 in more detail).

3.2. Implementation issues

This section deals with algorithmic issues that arise due to the nature of certain production rules in an SCFG. The arrays $\mathcal{R}_L$ and $\mathcal{R}_U$ that appear in the prediction and completion steps of Section 3.1 are defined below. The following discussion is necessary for the sake of reproducibility because the grammars we have defined in Section 2.3 contain production rules that can cause the Earley–Stolcke parser to undergo repeated prediction steps and cycling of completion steps that may result in an infinite loop. However, these situations can be fortunately dealt with in an efficient manner and we follow the treatment in [11,15]. Consider the grammar

$$S \to Sa|a,$$

where the | symbol represents an OR relation. The Earley–Stolcke parser will predict $S \to .Sa$ and $A \to .a$ at the first prediction step. This will make the parser consider these newly predicted states for further possible descent. As a result, the same states are produced again. The repeated prediction of the states results in an infinite summation for the parse probability. Such an infinite summation can be accounted for by computing a total recursive contribution for each left-corner relation. Two non-terminals X, Y are said to be in a left-corner relation $X \to_L Y$ if there exists a production for X that has Y on its RHS, such that $X \to Y\lambda$. We can thus define P_L which is the matrix of probabilities $P(X \to_L Y)$ defined as the total probability of choosing a production rule for X that has Y as a left-corner,

$$P(X \to_L Y) = \sum_{X \to Y\lambda} P(X \to Y\lambda).$$

Concurrently, we can also define a reflexive, transitive closure of the left-corner relation $X \to Y\lambda$. The closure is represented as the relation $X \to_L^* Y\lambda$ iff $X = Y$ or if $X \to Z\lambda$ and $Z \to_L^* Y\lambda$. We can now similarly define $\mathcal{R}_L$ as the matrix of probability sums $R(X \to_L^* Y)$. Each $R(X \to_L^* Y)$ is defined as a recurrence relation

$$R\big(X \to_L^* Y\big) = \delta(X, Y) + \sum_{Z} P(X \to_L Z)R\big(Z \to_L^* Y\big),$$

where $\delta(X, Y) = 1$ if $X = Y$ and $\delta(X, Y) = 0$ if $X \neq Y$. This can be conveniently written in matrix notation as $\mathcal{R}_L = I + P_L\mathcal{R}_L$, where I is an identity matrix of the same size as the number of non-terminals in the grammar. The closed form solution is thus given by $\mathcal{R}_L = (I - P_L)^{-1}$. A proof for the existence of $\mathcal{R}_L$ is given in [15]. The significance of the matrix $\mathcal{R}_L$ is that its elements are the sums of the probabilities of the potentially infinitely many prediction paths leading from a state $X_k \to \lambda.Zu$ to a predicted state $Y_i \to .v$, via any number of intermediate states. $\mathcal{R}_L$ can be computed once for each grammar and then stored for subsequent use in all prediction steps.

A similar problem of cyclical completion (infinite loop) arises due to unit production rules. Two non-terminals X, Y are said to be in a unit relation $X \to Y$ if there exists a production for X that has only Y on its RHS, such that $X \to Y$. Just as in the left-corner relation case, we define the matrix P_U as the matrix of probabilities $P(X \to Y)$. The reflexive and transitive closure of the unit relation leads to the relation $X \to^* Y$ iff $X = Y$ or if $X \to Z$ and $Z \to^* Y$. We can now similarly define $\mathcal{R}_U$ as the matrix of probability sums $R(X \to^* Y)$. Each $R(X \to^* Y)$ is defined as a recurrence relation

$$R\big(X \to^* Y\big) = \delta(X, Y) + \sum_{Z} P(X \to Z)R\big(Z \to^* Y\big).$$

This can be further written in matrix notation and solved to obtained the closed form solution $\mathcal{R}_U = (I - P_U)^{-1}$ which is then used in every completion step.

3.3. Data fusion in a multi-camera network

This section describes how to use the observations made by different sensors to make a final decision regarding the multiple hypotheses (different intents) being tested. A single camera in the network is only able to make a myopic inference on the intent of a person being tracked. For example, we know that a rectangular trajectory as seen in Fig. 3 consists of 4 linear segments. A camera observing only one of the linear segments of a persons trajectory is more likely to classify the trajectory as being a line. However, when the different parts of the trajectory as observed by different sensors is jointly considered, a better inference can be made.

In this paper, we take the following approach to data fusion. Each sensor transmits the posterior probability of each SCFG model given the observed string at that sensor node to a central fusion node. The central fusion node combines the posterior probabilities received from each sensor node to obtain a fused density which is used to determine the intent of the target being tracked. This is done by assigning the intent to the model with the highest posterior probability. The combination formula (also called a consensus rule) $C_F : [0, 1]^J \to [0, 1]$ is a mapping from the posterior probabilities computed by all the J camera sensors in the network to a fused posterior density obtained at the central node. Consider the case of J camera sensors each computing the posterior probability of a given SCFG model among M possible models provided with the observation string $n_1, \ldots, n_k$. This is the exact scenario depicted in Fig. 1. The posterior probability of the m-th model computed by the j-th sensor is given by $p_j(\theta_m|n_1, \ldots, n_k)$. With this setup, we examine two consensus rules viz., the linear opinion pool and the logarithmic opinion pool [25].

Linear opinion pool is a consensus rule which takes the form

$$C_F(p_1, p_2, \ldots, p_J) = \sum_{i=1}^{J} \alpha_i p_i, \tag{6}$$

where $\sum_i \alpha_i = 1$. The linear opinion pool is simple, it yields a probability distribution and the weights α_i can be used to impart a notion of sensor reliability to the individual sensors in the camera network. However, the linear opinion pool is not a Bayesian decision maker because the consensus rule is not derived from the joint probability using Bayes rules.

Logarithmic opinion pool is an alternative to the linear opinion pool and the consensus rule takes the form

$$C_F(p_1, p_2, \ldots, p_J) = \frac{\prod_{i=1}^{J} p_i^{\alpha_i}}{\int \prod_{i=1}^{J} p_i^{\alpha_i}}, \tag{7}$$

where it is usually assumed that $\sum_i \alpha_i = 1$. The logarithmic opinion pool treats the posterior probabilities coming from each sensor independently and results in a Bayesian decision maker. The main problem with this method is to decide how to assign the weights α_i.

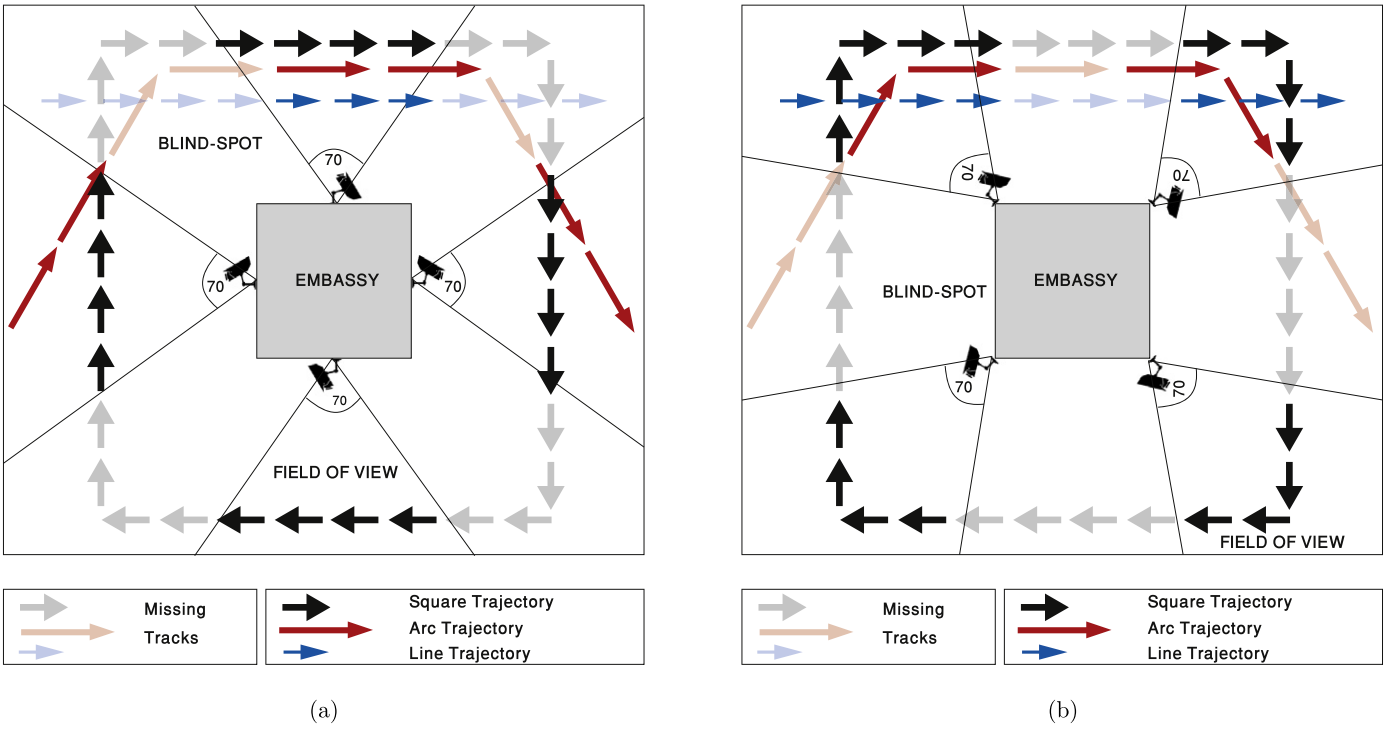

Fig. 6. An external building surveillance example. The camera placement is at the center of the building walls in (a) while the cameras are placed at the corners in (b).

4. Numerical examples of syntactic tracking

We present two examples of syntactic tracking using a distributed camera network. The first scenario is that of surveillance of the exterior of a building while the second scenario is that of surveillance inside the corridors of a building. Different people can traverse the same path with different speeds or different cameras may have different frame rates. These may lead to trajectories of different scale. However, the recursive self-embedding property of an SCFG imparts scale-invariance to the intent inference task. We highlight this as a key feature of the approach presented and demonstrate these concepts numerically in the following simulations.

4.1. An external building scenario

The first scenario proposed is that of an external building surveillance scenario. We initially simulate the case where the mode estimates are noiseless and then examine the more interesting noisy case. Consider the surveillance of a security-sensitive building (like an embassy) depicted in Fig. 6(a). An alternative placement of the cameras for the same scenario is shown in Fig. 6(b). The surveillance is carried out using a network of cameras on the exterior walls of the building. A typical CCTV camera has an angle of view of 70° which heavily restricts the observable field of view. In this simple scenario, we assume that the cameras do not have zoom, pan or tilt capabilities. With four cameras, each covering one side of the building, wide unobservable areas called blind spots exist where targets cannot be reliably tracked. The scenario shown in Fig. 6 depicts some different trajectories that a target can exhibit. The usefulness of the scale-invariance or self-embedding property of the SCFG model is apparent in the fact that the target can maneuver at different distances away from the building thus resulting in various scaled versions of the shapes being considered. The SCFG model can account for this scaling implicitly.

As explained in Section 3.1, the problem of observation gaps is tackled by assuming that the cameras are synchronized and the maneuvers of the target in unobservable areas are represented by a special symbol x which has a uniform distribution over all the possible emitting non-terminals.

4.1.1. Noiseless target mode estimates

We first consider noiseless syntactic tracking in the presence of large observation gaps and non-overlapping views of the cameras. The target is maneuvering around the building in a square trajectory and large parts of the trajectory are occluded from each sensors view. The actual intended trajectory is codified by the directional tracklet sequence $\{aabbccdd\}$. However, each of the sensors only observe a linear portion of the trajectory. For example, sensor 1 only observes the sequence $\{aaxxxxxx\}$ while sensor 3 observes the sequence $\{xxxxccxx\}$. The posterior probabilities computed at each of the sensors are shown in Figs. 7(a)–(d). Finally, the data fusion approach of a linear pool in (6) and a logarithmic pool in (7) as outlined in Section 3.3 is used to obtain a combined posterior distribution over each model in Figs. 7(e)–(f). The pooling is carried out assuming that each sensor is equally reliable and hence equal weights are assigned in the consensus rule. However, in scenarios where certain sensors are known to be more reliable or when certain sensors have larger fields of view, the weights can be appropriately chosen to reflect this reliability. Due to the larger computational times required for the parsing, we choose to work with small strings that aptly provide proof-of-concept.

We observe in Figs. 7(a)–(d) that the individual sensors cannot discern the intent of the trajectory because of the large observation gaps. Each individual sensor only sees a line and hence each individual sensor classifies the trajectory as a line in a specified direction. A peculiarity is observed for sensor 4 which has a higher posterior probability for the m-rectangle as compared to a line. This is due to the fact that a smaller m-rectangle turns out to have a higher likelihood than a long line. Nonetheless, in Figs. 7(e)–(f), we immediately notice that the initial classification is that of a line, however, as soon as the target appears to turn, the probability

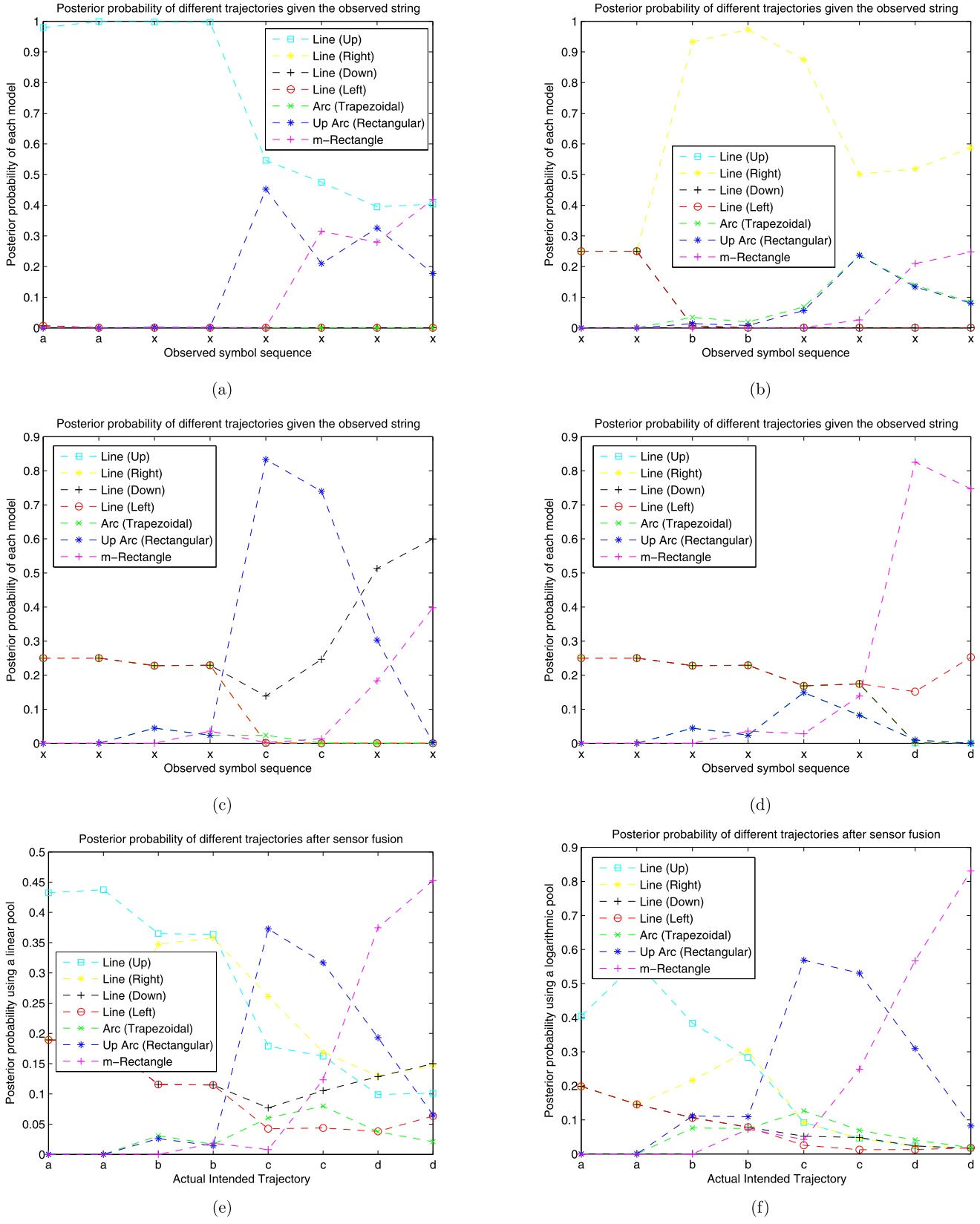

Fig. 7. The noiseless square trajectory in scenario 1. The posterior probabilities at each of the four sensors are shown in (a), (b), (c) and (d). The linearly pooled probabilities are shown in (e) and the logarithmically pooled probabilities are shown in (f). The parameters are specified in Sections 2.2, 2.3.

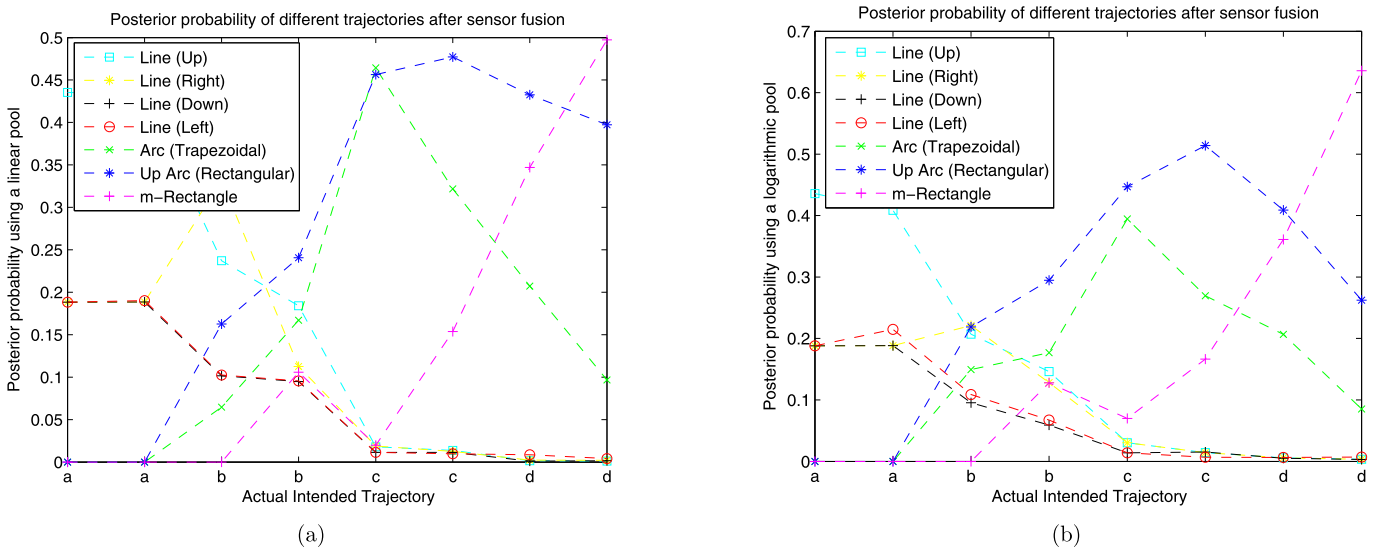

Fig. 8. The noisy square trajectory in scenario 1. The linearly pooled probabilities are shown in (a) and the logarithmically pooled probabilities are shown in (b). The parameters are specified in Sections 2.2, 2.3.

of the trajectory being a line decreases and the other trajectories take precedence. At the final instance, the m-rectangle receives the largest posterior probability which results in a correct classification of the target trajectory. Both the linear and logarithmically pooled probabilities appear similar and there is no significant difference worth commenting upon.

4.1.2. Noisy target mode estimates

Next, a noisy version of the same problem is simulated for the square trajectory by considering noisy estimates of the tracklets. For example, sensor 1 observes the string $\{ahxxxxxx\}$ and sensor 3 observes the string $\{xxxxfcxx\}$ while the intended actual trajectory is $\{aabbccdd\}$. The second symbol in the sensor 1 measurement is perturbed from a to h (this implies a perturbation of $\frac{\pi}{4}$ radians). A similar perturbation occurs in the other sensors. The posterior probabilities at each of the sensors are not useful for inferring the trajectory due to the large observation gaps. However, from the fused probabilities in Figs. 8(a)–(b), it can be observed that while the linearly pooled probabilities indicate the m-rectangle as having the largest posterior probability, the rectangular arc also receives a high score. Moreover, these posterior probabilities are below 0.5 which does not imply a high confidence of discernibility. On the other hand, the logarithmically pooled probabilities fare better by clearly indicating the m-rectangle as the model with the largest posterior probability. Due to the noisy estimates of the tracklets, we also notice that the trapezoidal arc also has a significant posterior probability in contrast with the noiseless case.

4.2. An internal hallway scenario

The second scenario that we consider is shown in Fig. 9(a). This scenario emulates that of surveillance in the interior of a building within its hallways. An example application could be monitoring of school or office hallways for suspicious behavior. Because of the narrow hallways, the field of view of cameras used indoors have to be larger. The usefulness of the scale-invariance property of SCFGs becomes clear when we consider that a target can traverse shapes of different sizes if different corridors are used. A larger square trajectory is shown in Fig. 9(b). Alternatively, different people could traverse the hallways at different speeds thus leading to the self-embedding of the directional tracklets. This occurs because the tracklets are simply unit directional vectors without any notion of magnitude. In both cases, the underlying SCFG modeling is able to

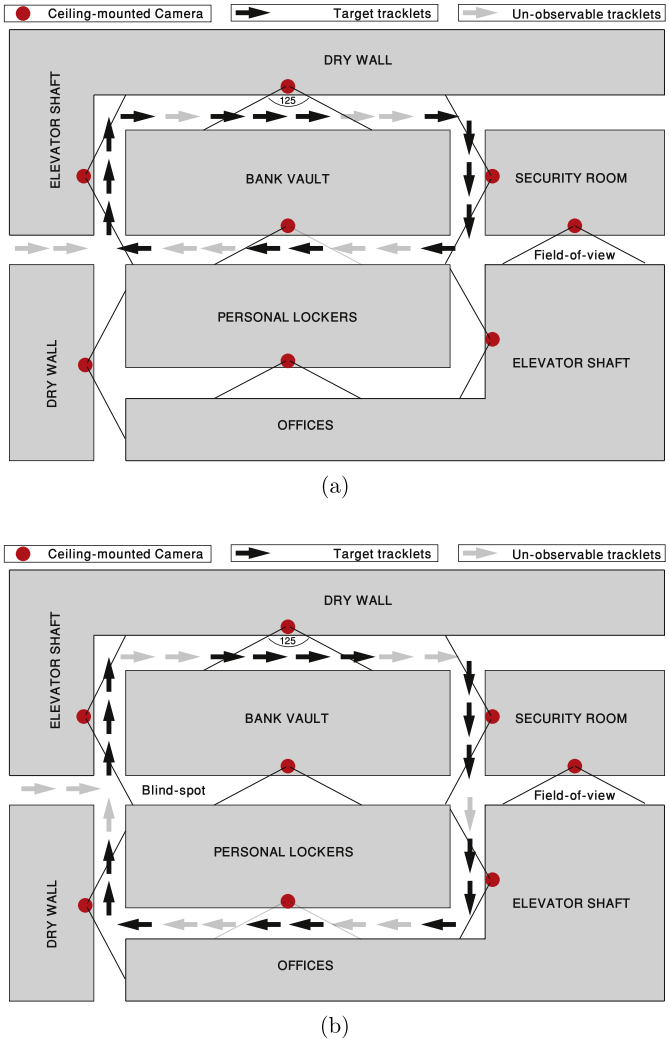

Fig. 9. An internal hallway surveillance example. A rectangular trajectory is shown in (a) and a square trajectory is shown in (b) which are both cases of the m-rectangle language. This example shows the scale invariance of the model due to different sized m-rectangles.

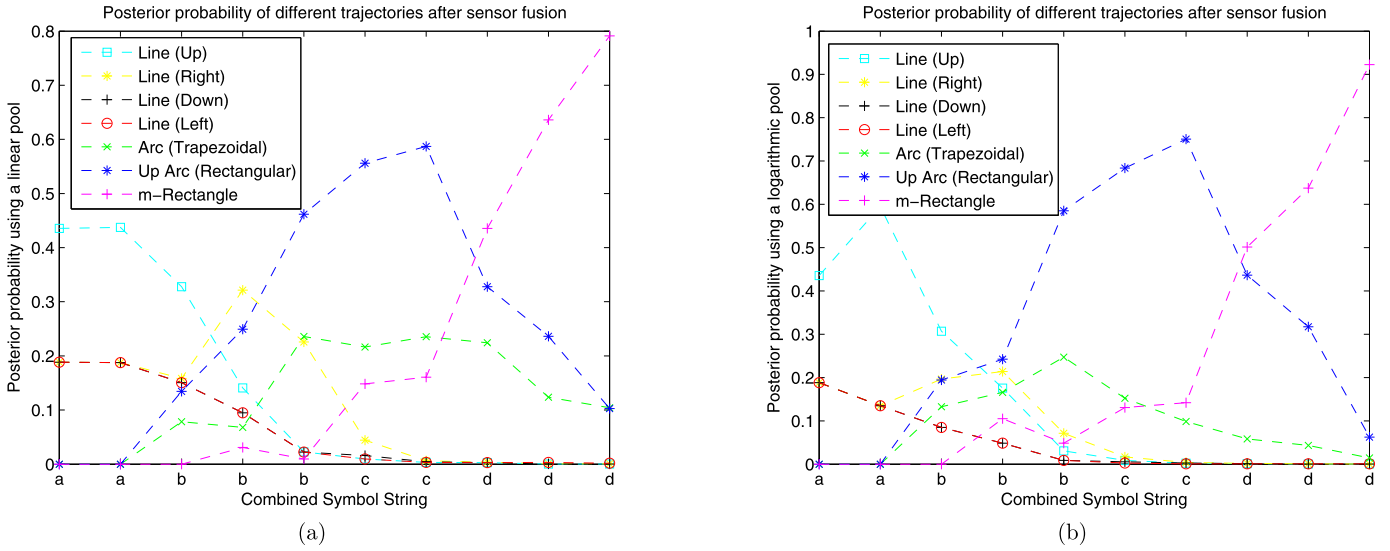

Fig. 10. The noiseless rectangular trajectory in scenario 2. The linearly pooled probabilities are shown in (a) and the logarithmically pooled probabilities are shown in (b). The parameters are specified in Sections 2.2, 2.3.

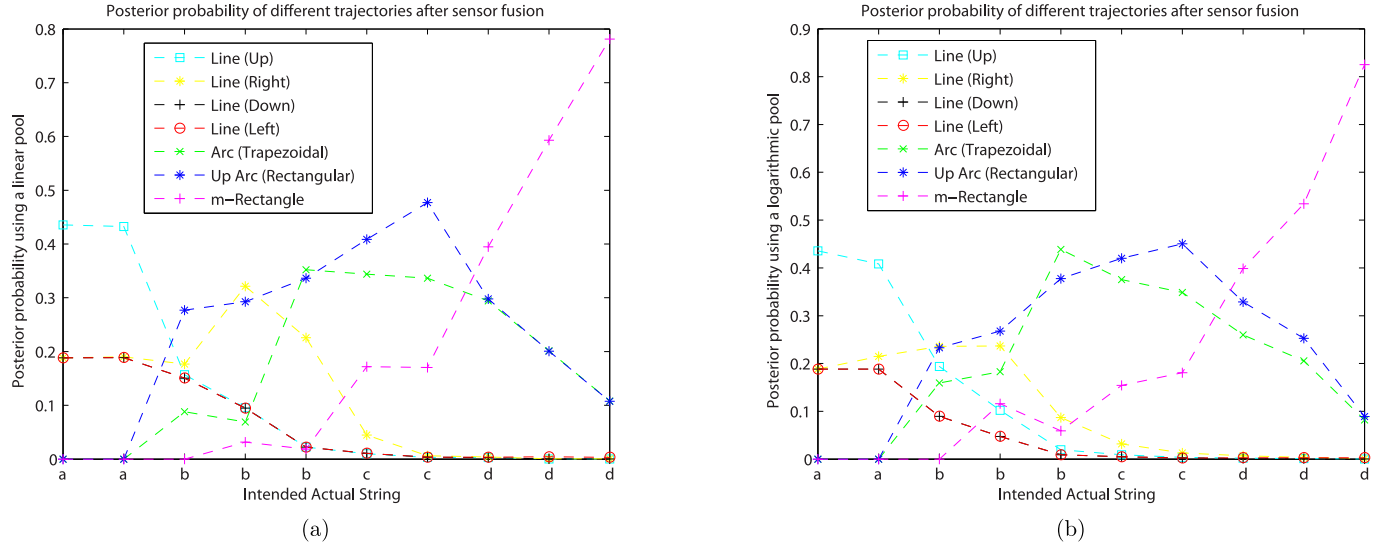

Fig. 11. The noisy rectangular trajectory in scenario 2. The linearly pooled probabilities are shown in (a) and the logarithmically pooled probabilities are shown in (b). The parameters are specified in Sections 2.2, 2.3.

recognize the trajectory without ad-hoc modifications to account for scale.

4.2.1. Noiseless target mode estimates

The noiseless case of an m-rectangle trajectory is considered in Fig. 10. The intended trajectory is given by the tracklet sequence {*aabbbccddd*}. This scenario is different from the earlier external building scenario because of the nature of the camera placement in indoor scenarios. Some of the cameras do not merely measure linear sections of the trajectory. For example, the string observed by sensor 1 is {*aabxxxxxxd*} which includes turn information as well as the tracklet at the end of the trajectory. Intuitively, we would expect a better intent inference which is apparent in the posterior probabilities in Fig. 10.

4.2.2. Noisy target mode estimates

Finally, a noisy version of the rectangular trajectory is simulated and the posterior probabilities are shown in Fig. 11. The intended trajectory is the same, viz., {*aabbbccddd*}. However, as before, random perturbations occur due to the noisy estimates provided by the tracklet estimator. For example, the tracklets observed at sensor 3 is {*xxxxxggcxx*}, where the sixth and seventh symbols have been perturbed from *c*'s to *g*'s (which corresponds to an error of $\frac{\pi}{4}$ radians). The resulting fused posterior probabilities are shown in Fig. 11. The fused probabilities in Figs. 11(a)–(b) are able to incorporate information from all the sensors and correctly classify the trajectory as an m-rectangle even in the presence of noise and large observation gaps.

5. Conclusions

The main highlight of this paper is the modeling of complex target trajectories representing a certain intent by the use of stochastic context-free grammars. SCFGs prove to be a useful modeling tool because of the scale invariance that they impart to trajectory identification. This is due to the fact that the same grammar can be used to model scaled as well as noisy versions of a specific trajectory. We also presented the Earley–Stolcke parser to perform Bayesian estimation of the posterior model probabilities. The extensions described in Section 3.1 allow us to extend

the Earley–Stolcke parser to deal with missing and noisy observations. The SCFG-modulated state space presented in Section 2.2 is a novel contribution of the paper to the person tracking application. Our proposed two-level procedure of tracking and inference is also advantageous because it allows the use of legacy trackers while building on the intent inference engine. Finally, numerical simulations were carried out in Section 4 that suitably demonstrate proof-of-concept. We showed that by fusing the posterior probabilities from individual sensors, an improved inference of the target trajectory can be made even in the presence of large observations gaps and noisy target mode estimates. We believe that the syntactic tracking conceptualized in this paper could be of considerable value to surveillance applications.

Appendix A. Proof that $\mathcal{L}_{\text{m-rectangle}}$ is not a regular grammar

A.1. Pumping lemma for regular grammars

Let L be a regular language. Then there exists a constant K such that if z is any string in L such that $|z|$ is at least K, then we write $z = vwx$, subject to the following conditions:

1. $|vw| \leqslant K$. This means that the substring to be pumped or the loop must occur within the first K symbols.
2. $w \geqslant 1$. Since w and x is the substring to be "pumped", this condition says that it cannot be empty.
3. For all $i \geqslant 0$, $vw^i x$ is in L. That is, the substring w may be "pumped" any number of times, including 0, and the resulting string will still be a member of L.

Lemma. *$L = \{a^m b^+ c^m d^+ | m \geqslant 1\}$ is not a regular language.*

Proof. Suppose L is a regular language. Let $z = a^K b^L c^K d^M$. The first condition dictates that $|vw| \leqslant K$ which implies that vw can only contain a's. If w contains even one b then $|vw| = K + 1$ (which would violate condition 1 of the pumping lemma), so consequently w must also contain only a's and at least one a (according to condition 2 of the pumping lemma). If we now pump w which contains at least one a, then the resulting string $z = vwx$ will contain more a's than c's. By contradiction, we can conclude that L is not regular. □

References

[1] P.H. Foo, G.W. Ng, K.H. Ng, R. Yang, Application of intent inference for surveillance and conformance monitoring to aid human cognition, in: 10th International Conference on Information Fusion '07, 2007, pp. 1–8.

[2] D. Polling, M. Mulder, M. van Paassen, Q. Chu, Inferring the driver's lane change intention using context-based dynamic Bayesian networks, in: IEEE International Conference on Systems, Man and Cybernetics '05, vol. 1, 2005, pp. 853–858.

[3] I. Hwang, C.E. Seah, Intent-based probabilistic conflict detection for the next generation air transportation system, Proceedings of the IEEE 96 (2008) 2040–2059.

[4] H. Sheng, C. Li, Q. Wei, Z. Xiong, Real-time detection of abnormal vehicle events with multi-feature over highway surveillance video, in: 11th International IEEE Conference on Intelligent Transportation Systems, ITSC 2008, pp. 550–556.

[5] C. Piciarelli, C. Micheloni, G. Foresti, Trajectory-based anomalous event detection, IEEE Transactions on Circuits and Systems for Video Technology 18 (2008) 1544–1554.

[6] C. Cohen, F. Morelli, K. Scott, A surveillance system for the recognition of intent within individuals and crowds, in: IEEE Conference on Technologies for Homeland Security, 2008, pp. 559–565.

[7] S. Hongeng, F. Bremond, R. Nevatia, Bayesian framework for video surveillance application, in: Proceedings 15th International Conference on Pattern Recognition, vol. 1, 2000, pp. 164–170.

[8] J. Muncaster, Y. Ma, Activity recognition using dynamic Bayesian networks with automatic state selection, in: IEEE Workshop on Motion and Video Computing, WMVC '07, 2007, pp. 30–34.

[9] V. Vu, F. Bremond, M. Thonnat, Automatic video interpretation: A recognition algorithm for temporal scenarios based on pre-compiled scenario models, in: International Conference on Computer Vision Systems, ICVS '03, 2003, pp. 523–533.

[10] M.S. Ryoo, J.K. Aggarwal, Recognition of composite human activities through context-free grammar based representation, in: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR '06), IEEE Computer Society, Washington, DC, USA, 2006, pp. 1709–1718.

[11] A. Bobick, Y. Ivanov, Action recognition using probabilistic parsing, in: IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 1998, pp. 196–202.

[12] A. Wang, V. Krishnamurthy, Signal interpretation of multifunction radars: Modeling and statistical signal processing with stochastic context free grammar, IEEE Transactions on Signal Processing 56 (2008) 1106–1119.

[13] V. Krishnamurthy, A. Wang, B. Balaji, Syntactic target tracking for GMTI systems, IEEE Transactions on Aerospace and Electronic Systems, in press.

[14] K. Lari, S.J. Young, The estimation of stochastic context-free grammars using the inside–outside algorithm, Computer Speech and Language 4 (1990) 35–56.

[15] A. Stolcke, An efficient probabilistic context-free parsing algorithm that computes prefix probabilities, Computational Linguistics 21 (1995) 165–201.

[16] K.S. Fu, Syntatic Pattern Recognition and Applications, Prentice–Hall, Inc., Englewood Cliffs, NJ, USA, 1982.

[17] R.I. Hartley, A. Zisserman, Multiple View Geometry in Computer Vision, 2nd edition, Cambridge University Press, 2004.

[18] J.S. Evans, R.J. Evans, Image-enhanced multiple model tracking, Automatica 35 (1999) 1769–1786.

[19] Z. Zhan, A flexible new technique for camera calibration, IEEE Transactions on Pattern Analysis and Machine Intelligence 22 (2000) 1330–1334.

[20] T. Svoboda, D. Martinec, T. Pajdla, A convenient multi-camera self-calibration for virtual environments, PRESENCE: Teleoperators and Virtual Environments 14 (2005) 407–422.

[21] P. Dupont, F. Denis, Y. Esposito, Links between probabilistic automata and hidden Markov models: Probability distributions, learning models and induction algorithms, Pattern Recognition 38 (2005) 1349–1371.

[22] D. Jurafsky, J.H. Martin, Speech and Language Processing: An Introduction to Natural Language Processing, Prentice–Hall, Inc., Englewood Cliffs, NJ, 2009.

[23] A.V. Aho, J.D. Ullman, The Theory of Parsing, Translation, and Compiling, Prentice–Hall, Inc., Upper Saddle River, NJ, USA, 1972.

[24] J. Earley, An efficient context-free parsing algorithm, Communications of the ACM 13 (1970) 94–102.

[25] J. Benediktsson, P. Swain, Consensus theoretic classification methods, IEEE Transactions on Systems, Man, and Cybernetics 22 (1992) 688–704.

Vikram Krishnamurthy is currently a professor and holds the Canada Research Chair at the Department of Electrical Engineering, University of British Columbia, Vancouver, Canada. His current research interests include computational game theory, stochastic dynamical systems for modeling of biological ion channels and stochastic optimization and scheduling. In 2009 and 2010 he serves as Distinguished lecturer for the IEEE Signal Processing Society. From 2010 he serves as Editor in Chief of IEEE Journal Selected Topics in Signal Processing.

Mustafa Fanaswala is currently a PhD student at the Department of Electrical Engineering, University of British Columbia, Vancouver, Canada. He received his Masters degree from the University of Otawa.