Agent-Oriented Incremental Team and Activity Recognition*

Daniele Masato Timothy J. Norman
Wamberto W. Vasconcelos
Department of Computing Science
University of Aberdeen
Scotland, UK

Katia Sycara
Robotics Institute
Carnegie Mellon University
Pittsburgh, PA USA

Abstract

Monitoring team activity is beneficial when human teams cooperate in the enactment of a joint plan. Monitoring allows teams to maintain awareness of each other's progress within the plan and it enables anticipation of information needs. Humans find this difficult, particularly in time-stressed and uncertain environments. In this paper we introduce a probabilistic model, based on Conditional Random Fields, to automatically recognise the composition of teams and the team activities in relation to a plan. The team composition and activities are recognised incrementally by interpreting a stream of spatio-temporal observations.

1 Introduction

In many real-life scenarios human teams need to cooperate towards the achievement of one or more goals during the enactment of a joint plan. Examples of such scenarios are disaster relief operations, crowd management at major outdoor events, and search-and-rescue missions [Sukthankar *et al.*, 2008]. Monitoring team activity is essential when time is a critical factor for the success the plan and the operating environment is uncertain. Monitoring allows teams to maintain awareness of each other's progress within the plan, it helps to keep interdependent tasks synchronised, and it enables anticipation of information needs. Monitoring has become easier to perform thanks to the development of wireless sensor networks that can collect large amounts of environmental data during plan execution. There is, however, a risk that processing that data may result in cognitive overload for humans, therefore hampering effective decision-making.

We propose the multi-agent architecture illustrated in Figure 1 to monitor team activity during plan execution. At the lower level, a software agent is associated with each team, working as that team's assistant. Team members usually employ teamwork when acting in a time-critical scenario, hence their movement patterns will be related in time and space. The team as a whole will also exhibit recognisable behaviour patterns (e.g., tactics or manoeuvres). Machine learning techniques can be applied to learn such patterns, so that the agent can infer the *team composition* and perform *activity recognition* based on the team members' locations. Recognised activities A_i are represented as hexagons in Figure 1.

At the upper level, activities A_i will be used as evidence about the plan progress by matching them against atomic plan actions. This process can identify active plan branches (bold grey path in Figure 1) but can also be exploited to detect deviations from the expected course of action. We assume that a software agent has some knowledge about the plan structure ($STask_i$) and dependencies (links between $STask_i$). These could be of any nature (e.g., ordering/time constraints, team size) and are usually specified as part of the joint plan.

In this paper we focus on team composition and activity recognition (lower level in Figure 1). Existing research on probabilistic models for activity recognition has mainly focused on single subjects, rather than teams. In addition, prediction is mostly performed on complete observation traces, rather than on a observation window (see Section 2). Even when classification is performed incrementally on a window, the computational cost depends on the window size, and little attention is dedicated to the cost of aggregating sensor data into features for efficient classification. These aspects are key to performing incremental prediction when software assistants are implemented into portable, resource-limited devices. In Section 4 we introduce a probabilistic model to recognise teams and activities incrementally from a stream of observations. The model is based on Conditional Random Fields (see Section 3). It can perform inference over an observation window in polynomial time independent of the window size, or on the entire sequence. It also scales well with respect to the number of observed subjects. In Section 5 we evaluate the model on spatio-temporal observations collected from artificial players acting in a virtual environment. We finally conclude in Section 6.

2 Related Work

The two main approaches to activity recognition are *template-based matching* and *probabilistic graphical models*.

*This research was sponsored by the U.S. Army Research Laboratory and the U.K. Ministry of Defence and was accomplished under Agreement Number W911NF-06-3-0001. The views and conclusions contained in this document are those of the author(s) and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Army Research Laboratory, the U.S. Government, the U.K. Ministry of Defence or the U.K. Government The U.S. and U.K. Governments are authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation hereon.

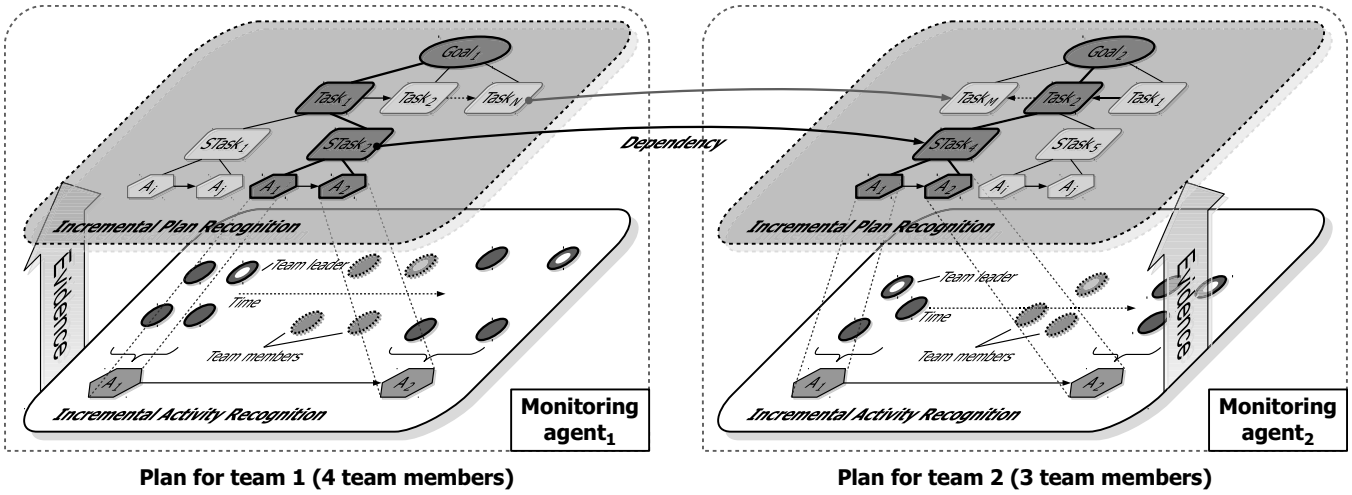

Figure 1: Incremental activity and plan recognition architecture

Template-based matching for team activity recognition [Sukthankar and Sycara, 2006] relies on a library of team behaviours, usually acquired by encoding experts' knowledge about the application domain. This approach attempts to match observed team features against the library in order to find one or more suitable labels for the observations. Template-based matching does not usually require machine learning because relationships between observations and labels are encoded in the library. Considerable effort, however, has to be put into eliciting the appropriate knowledge and encoding it into the library; in addition, particular care is required in designing matching algorithms that are robust to noise in the observations [Nguyen *et al.*, 2003].

In recent years, machine learning techniques, and in particular probabilistic graphical models, have received much attention because of their ability to effectively learn and infer activities from spatio-temporal observations. Conditional Random Fields (CRFs) are probabilistic models that were first applied to pattern recognition tasks in natural language processing, but they have become popular for activity recognition. Linear CRFs have been employed to identify the role of single robots in a game of Tag according to their locations [Vail *et al.*, 2007] and to recognise player roles in the *RoboCup* domain [Vail *et al.*, 2008], where two teams of computer-controlled robots compete in a football match. Relational Markov Networks [Liao *et al.*, 2007a] have been used to model a person's high-level activities (being at work, being at home, travelling) and significant places visited, based on GPS traces of that person's movements throughout the day. [Liao *et al.*, 2007b] has introduced a Dynamic Bayesian Network for learning and inferring transportation routines. That model was able to detect deviations from the routine, and to predict a person's location both in the short (e.g., change at a bus stop) and long term (e.g., travel to work or home).

Observations other than spatio-temporal traces can be used as well; in [Lian and Hsu, 2009], Factorial CRFs trained with annotated audio recordings were used to learn and classify chatting activity in meetings and public occasions. In [Yin *et al.*, 2009], Dynamic CRFs were shown to perform well in detecting events (e.g., presence of light) generated by a network of sensors distributed in an environment of interest. In common with to our approach, relationships among sensor readings in response to events were exploited to create meaningful features for classification. Most of that work relies on complex probabilistic models which may offer an increased accuracy in recognition, but required iterative, approximate inference algorithms whose running time cannot be characterised well in terms scalability with respect to the window/sequence size and the number of features.

3 Conditional Random Fields

Conditional Random Fields (CRFs) [Lafferty *et al.*, 2001] are probabilistic graphical models to perform relational learning on data sequences. Given a sequence of T observations $\mathbf{x} = (x_1, \ldots, x_T)$ and a sequence of corresponding hidden variables $\mathbf{y} = (y_1, \ldots, y_T)$, a CRF represents a conditional distribution $p(\mathbf{y}|\mathbf{x})$ as an undirected graph G. Nodes in G belong to the set $\{y_t\} \cup \{x_t\}$ and edges capture dependency relationships between observations and hidden variables. Hidden variables y_t assume values from a finite set of labels $\mathcal{Y}$. The distribution $p(\mathbf{y}|\mathbf{x})$ is computed as:

$$p(\mathbf{y}|\mathbf{x}) = \frac{1}{Z(\mathbf{x})} \prod_{c \in \mathcal{C}} \Psi_c(\mathbf{x}_c, \mathbf{y}_c), \; Z(\mathbf{x}) = \sum_{\mathbf{y}' \in \widehat{\mathcal{Y}}} \prod_{c \in \mathcal{C}} \Psi_c(\mathbf{x}_c, \mathbf{y}'_c) \tag{1}$$

In (1), $\mathcal{C}$ is the set of cliques in G, Ψ_c is a potential function, or *factor*, associated with clique c, $\mathbf{y}_c \subseteq \{y_t\}$, $\mathbf{x}_c \subseteq \{x_t\}$ and $\widehat{\mathcal{Y}} = \mathcal{Y}^T$ is the set of all possible label sequences. Z is the normalising partition function that guarantees $p(\mathbf{y}|\mathbf{x})$ is a valid probability distribution.

In our approach, we use Linear CRFs (LCRF, see Figure 2), which impose a first-order Markov assumption on the hidden variables and have a factor for each sequence position. In LCRFs, factors are defined as a linear combination of M *features* f_i weighted according to a real-valued vector $\boldsymbol{\lambda}$:

$$\Psi_{t,\boldsymbol{\lambda}}(\mathbf{x}, \mathbf{y}) = \exp\left(\sum_{i=1}^{M} \lambda_i f_i(y_{t-1}, y_t, \mathbf{x}, t)\right), 1 \leq t \leq T \tag{2}$$

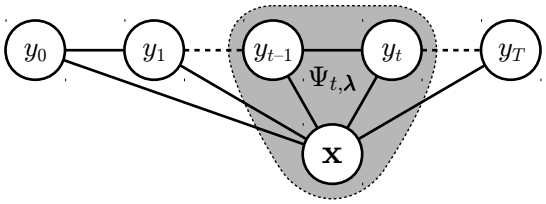

Figure 2: A Linear Conditional Random Field (LCRF)

Features f_i are real-valued functions that capture domain-specific relationships between graph nodes belonging to the same factor. They encode preferred label values for the hidden variables y_{t-1}, y_t according to some cues gathered from the observation sequence $\mathbf{x}$ at time t.

LCRFs are related to Hidden Markov Models (HMMs), but they relax the assumption that the observations in the sequence be mutually independent. This has two advantages:

- The model can accommodate overlapping, interdependent features. In addition, preferred label transitions can be made dependent on the features, making LCRF more suitable to perform inference on spatio-temporal traces [Vail *et al.*, 2007].[1]
- The forward-backward and Viterbi algorithms for efficient exact inference can be adapted to work for LCRFs [Sutton and McCallum, 2006].

Features are represented as a matrix random variable $F_t(\mathbf{x}) = \left[\Psi_{t,\boldsymbol{\lambda}}\left(\mathbf{x}, \mathbf{y}|_{(y_{t-1},y_t)=(y',y)}\right)\right]$, where $(y', y) \in \mathcal{Y}\times\mathcal{Y}$, and the forward and backward vectors $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ are defined as in (3).[2] Marginal probabilities are computed as in (4).

$$\alpha_1(y_1) = F_1(\mathbf{x})[\perp, y_1],\ \alpha_t(y_t) = \alpha_{t-1}(y_{t-1})\cdot F_t(\mathbf{x}) \quad (3a)$$

$$\beta_T(y_T) = \mathbf{1}^\top,\ \beta_t(y_t) = F_{t+1}(\mathbf{x})\cdot\beta_{t+1}(y_{t+1}) \quad (3b)$$

$$p(y_t = y|\mathbf{x}) = \frac{\alpha_t(y)\cdot\beta_t(y)}{Z(\mathbf{x})},\ Z(\mathbf{x}) = \textstyle\sum_{y\in\mathcal{Y}}\alpha_T(y) \quad (4)$$

Given a training set $\mathcal{E} = \{(\mathbf{x}_1, \mathbf{y}_1), \ldots, (\mathbf{x}_N, \mathbf{y}_N)\}$ of N observation/label pairs, a LCRF is trained by computing the optimal weights $\boldsymbol{\lambda}^*$ that maximise the log-likelihood $\ell_{\boldsymbol{\lambda}}(\mathcal{E})$:

$$\ell_{\boldsymbol{\lambda}}(\mathcal{E}) = \textstyle\sum_{i=1}^{N}\log p(\mathbf{x}_i|\mathbf{y}_i) - \sum_{i=1}^{M}\lambda_i^2/2\sigma^2 \quad (5)$$

$$\boldsymbol{\lambda}^* = \arg\max_{\boldsymbol{\lambda}}\ell_{\boldsymbol{\lambda}}(\mathcal{E}) \quad (6)$$

where σ is a regularisation parameter to avoid over-fitting. Because $\ell_{\boldsymbol{\lambda}}(\mathcal{E})$ is convex with respect to $\boldsymbol{\lambda}$, the maximisation can be performed via numerical gradient techniques. We employ the limited-memory BFGS algorithm [Zhu *et al.*, 1997].

4 Team and Activity Recognition Model

Our model is based on the intuition that the spatio-temporal traces of subjects working in a team will exhibit coordination patterns. These patterns can be exploited to learn and classify that team's members and activities. For example, teammates will try to maintain visual contact by remaining close over time; this can be captured by monitoring whether the distance among subjects falls within a particular threshold. Subjects' velocities and accelerations offer good cues as well; teammates usually adjust their speed to maintain uniformity throughout the performance of an activity. Team activities will also show temporal consistency between consecutive time steps. These intuitions are captured in Figure 2; the links between y_{t-1}, y_t and $\mathbf{x}$ encode those patterns, whereas the link between y_{t-1}, y_t guarantees temporal consistency.

In our scenario, S subjects act in a virtual environment. The observation sequence is $\mathbf{x} = \{\mathbf{p}_1, \ldots, \mathbf{p}_T\}$, where $\mathbf{p}_{t,s}$ contains the 2D position of the s-th subject at time t. $\mathcal{S} = \{s_1, \ldots, s_S, \ominus\}$ is the set of all subjects, where $\ominus$ is a fictitious subject that represents single-subject teams.

4.1 Features

Features play two crucial roles: during learning, they should favour relationships that achieve a good model accuracy; during classification, their computational cost should scale well according to the model size. We define our features as:

$$f_i(y_{t-1}, y_j, \mathbf{x}, s_o, t) = \mathbf{1}_{\{l_1\}}(y_{t-1})\times\mathbf{1}_{\{l_2\}}(y_t)\times\mathbf{1}_{\mathcal{O}}(s_o)\times$$
$$\times\ \text{score}_i\left((g_i^{(s_1,s_2)}(\mathbf{x}_{t-W}), \ldots, g_i^{(s_1,s_2)}(\mathbf{x}_{t+W}))\right) \quad (7)$$

In (7), $\mathbf{1}_{\mathcal{A}}(\cdot)$ is the indicator function which takes value 1 if $(\cdot)$ belongs to the set $\mathcal{A}$ and 0 otherwise; $l_1 \in \mathcal{Y}$ and $l_2 \in \mathcal{Y}$ are label values capturing preferred transitions for two consecutive hidden variables; $\mathcal{O} \subseteq \mathcal{S}$ is the set of subjects for which the feature is active; $s_o \in \mathcal{S}$ is the subject we wish to find a teammate/activity for; $g_i^{(s_1,s_2)}$ computes a spatio-temporal cue for the pair $(s_1, s_2) \in \mathcal{S}\times\mathcal{S}$; and score_i aggregates and rates the cues over an observation window of interest. The observation window is centred at the current observation and has length $2W + 1$.

Features for Team Recognition

Grouping subjects into teams is challenging because the number of possible teams grows exponentially with S, even if subjects cannot belong to multiple teams [Sukthankar and Sycara, 2006]. Considering all possible teams as label values $\mathcal{Y}$ would render inference infeasible. The same would happen if the cues g_i were defined over all subject subsets. For these reasons, we created features that match one subject with *one* of the possible teammates. Because there are only $O(S^2)$ subject combinations, we can introduce a $O(S^2)$ number of *simple* features, where the cost for individual feature is $O(1)$. Therefore, the feature computational cost scales well with respect to the team size. The cost can be reduced even further if features are computed locally by sensors attached to each subject and then collected centrally for classification.

We encoded the features listed in Table 1 in a *single* LCRF (rather than one per subject) where we set $\mathcal{Y} = \mathcal{S}$. The LCRF will learn to associate each subject s_o with a teammate. Note that cues in Table 1 are not based on raw subject positions $\mathbf{p}_{t,s}$. This would make the model dependent on the particular environment (e.g., the same team often appears at a particular location). Although this might improve the performance for that environment, it would also result in overfitting.

[1] For example, consecutive measurements of a human subject's position or velocity are dependent, assuming that the sampling is performed at a sufficiently high rate.

[2] $\perp$ is a special symbol that identifies the start of the sequence.

#	(l_1, l_2)	$\mathcal{O}$	$g_i^{(s_1,s_2)}$	Description
1.1	$(\text{any}, s), s \in \mathcal{S}$	$\mathcal{S}$	1	Transition probability
	$(s', s) \in \mathcal{S} \times \mathcal{S}$	$\mathcal{S} \setminus \{l_2\}$	1	Prior probability
1.2	$(\text{any}, s), s \in \mathcal{S}$	$\{l_2\}$	-1	Penalty for self-association
1.3	$(\text{any}, s_2), s_2 \in \mathcal{S}$	$\{s_1\}, s_1 \in \mathcal{S} \setminus \{l_2\}$	$\lVert \mathbf{distance}(s_1, s_2) \rVert$	Proximity intercept
1.4	(s_2, s_2) $s_2 \in \mathcal{S} \setminus \{\ominus\}$ and $(\text{any}, s_2), s_2 \in \mathcal{S}$	$\{s_1\}$ $s_1 \in \mathcal{S} \setminus \{l_2, \ominus\}$ and $\mathcal{S} \setminus \{l_1, l_2\}\, \forall t$	$\lVert \mathbf{distance}(s_1, s_2) \rVert$	Pairwise distance
1.5			$\lVert \mathbf{distance}'(s_1, s_2) \rVert$ $\angle\, \mathbf{distance}'(s_1, s_2)$	Distance velocity compatibility
1.6			$\lVert \mathbf{distance}''(s_1, s_2) \rVert$ $\angle\, \mathbf{distance}''(s_1, s_2)$	Distance acceleration compatibility
1.7			$\lVert \mathbf{velocity}(s_1) \rVert / \lVert \mathbf{velocity}(s_2) \rVert$ $\angle\, \mathbf{velocity}(s_1) - \angle\, \mathbf{velocity}(s_2)$	Pairwise velocity compatibility
1.8			$\lVert \mathbf{acceler}(s_1) \rVert / \lVert \mathbf{acceler}(s_2) \rVert$ $\angle\, \mathbf{acceler}(s_1) - \angle\, \mathbf{acceler}(s_2)$	Pairwise acceleration compatibility
1.9			$\frac{\mathbf{distance}(s_1, s_2)}{\lVert \mathbf{distance}(s_1, s_2) \rVert} \cdot \frac{\mathbf{velocity}(s_1)}{\lVert \mathbf{velocity}(s_1) \rVert}$	s_1 following s_2

Table 1: Features for team recognition

Features for Activity Recognition

Subjects can perform a finite set of behaviours $\mathcal{B}$ (see also Section 5), so we created a second *single* LCRF with $\mathcal{Y} = \mathcal{B}$. The LCRF will learn to associate each subject s_o to the activity the team s_o belongs to is performing. Features are similar to those listed in Table 1, where values for l_1 and l_2 belong to $\mathcal{B}$ rather than $\mathcal{S}$. We left out cues of type 1.3 and instead included label transition features linked to the subjects' individual speeds and accelerations. We also included label transition features accounting for each subject's team size.

Scores

Each cue $g_i^{(s_1,s_2)}$ in Table 1 is associated to a score $0 \leq \text{score}_i \leq 1$ expressing how likely is that s_1 and s_2 belong to the same team. For example, cues 1.3 and 1.4 will be given a proportionally higher score the closer s_1 and s_2 are. Similarly, cues 1.7 will be given a higher score the more similar the subjects' velocities are. It is essential that cue scores be aggregated incrementally over the observation window. In our model we simply average the cue scores. An average $\sum_{i=1}^{N} x_i/N$ can be incrementally computed by keeping numerator and denominator separate. When a new value x_{N+1} arrives, the numerator is updated by subtracting x_1 and adding x_{N+1}. Other scoring algorithms can be used as long as the incremental update property is guaranteed.

4.2 Incremental Forward-Backward

Classification on a moving window consists of assigning the best *individual label* to each hidden variable y_t. Given an unseen observation window $\mathbf{x}^*_{t,W} = (x^*_{t-2W}, \ldots, x^*_{t+2W})$, $y_t = \arg\max_{y' \in \mathcal{Y}} p(y' | \mathbf{x}^*_{t,W})$, where t is the current time step. This can be performed by applying the forward-backward algorithm in (3) and (4), where $\boldsymbol{\alpha}_t$ and $\boldsymbol{\beta}_t$ are:

$$\boldsymbol{\alpha}_t = \mathbf{1} \cdot A_t, \quad A_t = F_{t-W}(\mathbf{x}^*_{t,W}) \cdot \ldots \cdot F_t(\mathbf{x}^*_{t,W}) \tag{8a}$$

$$\boldsymbol{\beta}_t = B_t \cdot \mathbf{1}^\top, \quad B_t = F_{t+1}(\mathbf{x}^*_{t,W}) \cdot \ldots \cdot F_{t+W}(\mathbf{x}^*_{t,W}) \tag{8b}$$

with a time complexity of $O(W|\mathcal{Y}|^3)$ (window size times matrix multiplication). When a new observation arrives, y_{t+1} could be computed with an application of (8) to $\mathbf{x}^*_{t+1,W}$, but we compute it more efficiently by updating the forward-backward vectors as $\boldsymbol{\alpha}_{t+1} = \mathbf{1} \cdot A_{t+1}$, $\boldsymbol{\beta}_{t+1} = B_{t+1} \cdot \mathbf{1}^\top$ where A_{t+1} and B_{t+1} are the solutions of the linear systems:

$$F_{t-W}(\mathbf{x}^*_{t,W}) \cdot A_{t+1} = A_t \cdot F_{t+1}(\mathbf{x}^*_{t+1,W}) \tag{9a}$$

$$F_{t+1}(\mathbf{x}^*_{t,W}) \cdot B_{t+1} = B_t \cdot F_{t+W+1}(\mathbf{x}^*_{t+1,W}) \tag{9b}$$

with a time complexity of $O(|\mathcal{Y}|^3)$ (matrix multiplication and system solution computation). Both (8) and (9) require $O(W|\mathcal{Y}|^3)$ memory to store the F_i over the window. Because computing the features requires $O(|\mathcal{Y}|^2)$ time at each time step (see Section 4), inference complexity remains $O(|\mathcal{Y}|^3)$.

There is a limitation in this approach; if some of the F_i are ill-conditioned, the system solutions may be very difficult to compute, even for mainstream linear-algebra packages (e.g., LAPACK). When this happens, we regularise F_i using the Singular Value Decomposition (SVD) [Neumaier, 1998]. When F_i is ill-conditioned, some of its singular values are very close to zero, or zero. We set those to 1, so that F_i becomes full-rank and the system solutions can be computed. Computing the SVD requires $O(|\mathcal{Y}|^3)$ time, but has a significant hidden constant (22); however, because few F_i require regularisation, this cost can be amortised throughout the window. This approach does not completely avoid recomputing the window via (8); round-off errors will eventually make it necessary. This condition can be detected by checking A_{t+1} and B_{t+1} for negative entries. Those are invalid because A_{t+1} and B_{t+1} are products of matrices with non-negative entries (see (2)). We analyse how often this happens in Section 5.

4.3 Incremental Viterbi

Instead of assigning the best label to each y_t, the Viterbi algorithm in (10) can be used to compute the most likely *sequence of labels* $\mathbf{y}^* = \arg\max_{\mathbf{y} \in \widehat{\mathcal{Y}}} p(\mathbf{y} | \mathbf{x}^*)$, given an unseen sequence $\mathbf{x}^*$. In (10), $\delta_t(y)$ is proportional to the probability of the most likely label sequence $(y^*_1, \ldots, y^*_{t-1})$ ending with $y_t = y$; $\phi_t(y)$ encodes the optimal label transition path leading to $y_t = t$ given the optimal labels up to time $t - 1$.

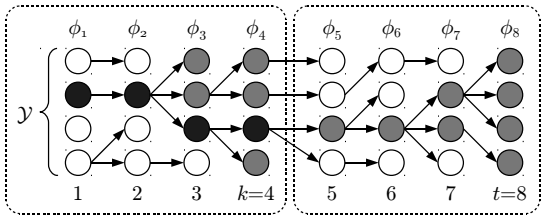

Figure 3: Path matrix P for incremental Viterbi

$$\delta_1(y_1) = F_1(\mathbf{x}^*)[\perp, y_1],\ \delta_t(y) = \max_{y' \in \mathcal{Y}} \delta_{t-1}(y') \cdot F_t(\mathbf{x}^*)[y', y]$$
$$\phi_1(y_1) = \perp,\ \phi_t(y) = \arg\max_{y' \in \mathcal{Y}} \phi_{t-1}(y') \cdot F_t(\mathbf{x}^*)[y', y] \quad (10)$$

The sequence $\mathbf{y}^*$ is retrieved by starting from the most likely final label $y_T^* = \arg\max_{y' \in \mathcal{Y}} \delta_T(y')$ and evaluating $y_t^* = \phi_{t+1}(y_{t+1}^*), T-1 \geq t \geq 1$. This, however, requires reaching the end of the observation sequence before obtaining any labelling. It also means that the memory required to store $\boldsymbol{\delta}$ and $\boldsymbol{\phi}$ may exceed that available on the host system.

We shall instead apply the incremental Viterbi algorithm introduced in [Bobbin, 2007]. The algorithm relies on the path matrix $P = [\boldsymbol{\phi}_1, \ldots, \boldsymbol{\phi}_k, \ldots, \boldsymbol{\phi}_t]$ shown in Figure 3. P is a collection of trees where nodes at time t represent label values for y_t. Tree paths in Φ identify the most likely label sequences ending with $y_t = y$. It may happen that at some time k, all paths are rooted at a unique initial label y_1^* (common root). When this happens again, let us assume at time t, it is possible to output the classification $(y_1^*, \ldots, y_k^*)$ up to time k (the black path in Figure 3). At this point the mechanism can restart on a new path matrix $P' = \left[\boldsymbol{\phi}_{k+1}, \ldots, \boldsymbol{\phi}_t, \ldots, \boldsymbol{\phi}_{t'}\right]$. The incremental Viterbi algorithm does not guarantee to output a new prediction for each new observation, and the worst-case complexity is $O(T|\mathcal{Y}|^2)$. Nonetheless, a partial classification can be obtained based on the frequency of the common roots, hence dramatically reducing memory usage.

5 Experimental Evaluation

We evaluated our recogniser on spatio-temporal traces collected from the shooter game OpenArena, an off-the-shelf game that provides a challenging environment for activity recognition.[3] Teams form and disband rapidly and players' behaviour is highly dynamic. We collected data from a *capture-the-flag* game; two opposing squads (red and blue) aim to get a flag located in the middle of the game scenario and to bring it to the opponent's base, while protecting their own base. When a squad succeeds, the flag returns to the middle.

We have instrumented the game to run simulated games using the embedded AI players (*bots*). Bots are implemented as finite state machines whose states identify basic actions (e.g., fight, swim). Bot squads can organise themselves in smaller teams pursuing higher level tasks (e.g., defend own base, get flag, attack enemy base). We trained two LCRFs on 30 minutes of data sampled every 2 seconds on a game played by two squads of 10 players each. For the first LCRF we labelled each subject with a teammate, whereas for the second we labelled each subject with the corresponding team activity. The results are based on three-fold cross validation.

[3] See `http://openarena.ws` for more information.

Team recognition

The team recognition performance is based on a comparison between the recognised teams and the ground truth at each time step. Given that team-recognition produces pairs of subjects and their probability of belonging to the same team, we need to reconstruct teams from these pairs. We order the subject pairs by decreasing probability, then we greedily group pairs into teams by merging pairs with common subjects according to the ordering. We order the recognised and actual teams by size, then we evaluate the average precision and recall. Given $\mathcal{P} = \{$*recognised team members*$\}$, $\mathcal{M} = \{$*actual team members*$\}$, Precision$= |\mathcal{P} \cap \mathcal{M}|/|\mathcal{P}|$ and Recall$= |\mathcal{P} \cap \mathcal{M}|/|\mathcal{M}|$. The results are shown in Table 2, for different sizes of the observation window W.

For $W = 0$ (no window) the model is similar to a HMM, because all features linked to label transitions (e.g., velocity, acceleration) are not considered. This results in a lower recall. The recall increases sharply as soon as the window opens. Precision and recall do not show large variations with respect to the window size, but they tend to decrease as the window size increases. This means that a shorter window better captures team dynamics than a longer one. When the window size grows, increasingly old observations become less relevant and degrade the model performance. Note also that the number of times it is necessary to recompute the window for the incremental forward-backward (W reset) decreases as the window size increases. Considering a larger window probably allows the linear system solver to perform better.

Activity recognition

Although bots can still perform different basic actions when in the same team, we are interested in recognising the higher level team activities. Therefore, we trained the activity-recognition LCRF by labelling each teammate with the corresponding team-level activity. The label set is $\mathcal{B} = \{$*attack enemy's base*, *stay at own base*, *patrol own base*, *get flag*, *recover flag when lost*, *bring flag to enemy's base*$\}$. We then evaluated the LCRF on the average number of correct label assignments using different window sizes. Results are shown in Table 3, for different sizes of the observation window W.

Given that the probability of randomly guessing a subject's activity is 16.67%, the model performs better than chance, but not extremely well. Activities are probably challenging to recognise using the features we introduced in Section 4 because they *depend* on that subject's raw position; for example capturing the flag often happens in the middle of the map, and the location of the bases offers a good indication about whether a subject is attacking or defending. There is, as before, a sharp improvement in the performance when $W > 0$ and a slight decrease as the window size increases.

6 Conclusion

In this paper we introduced a probabilistic model to perform incremental team composition and activity recognition on a stream of observations. We evaluated the model on spatio-

Incremental Forward-Backward					
W	**Squad**	**Precision**	**Recall**	**F-score**	**W reset**
0 sec.	red	63.33	45.64	53.04	n/a
	blue	63.87	44.41	52.39	
2 sec.	red	63.83	56.26	59.80	26.54
	blue	61.37	54.88	57.94	26.74
4 sec.	red	59.88	59.24	59.55	22.83
	blue	60.79	57.03	58.85	23.51
8 sec.	red	59.61	60.79	60.19	19.08
	blue	61.43	57.29	59.28	19.27
16 sec.	red	55.98	58.51	57.21	19.28
	blue	58.17	56.56	57.35	19.98
30 sec.	red	51.47	59.27	55.09	20.76
	blue	55.21	52.45	53.79	20.19

Incremental Viterbi				
W	**Squad**	**Precision**	**Recall**	**F-score**
0 sec.	red	48.63	78.76	60.13
	blue	47.77	77.75	59.17
2 sec.	red	58.51	55.81	57.12
	blue	55.11	54.77	54.93
4 sec.	red	57.89	56.86	57.37
	blue	57.92	56.72	57.31
8 sec.	red	55.98	59.61	57.73
	blue	56.75	54.43	55.56
16 sec.	red	50.93	59.46	54.86
	blue	51.78	56.35	53.96
30 sec.	red	49.29	57.80	53.20
	blue	54.11	54.14	54.12

Table 2: Team recognition performance (%)

temporal traces collected from artificial players acting in a highly dynamic environment. The model performed reasonably well when recognising team composition, whereas performance for activity recognition is not very conclusive. In the future, we shall investigate in more detail whether it is possible to improve the activity recognition performance. Finally, we plan to evaluate the performance of our model on traces collected during a simulation with human subjects.

Incremental Forward-Backward						
Squad	**0 sec.**	**2 sec.**	**4 sec.**	**8 sec.**	**16 sec.**	**30 sec.**
red	22.58	52.38	48.34	48.62	47.76	44.49
blue	29.64	43.44	44.24	39.65	39.65	41.32

Incremental Viterbi						
Squad	**0 sec.**	**2 sec.**	**4 sec.**	**8 sec.**	**16 sec.**	**30 sec.**
red	41.21	50.69	48.20	51.04	50.47	45.55
blue	39.67	44.18	43.85	41.07	39.85	41.26

Table 3: Activity recognition accuracy (%)

References

[Bobbin, 2007] J. Bobbin. An incremental Viterbi algorithm. In *Proc. of the Fourth Int. Conf. on Machine Learning and Applications*, Ohio, USA, 2007.

[Lafferty *et al.*, 2001] J. D. Lafferty, A. McCallum, and F. Pereira. Conditional Random Fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. of the Eighteenth Int. Conf. on Machine Learning*, Massachusetts, USA, 2001.

[Lian and Hsu, 2009] C. Lian and J. Hsu. Probabilistic models for concurrent chatting activity recognition. In *Proc. of the Twenty-First Int. Joint Conf. on Artificial Intelligence*, California, USA, 2009.

[Liao *et al.*, 2007a] L. Liao, D. Fox, and H. Kautz. Extracting places and activities from GPS traces using Hierarchical Conditional Random Fields. *The Int. Journal of Robotics Research*, 26(1):119–134, 2007.

[Liao *et al.*, 2007b] L. Liao, D. J. Patterson, D. Fox, and H. Kautz. Learning and inferring transportation routines. *Artificial Intelligence*, 171(5–6):311–331, 2007.

[Neumaier, 1998] A. Neumaier. Solving ill-conditioned and singular linear systems: A tutorial on regularization. *SIAM Review*, 40(3):636–666, 1998.

[Nguyen *et al.*, 2003] N. T. Nguyen, H. H. Bui, S. Venkatesh, and G. West. Recognising and monitoring high-level behaviours in complex spatial environments. In *Proc. of the IEEE Computer Society Conf. on Computer Vision and Pattern Recognition*, Wisconsin, USA, 2003.

[Sukthankar and Sycara, 2006] G. R. Sukthankar and K. Sycara. Robust recognition of physical team behaviours using spatio-temporal models. In *Proc. of the Sixth Int. Joint Conf. on Autonomous Agents and Multiagent Systems*, Japan, 2006.

[Sukthankar *et al.*, 2008] G. R. Sukthankar, K. Sycara, J. A. Giampapa, and C. Burnett. A model of human teamwork for agent-assisted search operations. In *NATO Research and Technology Organisation: Human Factors and Medicine Panel Symposium*, France, 2008.

[Sutton and McCallum, 2006] C. Sutton and A. McCallum. An introduction to Conditional Random Fields for relational learning. In *Introduction to Statistical Relational Learning*. MIT Press, 2006.

[Vail *et al.*, 2007] D. L. Vail, M. M. Veloso, and J. D. Lafferty. Conditional Random Fields for activity recognition. In *Proc. of the Sixth Int. Conf. on Autonomous Agents and Multiagent Systems*, Hawaii, USA, 2007.

[Vail *et al.*, 2008] D. L. Vail, M. M. Veloso, and J. D. Lafferty. Feature selection for activity recognition in multi-robot domains. In *Proc. of the Twenty-Third AAAI Conf. on Artificial Intelligence*, Illinois, USA, 2008.

[Yin *et al.*, 2009] J. Yin, D. Hao Hu, and Q. Yang. Spatio-temporal event detection using dynamic conditional random fields. In *Proc. of the Twenty-First Int. Joint Conf. on Artificial Intelligence*, California, USA, 2009.

[Zhu *et al.*, 1997] C. Zhu, R. H. Byrd, P. Lu, and J. Nocedal. Algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound-constrained optimization. *ACM Transactions on Mathematical Software*, 23(4):550–560, 1997.