Multi-State Discriminative Video Segment Selection for Complex Event Classification

Prithviraj Banerjee and Ram Nevatia

University of Southern California, Los Angeles, USA

Abstract. Recognizing long range complex events composed of a sequence of primitive actions is a challenging task, as videos may not be consistently aligned in time with respect to the primitive actions across video examples. Moreover, there can exist arbitrary long intervals between, and within the execution of primitive actions. We propose a novel multistate segment selection algorithm, which pools features from the discriminative segments of a video. We present an efficient linear programming based solution, and introduce novel linear constraints to enforce temporal ordering between segments from different states. We also propose a regularized version of our algorithm, which automatically determines the optimum number of segments to be selected in each video. Furthermore, we present a new, provably faster $O(N \log N)$ algorithm for the single state K-segment selection problem. Our results are validated on the Composite Cooking Activity dataset, containing videos of cooking recipes. [1]

1 Introduction

Human activity recognition is a fundamental problem in computer vision. The difficulty of the task varies greatly based on the complexity of the actions themselves, and the imaging conditions. It is useful to characterize the activities as falling into three broad classes. Firstly, is the class of relatively short period activities consisting of primitive actions such as shake, hug and wave [1], and snippets of sport activities like diving, golf-swing [2] *etc.* A second category of activities consist of complex events naturally described as a composition of simpler and shorter term primitive actions; examples include videos of cooking recipes and assembly instructions, which take place in a structured environment consisting of fixed camera and consistent background structures, such as kitchen tops and shelves. Lastly, there is the category of *videos in the wild*, such as amateur video uploads on YouTube, where there is significant variety in camera pose and challenging background environments.

The focus of this work lies on the second category of activities. Even though the three categories face common challenges caused by ambiguities inherent to

[1] We thank Marcus Rohrbach for his help in providing the MP2 Cooking Composite Activities dataset. This research was supported, in part, by the Office of Naval Research under grant #N00014-13-1-0493.

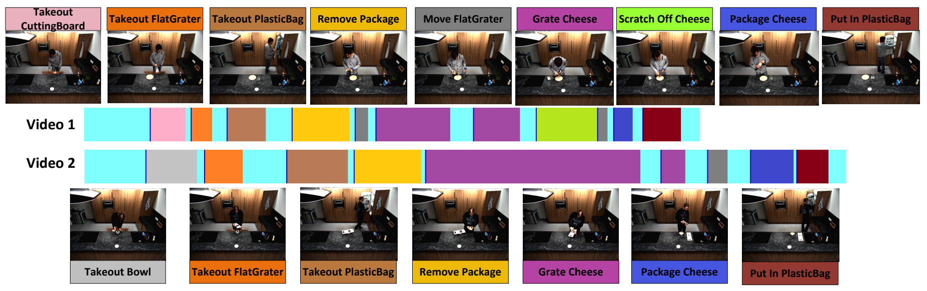


Fig. 1: Composite event *Grating Cheese* is composed of numerous primitive actions which occur with varying durations and arbitrary gaps between them, making it a challenge to learn a composite event classifier.

images and variations in the actor clothing, style and background variations, there are also significant differences. For the shorter-term activities, local features such as STIPs or Dense Trajectory (DT) features aggregated by methods such as Bag of Words [3, 4] or Fisher Vectors [5] have been found to give good performance. However, such methods are not adequate for the more complex events due to the much larger variations possible in such activities. Consider the example of *grating cheese* illustrated in figure 1, where the relatively straightforward task of retrieving cheese from the refrigerator, unwrapping it, grating it and replacing the cheese back, is performed in varying styles by two different actors. Some actors retrieve a bowl (or a tray) before (or after) retrieving the grater, resulting in only a loose temporal ordering between the primitive states of the activity, making it difficult to estimate a fixed temporal distribution of the states in a video sequence. Moreover, there are significant variations in the duration of each state due to different speeds and styles of the actors, and optional actions like *moving the grater* may cause large gaps between the essential states of the event. The task becomes even more challenging due to gaps appearing within the execution of a state, as actors tend to take breaks during long actions, like *grating cheese*, and hence violating the usual assumption of temporal continuity with respect to primitive action executions. However, we note that certain long range temporal relationships are not violated, such as retrieving the cheese before grating it, and a composite event recognizer should learn such relationships.

Furthermore, such videos may contain many intervals that are not of direct relevance to the activity of interest, for example, resting in between parts of the task. Some of these variations may be captured by spatio-temporal feature pooling strategies [3, 6], where the video is divided into spatio-temporal grids of histograms. However, rigid grid quantizations are not likely to be sufficient for un-aligned and un-cropped videos [4], and are sensitive to temporal variations in action executions. There has been work representing such activities by dynamical graphical models such as Dynamic Bayesian Networks [7, 8]; however, these representations can be highly sensitive to varying time intervals and spurious intervening activities.

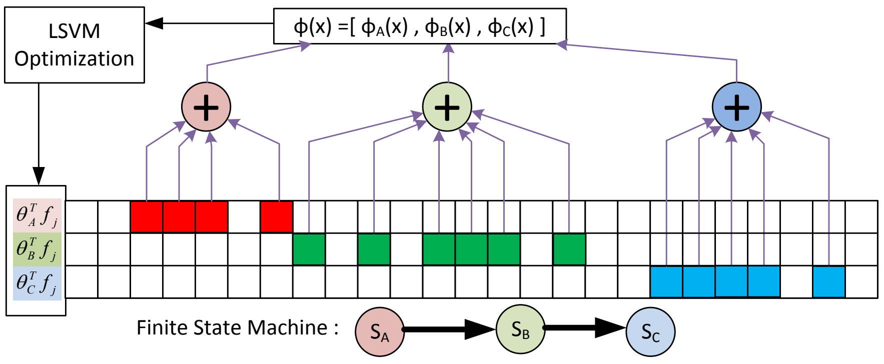


Fig. 2: Flow diagram for multistate dynamic feature pooling algorithm. A 3-state finite state machine is assumed, which specifies a temporal ordering as $S_A < S_B < S_C$. Segment classification scores are computed *w.r.t* each state, and the discriminative segments are selected by solving a linear programe. The aggregate feature statistics from the selected segments are pooled to compute a global feature statistic $\Phi(x)$. which is used for training a Latent-SVM classifier, and hence learn the optimal activity classification weights θ. The weight learning and feature poolling is repeated iteratively till convergence.

We suggest that dynamic pooling strategies offer a suitable compromise between static pooling and the rigid dynamical models. Dynamic pooling has been used in previous works, such as [2, 9, 10], which identify the discriminative video segments to pool features from, and therefore adapt the pooling strategy to the observed features in the video. However, such methods assume restrictive constraints on the temporal positioning of primitive actions, and do not allow for arbitrary length inter-action gaps, and furthermore restrict the hypothesis space for primitive actions to continuous video segments only.

Our work is closest in spirit to [11], which proposed a dynamic pooling algorithm that treats the locations of characteristic segments of the video as latent variables which are inferred for each video sequence in a LSVM framework. The possible set of segment selections is combinatorial in size, and they propose a fractional integer linear programming based solution to obtain the exact solution. However, [11] does not distinguish between the primitive actions in a video during pooling, and incorporates only weak pair-wise temporal associations by considering an exhaustive set of segment pairs during both training and testing phase, making it unsuitable for long-term complex event recognition.

We are motivated to extend the segment selection algorithm [11] to incorporate learning of temporal relationships between primitive actions, and propose a novel temporal segment selection algorithm for complex event recognition. We formulate the task of multi-state dynamic feature pooling as a linear programming optimization problem, where the discriminative weights for feature classification are learned iteratively in a latent support vector machine (LSVM). Figure 2 presents an overview of our approach while assuming a three state model. Aggregate statistics, such as histograms of spatio-temporal features are computed from overlapping segments of the video, and a classification score is assigned to each segment *w.r.t* each state. The discriminative segments are identified on a per-state basis, and a global feature vector Φ is pooled from the selected segments, which is used to train the LSVM classifier. Our algorithm is simultaneously (1) robust to arbitrary gaps between primitive states while respecting their temporal ordering, (2) robust to gaps within primitive states caused by varying durations of primitive action executions, and (3) selects the discriminative sub-segments of the video for classifying the composite event. Our algorithm eschews any manual annotations of the states during training, and instead automatically infers the important states from the training data.

Our key contributions are multi fold: (1) We present a new provably fast solution to the K-segment selection problem, which improves the running time of the previously proposed [11] fractional linear program based solution to $O(N \log N)$, (2) we present a regularized linear programming optimization for multi-state K-segment selection for dynamic feature pooling, and finally (3) propose a novel extension to our linear program for automatically determining the number of discriminative segments in a video, and hence avoiding pooling features from non-informative (*w.r.t* to the complex event label) video segments.

We validate our algorithm on the recently introduced Composite Cooking Dataset [12], which consists of a collection of cooking recipe demonstrations performed by multiple actors while following a weakly enforced script. Each recipe is a long term composite event, consisting of a series of primitive actions like opening cupboard, taking out toaster, peeling vegetables *etc*, and we show significantly improved results on the dataset, along with an analysis of the discriminative segments selected by our algorithm.

2 Related Work

We focus our survey on classification algorithms using different feature pooling strategies, and also briefly review algorithms modeling temporal structure of complex activities.

Static pooling algorithm model the (x,y,t) distribution of features in the 3-dimensional video volume, and fix the pooling strategy before commencing classifier training, and do not adapt dynamically to the features statistics observed during training. A popular class of pooling strategy divides the video into multiple spatio-temporal grids [3, 6], and concatenates the feature statistics computed from each of the grids into a global video-wide feature vector, while other approaches construct interest point centric histograms [13, 14, 4], where feature statistics are accumulated from static spatio-temporal grids centered around each detected interest point. However, the *static* nature of the feature pooling grids assumes that the underlying action is consistently aligned with the video, and makes it sensitive to time and duration shifts in action instances.

In real life videos, the actions of interest might occur in only small a fraction of the frames. Restricting the classifier to the important video segments

[15, 16] has been shown to improve recognition accuracy, emphasizing the importance of *dynamic* feature pooling algorithms, which only compute feature statistics from the important segments of a video, determined dynamically during training. While some algorithms ignore the temporal structure, and select only a single continuous subsequence [15, 17] of the video, others represent the video as a sequence of atomic segments [9, 2, 10] and either learn or manually define temporal relationships between the segments. A variety of atomic segment representations have been proposed such as histogram of codewords over temporal [2] and spatio-temporal volumes [10], poselet based representation of human actions [18], and scene context based video decomposition [19]. However they either ignore temporal ordering [19], or assume well cropped videos without arbitrary gaps between the atomic segments [2, 10], while [18] selects only a small number of discrete frames, and requires a manually defined set of relevant poselets, further restricting it to primitive single-human actions.

A popular class of approaches represent the complex event as a sequence of state transition of a finite state machine, where the state definitions can be semantic, such as human key-poses [7] or linear interpolations between 3D joint positions [8]. Other approaches favor non semantic state definitions based on low level feature, such as discriminative motion patches [20] and histogram of gradient/flow features [21]. The event label is inferred using either logic models [22], or generative [23] and conditional [24, 8] probabilistic graphical models like HMM and CRF. Such methods have shown robust performance on short duration activities like walking, jumping *etc.*, however due to their inherent Markovian nature, they are unable to handle long range dependencies between primitive action segments, and are sensitive to variations in duration and style of action execution. Recently, [21] proposed a conditional variable duration HMM model for action classification on youtube videos, however it does not distinguish between video segments based on relevance to the composite event, and instead attempts to model every segment as a valid state, making it susceptible to spurious features from unimportant segments.

3 Pooling Interest Point Features for Event Classification

Classical framework for event recognition using low level image features consists of a three stage process: detection, pooling and classification. The detection stage consists of computing a set of descriptors $\mathbf{x}_i = \{\boldsymbol{x}_k\}$ from spatio-temporal interest point detections in the i^{th} video. The pooling stage involves combining the multiple local feature descriptors into a single global feature $\Phi(\mathbf{x}_i)$ representation of the video. Lastly in the classification stage, discriminative classifiers like support vector machines are trained using the global features as training data.

Our contributions are in the feature pooling stage of the framework. We present a new, provably faster, algorithm to a previous segment selection algorithm by Li et al [11]. We further propose a novel algorithm for pooling features from the discriminative time intervals of a video, while modeling the temporal dynamics present in the activity in a joint optimization framework. We also present an extension to our algorithm which automatically determines the optimum number of segments to select.

3.1 Discriminative Segment Selection

We first describe the basic framework of discriminative segment selection, where the video is divided into N equal-length temporal segments. Let $\mathbf{f}_j$ be the locally pooled feature computed from the j^{th} segment, where the pooling criteria can be as simple as computing a histogram of codewords detected within the segment. The global feature descriptor is computed by pooling a subset of the segments:

$$\Phi(\mathbf{x},\mathbf{h}) = \frac{\sum_j \mathbf{f}_j h_j}{\sum_j e_j h_j} \quad , \quad \Phi^*_{\boldsymbol{\theta},\mathbf{x}} = \max_{\boldsymbol{h}} \boldsymbol{\theta}^T \Phi(\mathbf{x},\boldsymbol{h}) = \max_{\boldsymbol{h}} \frac{\sum_j \boldsymbol{\theta}^T \mathbf{f}_j h_j}{\sum_j e_j h_j} \tag{1}$$

where h_j is a binary variable indicating the selection of the j^{th} segment, and e_j is a strictly positive constant which normalizes the $\Phi(\cdot)$ with respect to the number of segments selected. A variety of representations can be chosen for $\mathbf{f}_j$ and e_j, for example, setting $\mathbf{f}_j$ as the histogram of codewords and e_j as the sum of codewords appearing in the j^{th} segment results in the classical BoW feature.

The discriminative weight vector $\boldsymbol{\theta}$ computes the score c_j corresponding to each segment, where the score is proportional to the importance of the segment in classifying the given event. The weight vector $\boldsymbol{\theta}$ is learned using a Latent-SVM optimization [25, 26]:

$$\min_{\boldsymbol{\theta}} \frac{1}{2}\|\boldsymbol{\theta}\|_2^2 + C\sum_{i\in\mathcal{P}} \max\left(1, \Phi^*_{\boldsymbol{\theta},\mathbf{x}_i}\right) + C\sum_{i\in\mathcal{N}} \max\left(0, 1+\Phi^*_{\boldsymbol{\theta},\mathbf{x}_i}\right) - C\sum_{i\in\mathcal{P}} \Phi^*_{\boldsymbol{\theta},\mathbf{x}_i} \tag{2}$$

where $\mathcal{P}$ and $\mathcal{N}$ are the set of positive and negative training examples.

3.2 $\mathcal{K}$-Segment Selection (KSS)

Inference of the latent variable $\boldsymbol{h}$ in equation 1 determines the important segments in a video, and the inference algorithm can be stated as a $\mathcal{K}$-segment selection problem, where the objective is to select the $\mathcal{K}$ most optimum segments from a video for classification purpose, and is equivalent to solving the following fractional integer linear program:

$$\text{KSS}(\mathcal{K}): \quad \text{maximize} \frac{\sum_{j=1}^N c_j h_j}{\sum_{j=1}^N e_j h_j} \quad \text{s.t.} \quad \sum_{j=1}^N h_j = \mathcal{K} \quad , \quad \forall_j h_j \in \{0,1\} \tag{3}$$

where $c_j = \boldsymbol{\theta}^T \mathbf{f}_j$ is the classification score value corresponding to the feature vector $\mathbf{f}_j$, pooled from the j^{th} segment. Li et al [11] proposed a relaxed linear programming solution to solve the above integer problem, where the fractional linear program is transformed to an equivalent standard linear program [27]. An optimal solution to the linear program is computed using the standard simplex algorithm, which has exponential worst case complexity, and on average polynomial time complexity. We next present a provably faster algorithm to solve the $\mathcal{K}$-segment selection problem.

3.3 Linear Time Subset Scanning

Our key observation is that selecting the optimum $\mathcal{K}$ segments in a video by optimizing equation 3, is equivalent to solving the Linear Time Subset Scanning (LTSS) problem [28]. We give a brief introduction to the LTSS problem, and refer the readers to [28] for the details. Let us define a subset $\mathcal{S} = \{j \ni h_j = 1\}$, and define the following two additive statistics over the subset: $X(\mathcal{S}) = \sum_{j \in \mathcal{S}} c_j$ and $Y(\mathcal{S}) = \sum_{j \in \mathcal{S}} e_j$. We further define a subset scoring function $F(\mathcal{S}) = F(X, Y) = \frac{X(\mathcal{S})}{Y(\mathcal{S})}$ according to which we want to select the best possible subset. For all scoring functions $F(\mathcal{S})$ satisfying the LTSS property, the optimal subset $\mathcal{S}$ maximizing the score can be found by ordering the elements of the set according to some *priority* function $G(j)$, and selecting the top $\mathcal{K}$ highest priority elements.

A scoring function $F(\mathcal{S})$ satisfies the LTSS property (Theorem 1 [28]) with priority function $G(j) = \frac{c_j}{e_j}$, if (1) $F(\mathcal{S}) = F(X, Y)$ is a quasi-convex function of two additive sufficient statistics of subset $\mathcal{S}$, (2) $F(\mathcal{S})$ is monotonically increasing with $X(\mathcal{S})$, and (3) all additive elements of $Y(\mathcal{S})$ are positive. In our case, the scoring function $F(\mathcal{S}) = \frac{\sum_{j \in \mathcal{S}} c_j}{\sum_{j \in \mathcal{S}} e_j} = \frac{\sum_j c_j h_j}{\sum_j e_j h_j} = F(\boldsymbol{h})$ is a ratio of linear functions in the segment selector variable vector $\boldsymbol{h}$, and hence can be shown to be quasi-convex [27] using a simple analysis of its α-sublevel sets. Monotonicity of $F(\boldsymbol{h})$ w.r.t. $X(\mathcal{S})$ is shown due to them being linearly related, and furthermore, e_j are strictly positive by design, as they represent the normalization factor for each segment. Hence, the $\mathcal{K}$-Segment Selection problem satisfies the LTSS property with priority function $G(j) = \frac{c_j}{e_j}$. To select the optimum $\mathcal{K}$ segments, we sort the set of segment scores: $\left\{\frac{c_j}{e_j}\right\}_{j=1}^{N}$, and select the segments corresponding to the top $\mathcal{K}$ scores. Therefore our algorithm reduces the $\mathcal{K}$-segment selection problem to a simple sorting problem with a time complexity of $O(N \log N)$, which is an order of magnitude faster than solving a linear program. Note, that we only require an unordered list of the top $\mathcal{K}$ segments, and hence, one can select the $\mathcal{K}^{th}$ largest element using the linear time median-of-medians selection algorithm, and select the top $\mathcal{K}$ segments through a linear traversal over all the segments, which solves the problem in $O(N)$ linear time.

4 Multistate K-Segment Selection (MKSS)

The $\mathcal{K}$-segment selection algorithm [11] does not consider the temporal relationships between the selected segments, and hence ignores the temporal ordering of primitive action sequences composing a complex event. In effect, it can be viewed as a *single state* segment selection. We are motivated to extend the $\mathcal{K}$-segment selection algorithm to a multi-state formulation, where each state corresponds to discriminative primitive actions present in the video. Let us consider a two state problem, where our objective is to select the optimum sub-segments for both state A and state B, such that state A occurs in the video before state B. This is equivalent to a finite state machine where state A transitions to state B. We define $\mathbf{h}^A, \mathbf{h}^B \in \{0,1\}^N$ as the segment selection indicator vectors corresponding to the two states. To ensure the temporal ordering in the selected subsegments, we construct the following constraint:

$$\mathcal{K} h_j^B + \sum_{t=j}^{N} h_t^A \leq \mathcal{K} \qquad 1 \leq j \leq N \tag{4}$$

The linear equation defines a mutual exclusion constraint on states A and B, such that if the j^{th} segment is assigned to state B, then all temporal successor segments appearing after j cannot be assigned to state A. Similar constraints can be placed on predecessor segments of state B when selecting the j^{th} segment for state A. Using equation 4, we can build any left to right transition finite state machine with arbitrary number of states. Note, that self transitions are implicitly modeled, as the selected segments of a particular state can be arbitrarily separated.

State duration models specify the expected time a Markovian system will spend in a particular state. Similar duration constraints can be placed on our state models by specifying a compactness constraint:

$$\mathcal{K} h_j^A + \sum_{t=1}^{j-\delta} h_t^A + \sum_{t=j+\delta}^{N} h_t^A \leq \mathcal{K} \qquad 1 \leq j \leq N \tag{5}$$

The compactness constraint specifies that all the segments selected for the state A, must lie within a temporal window of length 2δ, by placing a mutual exclusion constraint on the j^{th} segment and all other segments lying outside the 2δ window. We combine both the temporal constraints and the compactness constraints with the $\mathcal{K}$-segment selection problem, and formulate it as a relaxed linear programming optimization:

$$\begin{aligned}
\text{MKSS}\,(\mathcal{K}) = \quad & \text{Maximize} \quad \frac{\sum_{s\in\mathcal{S}} \sum_{j=1}^{N} c_j^s h_j^s}{\sum_{s\in\mathcal{S}} \sum_{j=1}^{N} e_j^s h_j^s} \quad + a \log \sum_{s\in\mathcal{S}} \|\mathbf{h}^s\|_1 && \qquad (6)\\
\text{s.t.} \quad C1: \quad & 0 \leq h_j^s \leq 1 && \forall s \in \mathcal{S}, \quad 1 \leq j \leq N \\
C2: \quad & \sum_{j=1}^{N} h_j^s = \mathcal{K} && \forall s \in \mathcal{S} \\
C3: \quad & \mathcal{K} h_j^{s_b} + \sum_{k=j}^{N} h_k^{s_a} \leq \mathcal{K} && \forall (s_a, s_b) \in \mathcal{T}, \quad 1 \leq j \leq N \\
& \mathcal{K} h_j^{s_a} + \sum_{k=1}^{j} h_k^{s_b} \leq \mathcal{K} && \forall (s_a, s_b) \in \mathcal{T}, \quad 1 \leq j \leq N \\
C4: \quad & \sum_{s\in\mathcal{S}} h_j^s \leq 1 && 1 \leq j \leq N \\
C5: \quad & \mathcal{K} h_j^s + \sum_{k=1}^{j-\delta} h_k^s + \sum_{k=j+\delta}^{N} h_k^s \leq \mathcal{K} && \forall s \in \mathcal{S}, \quad 1 \leq j \leq N
\end{aligned}$$

where $\mathcal{S}$ is the set of states in our model, and $(s_a, s_b) \in \mathcal{T}$ is the set of temporal order constraints where state s_b appears only after state s_a. C1 is the linear relaxation constraint over the binary indicator variables. C2 specifies that exactly $\mathcal{K}$ segments should be selected corresponding to each state. C3 and C5 correspond to the temporal and compactness constraints respectively. C4 ensures that no segment is counted twice during state assignments. The resulting optimization is a fractional linear program, where the objective function is a ratio of linear functions. Solving fractional linear programs is a well explored problem in operations research, and can be solved using a simple transformation [27] to an equivalent standard linear program:

$$\begin{aligned}&\text{Maximize } \tfrac{\mathbf{c}^T\mathbf{h}+d}{\mathbf{e}^T\mathbf{h}+f}\\ &\quad G\mathbf{h} \preceq \mathbf{m}\\ &\quad A\mathbf{h} = \mathbf{b}\\ &\quad \mathbf{e}^T\mathbf{h}+f > 0\end{aligned} \quad \Longleftrightarrow \quad \begin{aligned}&\text{Maximize } \mathbf{c}^T\mathbf{y}+dz\\ &\quad G\mathbf{y}-\mathbf{m}z \preceq 0\\ &\quad A\mathbf{y}-\mathbf{b}z = 0\\ &\quad \mathbf{e}^T\mathbf{y}+fz = 1, \quad z \geq 0\end{aligned}$$

where $\mathbf{y} = \frac{\mathbf{h}}{\mathbf{e}^T\mathbf{h}+f}$ and $z = \frac{1}{\mathbf{e}^T\mathbf{h}+f}$. The standard linear program can be efficiently solved using off the shelf solvers[2], as it consists of $O(N|\mathcal{S}|)$ variables and $O(N|\mathcal{S}|^2)$ constraints, which is polynomial in the number of segments and states.

The optimal $\mathbf{h}^s$ vector obtained by solving the linear program in equation 6 identifies the selected segments for pooling features corresponding to each state $s \in \mathcal{S}$. The global feature descriptor $\Phi(\mathbf{x})$ is defined as a concatenation of features pooled from the individual states. Assuming a 3-state model such as in figure 2, the global feature vector is computed as: $\Phi(\mathbf{x}) = [\Phi_A, \Phi_B, \Phi_C] = \left[\frac{\sum_j \mathbf{f}_j h_j^A}{\sum_j e_j h_j^A}, \frac{\sum_j \mathbf{f}_j h_j^B}{\sum_j e_j h_j^B}, \frac{\sum_j \mathbf{f}_j h_j^C}{\sum_j e_j h_j^C}\right]$, which is used to train an latent-SVM classifier.

5 Selecting Optimal Parameter $\mathcal{K}$

The parameter $\mathcal{K}$ in the $\mathcal{K}$-segment selection problem specifies the number of segments to be selected. However the appropriate $\mathcal{K}$ value can vary from video to video, and there is little intuition on how to compute an appropriate value. One feasible criteria is to select the best $\mathcal{K}$ value by iteratively solving the $\mathcal{K}$-segment selection problem : $\mathcal{K}^* = \arg\max_{\mathcal{K}} \text{KSS}(\mathcal{K})$. It can be shown that the optimum value of such an iterative procedure will always be $\mathcal{K}^* = 1$, *i.e.* only selecting the segment with the largest $\frac{c_i}{d_i}$ ratio. Consider the following inequality: $\frac{a_2}{b_2} \leq \frac{a_2+a_1}{b_2+b_1} \leq \frac{a_1}{b_1}$, which can be verified for all $b_1, b_2 \geq 0$ using simple algebraic operations. The inequality shows that any combination of multiple segments will always have a lower ratio value than the single segment with the highest $\frac{c_i}{d_i}$ ratio. A similar theoretical argument cannot be made for MKSS, however in our experiments we observe that the optimum solution is for each state to select a single best segment.

Previously, [11] addressed the problem by adding a logarithmic regularization function : $a\log(\|h\|_1)$, which favors choosing a larger number of segments. However choosing appropriate values of the hyper-parameter a is again non-trivial, and it is estimated through cross-validation for each action category. In effect, the regularization parameter a indirectly chooses the appropriate $\mathcal{K}^*$ value, and is equivalent to selecting $\mathcal{K}^*$ through cross-validation. We next present an extension to our multistate segment selection algorithm for automatically selecting the optimum number of segments.

[2] http://www.gnu.org/software/glpk/

5.1 Regularized Multistate Segment Selection (RMSS)

The segment selection criteria used in KSS and MKSS are such that the negatively scored segments will never be selected. On the other hand, the segments contributing a positively weighted score corresponds to the discriminative (*w.r.t* classifying the composite event) segments in the video, and hence an appropriate segment selection criteria should maximize the number of positively weighted segments selected while satisfying the multi-state constraints. We define a vector $r_j^{s_a} = 0.5 \times \left(c_j^{s_a} + |c_j^{s_a}|\right)$ which contains all the positive valued scores computed from the segments for state s_a, while the negative valued scores are set to zero. We further define a segment selection constraint $\sum_{j=1}^{N} r_j^{s_a} h_j^{s_a} \geq \Delta \sum_{j=1}^{N} r_j^{s_a}$ which ensures that at least a Δ fraction of the positively weighted segments will be selected as part of the optimum solution. The linear programming optimization for regularized multistate segment selection takes only the positive weight fraction Δ as input parameter, and is defined as follows:

$$\mathrm{RMSS}\,(\Delta) = \text{Maximize} \quad \frac{\sum_{s\in\mathcal{S}} \sum_{j=1}^{N} c_j^s h_j^s}{\sum_{s\in\mathcal{S}} \sum_{j=1}^{N} e_j^s h_j^s} \qquad , \quad \tilde{K} = \left\lfloor \frac{N}{\|\mathcal{S}\|} \right\rfloor \tag{7}$$

$$\begin{aligned}
\text{s.t.}\quad C1: &\quad 0 \leq h_j^s \leq 1 && \forall s \in \mathcal{S}, \quad 1 \leq j \leq N \\
C2b: &\quad \sum_{j=1}^{N} h_j^s \leq \tilde{K} && \forall s \in \mathcal{S} \\
C3: &\quad \tilde{K} h_j^{s_b} + \sum_{k=j}^{N} h_k^{s_a} \leq \tilde{K} && \forall (s_a, s_b) \in \mathcal{T}, \quad 1 \leq j \leq N \\
&\quad \tilde{K} h_j^{s_a} + \sum_{k=1}^{j} h_k^{s_b} \leq \tilde{K} && \forall (s_a, s_b) \in \mathcal{T}, \quad 1 \leq j \leq N \\
C4: &\quad \sum_{s\in\mathcal{S}} h_j^s \leq 1 && 1 \leq j \leq N \\
C5: &\quad \tilde{K} h_j^s + \sum_{k=1}^{j-\delta} h_k^s + \sum_{k=j+\delta}^{N} h_k^s \leq \tilde{K} && \forall s \in \mathcal{S}, \quad 1 \leq j \leq N \\
C6: &\quad \sum_{j=1}^{N} r_j^{s_a} h_j^{s_a} \geq \Delta \sum_{j=1}^{N} r_j^{s_a} && \forall s \in \mathcal{S} \\
&\quad r_j^{s_a} = 0.5 \times (c_j^{s_a} + |c_j^{s_a}|) && \forall s \in \mathcal{S}, \quad 1 \leq j \leq N \\
C7: &\quad \sum_{j=1}^{N} h_j^{s_a} \geq (1-\theta) \sum_{j=1}^{N} h_j^{s_b} && \forall s_a, s_b \in \mathcal{S} \\
&\quad \sum_{j=1}^{N} h_j^{s_a} \leq (1+\theta) \sum_{j=1}^{N} h_j^{s_b} && \forall s_a, s_b \in \mathcal{S}
\end{aligned}$$

where the parameter $\tilde{K}$ is the maximum number of segments which can be selected per state in a video with N frames. The optimization does not restrict each state to a constant $\tilde{K}$ number of segment selections; instead it relaxes the equality constraint C2 with C2b in equation 7, which only places an upper bound on the number of segments selected. It is also desirable that the number of segments selected corresponding to each state is equally balanced across states, so that a single state does not dominate the solution of segment selection. An additional constraint C7 is added to ensure a balanced selection of segments across states, within a margin of $\pm\theta$.

The parameter Δ determines the number of segments selected in the optimum solution. However the constraints in the $RMSS(\Delta)$ optimization can be rendered infeasible for certain values of Δ, in particular, if the Δ value is too high, it is likely that the state transition constraints cannot be satisfied for any combination of segment selection. We further observe that there exists a $\Delta_0 \geq 0$ such that $RMSS(\Delta)$ has a feasible solution for all $\Delta \geq \Delta_0$, and hence the optimization problem is monotonic in Δ with respect to its feasibility. The monotonic behavior suggests a simple binary search over Δ to find the optimal Δ_0 within an error margin of ϵ in $O\left(\log \frac{1}{\epsilon}\right)$ iterations.

6 Experiments and Results

We evaluate our algorithm on the recently introduced Composite Cooking Dataset [12]. The dataset contains 41 cooking recipe demonstrations like *prepare ginger*, *seperate an egg*, *make coffee* etc, where the videos are recorded with a fixed elevated camera recording the actors from the front preparing the dishes inside a kitchen. There are a total of 138 videos of ~16 hours containing actions performed by 17 different actors, and are shot at 29.4fps with 1624x1224 pixel resolution. In our experiments, we use the pre-computed histogram of codeword features for each frame, provided with the dataset. The codewords are computed over HoG, HoF, motion boundary histograms and trajectory shape features, extracted from densely sampled interest point tracks [29] in the videos.

We divide each video into overlapping segments of 100 frames each, as primitive events like opening cupboard/refrigerator, retrieving utensils can be reasonably captured using a 100 frame overlapping window. Next, we sum the histogram of codewords from each frame within the j^{th} segment, to construct a single accumulated histogram feature $\boldsymbol{f}_j$. To setup the fractional linear programming problems MKSS and RMSS, we normalize the features from each segment using its L1 norm, and compute the scores values $c_j^s = \theta_s \boldsymbol{f}_j$ for each state s using the current value of the weight vector from the LSVM classifier. The normalization constants e_j^s are set to 1, and in effect, normalize the features based on the number of segments selected. In our experiments, we avoided any action or dataset specific tunning and set the regularization parameter as $a = 5$ for all events, which we empirically observed to select a larger fraction of positively scored segments in the video, and hence contributing more towards classifying the composite event.

6.1 Comparisons with baselines

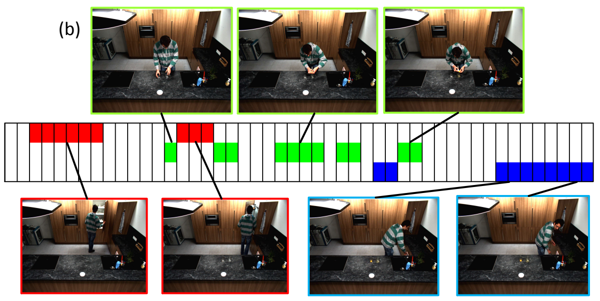
To establish a baseline, we implemented a bag of words based SVM classifier, where the codewords in the video are globally pooled into a single histogram. Figure 3 presents our results on the 41 composite cooking actions using a 6-fold cross validation as suggested by [12]. As the BoW features compute only globally aggregated statistics, their performance is quite low on complex long range activities. We also implement a temporally binned BoW classifier, as the one proposed by [3]. We experiment with three types of binning structures, where

Action Labels	BoW	TB-3	TB-5	TB-7	MKSS-3	MKSS-5	MKSS-7	RMSS-3	RMSS-5	RMSS-7
Chopping a cucumber	0.11	0.14	0.11	0.15	0.12	0.15	0.11	0.12	0.16	0.12
Prepare carrots	0.19	0.33	0.39	0.42	0.39	0.39	0.45	0.38	0.39	0.42
Prepare a peach	0.23	0.11	0.20	0.09	0.10	0.13	0.08	0.17	0.13	0.11
Slice a loaf of bread	0.58	0.61	0.73	0.66	0.75	0.71	0.87	0.77	0.72	0.85
Prepare cauliflower	0.40	0.43	0.42	0.36	0.60	0.65	0.35	0.34	0.63	0.33
Prepare an onion	0.16	0.46	0.26	0.10	0.42	0.19	0.10	0.35	0.20	0.09
Prepare an orange	0.14	0.68	0.47	0.38	0.30	0.42	0.36	0.61	0.44	0.37
Prepare fresh herbs	0.38	0.24	0.14	0.18	0.24	0.20	0.22	0.22	0.22	0.18
Prepare garlic	0.18	0.31	0.12	0.07	0.31	0.32	0.14	0.31	0.16	0.30
Prepare asparagus	0.02	0.03	0.04	0.04	0.03	0.04	0.05	0.04	0.05	0.05
Prepare fresh ginger	0.14	0.37	0.23	0.09	0.12	0.19	0.08	0.14	0.21	0.07
Prepare a plum	0.41	0.18	0.11	0.09	0.36	0.18	0.22	0.61	0.21	0.61
Zest a lemon	0.14	0.20	0.20	0.20	0.20	0.20	0.20	0.25	0.17	0.20
Prepare leeks	0.23	0.33	0.32	0.46	0.42	0.34	0.43	0.34	0.33	0.38
Extract lime juice	0.42	0.48	0.49	0.53	0.44	0.49	0.50	0.46	0.47	0.43
Prepare a pomegranate	0.39	0.81	0.94	0.56	0.53	0.65	0.78	0.48	0.70	0.78
Prepare broccoli	0.20	0.40	0.42	0.47	0.45	0.45	0.46	0.45	0.45	0.60
Prepare potatoes	0.23	0.11	0.15	0.11	0.23	0.22	0.14	0.24	0.23	0.17
Prepare a pepper	0.10	0.16	0.08	0.09	0.14	0.11	0.10	0.18	0.11	0.12
Prepare a pineapple	0.55	0.37	0.48	0.51	0.56	0.74	0.61	0.73	0.77	0.62
Prepare spinach	0.10	0.28	0.31	0.36	0.58	0.58	0.58	0.50	0.28	0.44
Prepare a fresh chilli	0.23	0.05	0.05	0.06	0.09	0.14	0.20	0.10	0.14	0.20
Cook pasta	0.26	0.53	0.45	0.54	0.38	0.49	0.64	0.53	0.47	1.00
Separate an egg	0.65	0.47	0.60	0.63	0.52	0.63	0.57	0.63	0.63	0.56
Prepare broad beans	0.14	0.68	0.18	0.29	0.68	0.51	0.68	0.47	0.52	0.52
Prepare a kiwi fruit	0.15	0.23	0.23	0.10	0.11	0.18	0.11	0.22	0.10	0.12
Prepare an avocado	0.13	0.07	0.13	0.07	0.05	0.08	0.07	0.05	0.10	0.07
Prepare a mango	0.06	0.16	0.30	0.22	0.15	0.29	0.32	0.21	0.29	0.33
Prepare figs	0.30	0.06	0.07	0.07	0.09	0.13	0.22	0.10	0.14	0.21
Use box grater	0.42	0.75	0.49	0.39	0.65	0.63	0.57	0.73	0.71	0.57
Sharpen knives	0.75	0.63	1.00	0.75	0.67	0.67	0.75	0.75	1.00	1.00
Use speed peeler	1.00	0.10	0.10	0.10	0.20	0.33	0.25	0.20	0.20	0.20
Use a toaster	0.16	0.57	0.39	0.35	0.42	0.43	0.31	0.52	0.34	0.20
Use a pestle-mortar	0.55	0.55	0.59	0.65	0.63	0.60	0.63	0.65	0.60	0.50
Use microplane grater	0.20	0.49	0.23	0.20	0.18	0.26	0.29	0.21	0.25	0.27
Make scrambled egg	0.22	0.45	0.57	0.56	0.50	0.65	0.61	0.53	0.78	0.72
Prepare orange juice	0.64	0.65	0.81	0.69	0.81	0.83	0.83	0.81	0.83	0.78
Make hot dog	0.05	0.30	0.18	0.59	0.21	0.21	0.44	0.40	0.40	0.44
Pour beer	0.56	0.10	0.06	0.03	0.56	0.53	0.53	0.53	0.55	1.00
Make tea	0.28	0.46	0.33	0.35	0.53	0.51	0.61	0.50	0.56	0.75
Make coffee	0.53	0.88	0.75	0.88	0.71	0.75	0.71	0.75	0.75	0.71
Splitwise Average MAP	0.30	0.40	0.36	0.35	0.39	0.41	0.41	0.42	0.41	0.41

Fig. 3: MAP result table for the Composite Cooking dataset. Column TB presents the MAP scores from a temporally binned BoW classifier [3] with 3, 5 and 7 temporal partitions. MAP scores for MKSS and RMSS algorithm for different number of states (3, 5 and 7) is also given for each action. The highest MAP score across states is highlighted.

the video is divided into 3, 5 and 7 equal length partitions, and histogram of codewords computed from each partition are concatenated together. We observe considerable improvement in MAP values compared to the BoW classifier, which we attribute to the temporal pooling of features which is important for complex event detection. However, the performance starts decreasing as the number of partitions is increased, which we attribute to the static nature of the partitions, making them sensitive to variations in the temporal location of primitive actions.

(a) Methodology	MAP
Bag of Words SVM	30.19%
SVM-MeanSGD [12]	32.30%
K-Segment Selection [11]	31.30%
Temporal Binning T-3 [3]	40.30%
Temporal Binning T-5 [3]	36.60%
Temporal Binning T-7 [3]	34.70%
Temporal Binning Best [3]	41.74%
MKSS : 3 states	39.40%
MKSS : 5 states	41.20%
MKSS : 7 states	41.00%
MKSS Best	43.57%
RMSS : 3 states	42.30%
RMSS : 5 states	41.00%
RMSS : 7 states	41.40%
RMSS Best	47.80%



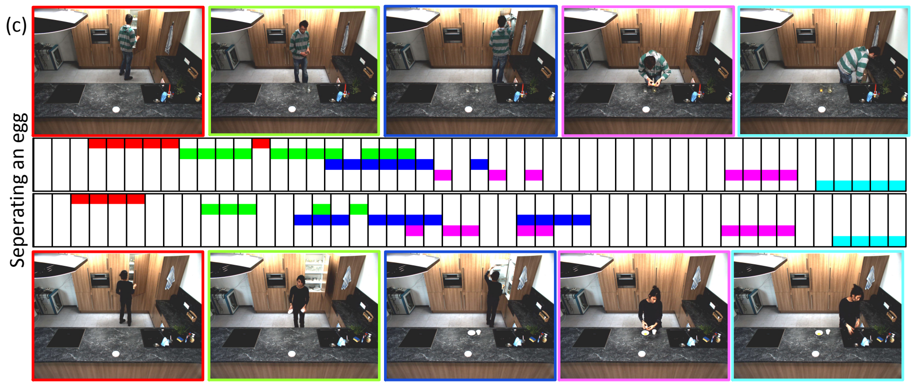


Fig. 4: (a) MAP result table. (b) 3-State segment selection result for *Seperating an egg* with representative frames of the selected segments. (c) 5-State segment selection result with comparisons of the frames selected by each state.

We next evaluate both the MKSS and RMSS algorithms for three different number of states: 3, 5 and 7. The optimal number of states is a function of the complexity of the underlying event in the video, and also the amount of variety present across videos of the same event class. Figure 4(a) shows the average MAP over all classes of the MKSS and RMSS algorithm for the different number of states, and also the average of the best performance. The MKSS and RMSS algorithms achieve on average an MAP score of 41.47% and 47.80% respectively. Our method outperforms the SVM-MeanSGD [12] algorithm, which learns a SVM classifier using chi-square kernels and reports a score of 32.30%. We note, that [12] also reports an MAP score of 53.9% using external textual scripts to guide the classifier training, however our focus is on purely computer vision based approaches, and expect our algorithm to also benefit from similar complimentary modalities. We also implemented the $\mathcal{K}$-segment selection [11] algorithm and evaluated it on the dataset. We observe only a minor improvement in performance over traditional BoWs, which we attribute to its lack of temporal structure modeling, which is crucial for classifying long term composite events.

6.2 Segment Selection results

Solving the MKSS and RMSS algorithms provides us with the optimum segment selection indicator vector $\boldsymbol{h}$, whose elements are real valued numbers between 0 and 1. For visualizing the segments assigned higher selection weights, we threshold the indicator values $\boldsymbol{h}$ such that atleast 40% of $\|\boldsymbol{h}\|_1$ is retained. Figure 4 (b) shows the results of the multistate segment selection algorithm for 3 states on a *seperate an egg* video. We note the clear decomposition of the selected segments into three temporal states, where state-A (red) appears before state-B (green), which is followed by state-C (blue). The states are learned automatically from the training data, and it is difficult to associate a single primitive action with each state. However, we can discern some interesting trends through visual inspection of the results. For example, state-B seems to correspond to working at the counter station, and as the video contains extended periods at the counter, our model only selects some parts of the video. State-A seems to correspond to moving to back of the room and opening a door, and is detected twice in the video where the person approaches the cupboard and the refrigerator. The intervening frames are not important to state-A and are ignored in its score computation. State-C seems to correspond to the person moving to the side to wash the dishes, or to keep them away.

Figure 4 (c,d) compares the segment selection applied to two different videos of the: *separate an egg* activity, where the algorithm assumes a 5-state model, and we see a correlation between the types of primitive actions each of states correspond to across the videos. We note that each state can represent a cluster of features, and hence may correspond to multiple primitive actions and scenes. This becomes more apparent in videos where the actor interchanges the actions of approaching the refrigerator and approaching the cupboard. As our states do not have a semantic understanding of what a refrigerator or a cupboard is, it only recognizes the gross spatio-temporal motions occurring in the scene.

7 Conclusions

We presented a novel multistate segment selection algorithm for pooling features from the discriminative segments of a video. We presented a solution based on efficiently solving a linear programming optimization, and formulate linear constraints to enforce temporal ordering among the states representing the primitive actions of an event. We also presented a provably faster solution to the single state K-segment selection problem [11] and improve the computation time to $O(N \log N)$. Finally, we presented a regularized version of the multistate segment selection algorithm, which automatically determines the number of segments to be selected for each state in a given video. We evaluated our algorithm on the Composite Cooking Activity dataset [12], and showed significantly improved results compared to other static and dynamic pooling algorithms. One promising approach for future work is to extend the algorithm to incorporate semantic mappings between the states and the underlying feature distributions, and explore automated methods of determining the optimal number of states.

References

1. Ryoo, M., Aggarwal, J.: Spatio-temporal relationship match: Video structure comparison for recognition of complex human activities. In: ICCV. (2009)
2. Niebles, J.C., Chen, C.w., Fei-fei, L.: Modeling Temporal Structure of Decomposable Motion Segments for Activity Classification. In: ECCV. (2010)
3. Laptev, I., Marszalek, M., Schmid, C., Rozenfeld, B.: Learning realistic human actions from movies. In: CVPR. (2008)
4. Kovashka, A., Grauman, K.: Learning a Hierarchy of Discriminative Space-Time Neighborhood Features for Human Action Recognition. In: CVPR. (2010)
5. Oneata, D., Verbeek, J., Schmid, C.: Action and event recognition with Fisher vectors on a compact feature set. In: ICCV. (2013)
6. Sun, J., Wu, X., Yan, S., Cheong, L., Chua, T., Li, J.: Hierarchical spatio-temporal context modeling for action recognition. In: CVPR. (2009)
7. Lv, F., Nevatia, R.: Single view human action recognition using key pose matching & viterbi path searching. In: CVPR. (2007)
8. Natarajan, P., Singh, V., Nevatia, R.: Learning 3D Action Models from a few 2D videos. In: CVPR. (2010)
9. Gaidon, A.: Actom sequence models for efficient action detection. In: CVPR. (2011)
10. Tian, Y., Sukthankar, R., Shah, M.: Spatiotemporal Deformable Part Models for Action Detection. In: CVPR. (2013)
11. Li, W., Yu, Q., Divakaran, A.: Dynamic Pooling for Complex Event Recognition. In: ICCV. (2013)
12. Rohrbach, M., Regneri, M., Andriluka, M.: Script data for attribute-based recognition of composite activities. In: ECCV. (2012)
13. Fathi, A., Mori, G.: Action recognition by learning mid-level motion features. In: CVPR, Ieee (2008)
14. Gilbert, A., Illingworth, J., Bowden, R.: Fast realistic multi-action recognition using mined dense spatio-temporal features. In: ICCV. (2009)
15. Schindler, K., Van Gool, L.: Action Snippets: How many frames does human action recognition require? In: CVPR. (2008)
16. Satkin, S., Hebert, M.: Modeling the Temporal Extent of Actions. In: ECCV. (2010)
17. Nowozin, S., Bakir, G., Tsuda, K.: Discriminative Subsequence Mining for Action Classification. In: CVPR, Ieee (2007)
18. Raptis, M., Sigal, L.: Poselet Key-framing: A Model for Human Activity Recognition. In: CVPR. (2013)
19. Vahdat, A., Cannons, K., Mori, G., Oh, S., Kim, I.: Compositional Models for Video Event Detection: A Multiple Kernel Learning Latent Variable Approach. In: ICCV. (2013)
20. Wang, Y., Mori, G.: Hidden Part Models for Human Action Recognition: Probabilistic vs. Max-Margin. PAMI (2010)
21. Tang, K., Fei-Fei, L., Koller, D.: Learning latent temporal structure for complex event detection. In: CVPR, Ieee (2012) 1250–1257
22. Brendel, W., Fern, A., Todorovic, S.: Probabilistic event logic for interval-based event recognition. In: CVPR. Number I (2011) 3329–3336
23. Duong, T., Bui, H., Phung, D., Venkatesh, S.: Activity recognition and abnormality detection with the switching hidden semi-markov model. In: CVPR. (2005)

24. Sminchisescu, Kanaujia, A., Li, Dimitris, M.: Conditional models for contextual human motion recognition. In: ICCV. (2005)
25. Yu, C.N.J., Joachims, T.: Learning structural SVMs with latent variables. In: ICML. (2009)
26. Felzenszwalb, P., McAllester, D.: A discriminatively trained, multiscale, deformable part model. In: CVPR. (2008)
27. Boyd, S., Vandenberghe, L.: Convex Optimization (2009)
28. Neill, D.: Fast subset scan for spatial pattern detection. Journal of the Royal Statistical Society: Series B (. . . **74** (2012) 337–360
29. Wang, H., Klaser, A.: Action recognition by dense trajectories. In: CVPR. (2011) 3169–3176