Generating Notifications for Missing Actions: Don't forget to turn the lights off!

Bilge Soran*, Ali Farhadi*†, Linda Shapiro*

*University of Washington
†Allen Institute for Artificial Intelligence
{bilge, ali, shapiro}@cs.washington.edu

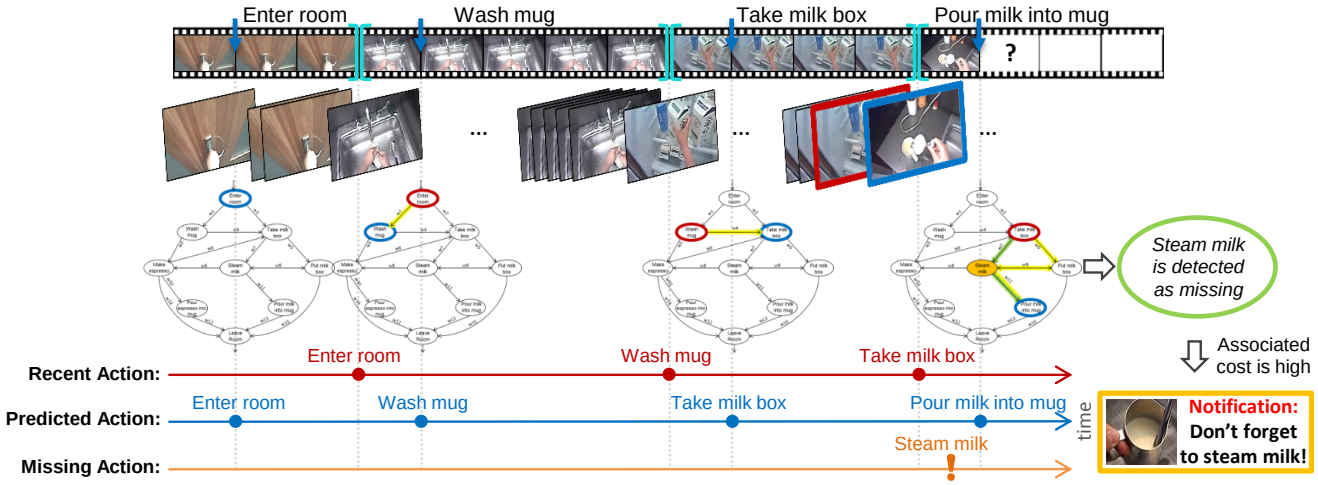


Figure 1: Our purpose is to issue notifications about missing actions given an unsegmented input stream of egocentric video. For the latte making sequence above, our system recognizes the actions that happened so far, predicts the ongoing action, reasons about missing actions and the associated cost, and generates notifications for the costly missing actions. In this figure, the brackets refer to segmented action boundaries, the blue arrows show the prediction points and the graphs below show the inter-action dependencies. The most recently completed action is marked in red, the predicted action is marked in blue, and the missing action is marked in orange. In this example, the actor is about to miss an important action: *steam milk*, and a reminder for that is given.

Abstract

We all have experienced forgetting habitual actions among our daily activities. For example, we probably have forgotten to turn the lights off before leaving a room or turn the stove off after cooking. In this paper, we propose a solution to the problem of issuing notifications on actions that may be missed. This involves learning about interdependencies between actions and being able to predict an ongoing action while segmenting the input video stream. In order to show a proof of concept, we collected a new egocentric dataset, in which people wear a camera while making lattes [1]. We show promising results on the extremely challenging task of issuing correct and timely reminders. We also show that our model reliably segments the actions, while predicting the ongoing one when only a few frames from the beginning of the action are observed. The overall prediction accuracy is 46.2% when only 10 frames of an action are seen (2/3 of a sec). Moreover, the overall recognition and segmentation accuracy is shown to be 72.7% when the whole activity sequence is observed. Finally, the online prediction and segmentation accuracy is 68.3% when the prediction is made at every time step.

1. Introduction

We all have witnessed prospective memory failures in our daily activities. How often do we forget to turn the

[1] A latte is a coffee drink made with espresso and steamed milk [1].

lights off before leaving a room? Or to turn the stove off after boiling water? Prospective memory failures are often devastating and sometime disastrous. Several researchers in cognitive sciences have explored the causes of prospective memory failures and shown effective medical practices [5]. One of the most effective medical practices is to use reminders [12] [30].

In this paper, we study the problem of automatically issuing reminders about actions that users might forget. These reminders can potentially prevent catastrophic events. Such notifications can also be useful for individuals with memory and cognition problem such as Alzheimer's disease. We also envision that these reminders can eventually be helpful in completing complex procedures.

Egocentric cameras are suitable for these kinds of frameworks, as they move with the user. For a proof of concept, we collected a latte making dataset using an egocentric camera, in which subjects wore a head-mounted camera to record their actions during the latte making activity in their own style.

We propose a novel system, which notifies people when they forget to perform an action before their current action finishes. Such a system needs to understand the ongoing action before it completes and decide whether there is a missing action or not, and if it is necessary give a notification about this missing action. For that, it needs to extract interaction dependencies and possess a decision mechanism that takes the importance of the missing action into account. Furthermore, predicting the ongoing actions as early as possible is needed. For example, when a *leave room* action is predicted, the system can check whether it needs to remind the user for *turning the stove off* by checking whether *the stove is set to on* and decide how important that reminder is by analyzing the associated cost.

2. Related Work

Egocentric Vision: Recently, egocentric activity recognition is gaining a lot of interest among the computer vision community. There are many works analyzing either the camera wearers actions [7, 8, 7, 33, 21, 29] or the people surrounding the camera wearer [38, 37]. For a review of egocentric vision please see [20].

Our notification system requires prediction of actions from partial observations. [37] uses the video recordings of a camera worn by a robot and *predicts the actions of the people around the robot*. To the best of our knowledge, we are the first to study action prediction in egocentric videos, where the *camera wearer's actions* are analyzed. This setting is especially suitable for daily living activity prediction.

Action Prediction: Action prediction is defined in [36] as a probabilistic process, which estimates the action in progress when only the beginning part of the action exists. [36] represents an action as an integral histogram of spatio-temporal features and uses a dynamic bag-of-words approach to predict. [37] predicts the human actions during human-robot interactions. They use previous actions as cues to infer the following ones, therefore requiring particular action pairings. It is different from our model in that, although there is a partial-order among the actions in our model, we do not require one action to be a predecessor of another; a person either steams the milk first and grinds the coffee next or vice-versa. [4] studies recognizing actions where the missing part of the action (temporal gap), can be anywhere. They used sparse coding to determine the likelihood that a certain type of activity is present in the partially observed video. [27, 28] propose an activity prediction framework; their purpose is to predict higher-level activities using actionlets, and the dependencies between them are represented by a probabilistic suffix tree.

A max-margin early event detector (MMED) is proposed by [13, 15] for training temporal event detectors for early detection of actions. Their model extends the structured output SVM (SOSVM) to accommodate sequential data. In general MMED is designed for *early event detection*, while for the problem we are tackling we need to *recognize the actions from partially observed videos*. [43] proposes a moving pose descriptor framework on depth sequences, which is also used for activity detection and low-latency recognition. After the successful application of a modified SOSVM model to an early event detection problem, the popularity of SOSVM-based models for action prediction has risen over the years. [23] proposed a multi-scale model for action prediction for already segmented videos. Their model uses both the global and the local history of features to learn temporal dynamics. Similarly, [24] used hierarchical "movemes" for describing human movements at multiple levels of granularities, ranging from atomic movements to the ones that exist in a larger temporal extent to predict actions in segmented videos. Both [23, 24] used modified versions of the SOSVM. [6] explored the trade-off between accuracy and observational latency by determining distinctive canonical poses of the subjects.

Activity forecasting aims to predict future actions rather than recognizing ongoing ones from partial observations. [22, 17] proposed methods for activity forecasting, which is out of the scope of our paper.

Event detectors in general suffer from a major fundamental drawback of overlapping detections for different actions. In our paper, besides prediction we also address temporal segmentation in a coupled, supervised setting.

Joint Segmentation & Recognition: Most studies in the literature perform recognition/prediction on pre-segmented videos. There exist techniques for coupled segmentation/recognition, but most of them rely on generative models [10, 2, 26, 44]. [14] proposed a maximum margin temporal clustering, which determines the start and the end

of each segment, and discriminates among temporal clusters by a multi-class SVM. [31] represented activities as temporal compositions of motion segments. To train their action classifier model, they used both temporal composition information and visual features of motion segments. [3] used spatio-temporal graphs of an activity class to parse a test video by matching its graph with the closest activity model. [34] achieved online parsing of videos to its actions and sub-actions using specialized grammars, which define temporal structure and can be parsed with a finite-state-machine. [16] jointly performed video segmentation and action recognition using a method based on a discriminative temporal extension of the spatial bag-of-words model. The most similar to our work, [9] segmented actions by using a model that encodes the change in the states of the world.

Orderings in an Activity Sequence: An action is defined by [35, 18] as a sequence of key poses of actors depicting key states. Their model assumes a partial ordering between the key frames, making it tolerant to action duration. [41] proposed a method to learn the partially ordered structure inherent in human everyday activities by analyzing the variability in the data. [40] represented each activity using partially ordered intervals and used Dynamic Bayesian Networks to represent the partial order. [11] extracted the storyline of a video using AND/OR graphs to represent the causal relationships among actions. [19] used co-occurrence graphs to represent the relations among low-level events. In our work we focus on the ordering among actions and propose to use directed graphs, where a relation between two nodes represents an action ordering but the relation defining the graph does not have to be antisymmetric as in the case of partial orders, allowing the inclusion of cycles. Strictly sequential models like HMM do not directly allow inferences about missing actions. P-Nets require temporal and logical relationships between nodes to be engineered manually, which also does not allow circles.

3. Approach

Our end goal is to notify users when they forget to perform an important action before the ongoing one ends. In order to do this we need to: (1) recognize what the user has already done so far, (2) determine what the user is about to do, (3) identify the missing actions, (4) compute the corresponding cost of missing them, and (5) use this cost to generate reminders. In this framework, the first two tasks require performing action recognition and prediction together with segmentation, (3) and (4) require extracting inter-action dependencies, and (5) requires a notification decision mechanism.

Suppose we are given the segmentation of actions, and we have a function $F(t)$ that for each small window L before time t can give us the action category ($F(t) = a_j \in A = \{a_1, a_2, ..., a_M\}$, the set of all actions), and a function $\psi(t, F(t))$, that for each time t and the given action category $F(t)$, provides the state of the progression of the action:

$$\psi(t, F(t)) = \begin{cases} -1 & if\ t \in B_{F(t)} \\ 0 & if\ t \in M_{F(t)} \\ 1 & if\ t \in E_{F(t)} \end{cases}$$

where B, M, E, represents the action beginning, middle and end states. Let t_+ be the next time step after t; $t_+ = t + L$. Then we can formulate a scoring function $\mathcal{Z}$ for the utility of issuing a reminder at time t_+ as:

$$\mathcal{Z}(t_+) = \Delta(t) * [\mathcal{C}(F(t_+), \mathcal{N}(F(t), F(t_+))) + \lambda * \alpha] \quad (1)$$
$$\Delta(t) = -\psi(t_+, F(t_+)) * \psi(t, F(t))$$

where $\mathcal{C}(a_j, a_q)$ is the cost of performing action a_q after a_j, $\mathcal{N}(a_i, a_j)$ returns the first missing action between the last completed action a_i and the predicted one a_j, $a_i, a_j, a_q \in A$. $\Delta(t)$ is a function that specifies the candidate times for issuing a reminder; these are the times a prediction is made after a completed action. α is a constant penalty for a reminder about an unnecessary action or for missing a required reminder, and λ is a trade off factor. The value of $\mathcal{Z}(t_+)$ at time t_+ determines if a notification should be given.

In order to obtain a list of missing actions and compute $\mathcal{N}$ and $\mathcal{C}$, we need to extract the dependencies between actions and calculate the cost of performing one action after another. We use action orderings to model the inter-action dependencies.

3.1. Extracting Dependencies Between Actions:

Every individual might follow a different order of actions while completing an activity. For example, one person might prefer to add milk after adding espresso, while the other might prefer to do the opposite. We represent the space of all dependencies between actions as a possibly cyclic, directed graph [2], which we call a *flexible ordered graph* [3].

A flexible ordered graph is a directed graph $G = (V, E)$, in which the vertices V represent actions and the weighted directed edges E represent ordering constraints (Figure 2). The flexible ordered graph is transitive, possibly cyclic but not reflexive. Having cycles does not hurt the process but rather gives the flexibility to complete some action sequences multiple times. For example, a person might decide to add more milk after having added some.

Flexible Ordered Graph Construction: Assume that we have a set of videos, $S = \{S_1, S_2, ..., S_K\}$. To construct a

[2] Different from partially ordered graphs, which do not allow cycles.

[3] To make the solution simpler, we do not allow self-loops although our model can be easily adjusted to handle them.

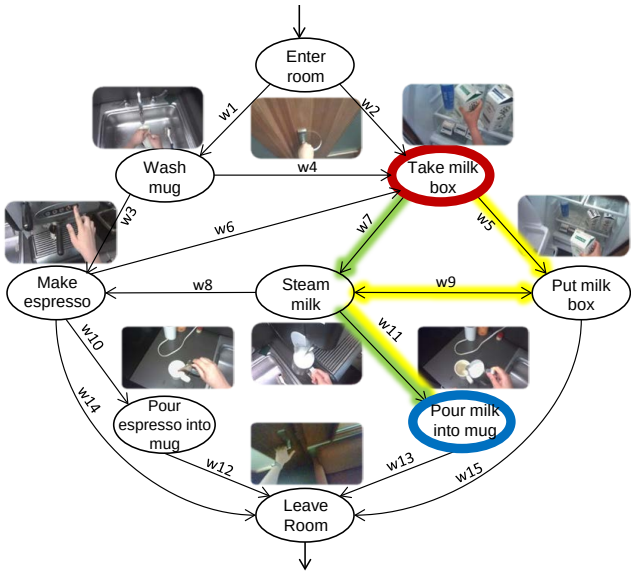


Figure 2: An example flexible ordered graph, where the most recent action is `take milk box`, marked in red, and the predicted action is `pour milk into mug`. The possible paths from the most recent action to the predicted one are marked in green and yellow.

flexible ordered graph, we use the *complete set of actions*, which is assumed to contain possible action orderings when the activity is completed *correctly*. Over this set, a flexible ordered graph is constructed by computing the transition probabilities, T, between every pair of actions (a_i, a_j). $T(a_i, a_j)$ shows the transition probability from action a_i to a_j. For each action pair (a_i, a_j), if a_j follows a_i, an edge $e_{ij} \in E$ is created from v_i to v_j; $v_i, v_j \in V$. The weight of each edge $w(v_i, v_j) = w(e_{ij}) = 1/T(a_i, a_j)$.

For the latte making activity, the flexible ordered graph has fixed entrance and exit states, namely, `entering the room` and `leaving the room`. An example flexible ordered graph is given in Figure 2.

Cost of performing one action after another ($\mathcal{C}$): We calculate the cost of taking action a_q after a_j, $\mathcal{C}(a_j, a_q)$ as the minimum weighted geodesic distance from a_j to a_q on the flexible ordered graph G:

$$\mathcal{C}_q^j = \mathcal{C}(a_j, a_q) = min_P(\sum_{e \in P} w(e))$$

where $P(a_j, a_q)$ is *any* path from a_j to a_q, and $e \in P$. The path corresponding to $\mathcal{C}(a_j, a_q)$ is represented as P_{jq}^*. Since the weights on the edges are always positive, taking circular paths repeatedly can never produce a value less than taking the direct path once.

Determining missing actions ($\mathcal{N}$): Assume that the most recently completed action is a_i, and our method predicted that the user is about to perform action a_j. We compute $\mathcal{C}(a_j, a_q)$, where a_q is the first action on the path P_{ij}^*. If the value of $\mathcal{C}_q^j$ is high enough, we report the actions on P_{ij}^* as missing. For example in Figure 2, after `take milk box`, our system predicted that the user is about to `pour milk into mug`. The action `steam milk` is then reported as a missing action, because it is on the minimum cost path between `take milk box` and `pour milk into mug` and has a high cost of being missing.

3.2. Coupled Prediction and Segmentation Module

Calculation of $\mathcal{Z}$, in Equation 1, requires F and ψ, which in turn requires action segmentation, recognition and, prediction available. So far, these were assumed. In this section we describe a model for predicting what the user is about to do from a video stream. This requires knowing the boundary of the previous action and predicting the ongoing action from the very first few frames.

Segmentation Using Action Part Classifiers: For a streaming unsegmented video sample, we designed a discriminative Hidden Markov Model (HMM), which infers the past (recognition) and the current (prediction) action categories, while segmenting the activity sequence. In addition, it provides the current progress of the predicted action: beginning, end or middle.

In this model, the states are different action parts belonging to all action categories. These actions are connected according to their reachability. Actions parts can be in one of the three categories: action beginning (corresponding to the first L frames of an action), action end (corresponding to the last L frames of an action) and action middle (corresponding to each L consecutive frames between the action beginning and end parts). For a particular action a_i, the probability of reaching to action middle (M_{a_i}) and action end (E_{a_i}) states from the action beginning (B_{a_i}) state depends on the length of the action. In addition to in-class reachability, the end state of every action class can reach to the beginning states of other action classes according to the transition probabilities obtained from the full training dataset, which consists of both complete and incomplete training samples. Figure 3 shows the states and the allowable transitions in the model, where the observation probabilities come from the action beginning, middle and end classifiers described in the next section.

Action Part Classifiers: For training *action part* classifiers, we first partition the action samples in the training data into parts: beginning, middle and end. Figure 4 shows this partition for different action part classifiers. N represents an arbitrary length of an action. We only use the first L

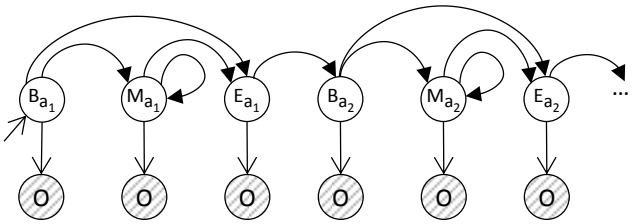


Figure 3: Our coupled HMM model for segmentation and prediction. The states are beginning, middle and end, and the observations come from action part classifiers.

frames of actions for training action beginning classifiers. For training the action end classifiers, we use the last L frames of each action, and we use all remaining L frames in each action as a separate sample for training the action middle classifiers. We use linear SVM's for the training of all action part classifiers and calibrate using the method of [39].

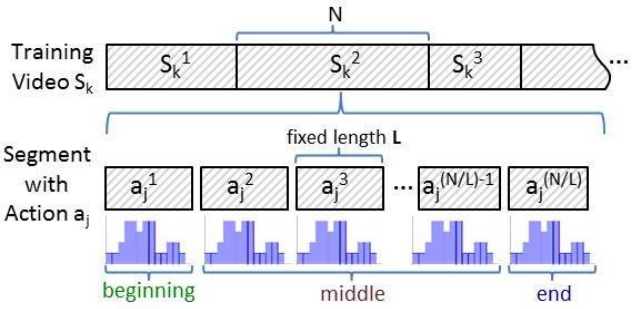


Figure 4: Different parts of training data are used for training different action part classifiers: beginning, middle, end.

Inference: During the inference process, we need to solve recognition, segmentation and prediction at the same time. A given input video S_k^{test} is divided into non-overlapping parts of length L and fed into the part classifiers. Given our model with the observations from the part classifiers, this problem is solved using the Viterbi algorithm. This procedure not only finds the action boundaries and recognizes past actions, but also predicts the ongoing action from partial observations. Figure 5 illustrates this procedure.

4. Experiments

Our purpose is to give proper reminders to users when they forget to perform some actions. For this purpose we design a model, which has two main functionalities: (1)segmenting, recognizing previous and predicting the current action at the same time, (2) making a notification decision based on the segmentation, recognition and prediction.

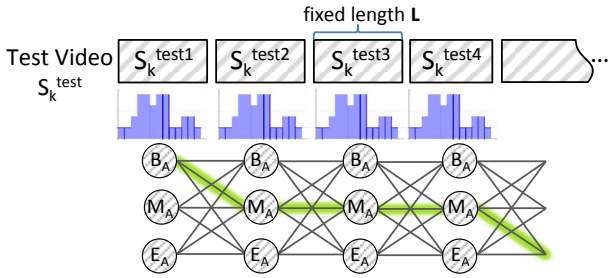


Figure 5: Inference: Each test video is divided into fixed length parts and feed to part classifiers. In the Viterbi trellis B_A, M_A, E_A nodes represent 29 class action beginning, middle and end classifiers, respectively, and every edge represents 29x29 connections. An example output path is highlighted in green.

Dataset: To the best of our knowledge, there is no existing dataset, which is designed for notification purposes. Egocentric settings are more suitable for understanding daily activities as the camera can move around with the person. For this reason, we collected a new dataset using a Looxcie HD camera mounted on a cap, which can record 1920x1080 resolution in $\sim 30fps$. We have recorded 41 videos of about 20 subjects, some of whom made lattes multiple times in different days. The total number of frames in the dataset is $344,173$. During data collection, all subjects were asked to make lattes according to their own preferences; therefore the action order and duration, and the actions in each video vary. Our only request to the subjects was to look at the action of interest while performing the action, in order to avoid irrelevant frames. However, in many cases the subjects still looked around, making the dataset more challenging. Another challenge in the dataset is simultaneous actions; we noticed that people tend to perform multiple actions at the same time. For instance, while making espresso, they may steam milk at the same time. Simultaneous action recognition/prediction is out of scope of this paper, and we labeled those parts of videos with only one of the simultaneous actions. Example frames from the collected dataset is given in Figure 6, and the full list of actions is given in Table 1. In our experiments, we downscaled the videos to 480x270 resolution and sampled one out of every two frames, resulting in a total of 108,159 frames after irrelevant ones were removed.

Metrics: We use a variety of metrics to evaluate the proposed notification system, and to analyze different parts and alternative approaches. Our goal is to be as precise as possible for each user. Therefore, our performance numbers are the average over the scores of each sample unless otherwise stated. Moreover, since recognizing each action is very im-

Figure 6: Example frames from the collected egocentric dataset of latte making activity.

clean portafilter	clean espresso pitcher
clean nozzle	clean milk pitcher
wash mug	close fridge
flatten milk box	steam milk
enter room	grind coffee
leave room	make espresso
open fridge	pour coffee into mug
take coffee portafilter	pour milk into mug
pour milk into pitcher	put jacket
put keys	put milk box back to fridge
throw milk box	put milk pitcher
take jacket	take keys
take milk box	take milk pitcher
turn off lights	turn on lights
stir	

Table 1: Actions of latte making activity.

portant for giving proper notifications, we believe average per class accuracy is more important than the average accuracy for the evaluation purposes (despite the fact that our model produces higher results for average accuracy). Note that, this is a multi-class problem and accuracy is calculated over all classes simultaneously.

Experimental Setup: In all of our experiments we use 23 complete samples, where there is no missing action as our fixed part of training data. We then use 18 incomplete samples with missing actions for testing in a leave-one-out-cross-validation fashion. We use our training data such that every sample of an action has exactly one action beginning part, exactly one action end part, and zero or more action middle parts, whose lengths are equal to 10 frames. During the inference time, a test video sequence is divided into fixed length parts of 10 frames.

Features: The visual data obtained from egocentric videos are drastically different from the data obtained from static cameras. Although the high camera movement causes jitter and requires camera motion handling and perspective distortions, there is an advantage to the egocentric

	Avg. Acc.	Avg. per Class Acc.
STIP	58.61	57.29
GIST	69.74	67.94

Table 2: Accuracies of the STIP vs the GIST features in recognizing actions using a Discriminative HMM.

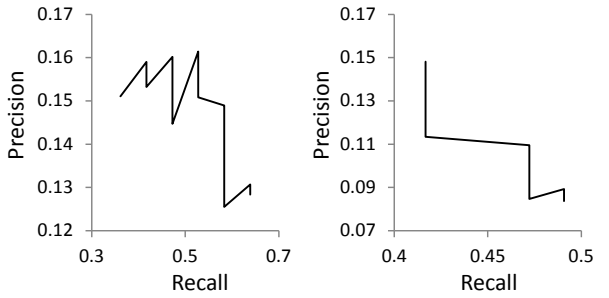
cameras: the scene changes with the action as the camera moves. Therefore, the scene-based representations such as GIST [32] features are useful. In our experiments we use GIST as our feature representation and extend it to video part representation using the bag of words (BoW) paradigm, with 500 words. To show the benefit of GIST, we also experiment with space-time-interest-point features (STIP) [25]. For comparison purposes, we use segmented videos and represent each action using BoW with a dictionary size of 500 words. Table 2 shows comparisons between GIST and STIP in action recognition on our dataset using discriminative HMM. Dense Motion Trajectory [42] features in general are not suitable in egocentric settings because of continuous drastic camera movement.

4.1. Evaluation

We evaluate the proposed system in two tasks: 1) Evaluation of the notification module. 2) Evaluation of the prediction and segmentation model.

Notification Module: Our system is designed to give appropriate notifications, while the users continue their activities. Since the activity of latte making always ends with the `leave room` action, we also check for the required actions that are not completed until a `leave room` action is detected. Some of these reminders can be previously given when a missing action is detected. The cost of any required action after performing the `leave room` action is set to infinity, always resulting in a notification when it is missed. Required actions are defined in two parts: the first part consists of the minimum set of actions, $A^{min} \subset A$, that are manually determined as required for latte making activity to be complete, and the second part is updated online at every new prediction, according to automatically calculated co-occurrences with other actions. (For example, an `open fridge` action always co-occurs with a `close fridge` action. When an `open fridge` action is detected, a `close fridge` action is added to the set of required actions.). For the calculation of co-occurrences, we use the complete set defined above. When a required action is completed, we remove it from the set.

The accuracy of our notification system can be measured in 3 ways: 1) Existence of a reminder: We measure if we give a reminder for the samples that need one regardless of



(a) Alarm correctness (b) Alarm correctness & timeliness

Figure 7: Evaluation of the notification module: (a) shows the precision-recall curve for notification correctness evaluation regardless of time. (b) shows the precision-recall curve for giving the correct notification at the right time.

time and action type. Our system gives a notification whenever it is required, precision and recall are equal to 1 for all samples. 2) Correctness of the type of the reminder: We measure if our method give the proper notification regardless of timeliness. 3) Timeliness of reminder: We measure both the correctness and timeliness of the notification. The timeliness of a notification is measured in a δ neighborhood. This means that if a reminder is given in the δ neighborhood of the groundtruth time, it is counted as correct. In our experiments, δ is set as 25 frames. Figure 7 represents the precision-recall curves for the last two evaluations, when $\alpha = 0.5, \lambda = 1$. Note that, issuing the correct and timely reminders is extremely challenging. As a baseline, we generated random notifications by making a random notification decision at the end of every action part. Over 10 random notification generations, the average precision and recall we could get are 0.0007 and 0.1, respectively.

Segmentation and Prediction Model: Table 3 shows the performance of our model when the whole sequence is already observed using different metrics, both in partwise and framewise. When we evaluate partwise, we assign the groundtruth label of a part with max pooling. The first column shows the average accuracy of all samples. The second column shows the accuracy calculated by concatenating the recognition labels and comparing with the concatenated groundtruth. This evaluation is biased towards the results of the longer sequences. First and second rows give average accuracy and average per class accuracy respectively, over parts. The third and fourth rows give the framewise average accuracy and framewise average per class accuracy, respectively.

Baselines: Tables 4, 5, 6 show different baselines.

To explore our model in different settings, we first calculate the prediction accuracy. For this purpose, we use the groundtruth locations of action beginning states for deciding prediction times. Figure 8 shows an example scenario, in which the prediction times are represented by arrows, brackets show groundtruth action boundaries, and the red boxes are the parts on which a prediction is made. The first line of Table 4 shows the results of the action beginning classifier, which does not use temporal information. The second line shows the results when we use an ordinary discriminative HMM, where the observations come from the action beginning classifier. In both of these experiments the segmentation is provided.

	Avg. Over Samples	Overall Avg.
Accuracy	72.66	73.82
Avg. per Class Acc.	63.39	64.78
Framewise Acc.	72.25	73.42
Framewise Avg. per Class Acc.	61.96	63.91

Table 3: Accuracies of our prediction and segmentation model when L=10 and the whole sequence is observed.

Row 3 of Table 4 shows the results for coupled prediction and segmentation. Prediction times are the same as the first two experiments and the red boxes in Figure 8 show the corresponding parts. Row 4 of Table 4 shows the results of online prediction. We measure the online prediction accuracies for the labels predicted at each time step. As more frames are observed, the earlier predictions might be updated, but the accuracies are computed only on the current predictions; the green boxes in Figure 8 show the corresponding parts. This is the evaluation suitable for a real life scenario, since we make decisions as we observe frames. As another experiment, we use the whole sequence to recognize and segment actions. The last row in Table 4 shows the corresponding results.

We evaluated our part classifiers to see how well each represented its class. Table 5 show their accuracies, when we assume the segmentation is given and the test input is from the related part of the action. The values in the last row are equal to the action prediction accuracies, when the segmentation is given and temporal information is not used (Table 4, row 1).

Our model is designed to predict actions for unsegmented videos. We compare our model with max-margin early event detectors (MMED) [13, 15], which is a modified SOSVM model, detecting the actions in an unsegmented video before they complete. At each step [13] detects only the first instance of an action. In order to detect multiple instances of the same class in a sample, we input the remaining video after every detection. We used the same features

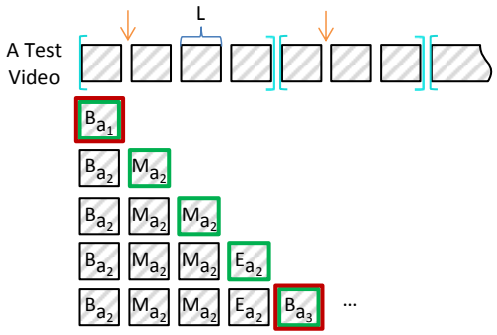


Figure 8: Prediction is evaluated on the action beginning parts, shown by red boxes. Online prediction evaluation is done over the predicted labels of parts shown by green boxes. The labels in the boxes show the predictions at each time step. Arrows show the times prediction is evaluated. Brackets show groundtruth action boundaries.

	Avg. Acc.	Avg. per Class Acc.
Prediction, not temporal *(segmentation is given)*	51.53	48.80
Prediction, temporal *(segmentation is given)*	72.04	70.04
Prediction, temporal *(segmentation is unavailable)*	46.19	44.06
Online Prediction, temporal *(segmentation is unavailable)*	68.32	56.18
Recognition, temporal *(segmentation is unavailable)*	72.66	63.39

Table 4: Accuracies of prediction models, in different settings. For all models, the prediction scores represent when only the first L=10 frames of actions are observed. Online prediction scores use the predictions after every part of L=10 frames.

	Avg. Acc.	Avg. per Class Acc.
Action End Classifier	48.77	46.27
Action Middle Classifier	59.25	47.09
Action Beginning Classifier (Prediction)	51.53	48.80

Table 5: Accuracies of Part Classifiers, when the segmentation is given, L=10.

as our method. The major drawback of the early event detection approach over coupled prediction and segmentation is that every detection of action is done separately for each class; therefore, there can be overlapping frames in the detected action locations. We measure the framewise F value

	Precision	Recall	F measure
MMED	0.32	0.25	0.25
Our Model	0.62	0.57	0.58

Table 6: Our model vs MMED. We use L=10 when evaluating our model.

of each class, and take the average over classes, then compare with that of our method. Our method showed superior performance in these experiments. The results are given in Table 6.

5. Conclusion

Prospective memory failures exist in all of our lives. Forgetting an action might cause serious and expensive consequences. In this paper, we introduce a framework that issues notifications on the actions that may be missed during a sequence of an activity. To show a proof of concept, we collected an egocentric dataset of people making lattes and showed that our method can produce action reminders. We evaluated our notification model by means of correctness and timeliness and showed promising results. To further analyze our model, we also evaluated our recognition, prediction and segmentation models and showed comparisons to state-of-the-art approaches.

Producing accurate and timely reminders is an important but extremely challenging problem and encourages new research directions. To reason about missing actions and their associated costs, we used temporal dependencies between actions represented by our flexible ordered graph. Our approach does not take into account semantics of dependencies between activities in terms of causalities, preconditions and effect. Future work involves deeper understanding of inter-action dependencies and discovering these challenging dependencies. Furthermore, producing action reminders requires solving segmentation, recognition, and prediction at the same time. We proposed one solution to this problem that couples these tasks at a higher level. Tighter coupling at lower levels is another promising direction to be explored. Finally, generating action reminders requires a complex decision making component. Our solution formulates this problem with a scoring function that trades off between the cost of missing action and the penalty for issuing a reminder. Devising suitable reward functions for more complex decision mechanisms (such as MDPs) that can take into account future possibilities is the next big step.

Acknowledgments: This work was partially supported by ONR N00014-13-1-0720, NSF IIS-1218683, NSF IIS-IIS- 1338054, and Allen Distinguished Investigator Award.

References

[1] Latte. http://www.oxforddictionaries.com/definition/english/latte. 1

[2] M. Brand and V. Kettnaker. Discovery and Segmentation of Activities in Video. *PAMI*, 2000. 2

[3] W. Brendel and S. Todorovic. Learning spatiotemporal graphs of human activities. In *ICCV*, 2011. 3

[4] Y. Cao, D. Barrett, A. Barbu, N. Siddharth, H. Yu, A. Michaux, Y. Lin, S. Dickinson, J. M. Siskind, and S. Wang. Recognizing human activities from partially observed videos. In *CVPR*, 2013. 2

[5] R. K. Dismukes. Prospective Memory in Workplace and Everyday Situations. *Curr. Dir. Psychol.*, 2012. 2

[6] C. Ellis, S. Z. Masood, M. F. Tappen, J. J. Laviola, Jr., and R. Sukthankar. Exploring the trade-off between accuracy and observational latency in action recognition. *IJCV*, 2013. 2

[7] A. Fathi, A. Farhadi, and J. M. Rehg. Understanding egocentric activities. In *ICCV*, 2011. 2

[8] A. Fathi, Y. Li, and J. M. Rehg. Learning to recognize daily actions using gaze. In *ECCV*, 2012. 2

[9] A. Fathi and J. M. Rehg. Modeling actions through state changes. In *CVPR*, 2013. 3

[10] E. Fox, E. Sudderth, M. Jordan, and A. Willsky. Bayesian nonparametric inference of switching dynamic linear models. *IEEE Signal Processing*, 2011. 2

[11] A. Gupta, P. Srinivasan, J. Shi, and L. Davis. Understanding videos, constructing plots learning a visually grounded storyline model from annotated videos. In *CVPR*, 2009. 3

[12] J. D. Henry, P. G. Rendell, L. H. Phillips, L. Dunlop, and M. Kliegel. Prospective memory reminders: A laboratory investigation of initiation source and age effects. *The Quarterly Journal of Experimental Psychology*, 2012. 2

[13] M. Hoai and F. De la Torre. Max-margin early event detectors. In *CVPR*, 2012. 2, 7

[14] M. Hoai and F. De la Torre. Maximum margin temporal clustering. In *AISTATS*, 2012. 2

[15] M. Hoai and F. De la Torre. Max-margin early event detectors. *IJCV*, 2014. 2, 7

[16] M. Hoai, Z.-Z. Lan, and F. De la Torre. Joint segmentation and classification of human actions in video. In *CVPR*, 2011. 3

[17] D.-A. Huang and K. M. Kitani. Action-reaction: Forecasting the dynamics of human interaction. In *ECCV*, 2014. 2

[18] N. Ikizler and D. A. Forsyth. Searching video for complex activities with finite state models. In *CVPR*, 2007. 3

[19] H. Izadinia and M. Shah. Recognizing complex events using large margin joint low-level event model. In *ECCV*, 2012. 3

[20] T. Kanade and M. Hebert. First-person vision. *Proceedings of the IEEE*, 2012. 2

[21] K. M. Kitani. Ego-action analysis for first-person sports videos. *IEEE Pervasive Computing*, 2012. 2

[22] K. M. Kitani, B. D. Ziebart, J. A. D. Bagnell, and M. Hebert. Activity forecasting. In *ECCV*, 2012. 2

[23] Y. Kong, D. Kit, and Y. Fu. A discriminative model with multiple temporal scales for action prediction. In *ECCV*, 2014. 2

[24] T. Lan, T. Chen, and S. Savarese. A hierarchical representation for future action prediction. In *ECCV*, 2014. 2

[25] I. Laptev. On space-time interest points. *IJCV*, 2005. 6

[26] B. Laxton, J. Lim, and D. Kriegman. Leveraging temporal, contextual and ordering constraints for recognizing complex activities in video. In *CVPR*, 2007. 2

[27] K. Li and Y. Fu. Prediction of human activity by discovering temporal sequence patterns. *PAMI*, 2014. 2

[28] K. Li, J. Hu, and Y. Fu. Modeling complex temporal composition of actionlets for activity prediction. In *ECCV*, 2012. 2

[29] T. McCandless and K. Grauman. Object-centric spatio-temporal pyramids for egocentric activity recognition. In *BMVC*, 2013. 2

[30] T. MORITA. Reminders supporting spontaneous remembering in prospective memory tasks1. *Japanese Psychological Research*, 2006. 2

[31] J. C. Niebles, C.-W. Chen, and L. Fei-Fei. Modeling temporal structure of decomposable motion segments for activity classification. In *ECCV*, 2010. 3

[32] A. Oliva and A. Torralba. Building the gist of a scene: the role of global image features in recognition. In *Progress in Brain Research*, 2006. 6

[33] H. Pirsiavash and D. Ramanan. Detecting activities of daily living in first-person camera views. In *CVPR*, 2012. 2

[34] H. Pirsiavash and D. Ramanan. Parsing videos of actions with segmental grammars. In *CVPR*, 2014. 3

[35] M. Raptis and L. Sigal. Poselet key-framing: A model for human activity recognition. In *CVPR*, 2013. 3

[36] M. S. Ryoo. Human activity prediction: Early recognition of ongoing activities from streaming videos. In *ICCV*, 2011. 2

[37] M. S. Ryoo, T. J. Fuchs, L. Xia, J. Aggarwal, and L. Matthies. Robot-centric activity prediction from first-person videos: What will they do to me? In *HRI*, 2015. 2

[38] M. S. Ryoo and L. Matthies. First-person activity recognition: What are they doing to me? In *CVPR*, 2013. 2

[39] W. J. Scheirer, A. Rocha, R. Michaels, and T. E. Boult. Meta-recognition: The theory and practice of recognition score analysis. *PAMI*, 2011. 5

[40] Y. Shi, Y. Huang, D. Minnen, A. Bobick, and I. Essa. Propagation networks for recognition of partially ordered sequential action. In *CVPR*, 2004. 3

[41] M. Tenorth, F. D. la Torre, and M. Beetz. Learning probability distributions over partially-ordered human everyday activities. In *ICRA*, 2013. 3

[42] H. Wang, A. Kläser, C. Schmid, and C.-L. Liu. Action Recognition by Dense Trajectories. In *CVPR*, 2011. 6

[43] M. Zanfir, M. Leordeanu, and C. Sminchisescu. The moving pose: An efficient 3d kinematics descriptor for low-latency action recognition and detection. In *ICCV*, 2013. 2

[44] F. Zhou, F. De la Torre, and J. K. Hodgins. Hierarchical aligned cluster analysis for temporal clustering of human motion. *PAMI*, 2013. 2