Monte Carlo Tree Search for Scheduling Activity Recognition

Mohamed R. Amer, Sinisa Todorovic, Alan Fern
Oregon State University
Corvallis, Oregon, USA
{amerm, afern, sinisa}@eecs.oregonstate.edu

Song-Chun Zhu
University of California Los Angeles
Los Angeles, California, USA
sczhu@ucla.edu

Abstract

This paper presents an efficient approach to video parsing. Our videos show a number of co-occurring individual and group activities. To address challenges of the domain, we use an expressive spatiotemporal AND-OR graph (ST-AOG) that jointly models activity parts, their spatiotemporal relations, and context, as well as enables multitarget tracking. The standard ST-AOG inference is prohibitively expensive in our setting, since it would require running a multitude of detectors, and tracking their detections in a long video footage. This problem is addressed by formulating a cost-sensitive inference of ST-AOG as Monte Carlo Tree Search (MCTS). For querying an activity in the video, MCTS optimally schedules a sequence of detectors and trackers to be run, and where they should be applied in the space-time volume. Evaluation on the benchmark datasets demonstrates that MCTS enables two-magnitude speed-ups without compromising accuracy relative to the standard cost-insensitive inference.

1. Introduction

Research on video parsing has made impressive progress [1]. Recent approaches usually model activity parts, their spatiotemporal relations, and context (e.g., [16, 14, 4]), as well as enable tracking of actors (e.g., [6]). For this they use highly expressive activity representations whose intractable inference and learning require approximate algorithms. However, as the representations are getting increasingly expressive, even their approximate inference becomes prohibitively expensive.

This paper presents a framework for cost-sensitive video parsing using a probabilistic model, such that the resulting accuracy is close to that of the model's cost-insensitive inference. Our videos show a number of individual and group activities co-occurring in a large scene, as illustrated in Fig. 1. The video resolution allows for digital "zoom-in", and thus analyzing objects, and fine video details. Also, the video footage is long enough to show all stages of temporally structured activities. To address challenges of this domain, we use a hierarchical, spatiotemporal AND-OR graph (ST-AOG). ST-AOG is a stochastic grammar [18] that models both individual actions and group activities, captures relations of individual actions within a group activity, accounts for parts and contexts, and enables their tracking. Fig. 1 illustrates how ST-AOG increases modeling complexity relative to prior work.

ST-AOG enables parsing of challenging videos by running a multitude of object/people/activity detectors, and tracking their detections. For some applications, such a video parsing may be prohibitively expensive. To address this issue, we enforce that ST-AOG inference is cost sensitive, and formulate such an inference as a scheduling problem. In particular, given a query about a particular activity class (e.g., where and when it occurs in the video), our inference first identifies the node in ST-AOG that represents that activity, and then runs a *sequence* of computational processes for that node, termed as in [17, 14, 4]:

- α process: detecting objects, actions, and group activities directly from video features;
- β process: prediction of an activity from its children individual actions or objects;
- γ process: prediction of actions or objects using the context of their parent group activity;
- ω process: tracking an activity or action based on detections in the previous time intervals.

As illustrated in Fig. 1, the scheduling of α, β, γ, and ω processes jointly defines: which activity detectors to run, and which level of activities to track, and where in the space-time video volume to apply the detectors and tracking. Thus, given a query, the scheduling specifies a sequence of triplets {(process, detector, time interval)} to be run, in order to efficiently answer the query. The sequence is executed until the allowed time budget. In this way, inference becomes efficient, optimizing the total number of detectors and trackers to be run, for a given time budget.

We use training examples to learn a close-to-optimal scheduling of ST-AOG inference. Since the best sequence of inference steps is learned for each query type, and ex-

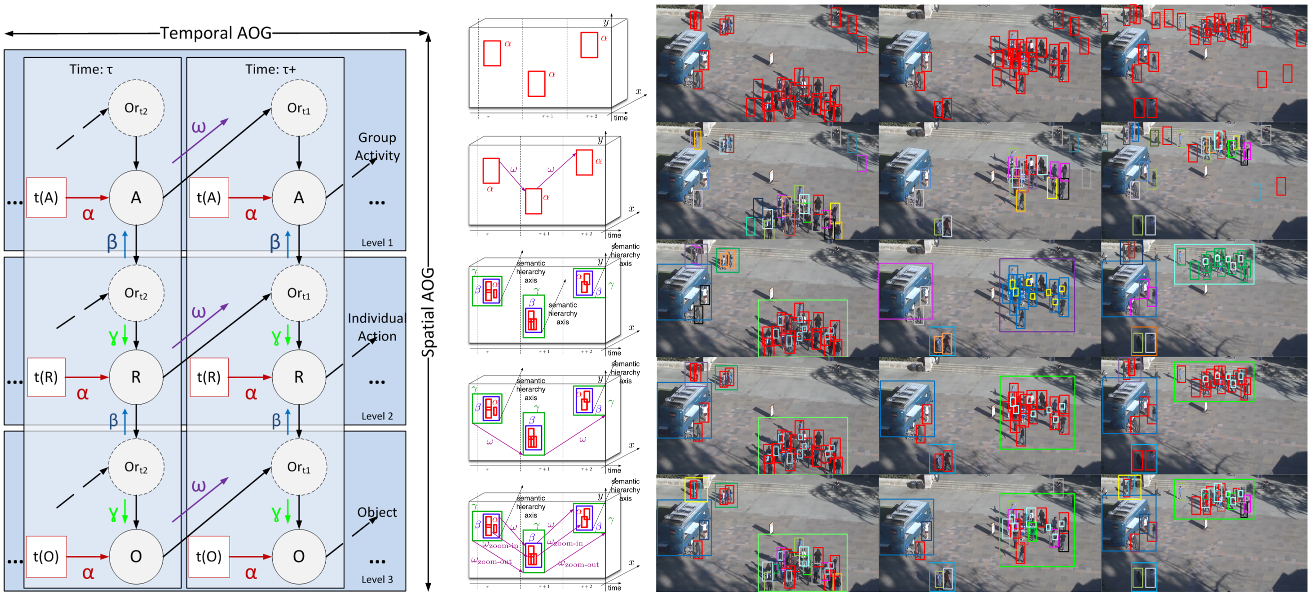

Figure 1. (*Left to right*) Our ST-AOG, inference processes (color-coded), and a video from the UCLA Courtyard dataset [4]. The rows show performance of typical activity representations, where modeling complexity increases top to bottom. The top two rows show detections of "walking", and tracking these detections for recognizing structured actions of each person, as in [9]; this approach may suffer from missed detections, identity switches, and false positives. The middle row increases complexity by accounting for both context and parts, as in [4]; this reduces the number of missed detections and false positives relative to the figure rows above; however, this approach cannot relate detections from different time intervals. The row above the bottom shows an approach that allows tracking but only at the group-activity level, as in [6]. The bottom row shows our performance for ST-AOG that models both individual actions and group activities, relations of individual actions within every group activity, context, and enables tracking at all semantic levels.

ecuted as-is during inference, our approach belongs to the general framework of open-loop planning. Inspired by recent advances in Monte Carlo planning [5], we use Monte-Carlo tree search (MCTS) to learn the scheduling of ST-AOG inference. MCTS iterates the exploration and exploitation steps. Exploration performs Monte-Carlo simulations for estimating the unknown distributions of rewards of various sequences of $\alpha, \beta, \gamma, \omega$. Based on these distribution estimates, exploitation conducts search for the best sequence with the highest reward.

In the following, Sec. 2 reviews prior work; Sec. 3 formulates our open-loop planning; Sec. 4 specifies MCTS; Sec. 5 defines ST-AOG; Sec. 6–7 present inference and learning; Sec. 8 describes implementation details; and Sec. 9 presents our results.

2. Prior Work and Our Novelty

Models: Prior work uses Conditional Random Fields (CRFs) [9, 6], Latent Structured SVM [12], Sum-Product Networks [3], and context-sensitive grammars [8, 16, 14, 4] for video parsing. Their inference is NP-hard, and requires approximations, such as, e.g., Branch-and-Bound [6], Network-flow [9], or the cutting-plane algorithm [12]. Also, grammar parsing, e.g., in [16, 14] uses dynamic programming to process only a few candidate detections.

Planning for Inference: Hidden Markov Decision Process (hMDP) is a planning framework used for predicting human trajectories [10]. However, hMDP was not used for activity recognition. MDP is not suitable for our problem, due to its restrictive assumptions that activities are governed by the stationary and first-order Markovian dynamics. In [4], Q-learning is used for scheduling the α, β, γ inference of a spatial AOG. In that work, Q-Learning simplifies the inference of a stochastic grammar by: i) Summarizing all current parse-graph hypotheses into a single state; and ii) Conducting inference as first-order Markovian moves in a large state space, following a fixed policy. The policy is estimated by Q-learning, where most of the states are simply ignored, because there is typically not enough training data to exemplify all states in the huge state space. Instead of estimating a single optimal move given a state via Q-learning, in this paper, we use MCTS for estimating an optimal *sequence* of moves in the state space.

Novelty: MCTS has never been used for learning how to schedule video parsing. ST-AOG extends the temporal AOG of [16, 14], and non-temporal AOG of [4]. Instead of Q-Learning, we use MCTS for estimating higher-order dependencies among a *sequence* of $\alpha, \beta, \gamma, \omega$.

3. Inference as Open-Loop Planning

Given a query for an activity in the video, and ST-AOG, a brute force approach to inference would be to run all detectors associated with $\alpha, \beta, \gamma, \omega$, and then compute the posterior of the query. This is prohibitive, and also unnecessary in most cases, since many detectors will provide little to no new information about the query. Our problem is to determine the best sequence of triplets $\{$(process, detector, time interval)$\}$ to run, in order to efficiently and accurately estimate the query. This problem can be viewed as a planning problem where each triplet is viewed as an inference step. Our goal is to select an optimal sequence of inference steps, given a time budget, that maximizes a utility measure.

One approach to selecting inference steps would be to follow a closed-loop planning, where at each step we run a planning algorithm to select the next step, based on the information from previous steps. However, in our setting, the computation required to select an optimal subset of detectors to run may be larger than just running the full set of detectors. Another closed-loop planning would be reinforcement learning (RL), e.g., used for grammar parsing in [4]. RL uses a policy that maps any inference state to an action (e.g., grammar inference step in [4]). However, since the number of inference states is enormous, such approaches require making significant approximations, e.g., ignoring most of the states. It is unclear how much the approximation is hurting accuracy.

We explore an alternative approach, based on open-loop planning. We pre-compute an explicit sequence of inference steps for each type of query that will be executed at inference time. The approach is open-loop in the sense that the sequence is executed without consulting previously obtained results. The assumption underlying our approach is that for each type of query there do exist high-quality open-loop sequences of inference steps. Our experimental results indicate that this is indeed true for ST-AOG inference. In particular, we demonstrate that our approach is more effective than prior work that used approximations in conjunction with closed-loop planning. In general, it is well-known that open-loop plans are quite effective, even compared to the best known closed-loop approximations.

We now formalize the open-loop planning problem. The steps available to our inference are $\{$(process, detector, time interval)$\}$ triplets. The process can be one of $\alpha, \beta, \gamma, \omega$, which invokes one of the corresponding detectors, at a certain time interval. In this paper, we use the set of 32 detectors. For each type of query, our objective is to produce a high utility sequence of inference actions $(a_1, \ldots, a_B)$, where B is the maximum budget available for applying actions during inference. Note that the exact observation sequence resulting from the action sequence will vary across videos. Thus, we take the utility of an action sequence to be the expected with respect to a distribution over videos of the log-likelihood of the parse graph, pg. As explained in Sec. 5 and Sec. 6, pg summarizes the current video parsing results given observations gathered from the applied action sequence. Note that this definition of utility cannot be computed exactly due to the expectation. Thus, in our approach, we estimate the expectation using an empirical average over a set of training data. In particular, we assume the availability of a set of training videos on which we can easily simulate the application of any action sequence and compute the required likelihoods. Next, we describe how to search for a high utility action sequence using MCTS.

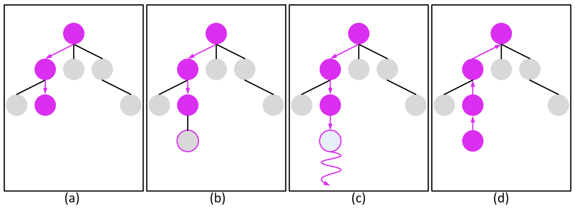

Figure 2. Steps for building MCTS tree. (a) Selection, where the selection function is applied recursively until a leaf node is reached. (b) Expansion, where a new node is added to the tree. (c) Simulation, where a sequence of moves is taken till the goal is reached. (d) Backpropagation, where the reward of the simulation is propagated to all the states along the tree path.

4. Monte-Carlo Tree Search

The number of potential action sequences is exponential in the budget B, and hence we use an intelligent search over potential action sequences, which is able to uncover high quality sequences in a reasonable amount of time. Our approach is based on the view that the set of all length B action sequences can be represented as a rooted tree, where edges correspond to actions, so that each path from the root to a leaf corresponds to a distinct length B action sequence. Our problem is to intelligently search this tree in order to quickly find a leaf corresponding to a high quality sequence.

To search the tree we use MCTS. An appealing aspect of MCTS is that it does not require the availability of a search heuristic function, which is generally required by other search methods, such as A*, and branch-and-bound methods. MCTS has shown considerable success in recent years, yielding state-of-the-art results in a variety of domains [5]. In this work, we focus on one of the most popular MCTS algorithms, called UCT [11].

MCTS is conceptually simple, as shown in Fig. 2. The search tree is built in an iterative manner. It is initialized to a single root node, and each iteration adds a single new leaf node to the current tree, and updates certain statistics of nodes in the tree. Each iteration, as illustrated in Fig. 2, begins by using a *tree policy* to follow a path of actions from the root until reaching a leaf node v of the current tree. A random action is selected at node v, and the resulting node v' is added to the tree as a new leaf, noting that v' corresponds to an action sequence from the root to v'. This action sequence is appended to by selecting random actions until reaching a depth of B, resulting in a sequence of B actions. The utility of the action sequence is then evaluated using the training videos, as described in Sec. 3. This evaluation is used to update the statistics of tree nodes along the path from the root to v'. Specifically, each node v in the tree maintains a count $n(v)$ of how many times the node has been traversed during the search, and the average utility $Q(v)$ of the length B actions sequences that have passed through the node so far during the search. Intuitively, the statistics at each tree node indicates the overall quality of the action sequences which have that node as a prefix.

After a specified number of iterations is reached, MCTS stops and returns the best depth B action sequence from the root. This is done by starting at the root and selecting the action that leads to the child node v with largest utility $Q(v)$. Then, from v the next action is the one that leads to the highest utility child of v. This repeats until we construct the entire path of B actions.

It remains to specify the tree policy which is the key ingredient in an MCTS algorithm as it controls how the tree is expanded. Intuitively, we would like the tree to be expanded toward more promising action sequences, which exploits information from previous information. However, at the same time the expansion to explore parts of the tree that have not been sampled much to avoid missing promising sequences. Different MCTS algorithms mainly differ in how they balance this trade-off between exploration and exploitation. We use the UCT algorithm that selects action a at node v as

$$\text{argmax}_a \left[Q(T(v,a)) + C\sqrt{\frac{2\log n(v)}{n(T(v,a))}} \right], \qquad (1)$$

where $T(v,a)$ denotes the tree node that is reached by selecting action a in node v. In (1), the exploitation term, $Q(T(v,a))$, favors actions that have been observed to have high average utility from v in previous iterations. The second exploration term in (1) is larger for actions that have been tried fewer times at v, since $n(T(v,a))$ is in the denominator. The parameter C thus serves to set the tradeoff between exploration and exploitation. We use $C = 1/\sqrt{2}$. Theoretically, by using this tree policy, UCT is guaranteed to converge to an optimal solution, if run long enough. In practice, UCT typically shows good anytime behavior.

In the next section, we specify our ST-AOG model. This will allow us to formalize, in Sec. 6, the $\alpha, \beta, \gamma, \omega$ processes that are scheduled by MCTS in inference for video parsing.

5. AOG Model

ST-AOG organizes domain knowledge in a hierarchical manner at three levels. Group activities are defined as a spatial relationship of a set of individual actions. They are represented by nodes at the highest level of ST-AOG. Individual actions are defined as punctual or repetitive motions of a single person, who may interact with an object. They are represented as children nodes of the group-activity nodes. Objects include body parts and tools or instruments that people interact with while conducting a individual action. Object nodes are placed at the lowest level of ST-AOG, and represent children of the individual-action nodes. Modeling efficiency is achieved by sharing children nodes among multiple parents, where AND nodes encode particular configurations of parts, and OR nodes account for alternative configurations. ST-AOG establishes temporal (lateral) edges between stages of the activity to model their temporal variations. Thus, ST-AOG accounts for both temporal and hierarchical context.

Formally, ST-AOG is a stochastic grammar, $\mathcal{G} = (\mathcal{V}_{NT}, \mathcal{V}_T, \mathcal{E}, \mathcal{P})$. $\mathcal{V}_{NT}$ is a set of non-terminal AND and OR nodes, denoted as $\wedge$ and $\vee$. $\mathcal{V}_T = \{t(\wedge) : \forall\wedge \in \mathcal{V}_{NT}\}$ is a set of terminal nodes connected to the corresponding non-terminal nodes, where each $t(\wedge)$ represents a detector applied to the video part associated with $\wedge$. $\mathcal{E}$ is a set of edges of $\mathcal{G}$. A parse graph, pg, is a valid, subgraph instance of $\mathcal{G}$. $\mathcal{P}$ is the family of pdf's characterizing $\mathcal{G}$.

$\mathcal{G}$ associates activity classes with $\wedge$ nodes, which are hierarchically organized in levels $l = 1, ..., L$ to encode their compositional relations. The root is at level $l = 1$. A parent of $\wedge_l$ is denoted as $\wedge_{l-}$. Similarly, the ith child of $\wedge_l$ is denoted as $\wedge_{il+}$. The hierarchical structure of $\mathcal{G}$ means that activity classes $\wedge_l$ are connected with directed *compositional* edges to their children sub-activities $\{\wedge_{il+}\}$. $\mathcal{G}$ uses $\vee_l$ nodes as switches that provide alternative, hierarchical definitions of the activities. As shown in Fig. 1, at a particular level l of $\mathcal{G}$, nodes $\vee_l$ are connected with *switching* edges to nodes $\wedge_l$, and nodes $\wedge_l$ are connected with *relational* edges to other nodes $\wedge_l$.

$\mathcal{G}$ also encodes temporal constraints between activities, as illustrated in Fig. 1. The video can be partitioned into a number of time intervals $\tau = 1, ..., T$, where τ_+ denotes the interval that follows after τ. We associate distinct subsets of nodes $\vee_l^\tau$, $\wedge_l^\tau$, and $t(\wedge_l^\tau)$ with time intervals τ. Temporal relations are encoded by two types of edges. Nodes $\vee_l^\tau$ are connected with *temporal switching* edges to nodes $\wedge_l^{\tau+}$ to express alternative temporal sequences of activities, similar to the compositional switching edges. Also, *temporal prediction* edges link nodes $\wedge_l^\tau$ and $\wedge_l^{\tau+}$.

The distribution of a parse graph of $\mathcal{G}$ is defined as $p(\text{pg}) = \frac{1}{Z}\exp(-E(\text{pg}))$, where Z is the partition function, and the total energy $E(\text{pg})$ is defined as:

$$\begin{aligned} E(\text{pg}) = -\textstyle\sum_{l,\tau} \Big[&\textstyle\sum_{\text{switch edges}} \log p(\wedge_l^\tau | \vee_l^\tau) \\ &+ \textstyle\sum_{\text{compositional edges}} \log p(X(\wedge_l^\tau) | X(\wedge_{l-}^\tau)) \\ &+ \textstyle\sum_{\text{relational edges}} \log p(X(\wedge_{il+}^\tau), X(\wedge_{jl+}^\tau)) \\ &+ \textstyle\sum_{\text{temporal-switch edges}} \log p(\vee_l^{\tau+} | \wedge_l^\tau) \\ &+ \textstyle\sum_{\text{prediction edges}} \log p(X(\wedge_l^{\tau+}) | X(\wedge_l^\tau)) \Big]. \end{aligned} \qquad (2)$$

where $X(\wedge)$ denotes a feature vector of the video part associated with node $\wedge$.

6. Video Parsing

The goal of video parsing is to detect and localize all instances of a given query activity. From (2), the query uniquely identifies the level l in ST-AOG, and its parent level l^- wherein the corresponding $\mathrm{pg} = \mathrm{pg}_l$ is rooted. A subgraph pg_l^τ of pg_l, associated with time interval τ, has a single switching node $\vee_l^\tau$ which selects $\wedge_l^\tau$ representing the query activity detected in interval τ of the video. The detected activity $\wedge_l^\tau$ can be explained as a layout of N_l^τ sub-activities, $\{\wedge_{il+}^\tau : i = 1, ..., N_l^\tau\}$. Also, the detected activity $\wedge_l^\tau$ can be predicted, given a preceding detection at the same level $\wedge_l^{\tau-}$. Thus, we have $E(\mathrm{pg}_l) = \sum_\tau E(\mathrm{pg}_l^\tau)$, where $E(\mathrm{pg}_l^\tau)$ is defined as

$$\begin{aligned}
-E(\mathrm{pg}_l^\tau) = {} & \underbrace{\log p(\wedge_l^\tau|\vee_l^\tau)}_{\text{spatial switch}} + \underbrace{\alpha(t(\wedge_l^\tau))}_{\alpha_l^\tau \text{ detector}} \\
& + \underbrace{\Big[\underbrace{\alpha(t(\wedge_{l^-}^\tau))}_{\alpha_{l^-}^\tau \text{ detector}} + \underbrace{\log p(X(\wedge_l^\tau)|X(\wedge_{l^-}^\tau))}_{\gamma_l^\tau \text{ spatial prediction}}\Big]}_{\text{zoom-out to the parent activity}} \\
& + \underbrace{p(N_l^\tau)\sum_{i=1}^{N_l^\tau}\Big[\underbrace{\alpha(t(\wedge_{il+}^\tau))}_{\alpha_{il+}^\tau \text{ detectors}} + \underbrace{\log p(X(\wedge_{il+}^\tau)|X(\wedge_l^\tau))}_{\gamma_{il+}^\tau \text{ spatial prediction}}\Big]}_{\text{zoom-in to the parts}} \\
& + \underbrace{p(N_l^\tau)\sum_{i,j=1}^{N_l^\tau}\underbrace{\log p(X(\wedge_{il+}^\tau), X(\wedge_{jl+}^\tau))}_{\beta_{ijl+}^\tau \text{ spatial relations}}}_{\text{zoom-in to the parts}} \\
& + \underbrace{\underbrace{\log p(\vee_l^\tau|\wedge_l^{\tau-})}_{\text{temporal switch}} + \underbrace{\log p(X(\wedge_l^\tau)|X(\wedge_l^{\tau-}))}_{\omega_l^\tau \text{ temporal prediction}}}_{\text{look-ahead in time}} .
\end{aligned} \tag{3}$$

The equation (3) specifies the four computational processes involved in inference of pg_l^τ – namely, α_l^τ, β_l^τ, γ_l^τ, ω_l^τ, illustrated in Fig. 1.

From (3), for each type of query, our inference first identifies the root node of pg. Then, it executes the maximum expected-utility inference sequence $(a_1, \ldots, a_b, \ldots, a_B)$, learned by MCTS, where B is the maximum time budget. Every inference action a_b, represents a triplet {(process, detector, time interval)}, where the process is one of $\{\alpha_l^\tau, \beta_l^\tau, \gamma_l^\tau, \omega_l^\tau : l = 1, 2, 3, \tau = 1, ..., T\}$, indexed by the time interval τ, and the detector is one from the set of detectors associated with the process.

7. Parse Graph Distributions and Learning

Parameters of the distributions, given by (3), are assumed independent of time intervals. Learning of these parameters is the same as in [17, 14, 4].

Learning spatial switching: $p(\wedge_l|\vee_l)$ is learned as the frequency of occurrence of pairs $(\wedge_l, \vee_l)$ in training parse graphs. The prior of the number of children nodes $p(N_l)$ is the exponential distribution, learned on the numbers of corresponding children nodes of $\wedge_l$ in training parse graphs.

Learning α: Positive examples $T_{\alpha_l}^+$ are labeled bounding boxes around group activities ($l = 1$), or individual actions ($l = 2$), or objects ($l = 3$). Negative examples $T_{\alpha_l}^-\}$ are random boxes. α_l uses Deformable Parts Model (DPM) of [Felzenszwalb et al. PAMI10], learned on $\{T_{\alpha_l}^+, T_{\alpha_l}^-\}$.

Learning γ: Training set T_{γ_l} consists of pairs of descriptor vectors, $\{(X(\wedge_l), X(\wedge_{l^-}))\}$, extracted from bounding boxes around individual actions (or objects), and their contextual group activities (or individual actions) occurring in the training videos. The descriptors $X(\cdot)$ capture the relative location, orientation, and scale of the corresponding pairs of training instances. T_{γ_l} is used for the ML learning of the mean and covariance, $(\mu_{\gamma_l}, \Sigma_{\gamma_l})$, of the Gaussian distribution $p(X(\wedge_l)|X(\wedge_{l^-}))$.

Learning β: Training set T_{β_l} consists of all pairs of descriptors, $\{(X(\wedge_{il}), X(\wedge_{jl}))\}$, of bounding boxes of individual actions (or objects) having the same parent. The descriptors capture the relative location, orientation, and scale of the corresponding pairs of training instances. T_{β_l} is used for the ML learning of the mean and covariance, $(\mu_{\beta_l}, \Sigma_{\beta_l})$ of the Gaussian distribution $p(X(\wedge_{il}), X(\wedge_{jl}))$.

Learning ω: Training set T_{ω_l} consists of pairs of descriptors $\{(X(\wedge_l^\tau), X(\wedge_l^{\tau+})) : \tau = 1, 2, ...\}$, of bounding boxes at intervals τ around group activities, or individual actions, or objects occurring in the training videos. The descriptors capture the relative location, orientation, and scale of the corresponding pairs of training instances. T_{ω_l} is used for the ML learning of the mean and covariance, $(\mu_{\omega_l}, \Sigma_{\gamma_l})$, of the stationary Gaussian distribution $p(X({\wedge_l^\tau}^+)|X(\wedge_l^\tau))$.

8. Implementation Details

Grid of Blocks: Each video is partitioned into a grid of 2D+t blocks, allowing inference action sequences to select optimal blocks for video parsing. We evaluate the effect of the varying grid size on our performance, for the total number of blocks $\{1, 5, 10, 15, 20, 25, 50\}$. Best results are obtained for 25 blocks.

Detectors: For each level l of ST-AOG, we define a set of α^l activity detectors. We specify three different types of detectors. All detectors have access to the DPM detector, and a multiclass SVM classifier for detecting a person's facing direction. The person detector is initially applied to each frame using the standard window scanning. A person's facing direction is classified by an 8-class classifier, learned by LibSVM on HOGs (the 5-fold cross-validation precision of orientation is 69%).

For detecting objects: We train the DPM on bounding boxes of object instances annotated in training videos, and apply this detector in a vicinity of every people detection. For each object detection, we use SVM to identity the object's orientation. We obtain the tracklets using [15].

For detecting individual actions: We apply the motion-appearance based detector of [13] in a vicinity of every person detection. From a given window enclosing a person detection, we first extract motion-based STIP features, and describe them with HOG descriptors. Then, we extract KLT tracks of Harris corners, and quantize the motion vectors along the track to obtain a descriptor called the Sequence Code Map. The descriptors of STIPs and KLT tracks are probabilistically fused into a relative location probability table (RLPT), which captures the spatial and temporal relationships between the features. Such a hybrid descriptor is then classified by a multiclass SVM to detect the individual actions of interest.

For detecting group activities: We compute the STV (Space-Time Volume) descriptors of [7] in a vicinity of every person detection, called an anchor. STV counts people, and their poses, locations, and velocities, in space-time bins surrounding the anchor. Each STV is oriented along the anchor's facing direction. STVs calculated per frame are concatenated to capture the temporal evolution of the activities. Since the sequence of STVs captures a spatial variation over time, the relative motion and displacement of each person in a group is also encoded. Tracking STVs across consecutive frames is performed in 2.5D scene coordinates. This makes detecting group activities robust to perspective and viewpoint changes. The tracks of STVs are then classified by a multiclass SVM to detect the group activities of interest.

9. Results

Datasets: For evaluation, we use datasets with multiple co-occurring individual actions and group activities, such as the UCLA Courtyard Dataset [4], Collective Activity Dataset [7], and New Collective Activity Dataset [6]. These datasets have multiple layers of annotations in terms of group activities, individual actions, and objects. They test our performance on a collective behavior of individuals under realistic conditions, including background clutter, and transient occlusions. Other existing benchmark datasets are not suitable for our evaluation. Major issues include: (1) unnatural, acted activities in constrained scenes; (2) limited spatial and temporal coverage; (3) limited resolution; (4) poor diversity of activity classes (particularly for multi-object events); (5) lack of concurrent events; (6) lack of detailed annotations; and (7) primarily aimed at evaluating video classification rather than video parsing.

UCLA Courtyard Dataset [4] consists of a 106-minute, 30 fps, 2560×1920-resolution video footage. A bounding box is annotated with the orientation and pose, where we use 4 orientation classes for groups, 8 orientations for people, and 7 poses for people. The following annotations are provided, 6 group activities, 10 individual actions, and 17 objects. For each group activity or individual action, the dataset contains 20 instances, and for each object the dataset contains 50 instances. We split the dataset 50-50% for training and testing.

Collective Activity Dataset [7] consists of 75 short videos of crossing, waiting, queuing, walking, talking, running, and dancing. For training and testing, we use the standard split of $2/3$ and $1/3$ of the videos from each class. The dataset provides labels of every 10th frame, in terms of bounding boxes around people performing the activity, their pose, and activity class. Recently [6] released a new collective activity dataset which has interactions.

New Collective Activity Dataset [6] is composed of 32 video clips with 6 collective activities, with 9 interactions, and 3 individual actions. The annotations include 8 poses. We use the same setup for splitting the training and testing for testing. The dataset is divided into 3 subsets and run 3-fold training and testing.

For each type of query, our inference identifies the root node of pg, then executes the associated inference sequence $(a_1, \ldots, a_b, \ldots, a_B)$, where every inference action a_b, represents a (process, detector, time interval) triplet. The process is one of $\{\alpha_l^\mathcal{T}, \beta_l^\mathcal{T}, \gamma_l^\mathcal{T}, \omega_l^\mathcal{T}\}$, the detector is one from the set of detectors associated with the process, described in Sec. 8.

Variants: We evaluate three variants of our model for different time budgets B. $B = \infty$ means that we run full inference with unlimited number of inference steps. **V1**(B) is our default ST-AOG specified in Sec. 5. Inference of V1(B) has time budget B and accounts for ω process at all semantic levels. **V2**(B) is a variant of our ST-AOG, whose inference accounts for ω process only at the query level. The comparison of V1(B) and V2(B) tests the performance gain of tracking jointly groups and individuals versus either groups or individuals. This variant is similar to the temporal AOG of [14], and thus V2(B) may be used for comparison with [14]. A direct comparison with [14] is not possible, because their temporal AOG models human intentions, which is beyond our scope. **V3**(B) is a variant of our AOG that does not model temporal relations, and thus becomes a spatial AOG (S-AOG). Consequently, its inference does not account for ω process. This variant is similar to S-AOG presented in [4]. The comparison of V3(B) and S-AOG of [4] tests the performance gain of using MCTS versus Q-Learning for learning the inference policy.

We compare our activity recognition with that of the state of the art [4, 6, 9], and our tracklet association accuracy with that of [6]. For the UCLA Courtyard dataset, performance is evaluated in terms of precision and false positive rate of per-frame activity recognition. For the Collective Activities and New Collective Activities datasets, performance is evaluated in terms of video classification accuracy. Our tracking evaluation uses the same setup as presented in [6]. We use the same metric – namely, Match Error Correction Rate (MECR), (error in tracklet $-$ error in result)$/$error in tracklet), which counts the number of errors made during data association in tracking. MECR can effectively capture the amount of fragmentation and identity switches in tracking. In the case of a false alarm tracklet, any association with this track is considered to be an error.

Experiments: Tables 1–2 show the results of the three variants of our approach on the UCLA Courtyard dataset. Table 3 shows our performance on the Collective Activities dataset. Tables 4–5 show our performance on the New Collective Activity dataset. From the tables, we derive the following empirical conclusions.

The comparison of $\mathrm{V3}(B)$ and S-AOG of [4] in tables 1–2 demonstrates that the use of MCTS significantly improves per-frame activity recognition, and reduces the over all computational time, since we operate per blocks of video rather than the entire video.

When time budget $B = \infty$, our approach achieves the best results in Tables 1–2, since it is able to run as many inference steps as needed. From Tables 1–2 we see that as B decreases, our performance gracefully downgrades. For a sufficiently large B our performance is very close to that obtained for $B = \infty$.

The comparison of $\mathrm{V2}(B)$ and $\mathrm{V3}(B)$, and the comparison of $\mathrm{V2}(B)$ with recent work of [4, 9, 6] in Tables 1–2 and Tables 3–4 demonstrate that accounting for temporal relations between activities across the video improves performance. From the four tables, the comparison of $\mathrm{V1}(B)$ and $\mathrm{V2}(B)$ demonstrates that accounting for temporal relations at all semantic levels leads to performance gains relative to the approach that only tracks individuals.

As can be seen in Table 5, the comparison of our MCER tracking accuracy with that of [6] demonstrates that we are able to correct more trajectories than their approach.

Qualitative Results: Fig. 1 shows the results of the different variants of our approach on UCLA dataset. As you can see, starting from the top to bottom, the number of false detections was reduced by using $\mathbf{V3}(\infty)$. Then, tracking the root node by using $\mathbf{V2}(\infty)$ further reduces false detections. At the bottom of Fig. 1, $\mathbf{V1}(\infty)$ tracks activities at multiple levels and achieves the best performance.

10. Conclusion

To address challenges of parsing complex videos, we have used a stochastic grammar, called spatiotemporal And-Or-Graph (ST-AOG). In our setting, common approaches to AOG inference (e.g., using dynamic programming) may be prohibitively expensive, since video parsing requires running a multitude of object and activity detectors in the long video footage. To address this issue, we have formulated inference of ST-AOG as open-loop planning, which optimally schedules inference steps to be run until the allowed time budget. For every query type, our inference executes a maximum utility sequence of inference processes. These optimal inference sequences are learned using Monte Carlo Tree Search (MCTS). MCTS efficiently estimates the expected utility of inference steps by using an empirical average over a set of training data. MCTS accounts for higher-order dependences of inference steps, and thus alleviates drawbacks of Q-Learning and Markov Decision Process used in related work for inference. Our results demonstrate that the MCTS-based scheduling of video parsing gives similar accuracy levels under two-magnitude speed-ups relative to the standard cost-insensitive inference with unlimited time budgets. Also, the extended expressiveness of ST-AOG relative to existing activity representations leads to our superior performance on the benchmark datasets, including the UCLA Courtyard, Collective Activities, and New Collective Activities datasets.

Acknowledgements

This research has been sponsored in part by grants DARPA MSEE FA 8650-11-1-7149, ONR MURI N00014-10-1-0933, and NSF IIS 1018490.

References

[1] J. Aggarwal and M. Ryoo. Human activity analysis: A review. *ACM Comput. Surv.*, 43:16:1–16:43, 2011.

[2] M. Amer and S. Todorovic. A Chains model for localizing group activities in videos. In *ICCV*, 2011.

[3] M. Amer and S. Todorovic. Sum-product networks for modeling activities with stochastic structure. In *CVPR*, 2012.

[4] M. Amer, D. Xie, M. Zhao, S. Todorovic, and S.-C. Zhu. Cost-sensitive top-down/bottom-up inference for multiscale activity recognition. In *ECCV*, 2012.

[5] C. Browne, E. J. Powley, D. Whitehouse, S. M. Lucas, P. I. Cowling, P. Rohlfshagen, S. Tavener, D. Perez, S. Samothrakis, and S. Colton. A survey of monte carlo tree search methods. *IEEE Trans. Comput. Intellig. and AI in Games*, 4(1):1–43, 2012.

[6] W. Choi and S. Savarese. A unified framework for multi-target tracking and collective activity recognition. In *ECCV*, 2012.

[7] W. Choi, K. Shahid, and S. Savarese. What are they doing? : Collective activity classification using spatio-temporal relationship among people. In *ICCV*, 2009.

[8] A. Gupta, P. Srinivasan, J. Shi, and L. Davis. Understanding videos, constructing plots learning a visually grounded storyline model from annotated videos. In *CVPR*, 2009.

Variant	Line	Tour	Disc.	Sit	Walk	Wait	Avg	**Time**
V1(5), Precision	75.1	77.2	76.2	81.4	80.1	73.2	77.2	15
V1(5), FP	4.2	4.9	9.8	11.2	8.3	10.1	8.0	15
V2(5), Precision	73.2	76.1	74.9	78.3	76.1	68.3	74.5	15
V2(5), FP	6.8	4.6	10.4	12.2	9.8	11.0	9.1	15
V3(5), Precision	68.9	71.2	72.8	73.2	75.6	61.3	70.5	15
V3(5), FP	9.2	5.1	11.9	13.6	11.2	12.6	10.6	15
QL(5), Precision	64.1	65.4	68.3	66.5	69.8	63.1	66.2	25
QL(5), FP	8.6	3.1	9.2	9.9	8.3	10.5	8.26	25
V1(15), Precision	77.8	80.2	78.9	84.4	85.1	76.5	80.4	35
V1(15), FP	8.2	4.9	11.8	13.2	10.3	12.1	10.0	35
V2(15), Precision	75.8	79.6	75.6	81.3	77.1	70.2	76.6	35
V2(15), FP	9.6	5.4	12.5	14.1	11.8	12.9	11.0	35
V3(15), Precision	71.2	71.8	73.5	73.9	76.3	67.0	72.2	35
V3(5), FP	10.2	5.9	12.8	14.6	12.5	13.7	11.6	35
QL(15), Precision	65.4	66.1	69.0	68.7	70.3	66.5	67.6	75
QL(15), FP	10.1	4.7	11.1	11.1	8.7	10.9	9.4	75
V1(∞), Precision	80.4	83.5	81.5	87.2	88.6	80.1	83.7	170
V1(∞),FP	8.9	5.7	12.4	14.2	11.1	12.9	10.9	170
V2(∞), Precision	77.4	82.2	77.2	84.2	79.3	72.9	78.8	170
V2(∞), FP	9.9	6.3	12.8	14.6	12.5	13.3	11.6	170
V3(∞), Precision	74.8	73.5	77.1	75.8	80.1	71.0	75.4	170
V3(∞), FP	11.2	6.4	13.1	14.8	13.0	13.9	12.1	170
QL(∞), Precision	68.0	70.2	75.1	71.4	78.6	72.6	72.7	230
QL(∞), FP	13.6	10.3	17.1	13.7	10.1	12.2	12.8	230

Table 1. Average precision and false positive rates on the UCLA Courtyard Dataset for group activities. The larger the allowed number of actions, the better precision. Results are shown in %, and time is in seconds.

Class	**V1**(∞) Prec.	**V1**(∞) FP	**V2**(∞) Prec.	**V2**(∞) FP	**V3**(∞) Prec..	**V3**(∞) FP	**QL**(∞) Prec.	**QL**(∞) FP
Walk	80.0	17.1	76.9	17.5	73.2	18.1	69.1	18.7
Wait	78.4	18.8	74.1	20.1	69.2	20.4	67.7	20.2
Talk	76.8	16.6	74.8	17.3	72.9	17.5	69.6	17.9
Drive	82.1	8.1	78.3	9.0	75.4	9.6	70.2	9.7
Surf	79.8	15.4	75.2	16.1	73.1	17.0	71.3	17.1
Scoot	81.8	14.1	76.8	14.8	73.3	15.9	68.4	16.3
Bike	76.9	12.2	71.5	13.0	68.2	13.2	61.4	12.3
Read	79.6	10.1	75.6	11.2	73.9	11.8	67.3	12.1
Eat	82.3	6.5	77.4	7.1	74.1	7.5	71.3	7.7
Sit	75.5	8.1	71.2	8.8	68.3	9.3	64.2	9.0
Avg	79.3	12.7	75.1	13.5	72.2	14.0	68.0	14.1
Time	210	210	210	210	210	210	330	330

Table 2. Average precision, and false positive rates on the UCLA Courtyard Dataset for individual actions. The larger the allowed number of actions, the better precision. Results are shown in %, and time is in seconds.

Class	**V3**(∞)	[4]	[2]	[12]	[7]	**V1**(∞)	**V2**(∞)	[6]	[9]
Walk	78.1	74.7	72.2	80	57.9	83.4	79.3	65.1	61.5
Cross	79.4	77.2	69.9	68	55.4	81.1	80.0	61.3	67.2
Queue	95.3	95.4	96.8	76	63.3	97.5	96.3	95.4	81.1
Wait	81.5	78.3	74.1	69	64.6	83.9	82.4	82.9	56.8
Talk	98.1	98.4	99.8	99	83.6	98.8	98.4	94.9	93.3
Avg	86.5	84.8	82.5	78.4	64.9	88.9	87.2	80	72
Time	120	165	55	N/A	N/A	180	150	N/A	N/A

Table 3. Average classification accuracy, and running times on the Collective Activity Dataset [7]. We use $B = \infty$. Results are shown in %, and time is in seconds.

Class	**V3**(∞)	**QL**(∞)	[7]	**V1**(∞)	**V2**(∞)	[6]
Gathering	48.1	44.2	50.0	48.9	42.8	43.5
Talking	81.3	76.9	72.2	86.5	82.4	82.2
Dismissal	55.3	50.1	49.2	84.1	81.2	77.0
Walking	89.1	84.3	83.2	92.5	89.9	87.4
Chasing	95.9	91.2	95.2	96.5	95.3	91.9
Queuing	96.7	92.2	95.9	97.2	96.1	93.4
Avg	77.7	74.8	77.4	84.2	80.1	79.2
Time	130s	150s	N/A	180s	170s	N/A

Table 4. Average classification accuracy, and running times on the New Collective Activity Dataset [6]. We use $B = \infty$. Results are shown in %, and time is in seconds.

Dataset	Match	Linear [6]	Quadratic [6]	**V2** (∞)	**V1**(∞)
Collective Activity [7]	28.73	37.40	42.54	46.9	48.7
New Collective Activity [6]	81.79	82.28	82.78	83.1	83.9

Table 5. Tracking results on [7] and [6]. All results are shown in %.

[9] S. Khamis, V. I. Morariu, and L. S. Davis. Combining per-frame and per-track cues for multi-person action recognition. In *ECCV*, 2012.

[10] K. M. Kitani, B. D. Ziebart, J. A. Bagnell, and M. Hebert. Activity forecasting. In *ECCV*, 2012.

[11] L. Kocsis and C. Szepesvari. Bandit based monte carlo planning. In *ECML*, 2006.

[12] T. Lan, Y. Wang, W. Yang, S. Robinovitch, and G. Mori. Discriminative latent models for recognizing contextual group activities. *TPAMI*, 34(8):1549–1562, 2012.

[13] P. Matikainen, M. Hebert, and R. Sukthankar. Representing pairwise spatial and temporal relations for action recognition. In *ECCV*, 2010.

[14] M. Pei, Y. Jia, and S.-C. Zhu. Parsing video events with goal inference and intent prediction. In *ICCV*, 2011.

[15] H. Pirsiavash, D. Ramanan, and C. Fowlkes. Globally-optimal greedy algorithms for tracking a variable number of objects. In *CVPR*, 2011.

[16] Z. Si, M. Pei, B. Yao, and S.-C. Zhu. Unsupervised learning of event AND-OR grammar and semantics from video. In *ICCV*, 2011.

[17] T. Wu and S.-C. Zhu. A numerical study of the bottom-up and top-down inference processes in and-or graphs. *IJCV*, 93:226–252, June 2011.

[18] S.-C. Zhu and D. Mumford. A stochastic grammar of images. *Found. Trends. Comput. Graph. Vis.*, 2(4):259–362, 2006.