Learning Convolutional Action Primitives from Multimodal Timeseries Data*

Colin Lea[1], René Vidal[2], and Gregory D. Hager[1]

***Abstract*— Fine-grained action recognition is important for many applications such as human-robot interaction, automated skill assessment, and surveillance. The goal is to predict what action is occurring at any point in a timeseries sequence. While recent work has improved recognition performance in robotics applications, these methods often require hand-crafted features or large use of domain knowledge. Furthermore these methods tend to model actions using pointwise estimates of individual frames or statistics on collections of frames. In this work we develop a notion of an action primitive that models how generic features transition over the course of an action. Our Latent Convolutional Skip Chain Conditional Random Field (LC-SC-CRF) model learns a set of interpretable and composable action primitives. We apply our model to the cooking and robotic surgery domains using the University of Dundee 50 Salads dataset and the JHU-ISI Gesture and Skill Assessment Working Set (JIGSAWS). Each consists of multimodal timeseries data including video, robot kinematics, and/or kitchen object accelerations. Our recognition performance on 50 Salads and JIGSAWS are 18.0% and 5.3% higher than the state of the art. Our model has the advantage of being more general than many recent approaches and performs well without requiring hand-crafted features or intricate domain knowledge. Upon publication we will release our LC-SC-CRF code and the features used on the two datasets.**

I. Introduction

Fine-grained action recognition is important for many applications like human-robot interaction, automated skill assessment of complex tasks, and surveillance. These systems have the potential to change the way people interact with the world and with each other by automatically recognizing and evaluating human actions. The focus of this work is to predict a sequence of actions given video, robot kinematics, and/or other sensor information. For example, in a cooking task, the activity `make_sandwich` might start with the action *put_bread_on_plate*, transition into *add_mustard_to_bread*, then *place_meat_on_bread*, and finish with *place_bread_on_sandwich*.

There are many subtleties to fine-grained action recognition that make it a difficult problem. There is often large variability in how an action is performed, how long it takes to perform that action, and in what order actions take place. In the cooking example different users may add different ingredients to the sandwich. Some actions may require a specific order while others may have several common variations.

This work was supported in part by grants NRI-1227277, NSF-1218709, NSF-1335035 and ONR-N00014131011

[1]CL and GDH are with the Department of Computer Science and [2] RV is with the Department of Biomedical Engineering, Johns Hopkins University, 3400 N. Charles, Baltimore, MD, USA. Emails: `clea1@jhu.edu`, `rvidal@cis.jhu.edu`, `hager@cs.jhu.edu`

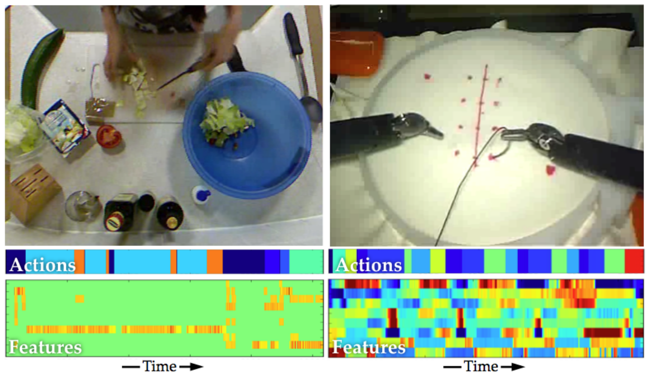

Fig. 1. (left) A sequence from the 50 Salads dataset (right) A sequence from the Suturing task in the JHU-ISI Gesture and Skill Assessment Working Set (JIGSAWS). Directly below each photo is the sequence of actions performed throughout each video. The bottom depicts the stream of timeseries features where each row is a different feature type.

In efforts to model how the scene changes over time we introduce a notion of an action primitive that is composed of convolutional filters. These non-linearly model how the features (e.g. positions, object state) transition through an action. Our action primitives are motivated by recent work in deep learning where similar filters are learned in a Convolutional Neural Network (CNN) to perform tasks like object classification. One important difference here is that our filters are more interpretable because we learn them per action instead of among all actions.

We introduce the Latent Convolutional Skip-Chain Conditional Random Field (LC-SC-CRF) for joint temporal segmentation and action classification. This is a generalization of the Skip Chain Conditional Random Field (SC-CRF) of Lea et al. [6] which achieves high performance on surgical action recognition. Our variation models multiple variations on each action. For example the action *peeling* may now consist of three action primitives: *pick_up_peeler*, *peel_cucumber*, and *put_down_peeler*.

Lastly, we suggest the use of two evaluation metrics that are important for practical applications of action recognition. The first is a modified overlap score that gives a high score for correctly predicting the temporal segmentation. The second is a segmental edit score which gives a high score for predicting the correct order of actions in a sequence.

We target applications in cooking and robotic surgery and evaluate on two datasets. For cooking we use the University of Dundee 50 Salads dataset to recognize actions, such as *cutting*, *mixing*, *peeling*, performed while making a salad. This dataset includes data from an RGBD sensor and a set of accelerometers located on kitchen utensils. For robotic surgery we evaluate on the JHU-ISI Gesture and Skill Assessment Working Set (JIGSAWS) which contains videos and robot kinematic data of users performing training tasks like suturing, needle passing, and knot tying. These were collected from a DaVinci surgical robot and are also used for automated skill assessment.

We make the following contributions:

- we model a set of interpretable action primitives using convolutional filters
- we develop the Latent Convolutional Skip Chain CRF
- we suggest two important evaluation metrics for practical robotics applications.

Our LC-SC-CRF code will be released as an opensource library called Struct.jl.[1]. We will also release the features used on 50 Salads and JIGSAWS.

II. Related Work

Some recently applications of fine-grained action recognition are in the domains of cooking [7], [12], [14], [15], robotic surgery, human-robot assembly [20], [3], household tasks [5], [4], and sports [8], [11],

Cooking: Ramirez-Amaro et al. [12] look at two cooking tasks in a setup where they can easily detect relevant objects and the hands of a user. They claim that complicated temporal models are unnecessary in situations where the features are highly accurate. They use semantic reasoning (via a decision tree) with simple predicates like *hand_moving* and *object_in_hand* to classify fine-grained cooking actions. While their claim may be true in specific applications, contrary to our approach, their method may not work well in domains where there is high variability in features. Lei et al. [7] approach a similiar problem to ours except that they assume known temporal segmentation. They recognize a set of cooking tasks like "'making a cake" by modeling the objects and user's hand trajectories. Their approach is based on histograms of trajectory features within each action segment. Stein and McKenna [14] introduce the 50 Salads dataset. They derive a set of features based on object locations and object use and set a baseline using standard frame-wise models like Naive Bayes and Random Decision Forests.

Surgical action recognition: recent models include variations on HMMs [16], [19], [13], Linear Dynamic Systems (LDS) [19], and Linear Chain CRFs [17], [6]. These HMMs and CRFs assume that each action can be classified solely using the features at an individual frame. The Switching LDS [19] partitions the data into chunks (e.g. 15 frames), fits an LDS this chunk, and transitions between states. The Switching LDS is similar in spirit to the our action primitives, however, our convolutional filters can learn nonlinear changes within an action. Tao et al. [17] propose a Markov Semi-Markov model for doing joint segmentation and action recognition. They infer action segments as a function of many frames. However the segments are modeled simply using the mean of the features within a segment.

Human Robot Assembly: Vo and Bobick [20] introduce the Sequential Interval Network for timeseries prediction of human-robot interaction applications. They develop a segmental model that requires a user-defined task plan. Their model takes the task model in the form of a Context Free Grammar and converts it into a Dynamic Bayes Network for predicting the start, stop, and action type of each segment. They apply it to a toy assembly task dataset and in [3] apply it to human robotic interaction. Their features are based on detected hand positions and a set of known bin locations at the start and end of each action and the duration of a segment.

Other areas: Koppula and Saxena [5] propose a CRF model that captures the relationships among objects and between actions and objects for activities of daily living. These interactions are defined between different temporal segments which are generated by sampling and evaluating many different temporal and object relationship graph structures. Hu et al. [4] work in this same domain but uses a latent Conditional Random Field where nodes correspond to the data, actions, and action primitives. They perform inference in their loopy graph efficiently by collapsing the latent variable to make the graph a linear chain. Yezhou et al [**?**], [**?**] learn how objects change as a consequence of an action. For example, in a cutting task they model the transition from a whole cucumber to two halves as a change from one segment to two. While this work provides interesting insights into action-object relationships, it is unclear if it would work in complicated environments.

III. Activity Model

In this section we start by introducing our action primitive representation and describe the temporal model. Then we discuss how to learn the parameters of our model using a Latent Structural Support Vector Machine and how to perform inference. Let the variable X_t be a set of features (e.g. positions, velocities) at time t for $t \in \{1, \ldots, T\}$ and Y_t be the corresponding action (e.g. cutting, peeling). Let y be any label in our action set $\mathcal{Y}$.

A. Action Primitives

Our goal is to learn a representation for an action that describes how the world changes over the course of an action. Here we describe three models: a common pointwise model, an action primitive, and a set of latent action primitives. These are displayed in figure 2.

Common timeseries models (e.g. HMMs, CRFs) typically use pointwise estimates of an action. We say this is pointwise because it only captures a statistic on the data; it does not

[1]Code will be available here before publication: `https://github.com/colincsl/struct.jl`

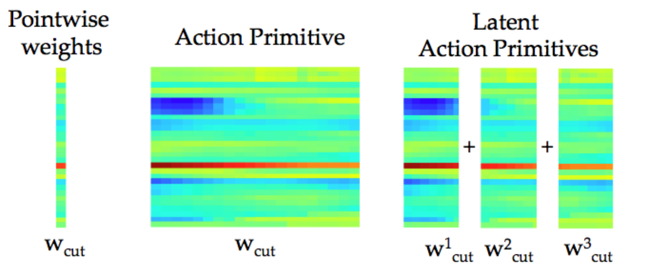

Fig. 2. Action primitives for the class *cutting* (left) traditional weight vector applied to a single frame (middle) our convolutional action primitives (right) our latent action primitive that models an action using multiple parts.

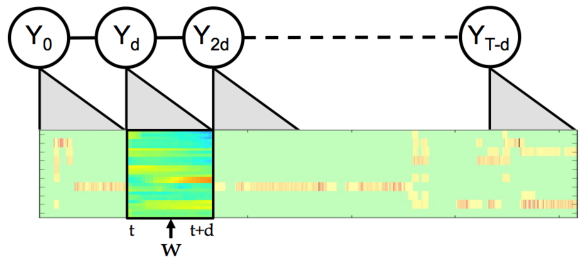

Fig. 3. Our Latent Convolutional Skip Chain CRF. We depict an example action primitive overlaid from t to $t+d$. Note that for clarity we only depict nodes for intervals 0, d, and $2d$. There are additional chains covering the rest of the timesteps.

capture how the features change over time. We compare against the most common pointwise function which is simply a linear combination of weight vector w for action y and the data at time t:

$$\phi(X, y, t) = w_y X_t \tag{1}$$

We define an action primitive as a convolutional filter which we learn. These filters nonlinearly model how features change over the course of a specific action. We define a weight matrix ("filter") w that represents a given action.[2] For each action we learn a single filter of size $F \times d$ where F is the number of features and d is the duration of the primitive. The column of each filter corresponds to the features at each timestep within an action. Ideally the duration would cover the entire length of each action. However, because actions of widely varying lengths we choose it to be around the average length of a segment. The score of our classifier is given by the following where $\star$ is the convolution operator:

$$\phi(X, y, t) = w_y \star X_{t:t+d} \tag{2}$$

This 2D convolution results in a scalar score for time t. Note for later that this convolution can be rewritten as a dot product of the vectorized filter w and data $X_{t:t+d}$.

In practice actions can last different amounts of time. For example, in a *cutting* action, one person may pause in between picking up a knife and cutting a vegetable. Thus, it may be advantageous to learn a separate model for different parts of an action such as the start, middle, and end. We now define a set of composable action primitives using latent variables which represent these temporal subsets of an action.

Let h_t be the latent state at time t. We define a new set of filters w_y^h for $h = 1 \dots H$ and each class y. These will be initialized by dividing each action into H segments and learning an initial set of parameters. We want our action score to be high regardless of which latent state we are in. The score for any hidden state is:

$$\phi(X, y, h, t) = w_y^h \star X_{t:t+d} \tag{3}$$

Our energy function will maximize over the best scoring filters h_t. In practice we find that the optimal number of hidden states H is less than 5 for our applications.

[2]Here we define the weights and data in terms of matrices to help build intuition for the reader. For learning these terms are vectorized.

In Section IV we compare the traditional, action primitive, and latent action primitive models.

B. Temporal Model

The LC-SC-CRF adds a cost for transitioning from action-to-action and a cost for an action that occurs at specific points in time. For a depiction of our model see figure 3.

Given a sequence of data X and labels Y we define an energy $E(X, Y)$ as

$$E(X, Y) = \max_h \sum_{t=1}^{T} \phi(X, Y, h, t) + \psi(Y, h, t) + \pi(Y, t) \tag{4}$$

where ϕ, ψ, and π are the scores for action primitives, pairwise skip-frame actions, and temporal priors respectively. In our experiments we evaluate using each variation of ϕ. The pairwise and prior terms are described below.

Note that in a probabilistic setting this model can be viewed as a Latent Conditional Random Field using the Gibbs distribution $P(Y|X) = \frac{1}{Z}\exp(-E(X, Y))$ with partition function Z. Sometimes this type of model is also simply referred to as a Latent Structural Support Vector Machine.

1) Pairwise Skip-frame Score: The pairwise skip-frame term is a generalization of the common Markov class transition potential. In activity data, actions do not change frequently and thus the typical Markov assumption biases the predictions to stay in the same class from time $t-1$ to t. Instead we model the class transitions from time $t-d$ to t. Using this model the probability of an action changing from class a to class b between timesteps is much higher. Parameter d is called the skip length and is chosen via cross validation. Empirically this has shown to have a large effect on accuracy.

The skip-frame term is modeled as a function where index (y_{t-d}, y_t) is the corresponding skip transition weight and 0 everywhere else. The score is

$$\psi(y, y') = w_{y,y'} \tag{5}$$

In the latent variable case we use pairwise transitions that are a function of the latent variables corresponding to each class. For example we compute the transition from "start" to "end" variables in *cutting*.

2) Boundary Priors: In fine-grained applications there are often many different sequence orderings that all constitute the same activity. For example, if you are making a peanut butter and jelly sandwich you can spread peanut butter on bread before jelly or after. However, often there are only one or two ways to start and end a sequence. For example, you must start the sandwich making activity by picking up the bread.

To better classify these boundary actions we add a start bias π_s for what action class occurs at the beginning of a sequence and an end bias π_e for the action at the end of the sequence. These are modeled as follows where $\mathbf{1}[a]$ is 1 if a is true and 0 if it is false:

$$\pi_s(y,t) = w_y \mathbf{1}[t < d] \tag{6}$$

$$\pi_e(y,t) = w_y \mathbf{1}[T-d < t \le T] \tag{7}$$

In practice we add the priors to frames $t = 1 \dots d$ and $t = T - d \dots T$.

We also add a per-class bias at each timestep to weigh the likelihood of any class occurring:

$$\pi_c(y) = w_y \tag{8}$$

If a class occurs more frequently then it will have a higher bias. We collect all prior terms into $\pi(y,t) = \pi_s(y,t) + \pi_e(y,t) + \pi_c(y,t)$.

C. Learning

All of the aforementioned terms can be written as a linear function of w:

$$E(X,Y) = \max_h \sum_{t=1}^{T} w^\top \Psi(X,Y,h,t) \tag{9}$$

Ψ is the concatenation of all unweighted and vectorized energy terms for a given timestep. Using the convolutional action primitives this is:

$$\Psi(X,Y,h,t) = \sum_{t=1}^{T} w^\top \begin{bmatrix} X_{t:t+d} \\ \mathbf{1}[y = Y_{t-d}]\mathbf{1}[y' = Y_t] \\ \mathbf{1}[t < d] \\ \mathbf{1}[T-d < t \le T] \\ \mathbf{1} \end{bmatrix} \tag{10}$$

Recent work has shown that jointly learning parameters w of a CRF using the Structural Support Vector Machine (SSVM) objective [18] often achieves superior accuracy compared to probabilistic alternatives [21], [9]. Our action primitives contain latent variables so we use the Latent Structural Support Vector Machine [22]. This objective models the upper bound on the empirical risk of our data using the loss function Δ. We define $\Delta(Y^*, Y, h^*, h)$ as the Hamming distance given the ground truth labeling Y^* and any arbitrary labeling Y:

$$\Delta(Y^*, Y, h^*, h) = \sum_{t=1}^{T} \mathbf{1}[Y_t^* \ne Y_t] \tag{11}$$

This loss is analogous to the framewise accuracy we evaluate with later.

We minimize the LSSVM using the Convex Concave Procedure [22]. This is an alternating minimization method that alternates between updating the hidden states h_t at each timestep and then updating the weights using gradient descent. We use Stochastic Gradient Descent where the step size is computed dynamically with Adagrad [1]. Mathemtical details of the LSSVM are beyond the scope of this paper. For a recent overview on these models and methods see [10]. Note that for each term in our energy we initialize the latent weights by dividing each action into H pieces (corresponding to the H latent variables per class) and performing KMeans clustering. In the case where we have two latent states this encourages the latent parameters to fit to the start and end of each action.

D. Inference

We infer the best possible labeling $Y = \arg\max_Y E(X,Y)$ by maximizing the cost of all potentials Ψ at each frame using a slightly modified version of Viterbi decoding. During inference, we compute a score $V_{t,y}$ for each label y at all timesteps t. V is a table of size $T \times C$ where T is the total time and C is the number of classes. This is a dynamic programming problem where $V_{t,y} = \max_{y'} V_{t-d,y} + w^\top \Psi(X, y, y', t)$. We output the best labeling Y by backtracking through the table of scores to find the best scoring sequence. The difference between traditional Viterbi decoding and our algorithm is that we keep a separate table for each skip chain. We then interlace each chain to compute our final prediction. See [6] for more details. Computational complexity is on the order of $O(TC^2)$ operations and $O(TC)$ memory.

Viterbi decoding can be easily modified to run in an online fashion because each step in our energy function is simply a function of the current chunk of data and the previous action class. Instead of backtracking from T to 1 we backtrack from time t to 1 to find the best sequence so far.

1) Filtering: One caveat to using a frame-wise inference method, like Viterbi, is that it operates on a frame-by-frame basis. Coupled with our skip frame transition features we find it prone to over-segmentation; while these features provide valuable information about action transitions, they occasionally oscillate between states along what should be long contiguous segments. We experimented with sophisticated remedies but found that simply applying a median filter to the predictions works best.

IV. Evaluation

In this section we define the evaluation metrics, datasets, and experimental setups.

A. Metrics

We suggest two evaluation metrics that we find important for practical applications of action recognition. These suggestions correspond to two types of errors which are depicted in figure 4. We also compare our results with prior work using accuracy, precision, and recall.

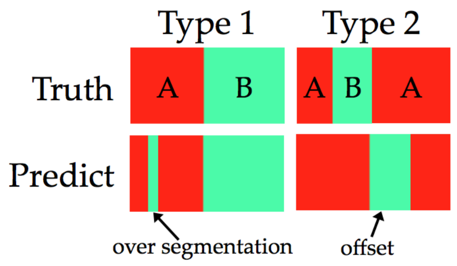

Fig. 4. Our suggested metrics are used to measure two types of errors. The first is oversegmentation, which is when there are multiple different action segments contained within what should be one contiguous segment. The second accounts for temporal offsets in our data. They offsets are sometimes caused by inter-reviewer variability and should not negatively impact our performance.

Low-level actions	Accuracy	Overlap	Edit
SC-CRF	44.04	27.69	26.0
LC-SC-CRF (H=1)	44.76	32.17	29.45
LC-SC-CRF (H=3)	**46.28**	**34.3**	**31.71**
Mid-level actions	Accuracy	Overlap	Edit
SC-CRF	51.47	32.98	20.62
LC-SC-CRF (H=1)	52.36	34.4	26.33
LC-SC-CRF (H=3)	**55.05**	**38.42**	**29.02**
High-level actions	Accuracy	Overlap	Edit
SC-CRF	92.85	60.86	57.9
LC-SC-CRF (H=1)	93.26	59.59	**64.27**
LC-SC-CRF (H=3)	**94.06**	**64.64**	63.24

TABLE II
EVALUATION ON THE 50 SALADS DATASET USING THE LOW-LEVEL, MID-LEVEL, OR HIGH-LEVEL ACTIONS DESCRIBED IN THE TEXT.

The first score measures how much overlap there is between the ground truth segments and the predicted segments. A common metric for this is the Jaccard Index. While this does a good job at measuring coverage, it does not penalize for oversegmentation errors. Predictions from models like the Skip Chain CRF will sometimes add spurious false labels within what should be a long contiguous segment. We penalize these errors by defining an overlap score that is a function of the longest contiguous predicted segment for a given ground truth segment. The score is:

$$s_o(G, P) = \frac{100}{N} \sum_{i=1}^{M} \max_j \frac{G_i \cap P_j}{G_i \cup P_j} \tag{12}$$

This score lies in $[0, 100]$ and a higher overlap score is better.

The second metric accounts for slight temporal shifts in the prediction. In many domains there is large inter-annotator variability in labeling where temporal boundaries should be located. In applications like surgical skill evaluation the ordering of the actions may matter more than the precise location of the segments. To measure this error we use a segmental edit score. For each ground truth and predicted labeling we first output the sequence of actions (independent of segmentation boundaries). We then compute a normalized edit distance, $s_e(G, P)$, using the Wagner-Fischer algorithm. To make it easier to interpret the results we show scores for $(1 - s_e(G, P)) * 100$. The best score 100 and the worst is 0.

We also use accuracy, precision, and recall to compare against prior work. These are defined using TP, FP, TN, and FN for true positives, false positives, true negatives, and false negatives.

$$\text{Accuracy} = \frac{TP + TN}{TP + FP + FN + TN} \tag{13}$$

$$\text{Precision} = \frac{TP}{TP + FP} \tag{14}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{15}$$

B. 50 Salads

50 Salads [14] is a multimodal dataset that includes time-synchronized video, depth, and accelerometer data. There are 50 five- to ten-minute samples of a user making a salad. In total there are 25 users – each of whom makes a salad in two different videos. As shown in figure 1 a static RGBD camera is mounted facing down looking at the user preparing the salad. Each kitchen utensil has an accelerometer attached that records motion. These can be used to indicate which tools is being used at any given time. In total there are 10 accelerometers. These are located on the following kitchen tools: *plate, pepper_dispenser, bowl, oil_bottle, large_spoon, dressing_glass, knife, peeler, small_spoon, chopping_board*.

This dataset contains multiple granularities of action labels. At the coarsest there are three actions: *cut_and_mix_ingredients, prepare_dressing, serve_salad.* There are 17 mid-level actions including *add_vinegar, cut_tomato, mix_dressing, peel_cucumber, place_cheese_into_bowl, serve_salad.* and others. At the low-level each mid-level action is decomposed into "start," "core," and "end." Results using the low-, mid-, and high-level action sets are shown in Table II. For all of these label sets we also add a background class for when no action is occurring.

Following the work of [14] we also evaluate using 10 labels: *add_dressing, add_oil, add_pepper, cut, mix_dressing, mix_ingredients, peel, place, serve_salad_onto_plate,* and *none*. This is a hybrid action set that combines variations on cutting and placing actions. For example cutting a cucumber and cutting a tomato belong to the same action.

In our experiments we only use the accelerometer data. We preproccess the data by using the absolute value of each signal. We evaluate using 5-fold validation where we train on 20 users (40 videos) and test on 5 users (10 videos). We use a skip length and filter length of 200 frames. Table I (left) shows results using our methods.

C. JHU-ISI Gesture and Skill Assessment Working Set (JIGSAWS)

The JIGSAWS dataset [2] has three fine-grained activities common for robotic surgery training. These activities are perfomed on benchtop phantoms and include suturing, needle passing, and knot tying. See figure 1 for an example of suturing. These activities are each decomposed into about 10 unique action primitives such as *inserting_needle_into_skin,* 50 Salads

Models	Accuracy	Overlap	Edit
Linear Chain CRF	71.54	48.40	44.82
SC-CRF (no filter)	76.61	33.05	4.77
SC-CRF (H=1)	77.47	59.42	51.92
SC-CRF (H=2)	79.23	62.26	54.25
SC-CRF (H=3)	78.83	60.12	52.6
SC-CRF (H=4)	79.23	62.39	52.86
LC-SC-CRF (H=1)	81.70	64.24	56.89
LC-SC-CRF (H=2)	81.69	64.67	56.87
LC-SC-CRF (H=3)	81.39	64.55	**58.46**
LC-SC-CRF (H=4)	**81.75**	**64.90**	58.08

JIGSAWS

Models	Accuracy	Overlap	Edit
Linear Chain CRF	74.55	77.58	62.92
SC-CRF (no filter)	78.57	74.47	21.74
SC-CRF (H=1)	79.25	86.00	71.34
SC-CRF (H=2)	82.10	87.36	75.18
SC-CRF (H=3)	81.48	87.55	74.94
SC-CRF (H=4)	82.55	88.32	72.71
LC-SC-CRF (H=1)	81.69	88.55	**78.9**1
LC-SC-CRF (H=2)	83.18	88.78	78.69
LC-SC-CRF (H=3)	82.37	86.77	77.57
LC-SC-CRF (H=4)	**83.45**	**88.88**	76.86

TABLE I

RESULTS ON 50 SALADS AND JIGSAWS. SC-CRF IS THE SKIP FRAME CRF, LC-SC-CRF IS THE LATENT CONVOLUTIONAL SKIP CHAIN CRF, $H = h$ DEFINES THE NUMBER OF LATENT VARIABLES, NO FILTER IMPLIES SMOOTHING IS NOT USED.

tying_a_knot, transferring_needle_between_tools, and *dropping_the_needle_at_the_end_point.* Each task has between 26 and 39 trials performed by up to 8 users. Videos are around two minutes long and contain 15 to 20 action primitives per video.

The data includes video and robot kinematics from a DaVinci medical robot. In this paper we use the robot kinematics and vision-based features as described in [6]. The features are: left and right tool positions, velocities, and gripper angles as well as the distance from the tools to the closest object in the scene from the video.

We evaluate on the suturing task using Leave One User Out as described in [2]. In each split we train on seven users and test on the left out user. We use a skip length and action primitive duration of 100 frames. Table I (right) shows the results using our methods.

V. DISCUSSION

Results: On the 50 Salads dataset we achieve 18.0% higher precision and 16.5% higher recall than the state of the art [14] using the LC-SC-CRF. Our precision and recall for the action primitive model with four hidden states are 79.05% and 79.23%. On 50 Salads our LC-SC-CRF without latent variables (H=1) tends to achieve about 5% better accuracy than the SC-CRF. Interestingly the latent variations tend to perform about the same for various numbers of latent variables with this model. For the SC-CRF the latent variables tend to improve performance by a few percent.

Table II depicts our best results on each label set provided in the 50 Salads dataset. As far as we know there are no published results using these labels so we can not compare against prior work. Our performance on the low granularity labels is very poor. However, given that there are 52 low level actions we do substantially better than chance (random accuracy=1.9%). This is the set that includes "start", "core", and "end" subactions for each action. Some of these subactions, such as *start_cutting_cucumber* and *start_peeling_cucumber*, are indistinguishable when looking at the accelerometer data. On the mid-level actions there are 18 classes (random accuracy=5.5%) and we perform somewhat better. Again, in this set there are actions such as *cutting_cucumber* and *cutting_cheese* that are indistinguishable using the accelerometer data. In the high-level actions there are only four labels so it is not surprising performance is very high. Future work should look into using the video and depth data to help differentiate between actions using different ingredients like cutting a tomato versus cutting a cucumber.

On JIGSAWS we achieve 5.3% higher accuracy than the state of the art [6]. Contrary to 50 Salads, the latent variables have a larger affect on performance. The overlap scores improves by 3.4%. and 1.0% for the chain and convolutional models respectively. Accuracy for the SC-CRF with four latent variables is only about 1% worse than for the LC-SC-CRF. The features around the action transitions in this data tend to vary smoothly, so it is likely that the latent variables are appropriately capturing the difference between the start, middle, or end of an action.

In both cases filtering plays an important role in smoothing out the results. On 50 Salads the overlap scores improve from 33.05 to 59.42 and 4.77 to 51.92. There is a smaller but also significant improvement on JIGSAWS. This improvement is due to the large number of extra segments created by the skip chain model. These results exemplify why our proposed metrics do a superior job highlighting errors in practical applications.

Predictions: Figure 5 shows example prediction sequences for the two datasets. The top of each figure depicts the timeseries features from time $t = 1$ to T. Red is high, green is neutral, and blue is low. On 50 Salads dark blue corresponds to the value 0. For a listing of the features see figure 6. In both datasets it is clear that there are repeating patterns in the features that are indicative of each action class. In the Salads dataset change in tool usage tends to correlate with change in action. In JIGSAWS the features are sometimes blurred during the transitions between different actions.

An example sequence from each dataset is shown in figure 5. The top plot for each shows the raw features over time. The consecutive plots show the ground truth and predicted action sequences for several variations of our model. Each color in a label sequence indicates a unique action. it is clear that the model without filtering contains many incorrect frames. The difference between the SC-CRF and LC-SC-CRF are less extreme but still noticeable. The segment boundaries tend to be better aligned with the ground truth and there are fewer oversegmentation issues.

In the Salads dataset it is interesting to see that there are occasions where the features do not appear to be synced up exactly with the accelerometer values. For example, the ground truth labels may indicate that *peeling* begins when the user starts to reach for the peeler object. Our method does not immediately pick up on this action because we rely solely on the accelerometer data. If we extracted information from the video it is possible that we would pick up on these events earlier.

Learned Action Primitives: Some action primitives learned on 50 Salads and JIGSAWS are shown in Figure 6. In 50 Salads each object has 3 values for the X, Y, and Z component of the accelerometer. It is clear that there is a dominant object that corresponds to each of the actions. For example in *cutting* the knife is dominant; in *peeling* the peeler is dominant. Notice that other objects sometimes vary in shade of blue/green. This indicates that they are sometimes used for a task (perhaps at the beginning or end) but not in all cases.

In JIGSAW it is common for an action to contain a change in gripper state. For example when reaching for the needle the first gripper transitions from open to closed (red to green). Other actions are often characterized by transitions in tool positions. See the gradients in the "Position (right)" features in *position_needle* and *push_through_tissue*. The features in this case are from the robot kinematics (position, velocity, and gripper angle) and the relative location of the tools to the surgical

In both cases it is clear that these action primitives provide an interpretable way for learning how features transition throughout an action.

Model: Our action primitives clearly provide a much richer model of how features change over the course of an action compared to the traditional pointwise model. Note, however, that it does not always achieve substantially improved accuracy. The intuition is as follows. If there is substantial change in the features over the course of an action then our action primitive representation should perform much better than the common pointwise estimate. If the features throughout an action tend to be similar then our model may not impact performance. In all datasets we have evaluated (including data not described here) we have seen at least a marginal improvement and sometimes a more substantial improvement.

Failed experiments: Throughout the course of this research we also looked at several methods for temporal segmentation and then recognition. In our experience there is not a single temporal segmentation method that works well all of the time. One method may work moderately well on data like 50 Salads but not on JIGSAWS. Furthermore, it is common to face issues of gross over- (or under-) segmentation by performing these tasks independently. If you perform them jointly then you can leverage more information about each of the action classes in ways that you typically do not for pure segmentation.

In many domains sparse regularization has been used to prevent overfitting. We tested our model using the $L2$, $L1$, and $L*$ norms on the weight vector in the SSVM. While the learned weights using the $L1$ and $L*$ variants were more sparse the overall accuracy was several percent worse. Further investigation into structured sparsity could be used to find a small subset of features (e.g. cooking utensils) that are most important for an action.

VI. Conclusion

In this paper we introduced a notion of an action primitive that provides a set of interpretable filters that describe how features change over time. We showed that our Latent Convolutional Skip Chain CRF achieves notably higher performance compared to the models of [6] and [14] on all metrics. In efforts to promote collaboration we will release our code and corresponding features for each dataset.

References

[1] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2010-24, Mar 2010.

[2] Y. Gao, S. S. Vedula, C. E. Reiley, N. Ahmidi, B. Varadarajan, H. C. Lin, L. Tao, L. Zappella, B. Bejar, D. D. Yuh, C. C. G. Chen, R. Vidal, S. Khudanpur, and G. D. Hager, "The jhu-isi gesture and skill assessment dataset (jigsaws): A surgical activity working set for human motion modeling," in *Medical Image Computing and Computer-Assisted Intervention M2CAI - MICCAI Workshop*, 2014.

[3] K. P. Hawkins, S. Bansal, N. N. Vo, and A. F. Bobick, "Anticipating human actions for collaboration in the presence of task and sensor uncertainty," in *2014 IEEE International Conference on Robotics and Automation, ICRA 2014, Hong Kong, China, May 31 - June 7, 2014*, 2014, pp. 2215–2222.

[4] N. Hu, G. Englebienne, Z. Lou, and B. Kröse, "Learning latent structure for activity recognition," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2014.

[5] H. S. Koppula and A. Saxena, "Learning spatio-temporal structure from rgb-d videos for human activity detection and anticipation," in *International Conference on Machine Learning (ICML)*, 2013.

[6] C. Lea, G. D. Hager, and R. Vidal, "An improved model for segmentation and recognition of fine-grained activities with application to surgical training tasks," in *2015 IEEE Winter Conference on Applications of Computer Vision, WACV 2014, Waikoloa, HI, USA, January 5-9, 2015*, 2015, pp. 1123–1129.

[7] J. Lei, X. Ren, and D. Fox, "Fine-grained kitchen activity recognition using RGB-D," in *ACM Conference on Ubiquitous Computing (Ubicomp)*, 2012, pp. 208–211.

[8] V. I. Morariu and L. S. Davis, "Multi-agent event recognition in structured scenarios," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2011, pp. 3289–3296.

[9] A. Müller and S. Behnke, "Learning a Loopy Model For Semantic Segmentation Exactly," *arXiv preprint arXiv:1309.4061*, no. 2006, 2013.

[10] S. Nowozin, "Structured Learning and Prediction in Computer Vision," *Foundations and Trends in Computer Graphics and Vision*, vol. 6, no. 3-4, pp. 185–365, 2010.

[11] H. Pirsiavash and D. Ramanan, "Parsing videos of actions with segmental grammars," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014, pp. 612–619.

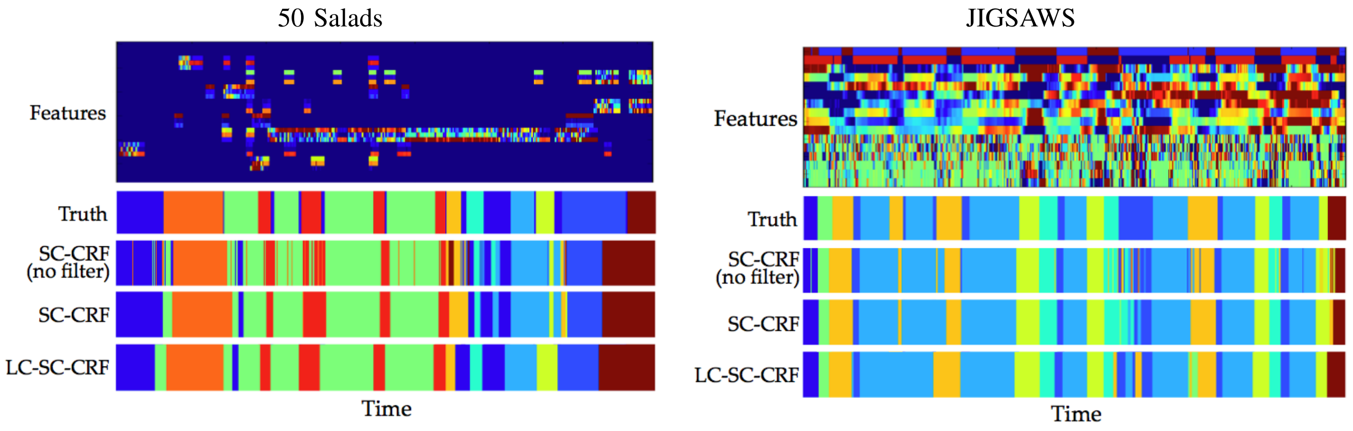

Fig. 5. Visualization of the ground truth and predicted labels for example sequences in the 50 Salads (left) and JIGSAWS (right) datasets. The top row in each shows the raw input features over time. Red is high, green is neutral, and blue is low. The subsequent rows depict the ground truth and predicted labels for several models ($H = 1$). Each color corresponds to a different class.

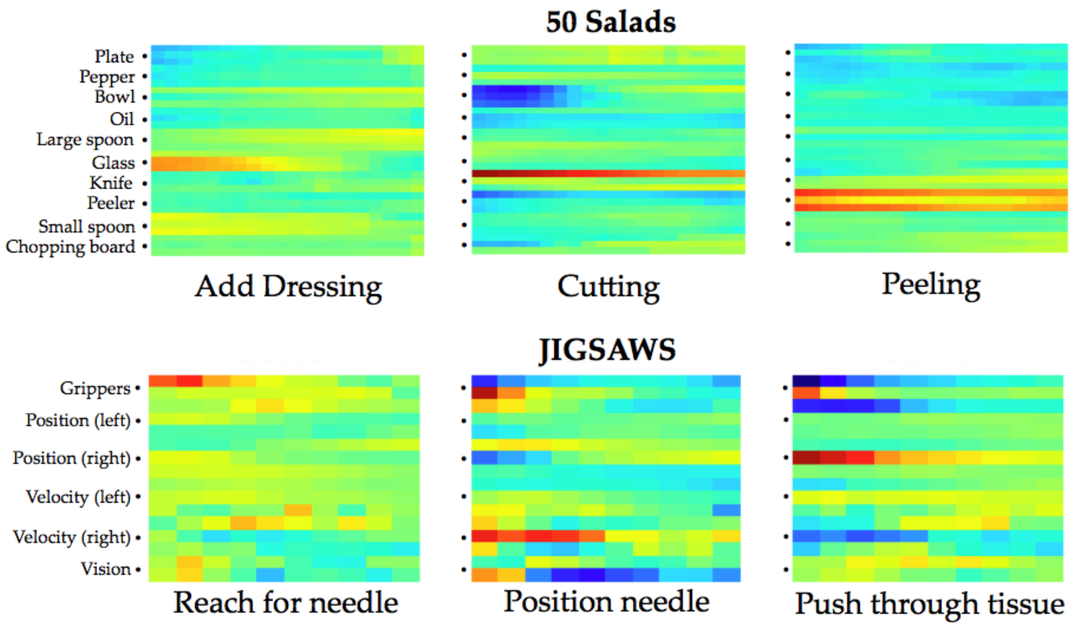

Fig. 6. Example action primitives learned on 50 Salads and JIGSAWS. The rows of each plot show the values of each feature over the course of an action. The labels along the y axis are the corresponding feature s for each row. For 50 Salads there are X, Y, and Z values for each object.

[12] K. Ramirez-Amaro, M. Beetz, and G. Cheng, "Automatic segmentation and recognition of human activities from observation based on semantic reasoning," in *IEEE Conference on Intelligent Robots and Systems (IROS)*, 2014.

[13] J. Rosen, J. Brown, L. Chang, M. Sinanan, and B. Hannaford, "Generalized approach for modeling minimally invasive surgery as a stochastic process using a discrete markov model," *IEEE Transactions on Biomedical Engineering*, vol. 53, no. 3, pp. 399–413, March 2006.

[14] S. Stein and S. J. McKenna, "Combining embedded accelerometers with computer vision for recognizing food preparation activities," in *Proceedings of the 2013 ACM International Joint Conference on Pervasive and Ubiquitous Computing (UbiComp 2013), Zurich, Switzerland*. ACM, September 2013.

[15] S. Stein and S. J. McKenna, "User-adaptive models for recognizing food preparation activities," in *ACM International Conference on Multimedia, 5th International Workshop on Multimedia for Cooking and Eating Activities (CEA'13)*. ACM, October 2013.

[16] L. Tao, E. Elhamifar, S. Khudanpur, G. D. Hager, and R. Vidal, "Sparse hidden markov models for surgical gesture classification and skill evaluation." in *International Conference on Information Processing in Computer-Assisted Interventions (IPCAI)*, ser. Lecture Notes in Computer Science, vol. 7330. Springer, 2012, pp. 167–177.

[17] L. Tao, L. Zappella, G. D. Hager, and R. Vidal, "Surgical gesture segmentation and recognition." in *Medical Image Computing and Computer Assisted Intervention (MICCAI)*, 2013, pp. 339–346.

[18] I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun, "Large Margin Methods for Structured and Interdependent Output Variables," *Journal of Machine Learning Research*, vol. 6, pp. 1453–1484, 2005.

[19] B. Varadarajan, "Learning and inference algorithms for dynamical system models of dextrous motion," Ph.D. dissertation, Johns Hopkins University, 2011.

[20] N. N. Vo and A. F. Bobick, "From stochastic grammar to bayes network: Probabilistic parsing of complex activity," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014, pp. 2641–2648.

[21] Y. Wang and G. Mori, "Hidden part models for human action recognition: Probabilistic vs. max-margin," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 7, pp. 1310–1323, 2011.

[22] C.-N. Yu and T. Joachims, "Learning structural svms with latent variables," in *International Conference on Machine Learning (talk)*, 2009.