

Modeling Actions through State Changes

Alireza Fathi and James M. Rehg
College of Computing
Georgia Institute of Technology
`{afathi3, rehg}@gatech.edu`

Abstract

In this paper we present a model of action based on the change in the state of the environment. Many actions involve similar dynamics and hand-object relationships, but differ in their purpose and meaning. The key to differentiating these actions is the ability to identify how they change the state of objects and materials in the environment. We propose a weakly supervised method for learning the object and material states that are necessary for recognizing daily actions. Once these state detectors are learned, we can apply them to input videos and pool their outputs to detect actions. We further demonstrate that our method can be used to segment discrete actions from a continuous video of an activity. Our results outperform state-of-the-art action recognition and activity segmentation results.

1. Introduction

What makes an action (e.g. “open the jar”) identifiable? How can we tell if such an action is performed? Over the last two decades various cues have been used to model and understand actions in computer vision: holistic shape and motion description [2, 5], space-time interest points [10], feature tracks [18], object and hand interaction [6, 9, 29] and various other techniques. The common theme among all these works is that they model an action by encoding motion and appearance throughout the interval in which it is performed.

However, in order to fully understand actions we must understand their purpose [25]. Actions with similar motion patterns and hand-object relationships can have a different meaning because they accomplish a different goal. For example, “open coffee jar” and “closed coffee jar” are two different actions, in fact they are inverse. However, they produce similar motion patterns and involve the same object. The key to distinguishing these two actions is to be able to detect the **state** of the “coffee jar” (open vs. closed) and how it changes by these actions. For example, the opening action changes the state of an object from closed to open.

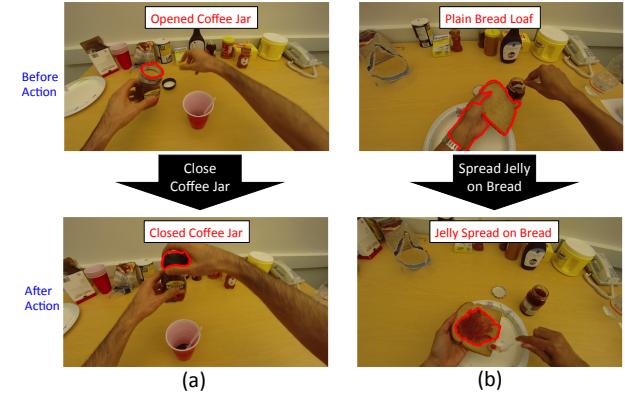


Figure 1: By comparing the initial and final frames of an action, and exploiting the action label, we can learn to detect the meaningful changes in the state of objects and materials produced by the action. In example (a), our method recognizes the action of “close coffee jar”, as a result of detecting regions corresponding to open and closed coffee jar. Similarly in example (b), the action of “spread jelly on bread” is recognized by detecting the regions corresponding to plain bread loaf and jelly spread on bread respectively in the initial and final frames of the action.

Based on this observation, we introduce a method for recognizing daily actions by recognizing the changes in the state of objects and materials. Most actions can be performed only if certain preconditions are met. Moreover, their execution causes some existing conditions to change. For instance, the action “spread jelly on bread using knife” requires jelly to be on the knife but not on the bread when it is applied. This action changes the state of the jelly from being on the knife to being spread on the bread. Or for example, “take cup” is an action before which the cup is not being held by the hand, but once it is performed the cup is grasped by the hand¹.

¹Note that this notion of actions as state changing processes holds for most cases, however, there are exceptions such as “dancing” that do not create any describable or observable changes in the environment. In this paper our focus is on goal-oriented object-manipulation tasks which are

We are interested in two kinds of changes: 1) changes in the state of objects (e.g. coffee-jar becoming open or closed) and the transformation of stuff (e.g. coffee powder mixing with water, jelly getting spread on bread, egg getting scrambled). For example, the following actions take place during the activity of “making coffee”: (open coffee jar), (scoop coffee using spoon), (pour coffee into cup) and (put hot water into cup), (close coffee jar). Throughout these actions, the coffee jar changes states from closed to open and again to closed. Likewise, the coffee powder changes state from being in the coffee jar to being on the spoon, and then being in the cup, and finally dissolving into hot water.

Following our previous works [8, 6, 7] and like many other recent works [17, 22], we adopt egocentric paradigm for recognizing daily activities and actions. Analyzing the details of hand-object interaction is challenging in third-person view videos due to insufficient resolution of hands and objects. In contrast, the egocentric view puts the environment into the center of the action interpretation problem. In this view, the subject often naturally avoids occlusion which results in high resolution and detailed images of handled objects. We leverage the egocentric paradigm to build fine-grained representations of the object states and materials in order to describe object manipulation tasks.

In this paper, we propose a weakly supervised method for learning the object and material states that are necessary for recognizing daily actions. Once these state detectors are learned, we run them at each frame of the videos and describe the environment at each moment in time based on the existence or absence of detected object and material states. We introduce methods that leverage the changes in the state of the environment to recognize actions and segment activities. Our results outperform state-of-the-art action recognition and activity segmentation results. Our contributions in this paper are: 1) We present a model for actions based on the changes in the state of the environment, 2) we introduce a method for weakly supervised discovery of state-specific regions from action videos and 3) we provide an activity segmentation method by verifying the consistency of the environment state with the beginning and ending conditions of the actions.

2. Related Work

Action recognition has been the subject to a large body of work in computer vision [15, 24]. Much of the initial work on this area has been focused on understanding human body movement patterns such as walking, running, ballet moves, etc. [2, 5] and have resulted in near perfect performance on simple standard datasets such as KTH [21]. In contrast to these early datasets, people manipulate objects as a natural part of performing realistic daily activities. In these actions,

often intended to accomplish a particular goal.

the interaction between hand and object is an important part of visual evidence that should be considered.

There has been various attempts in the past to model object context for action and activity recognition. Wilson and Bobick [27] propose parametric Hidden Markov Model for recognizing human actions. Their method indirectly models the effect of object attributes on human actions. Mann et al. [13] use a physics based approach to describe kinematic and dynamic properties of hands and objects and understand their interactions. Li and Fei-Fei [11] classify events based on the object categories that appear in an image. Wu et al. [28] recognize activities based on temporal patterns of object use. They use RFID-tagged objects to bootstrap the appearance-based classifiers. Marszalek et al. [14] use scene context to improve action recognition performance. Yao and Fei-Fei [29] recognize activities in images based on the mutual context of objects and human pose.

All of these methods use detected objects as context for recognizing actions. In addition, some of them model the actions by considering the mutual relationship between human pose and object locations. However, none of these methods leverage the state of the objects for recognizing human-object interaction. Vaina and Jaulent [25] suggested that a comprehensive description of an action requires understanding its goal. They refer to the conditions necessary for achieving the goal of an action as action requirements and model the compatibility of an object with those goals. Gupta et al. [9] look at the change of intensity in the area around hands in a constrained setting to recognize subtle object reactions like the turning on a flashlight. However, change of intensity cannot describe complex state changes caused by an action like opening an object. In this paper, we describe a method that discovers state-specific regions from the training videos and models their changes to recognize actions during the testing phase.

In the AI and robotics literature, an action is performed by an intelligent agent through actuators and results in particular changes to the environment [19]. Such changes can be perceived by agent’s sensors, and lead to decision making, resulting in a perception and action loop. However, in robotics often the focus has been on the planning problem rather than recognizing actions. In this work, we focus on learning visual representations for object and material states and modeling the actions based on their changes.

3. Method

Our task is to model daily actions via the changes they induce in the state of the environment. We formulate this problem as the discovery of **changed regions** that either correspond to a specific state of an object (e.g. open mouth of the bottle of water) or represent a particular material (stuff, e.g. coffee powder on spoon). We represent actions based on the changes they make in objects and materials. In

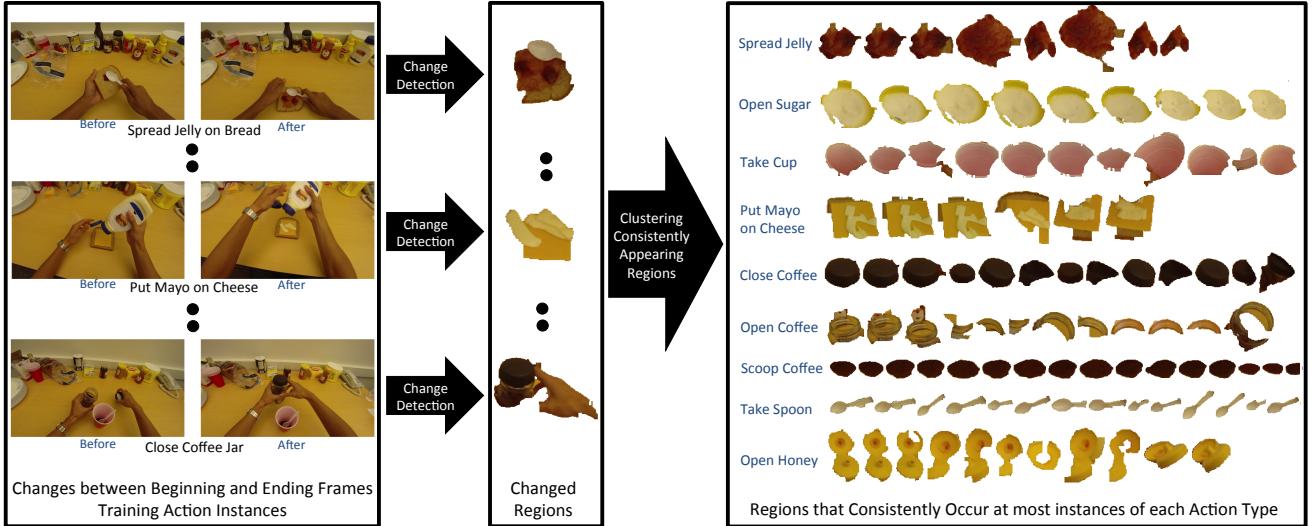


Figure 2: Stages of our state-specific region discovery framework are shown. This procedure only takes place during the training phase. First for each action instance, we compare its initial frames with its final frames to extract the regions that are changed. In the second stage we discard the changes that are not common over the examples of their corresponding action type. In the final stage, we learn a detector for each group of consistent regions. During the testing phase we apply the trained region detectors to describe actions and states.

order to find the regions that are changed as a result of an action, we compare their appearance before the action starts to their appearance after the action ends. The changed regions often correspond to either the state of the objects, or to the materials. Using our method, we show significant gains in action recognition and video segmentation performance.

During the training phase, we are given a set of activity videos. An activity like making peanut-butter and jelly sandwich, consists of a sequence of atomic actions (e.g. take bread, open peanut-butter jar, spread peanut-butter on bread using knife, etc.). For each training activity video, the actions are annotated. The annotation for each action contains its start frame, end frame, a verb (e.g. scoop), and a set of nouns (e.g. coffee, spoon). We emphasize that we are not provided with any object location or mask. In Sec 3.1, we introduce a method for discovering regions that correspond to object states and materials from video images. We further learn various state-specific region detectors from the set of discovered regions. In Sec 3.2 and 3.3, we propose a method for recognizing actions based on the change in the detected object states and materials. Finally, in Sec 3.4, we introduce a method for segmenting a new video into a sequence of actions by localizing their initial and final frames.

3.1. Discovering State-Specific Regions

The first step in our training phase identifies regions that are representative of the state of an object or existence of a material. In this stage, we make two assumptions: (1) an object state or material does not change unless an action

is performed and (2) an object state or material change is associated with an action only if it consistently occurs at all instances of that action. Fig 2 illustrates our three stage approach to discovering the state-specific regions. In the first stage, we identify regions that either appear or disappear as a result of the execution of each action instance. For example, the region corresponding to the lid of the coffee jar will change as a result of performing the action of open coffee. However, there may be other irrelevant changes in addition. For example, a change in the appearance of hand as a result of its movement. In the second stage, we prune changes that are not consistently associated with an action. Finally, in the third stage we learn a detector for each group of discovered state-specific regions. We use these detectors to classify unknown regions during the testing phase.

Change Detection: In this stage, we find the regions that either appear or disappear throughout each action instance in the training set. Each action instance corresponds to a short interval which is a sub-part of a longer activity video. For each action instance, we sample a few frames from its beginning and a few frames from its end. We compare the beginning and ending frames to find their differences. For each pair of beginning and ending images, we match their pixels using large displacement optical flow [3]. Then for each pair of matched pixels, we compute change based on their color difference, similar to the method of Sand and Teller [20]. We calculate the significance of change for each region based on the average amount of change in its pixels. The regions that we use in our algorithm are acquired using

the method in Arbelaez et al. [1].

These appearance and disappearance patterns often correspond to changes in object states or the creation of new materials. For example, pouring water into a cup containing coffee powder results in the appearance of a new dark brown liquid region in the cup. Of course there will be many other irrelevant changed regions due to occlusion, lighting effects, and other factors. To overcome such mistakes, we compare each beginning (ending) image to multiple ending (beginning) images. We set the amount of change to the minimum amount computed among all the comparisons. A few examples of the results of this stage are shown in the second column of Fig 2. After this pruning procedure, still there are often regions left that do not correspond to state changes and materials. The next step is to remove them.

Consistent Regions: In the previous stage, we extract regions that have changed between the initial and final frames of each action instance. Now in this stage, we only keep the subset of those regions that consistently occur across the instances of an action type. For example, a region that corresponds to coffee jar's lid consistently appears at the beginning of the “open coffee”, but a spurious region would not. A region r consistently occurs at an action class \mathcal{A} , if there is a region \hat{r} similar to r at each instance a of that action class ($a \in \mathcal{A}$). Here we suggest an algorithm that extracts the consistent regions, and further groups them based on their similarity. Inspired by the source constrained clustering method of Taralova et al. [23], we cluster the N extracted regions from instances of action class \mathcal{A} into k sets $\{\mathcal{S}_1, \mathcal{S}_2, \dots, \mathcal{S}_k\}$ by enforcing the regions in each cluster to be drawn from the majority of action instances. We further add an additional constraint that each action instance can at most contribute one example to each cluster. This constraint prevents us from adding regions that correspond to non-relevant object parts but have similar appearance to the cluster. We do this by optimizing the following objective function:

$$\begin{aligned} \arg \min_{\mathcal{S}} \quad & \sum_{i=1}^k \sum_{x_j \in \mathcal{S}_i} \|x_j - \mu_i\|^2 \\ \text{subject to :} \quad & \sum_{a \in \mathcal{A}} \delta(\mathcal{S}_i, a) \geq h \\ & \delta(\mathcal{S}_i, a) \leq 1 \end{aligned} \quad (1)$$

where x_j is a feature vector representing region j , μ_i is the mean of points in \mathcal{S}_i , a is an instance from the set of all instances of the action class \mathcal{A} , and $\delta(\mathcal{S}_i, a)$ is a function that returns the number of regions from action instance a in cluster \mathcal{S}_i , and h is a scalar. This objective function is similar to the objective function of k-means with two additional constraints that enforce a cluster \mathcal{S}_i to have samples from at least h action instances.

We approximately minimize the objective function in Eq 1 through a simple iterative approach. In each iteration, we

pick the best set of h regions with minimum distance from each other and return them. We make sure each of these regions is picked from a different action instance. In the next iteration, we remove the previously returned regions from the set of remaining regions and repeat the procedure. We continue this until either k clusters are returned or there are less than h action instances with regions left in them. See Algorithm 1 for the details. In our experiments, we only use the first few clusters which have the highest self-similarity. We have shown examples of such clusters in the right-most column of Fig 2.

Algorithm 1 One iteration of selecting consistent regions

```

set of best  $h$  regions  $R = \{\}$ 
total intra-region distance  $b = \infty$ 
a temporary set for keeping regions  $\bar{R} = \{\}$ 
for (every region  $r$  in every action instance  $a \in \mathcal{A}$ )
    for (each  $\hat{a} \neq a, \hat{a} \in \mathcal{A}$ )
        pick the closest region in  $\hat{a}$  to  $r$  and add it to  $\bar{R}$ 
    end
    select a subset of  $h$  regions in  $\bar{R}$  with min total distance  $d$ 
    if ( $d < b$ )
        set  $R$  to the subset of  $\bar{R}$ 
        set  $b$  to  $d$ 
    end
end
return  $R$  as a cluster
remove  $\bar{R}$  from the set of extracted regions

```

State-Specific Region Detectors: In this stage, we learn a detector for each of the region clusters. We train a linear SVM by using the regions belonging to the cluster as the positive set and all the regions in activities that do not contain the action as the negative set. We describe each region with color, texture and shape features. For each region, we build a 128 dimensional color histogram by quantizing the color values of pixels into clusters. In addition, we build a texture histogram by computing texture descriptors [26] for each pixel and quantizing them to 256 centers. We further compute a 16 dimensional shape feature vector for each region. Our shape features are similar to HOG [4] features, but instead of computing them on patches, we compute them on the whole region. We compute the gradient at all pixels inside the region and quantize them into 16 orientations. We count the occurrence of gradients in each orientation. We concatenate these three features together into a 400 dimensional feature vector that we use to represent regions.

3.2. States as Action Requirements

An action can be performed only if certain conditions are satisfied in the environment. For example, “clean the table” is an action that requires the table to be dirty before

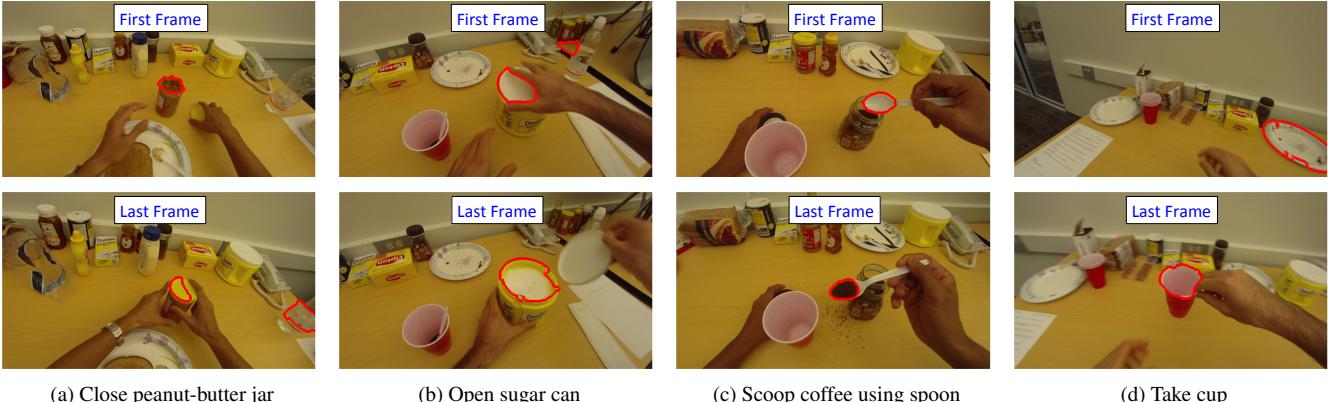


Figure 3: In contrast to the features of conventional action recognition methods, our features are meaningful to humans. Regions that correspond to state-specific detectors with a classifier weight higher than a threshold are shown with a red boundary. For example in (a), in the first frame SVM puts a high weight on the open mouth of peanut-butter jar and in the last frame puts a high weight on peanut-butter jar’s lid.

the action is performed, and clean afterwards. Thus, the key to recognizing a goal-oriented action is to be able to recognize the state of the environment both before and after that action.

We represent the environment state based on two criteria: (1) existence or absence of state-specific regions and (2) whether or not an object (region) is grasped and is being manipulated by the hands. To model the first criteria, we represent each frame of the test video by the response vector of the trained state-specific region detectors (Sec 3.1). For each detector, we run it on all the regions of the test frame and pick the highest classification score as its response. We set the responses that are higher than a threshold to 1, and the ones that are lower than a threshold to -1 . We set the rest of the responses to 0. This quantization helps us to avoid overfitting. In order to model if the regions are being grasped by the hands or not, we use the foreground segmentation method of [6] which identifies if a region is being moved by the hands or not. We build a similar vector based on the responses of the detectors on the foreground regions, instead of applying them to all regions. We represent each frame by the concatenation of its response vectors.

3.3. Modeling Actions through State Changes

The majority of common action recognition approaches rely on analyzing the motion and appearance content of the action intervals. Movement patterns are crucial for recognition of many actions, in particular body movements such as running, walking, dancing, etc. However, most daily object-manipulation tasks are goal-oriented actions that are defined by the changes they cause to the state of the environment.

Given a test action interval, we build two response vectors. One is based on the response of the detectors on its beginning frames, and the other is based on the responses

on its ending frames. We represent the interval by concatenation of these two vectors. We use linear SVM to train a classifier for each action type. Since we have concatenated the vectors of beginning and ending frames, linear SVM can model the change of an object state or material by putting weights on its corresponding responses. Linear SVM will put higher weights on state-specific regions that are consistently created either at the beginning or at the end of the action, and lower weights on the ones that do not relate to the action. We show visualizations of the classifier weight vectors for few action instances in Fig 3.

3.4. Activity Segmentation

Activity segmentation of a test video is the task of breaking a long activity video into a sequence of short actions. In order to do so, often one takes all the action detection scores as input and infers the frames that are assigned to each action in that video. In order to handle detection errors, a common strategy is to apply the action classifiers to every possible interval, and then use non-maximum suppression or dynamic programming [16, 6].

Here instead we leverage the capability of our framework for detecting environment states to segment activity videos. In state detection, different than action recognition, the problem is to assign a state label to each frame of the video. The possible set of states are: 1) before a particular action starts, during that action, after that action ends. Our method is as follows. For each action class (e.g. open coffee), we train two state detectors, one using its beginning frames and one using its ending frames. The state detectors are learned on top of the frame’s responses to pre-trained state-specific region detectors (Sec 3.3).

The state detectors are trained using linear SVM by taking the action’s beginning or ending frames as positive set

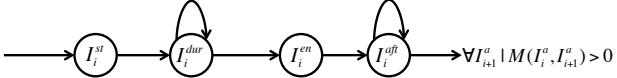


Figure 4: Possible transitions are shown for states of an interval I_i .

and all the other training frames as negative set. Given a test activity video, we apply all the trained beginning and ending state detectors on its frames. This results in two $|\mathcal{A}| \times T$ matrices S_B and S_E respectively, where $|\mathcal{A}|$ is the number of action types, T is the number of frames in the test activity video, and $S_B[a, t]$ and $S_E[a, t]$ respectively contain the classification scores of detecting the initial and final frames of action a at frame t .

We segment an activity video into a sequence of intervals $\mathcal{I} = \{I_1, \dots, I_{|\mathcal{I}|}\}$. An interval I_i has a few properties: I_i^a identifies its action label, I_i^{st} identifies its initial frame number, and I_i^{en} identifies its final frame number. We segment the activity video by optimizing the following objective function:

$$\begin{aligned} \arg \max_{\mathcal{I}} \quad & \sum_{I_i \in \mathcal{I}} S_B[I_i^a, I_i^{st}] + S_E[I_i^a, I_i^{en}] \quad (2) \\ \text{subject to :} \quad & I_i^{st} < I_i^{en} \\ & (I_i^{st} - I_j^{st}) \cdot (I_i^{en} - I_j^{en}) > 0 \\ & M(I_i^a, I_{i+1}^a) > 0 \end{aligned}$$

where M is a binary transition matrix. The objective function aims at finding the best set of intervals where the total sum of scores is maximized. The score of interval $I_i : \{I_i^a, I_i^{st}, I_i^{en}\}$ is computed by adding the response of the detector corresponding to the initial frame of action I_i^a on frame I_i^{st} with the response of the detector corresponding to the final frame of action I_i^a on frame I_i^{en} . There are three constraints involved in the optimization. The first two constraints prevent action intervals from overlapping with each other. The third constraint limits the possible transitions between actions. For example, it is not possible to pour milk after close milk is performed. We train the matrix M based on observed action transitions in training activities.

We can model this problem as a finite state sequential process and optimize it using dynamic programming. For this purpose, we have to assign a state to each frame of the video. In order to do this, in addition to the first frame of the interval I_i^{st} and its last frame I_i^{en} , we add two auxiliary states for it: during I_i^{dur} and after I_i^{aft} . The score of entering these states is zero, and they are only used to enforce the constraints of the Eq 2. For example, it is only possible to transition from the first frame of an interval to its during state, and then, either stay in its during state or move to its

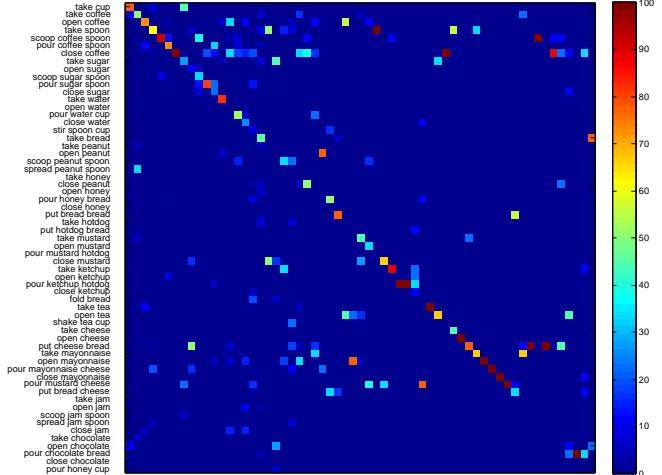


Figure 5: Confusion matrix for recognizing actions using our method is shown. The average accuracy is 39.7% on 61 classes of action which is significantly higher than the baseline (23%). Random classification chance is 1.6%.

ending frame. The set of possible state transitions for an action are shown in Fig 4.

4. Results

To validate our model of actions based on state changes, we show extensive qualitative and quantitative results on two tasks: (a) action recognition: assigning an action to a given interval and (b) activity segmentation: decomposing an activity into a sequence of actions by detection and decoding. We compare our results to state-of-the-art performance and different baselines.

4.1. Action Recognition

We evaluate our method on our GeorgiaTech Egocentric Activity (GTEA) dataset [8]. This dataset consists of 7 types of activities, where each activity is performed by 4 subjects. There are 61 actions in this dataset, after omitting the background action classes and fixing some of the mistakes in the original annotation. Training and testing sets are chosen as is done in [6].

Baselines: we compare our method to three baselines. The first baseline trains a linear SVM on concatenation of STIP [10] bag of words built from the first and second half of each interval. This STIP baseline performs poorly on this dataset, resulting in 11.6% accuracy in recognizing 61 classes. We believe the reason is that in an egocentric setting the camera is continuously moving, which makes space-time interest points fire at areas that do not relate to the action. The second baseline trains a linear SVM on two SIFT [12] bag of words built for each interval. SIFT features perform better than STIP, however, they still perform

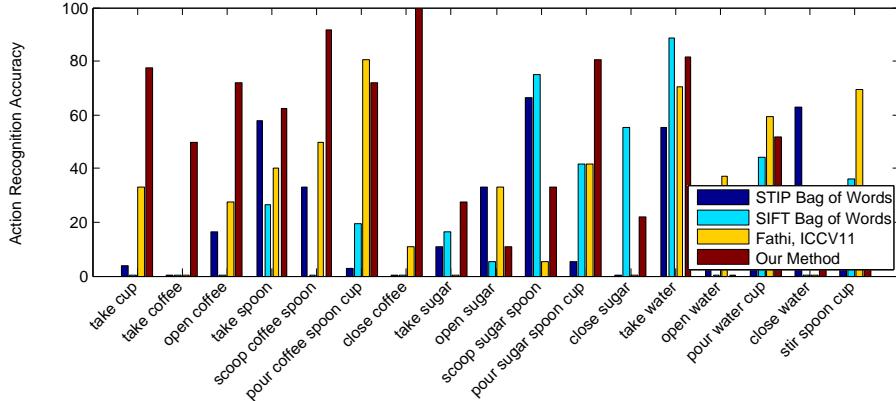


Figure 6: We compare the performance of our method with various baselines. STIP bag of words results in 11.6% accuracy, SIFT bag of words results in 19% accuracy, and the method of [6] results in 23% accuracy. Our method significantly outperforms these baselines by achieving 39.7% accuracy on 61 classes, where the random chance is 1.6%. We show the comparison on the actions of the activity of making coffee.

poorly, resulting in 19% accuracy. This is because many of the daily objects used in these activities are textureless and they produce no corners. Finally we compare our method to our previous work [6] which uses hand motion, hand location, objects in foreground and hand pose to recognize actions. This method achieves 23% accuracy on this dataset for 61 classes².

Our Method: Given an action interval, we build two response vectors: one using its beginning frames and one using its ending frames, as described in Sec 3.3. We have 610 state-specific region detectors. We train 10 detectors from the consistent changes of each action type. We describe a frame by applying each of these detectors on its regions and taking its highest response. We set the responses greater than 0.5 to 1, responses smaller than -0.5 to -1, and the responses between -0.5 and 0.5 to 0. We further build similar response vectors from foreground regions. Finally we represent each interval with a $610 \times 2 \times 2 = 2440$ dimensional feature vector, and train a linear SVM for each action type. Our method achieves 39.7% accuracy on 61 classes, where chance is 1.6%. This represents a significant improvement over previous baselines. We have shown the confusion matrix for our method in Fig 5. We have compared the accuracy of these methods in Fig 6.

Classifier Visualization: Many conventional action recognition methods rely on features such as corners, point tracks, etc. that are not meaningful to humans. In contrast,

²These numbers are different than the ones reported in my previous work [6]. The action recognition results in [6] are based on recognizing action verbs such as pouring, opening and closing. However, in this paper, the action labels contain both verbs and object names. For example, pouring mayonnaise on the cheese, opening coffee jar and opening honey. There are 11 action verbs in the GTEA dataset, while the number of action labels is 61.

our features are state-specific regions that correspond to object parts or materials, and they can be easily interpreted. The weights of the linear SVM classifier trained on examples of an action determines the state-specific parts and materials that should exist in its initial or final frames. We have shown regions that correspond to state-specific detectors with high classifier weights in Fig 3.

4.2. Activity Segmentation

Given a video, we use our activity segmentation method described in Sec 3.4 to segment it into different actions. We compare our method to the detection results from our previous method [6]. In our previous work, we train a CRF for each activity and apply that on action scores to force transition constraints. That method enforces much harder constraints in comparison to our new approach. Using our new method we achieve 42% accuracy and outperform our previous results of 33% accuracy. The results are computed by counting the percentage of the frames that are correctly labeled in the test activities. We show segmentation results in Fig 7 on two test activity videos. Our results are in general smoother in comparison to our previous results [6].

5. Discussion

In this paper, we present a model for actions based on the changes in the state of objects and materials. We show significant gains in both action recognition and detection results. We further introduce a simple temporal and logical encoding method for activity segmentation that outperforms the results of previous state-of-the-art methods. In addition, we introduce a method for discovering state-specific regions from the training action examples.

We believe an interesting future direction for research



Figure 7: Activity segmentation results are shown for two test activity videos. The horizontal axis represents time (frames). Different colors represent different action labels assigned to the frames. In each example, the bottom row shows the ground-truth results, the top row shows the results of our previous method[6], and the middle row shows our current results. We correctly assign true labels to 42% of frames and outperform the results of [6] with 33% accuracy. In addition, our labels are smoother in comparison to their method.

involves building a taxonomy of possible states of objects and materials. Modeling these states would require richer features that capture shape, physical properties and affordances of things and stuff. A potential challenge would be modeling the changes that do not correspond to observable visual patterns. The benefit of our current weakly supervised approach is that it is sensitive to distinguishable visual changes in objects and materials, and implicitly ignores the ones which can't be observed in videos. Combining this benefit with stronger domain models could be valuable.

6. Acknowledgment

Portions of this work were supported in part by NSF award IIS-1016772 and by ARO MURI award W911NF-11-1-0046.

References

- [1] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik. From contours to regions: an empirical evaluation. In *CVPR*, 2009. 4
- [2] A. F. Bobick and J. Davis. The recognition of human movement using temporal templates. *PAMI*, 23(3):257–267, 2001. 1, 2
- [3] T. Brox, C. Bregler, and J. Malik. Large displacement optical flow. In *CVPR*, 2009. 3
- [4] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *CVPR*, pages 886–893, 2005. 4
- [5] A. A. Efros, A. C. Berg, G. Mori, and J. Malik. Recognizing action at a distance. In *ICCV*, 2003. 1, 2
- [6] A. Fathi, A. Farhadi, and J. M. Rehg. Understanding egocentric activities. In *ICCV*, 2011. 1, 2, 5, 6, 7, 8
- [7] A. Fathi, Y. Li, and J. M. Rehg. Learning to recognize daily actions using gaze. In *ECCV*, 2012. 2
- [8] A. Fathi, X. Ren, and J. M. Rehg. Learning to recognize objects in egocentric activities. In *CVPR*, 2011. 2, 6
- [9] A. Gupta, A. Kembhavi, and L. S. Davis. Observing human-object interactions: using spatial and functional compatibility for recognition. In *PAMI*, 2009. 1, 2
- [10] I. Laptev, M. Marszalek, C. Schmid, and B. Rozenfeld. Learning realistic human actions from movies. In *CVPR*, 2008. 1, 6
- [11] L. J. Li and L. Fei-Fei. What, where and who? classifying event by scene and object recognition. In *CVPR*, 2007. 2
- [12] D.G. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 60(2):91–110, 2004. 6
- [13] R. Mann, A. Jepson, and J. M. Siskind. Computational perception of scene dynamics. In *ECCV*, 1996. 2
- [14] M. Marszalek, I. Laptev, and C. Schmid. Actions in context. In *CVPR*, 2009. 2
- [15] T. B. Moeslund, A. Hilton, and V. Kruger. A survey of advances in vision-based human motion capture and analysis. In *CVIU*, 2006. 2
- [16] M. H. Nguyen, Z. Lan, and F. De la Torre. Joint segmentation and classification of human actions in video. In *CVPR*, 2011. 5
- [17] H. Pirsiavash and D. Ramanan. Detecting activities of daily living in first-person camera views. In *CVPR*, 2012. 2
- [18] M. Raptis and S. Soatto. Tracklet descriptors for action modeling and video analysis. In *ECCV*, 2010. 1
- [19] S. Russel and P. Norvig. *Artificial Intelligence, A Modern Approach*. Prentice-Hall, 2002. 2
- [20] P. Sand and S. Teller. Video matching. In *SIGGRAPH*, 2004. 3
- [21] C. Schuldt, I. Laptev, and B. Caputo. Recognizing human actions: A local svm approach. In *17th International Conference on Pattern Recognition*, 2004. 2
- [22] E. H. Spriggs, F. De La Torre, and M. Hebert. Temporal segmentation and activity classification from first-person sensing. In *Egovision Workshop*, 2009. 2
- [23] E. Taralova, F. De la Torre, and M. Hebert. Source constrained clustering. In *ICCV*, 2011. 4
- [24] P. Turaga, R. Chellappa, V. S. Subrahmanian, and O. Udrea. Machine recognition of human activities: a survey. In *IEEE Transaction on Circuits and Systems for Video Technology*, 2008. 2
- [25] L. Vaina and M. Jaulent. Object structure and action requirements: a compatibility model for function recognition. In *Int'l J. Intelligent Systems*, 1991. 1, 2
- [26] M. Varma and A. Zisserman. A statistical approach to texture classification from single images. In *IJCV*, 2005. 4
- [27] A. D. Wilson and A. F. Bobick. Parametric Hidden Markov Models for Gesture Recognition. *PAMI*, 21(9):884–900, 1999. 2
- [28] J. Wu, A. Osuntogun, T. Choudhury, M. Philipose, and J. M. Rehg. A scalable approach to activity recognition based on object use. In *CVPR*, 2007. 2
- [29] B. Yao and L. Fei-Fei. Modeling mutual context of object and human pose in human-object interaction activities. In *CVPR*, 2010. 1, 2