A method for online Analysis of structured Processes using Bayesian Filters and Echo State Networks

Dimitrios I. Kosmopoulos and Fillia Makedon

University of Texas at Arlington,
Computer Science and Engineering, TX 76013, USA
dkosmo@ieee.org,makedon@uta.edu

Abstract. We propose a Bayesian filtering framework for online analysis of visual structured processes, which can be combined with the Echo State Network (ESN) to capture prior information. With the proposed method we mitigate the effective Markovian Behavior of the ESN. We are able to keep a set of hypotheses about the entire history of behaviors and to evaluate them online based on new observations. The performance is evaluated under two complex visual behavior understanding scenarios using public datasets: a visual process for a kitchen table preparation and a real life manufacturing process.

1 Introduction

Lately the visual analysis of workflows as part of industrial or other processes has been gaining momentum, see, e.g., [1], [2], [3], [4], [5]. A workflow is a structured process that occurs repetitively and consists of a sequence of discrete tasks that need to be recognized. The task order follows some statistically consistent patterns, which can be modelled as priors.

Our goal is the online labeling of each video frame given a set of available task labels. The online classification is performed by combining a probabilistic framework with an online supervised time series classifier such as the Echo State Network (ESN) [6]. The sole use of a probabilistic framework can only capture the dynamics between tasks but cannot solve the problem, since we need to know the statistics of each individual task. On the other hand, the sole use of ESNs can model adequately each individual task but cannot model accurately the interaction between tasks because the ESNs have a behavior close to Markovian (see, e.g., [7]), which is not the case for the sequence of industrial tasks.

Contribution We present a method to enhance the performace of the ESN classifier for workflows. Specifically, we combine the advantages of the ESN and of a probabilistic framework based on particle filtering and we exploit prior knowledge about task dynamics and workflow hierarchy. We aim to show that this framework works better than (a) a classifier that makes an explicit Markovian assumption (Hidden Markov Model - HMM) and (b) a classifier that effectively behaves as a Markovian one (ESN). The approach is applicable to any online classifier that calculates observation probabilities.

2 Related Work

A popular framework for time series analysis is the HMM (see e.g., [8]). It is efficient for application in previously segmented sequences (see e.g., [1]), however when the boundaries of the sequence are not known the search very inefficient [9]. In [10] a dynamic programming algorithm of (restrictive) cost T^3, is used to segment and then classify the sequences. In [11] action sequences are segmented using a margin criterion, however Markovian behavior is assumed.

Here we examine workflows which are composed of tasks and therefore are of hierarchical nature. Typical approaches that exploit the hierarchical structure of time series, are the hierarchical HMMs [12] or the layered hidden Markov model (LHMM) ([13]). Examples of such approaches for visual workflows can be found in [3]. In many workflows, the whole history of tasks is required, because it affects the appearance of future tasks (tasks that are executed are not expected and tasks not already executed are expected to appear later); therefore the Markovian assumption is not correct and this fact motivates methods that relax this assumption. In [14] Rao Blackwell particle filters were used along with a dynamic Bayesian network for tracking of hierarchical events. In [15] a method based on a Bayesian filter and HMMs was proposed for visual analysis. In [16], [17] the utility of particle filters in combination with an HMM is noticed, however, in these works observation predictions are performed, which is different from our online classification problem.

The ESN, see, e.g., [18], is a very promising method for general online classification of time series. It has been proved to be more robust than the HMM (see, e.g., [19], and TDNN [20]. It offers several benefits such as (i) fast and simple learning of many outputs simultaneously, (ii) possibilities of both off-line and on-line learning and testing, (iii) ability to learn complex dynamic behaviors, and (iv) directly dealing with high dimensional input data.

The ESN appears to be an attractive option for analysis of workflows (see e.g., [2]), since it does not make any explicit Markovian assumptions. However, it was shown in [7] that it effectively behaves as a Markovian classifier, i.e., recent states have a far larger influence on the predicted state. Therefore some additional methods are required for workflow analysis, to mitigate this effect.

In [21] an ESN was used to classify a whole sequence after the tasks get completed and segmented. Unlike in [21] here we assign a task label for each frame online, without waiting for the tasks to finish and we do not need to perform any explicit segmentation (thus no related training is required).

3 The online Classification Framework

We denote as $\mathbf{x}_t$ the state vector including the label l_t from the L classes (tasks) that has to be assigned to frame t. Our goal is to calculate the posterior $p\left(\mathbf{x}_{0:t}|\mathbf{o}_{1:t}\right)$ at every t, given the measurements $\mathbf{o}_{1:t}$(visual observations) up to t. *We emphasize that the* $\mathbf{x}_{0:t}$ *denotes the label sequence for the whole workflow history, and our ultimate goal is to calculate this exact sequence.*

A very attractive option is the ESN, which is a discrete time, continuous state, recurrent neural network proposed in [6]. Learning complexity is kept low while good generalization can be achieved on various dynamic problems. The hidden layer consists of N randomly connected neurons (N is typically in the order of a few hundred or several thousands). If the connectivity is low, this layer provides independent output trajectories. For this reason, the hidden layer is also called a “reservoir”. Furthermore, there are neurons which are connected to cycles in the reservoir, so that past states “echo” in the reservoir.

The big advantage of the ESN is that it can be considered as a black box that can label the data sequences in a online fashion, i.e., given a sequence of observations $\mathbf{o}_{1:t}$, it can calculate the label $\mathbf{x}_{t+1}$ for the current observation $\mathbf{o}_{1:t}$ (it is assumed that the nework is trained properly). To mitigate the effectively Markovian behavior of the ESN, i.e., recent states play far more important role, we propose to use particle filters to capture the entire label history. A possible method to calculate $p\left(\mathbf{x}_{0:t}|\mathbf{o}_{1:t}\right)$ is by employing a Bayesian filter, commonly expressed as:

$$p\left(\mathbf{x}_{0:t}|\mathbf{o}_{1:t}\right) = p\left(\mathbf{x}_{0:t-1}|\mathbf{o}_{1:t-1}\right) \frac{p\left(\mathbf{o}_t|\mathbf{x}_{0:t}, \mathbf{o}_{1:t-1}\right) p\left(\mathbf{x}_t|\mathbf{x}_{0:t-1}, \mathbf{o}_{1:t-1}\right)}{p\left(\mathbf{o}_t|\mathbf{o}_{1:t-1}\right)} \quad (1)$$

To estimate the fraction term of the right part of equation (1) we proceed as follows. *First*, the term $p\left(\mathbf{o}_t|\mathbf{o}_{1:t-1}\right)$ is independent of the class to which the current observations should be assigned to, so it can be omitted from the following calculations. *Second*, it is reasonable to assume that the current observation $\mathbf{o}_t$ depends only on the current task $\mathbf{x}_t$, so we simplify $p\left(\mathbf{o}_t|\mathbf{x}_{0:t}, \mathbf{o}_{1:t-1}\right)$ to $p(\mathbf{o}_t|\mathbf{x}_t)$. *Third*, for the term $p\left(\mathbf{x}_t|\mathbf{x}_{0:t-1}, \mathbf{o}_{1:t-1}\right)$ we propose an alternative expression, which is simply $p(\mathbf{x}_t|\mathbf{x}_{0:t-1})$; the latter holds because if the task history is known then the observation history does not affect the appearance of the next task. It is also reasonable to assume that in an industrial setting each task has a duration, which can be expressed by a probabilistic function, which can be learned. Therefore the state vector $\mathbf{x}_t$ is decomposed to $\mathbf{x}_t = (x_t^l, x_t^D)$, where x_t^l is the label of the current observations (in other words the task/class to which the current observations are assigned) and x_t^D is the residual duration of the current task.

Under these assumptions equation (1) simplifies to:

$$p\left(\mathbf{x}_{0:t}|\mathbf{o}_{1:t}\right) \propto p\left(\mathbf{x}_{0:t-1}|\mathbf{o}_{1:t-1}\right) \cdot p\left(\mathbf{o}_t|\mathbf{x}_t\right) p\left(\mathbf{x}_t|\mathbf{x}_{0:t-1}\right) \quad (2)$$

As observed in equation (2), the posterior probability $p\left(\mathbf{x}_{0:t}|\mathbf{o}_{1:t}\right)$ is proportional to a) the recurrent term $p\left(\mathbf{x}_{0:t-1}|\mathbf{o}_{1:t-1}\right)$, b) the probability $p(\mathbf{o}_t|\mathbf{x}_t)$, which expresses the observation model for each task, i.e., observations probability coming from the time series classifier c) the $p(\mathbf{x}_t|\mathbf{x}_{0:t-1})$, which expresses our a priori knowledge about task duration and transition from one state to another. Given the simplification of the general framework in (2), in the following we propose ways to integrate in a real time classification framework observations from multiple streams (see (b)) and prior knowledge (see (c)).

3.1 Integrating a-priori knowledge

The estimation of the probability $p\left(x_{0:t}|\mathbf{o}_{1:t}\right)$ requires the a priori knowledge about task *duration* and *transition* between tasks.

The *duration* d of a task k is stochastic and can be represented as a pdf $p_k(d)$. We used offline trained Gaussian mixture models to represent prior information about expected duration of tasks. In our notation we use the concept of residual duration of a task (the remaining time until the finalization of the task). The residual duration is denoted as x_t^D. As presented in (3) when we first enter task k, x_t^D is set to a value sampled from the pdf $p_k(d')$; then the residual duration decrements to zero. When x_t^D becomes zero then the task will change according to the probabilities given by a decision tree $T_{\mathbf{x}_{0:t}}$ (see (4)).

$$P(x_t^D = d'|x_{t-1}^D = d, x_t^l = k) = \begin{cases} p_k(d') & if\ d = 0\ (reset) \\ \delta(d', d-1)\ if\ d > 0\ (decrement) \end{cases} \tag{3}$$

$$P(x_t^l = j|x_{t-1}^l = i, x_{t-1}^D = d) = \begin{cases} \delta(i,j)\ if\ d > 0\ (same\ task) \\ T_{\mathbf{x}_{0:t}} \quad if\ d = 0\ (task \quad transition) \end{cases} \tag{4}$$

$T_{x_{0:t}}$ is a decision tree holding the priors about the *transition* between different tasks given the previous history of tasks. Assume that k_o is the task that appears in order o, given all previous tasks. Then we denote the associated pdf as $p(k_o|k_{0:o-1})$. In case of non zero value for that probability there will be a node at the o-th level of the tree with value equal to k_o; also it will be connected to its parent via an edge of weight equal to $p(k_o|k_{0:o-1})$; the parent (on the $(o-1)$-th level of the tree) has value k_{o-1}. The root of the tree has a virtual task value k_0, while its children indicate the tasks that may start the workflow (the edges are the associated priors). A complete task sequence is represented by a path connecting the root with any of the tree leaves. Given a path P, of L tasks going from k_0 to k_L the probability of $p(P)$ is given by

$$p(P) = \prod_{l=0}^{L} p(k_l|k_{0:l-1}) \tag{5}$$

which is the product of the associated probabilities for all the links in the path. Such a tree can be learned by using a training set of full workflows, and therefore the "legitimate" paths and their probabilities can be specified. More specifically, for each parent node we find the possible successors (descendants) and based on the normalized frequency that a specific child is selected as next task, we assign a probability value to the connecting edge. Since the history of all previous different tasks is maintained by using such a tree, we do not rely on the Markovian assumption. At this point we clarify that $p(k_o|k_{0:o-1})$ by definition indicates only the transitions between different tasks ($k_o \neq k_{o-1}$).

In summary, we devised a representation of $p(\mathbf{x}_t|\mathbf{x}_{t-1})$ by (3), (4), which includes our prior knowledge about the duration and the transition between tasks. That representation is valuable for sampling and hypotheses evaluation, as will be explained in the following.

3.2 Evaluating hypotheses

The main difficulty of solving equation (2) is that it involves dependencies from previous observations (visual descriptors) and classification (observation assignment to one of the L available classes). To handle the dependencies of the posterior probability $p\left(\mathbf{x}_{0:t}|\mathbf{o}_{1:t}\right)$ with the previous frame states (i.e., frame classification), we need first to introduce our set of hypotheses and then to validate them under a probabilistic framework.

Let us assume that we have a set of H available hypotheses (particles). A hypothesis describes a particular combination of the classes that the previous frames have been assigned to, in other words it is a particular assignment of labels for $\mathbf{x}_{0:t}$. Every hypothesis is evaluated through the Bayesian filters, which are estimation methods based on simulation and previous observations [22].

Weights are associated to the hypotheses, expressing the significance degree to the modeling process. Therefore given H hypotheses we have H weighted particle trajectories $\left\{\mathbf{x}_{0:t-1}^{(n)}, w_{0:t-1}^{(n)}\right\}_{n=1}^{H}$. Each of these trajectories approximates the posterior probability $p\left(\mathbf{x}_{0:t-1}|\mathbf{o}_{1:t-1}\right)$ up to time $t-1$. The current weight $w_t^{(n)}$ for the n^{th} hypothesis at the current frame t is estimated through the distribution $w_t^{(n)} = p(\mathbf{o}_t|x_t^{(n)})$

We can estimate the weights $w_t^{(n)}$ if we know a hypothesis, i.e., we know the class x_t^l to which observation t belongs, as well as the respective duration x_t^D. The pdf of the weights derives from the ESN output (the L outputs are normalized to express a probability). The hypothesis about the value of $\mathbf{x}_t$ requires a priori knowledge regarding task duration and order of tasks and is generated by sampling the pdf $p(\mathbf{x}_t|\mathbf{x}_{t-1})$ using (3), (4). The related pdfs are estimated offline (see subsection 3.1).

The hidden system state space (current task-related label) is one-dimensional and discrete, with low number of possible values (equals to the number of possible tasks). Also the duration is actually sampled only during task transition and in the other cases the sampling is trivial because it decrements with probability equal to one. Therefore the method is efficient because only a relatively low number of particles is required. An overview is given in Algorithm 1.

4 Experiments and Results

4.1 The workflow recognition dataset

The workflow recognition dataset [23] depicts real workers on the production line and includes tasks of picking several parts from racks and placing them on a designated cell some meters away, where welding took place. The behaviors (tasks) we were aiming to model are: (1) One worker picks part #1 from rack

Algorithm 1 Proposed Method

```
{OFFLINE TRAINING}
{Decision tree learning}
$T_{x_{0:t}}$ = LearnTree(AllTaskPaths)
{Supervised task learning through ESN}
for $k = 1$ to $NumberOfTasks$ do
    $p_k(d)$ = LearnDuration(AllTaskInstances,$k$)
end for
esn = TrainESN(TrainingStream, Labels)
{ONLINE CLASSIFICATION FRAMEWORK}
while input==TRUE do
    {The loop executes for all $t$}
    F=AcquireFrame()
    $\mathbf{o}_t$ = ProcessFrame(F); {extraction of observations}
    for $k = 1$ to $NumberOfTasks$ do
        $p(\mathbf{o}_t|\mathbf{x}_t = s)$ = CalcObservationProb(esn, $\mathbf{o}_{1,t}, ..., \mathbf{o}_{M,t}$)
    end for
    {for every Hypothesis do}
    for $n = 1$ to $H$ do
        $\mathbf{x}_t^{(n)} = Sample\ p\left(\mathbf{x}_t|\mathbf{x}_{0:t-1}^{(n)}\right)$ {use eq. (3), (4) }
        $Weight\ \mathbf{x}_t^{(n)}$ by: $w_t^{'(n)} = p\left(\mathbf{o}_t|\mathbf{x}_t^{(n)}\right)$
    end for
    for $n = 1$ to $H$ do
        $Normalize$ the weights by: $w_t^{(n)} = \frac{w_t^{'(n)}}{\sum_{n_1=1}^{H} w_t^{'(n_1)}}$
    end for
    Switching-state particles with low weight are set back to previous state.
    for $n = 1$ to $H$ do
        Update $p(\mathbf{x}_{0:t}^{(n)}|\mathbf{o}_{1:t})$ {use eq. (2) }
    end for
    The winner is the particle $n_0 : p(\mathbf{x}_{0:t}^{(n_0)}|\mathbf{o}_{1:t}) \geq p(\mathbf{x}_{0:t}^{(n)}|\mathbf{o}_{1:t}), \forall n \in \{1, ..., H\}$
end while
```

#1 and places it on the welding cell, (2) Two workers pick part #2a from rack #2 and place it on the welding cell. (3) Two workers pick part #2b from rack #3 and place it on the welding cell. (4) A worker picks up parts #3a and #3b from rack #4 and places them on the welding cell. (5) A worker picks up part #4 from rack #1 and places it on the welding cell. (6) Two workers pick up part #5 from rack #5 and place it on the welding cell. The most common task sequences were 1-2-3-4-5-6, 1-2-3-5-4-6 and 1-2-3-4-6-5.

For our experiments, we have used 20 sequences representing full assembly cycles, each one containing each of the defined behaviors and camera 32 from workflow 1. The length of each sequence ranged from 2000 to 4000 frames. We used the provided annotation. The dataset involves several occlusions and partially overlapping views. The features in the dataset were Zernike moments of motion history images with dimension of 31 see, e.g., [1] for details.

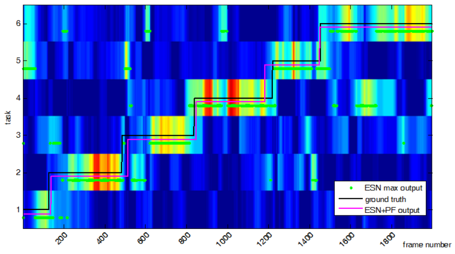


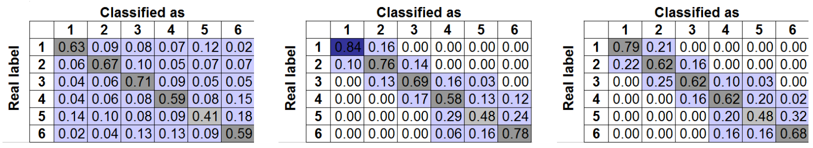
Fig. 1. Heat diagram representing the responses of the 6 ESN outputs for an industrial scenario. Values close to zero are colored deep blue, values close to one are colored red, and intermediate values by the corresponding colors. The maximum outputs are marked by green dots. Superimposed is the ground truth and the ESN+PF response.

The ESN had a linear regression reservoir (changing the ESN type did not favor any of the compared methods) of 500 plain nodes, which was efficient for real time execution, small enough to avoid overfitting and effective. Increasing the number of nodes would result in high memory requirements without real benefit. We had six output nodes, corresponding to six tasks. The median of the last 31 estimations was taken for lower output jitter. We have used the Matlab toolbox provided by the authors [18] using spectral radius 0.60, input scaling 0.3 and smoothing of noise level 0.0003 for optimal results. We trained the ESNs with the entire workflows and applied five-fold cross validation.

For the particle filter we used only 200 particles, which was a good trade-off, and we were able to perform the whole processing at a rate of about 20Hz, the most costly of which was the feature extraction. The confusion matrix per task for a typical case is given in Fig. 2. The learning phase included learning the task durations using a Gaussian mixture model, the task trajectories using a decision tree and the task models using HMMs.

In figure 1 we display the response of the ESN as a heat diagram and we compare it to the ground truth as well as to our ESN+PF approach. More specifically, the 6 outputs of the ESN (corresponding to the six tasks) are normalized, so that they constitute a probability function. Values close to zero are represented by deep blue, values close to one are represented by red, and the intermediate values by the corresponding colors.

The particle that was able to explain best the sequence according to (2) was considered to be the winner. In all cases the work cycle, which consisted of all tasks 1 to 6 was successfully recognized. Clearly our method (ESN+PF) outperforms the simple ESN. The latter tends to assign labels based almost only on the current and the very recent observations ignoring the history of tasks. As expected it behaves efficiently as a Markovian classifier, therefore a task can be misinterpreted as any other task as revealed by in Fig.2. On the contrary the ESN+PF makes estimations which even if not correct they are very close to the currently executed task, giving a better estimation of the executed task sequence.



	Classified as					
Real label	1	2	3	4	5	6
1	0.63	0.09	0.08	0.07	0.12	0.02
2	0.06	0.67	0.10	0.05	0.07	0.07
3	0.04	0.06	0.71	0.09	0.05	0.05
4	0.04	0.06	0.08	0.59	0.08	0.15
5	0.14	0.10	0.08	0.09	0.41	0.18
6	0.02	0.04	0.13	0.13	0.09	0.59

(a) ESN

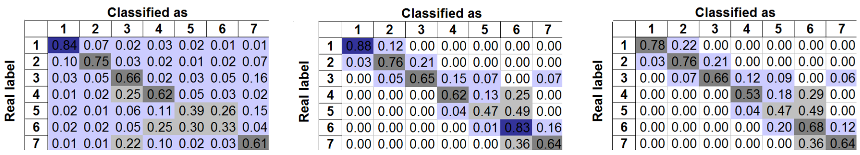
	Classified as					
Real label	1	2	3	4	5	6
1	0.84	0.16	0.00	0.00	0.00	0.00
2	0.10	0.76	0.14	0.00	0.00	0.00
3	0.00	0.13	0.69	0.16	0.03	0.00
4	0.00	0.00	0.17	0.58	0.13	0.12
5	0.00	0.00	0.00	0.29	0.48	0.24
6	0.00	0.00	0.00	0.06	0.16	0.78

(b) ESN + PF

	Classified as					
Real label	1	2	3	4	5	6
1	0.79	0.21	0.00	0.00	0.00	0.00
2	0.22	0.62	0.16	0.00	0.00	0.00
3	0.00	0.25	0.62	0.10	0.03	0.00
4	0.00	0.00	0.16	0.62	0.20	0.02
5	0.00	0.00	0.00	0.20	0.48	0.32
6	0.00	0.00	0.00	0.16	0.16	0.68

(c) HMM+HMM

Fig. 2. The WR dataset: Comparison of confusion matrices



	Classified as						
Real label	1	2	3	4	5	6	7
1	0.84	0.07	0.02	0.03	0.02	0.01	0.01
2	0.10	0.75	0.03	0.02	0.01	0.02	0.07
3	0.03	0.05	0.66	0.02	0.03	0.05	0.16
4	0.01	0.02	0.25	0.62	0.05	0.03	0.02
5	0.02	0.01	0.06	0.11	0.39	0.26	0.15
6	0.02	0.02	0.05	0.25	0.30	0.33	0.04
7	0.01	0.01	0.22	0.10	0.02	0.03	0.61

(a) ESN

	Classified as						
Real label	1	2	3	4	5	6	7
1	0.88	0.12	0.00	0.00	0.00	0.00	0.00
2	0.03	0.76	0.21	0.00	0.00	0.00	0.00
3	0.00	0.05	0.65	0.15	0.07	0.00	0.07
4	0.00	0.00	0.00	0.62	0.13	0.25	0.00
5	0.00	0.00	0.00	0.04	0.47	0.49	0.00
6	0.00	0.00	0.00	0.00	0.01	0.83	0.16
7	0.00	0.00	0.00	0.00	0.00	0.36	0.64

(b) ESN + PF

	Classified as						
Real label	1	2	3	4	5	6	7
1	0.78	0.22	0.00	0.00	0.00	0.00	0.00
2	0.03	0.76	0.21	0.00	0.00	0.00	0.00
3	0.00	0.07	0.66	0.12	0.09	0.00	0.06
4	0.00	0.00	0.00	0.53	0.18	0.29	0.00
5	0.00	0.00	0.00	0.04	0.47	0.49	0.00
6	0.00	0.00	0.00	0.00	0.20	0.68	0.12
7	0.00	0.00	0.00	0.00	0.00	0.36	0.64

(c) HMM+HMM

Fig. 3. The TUM kitchen dataset: Comparison of confusion matrices

The output of the simple ESN was used for the ESN+PF, so the comparison of the two methods is fair.

To show the effect of the invalidity of the Markovian assumption we also used a method based on the HMM. Firstly we *segmented* the sequences into tasks and then we *classified* each segment. For the *segmentation* task we trained an HMM to detect the task transitions, within a time widnow similarly to [21].

As for the classification task, we trained one HMM per task (L task-HMMs). Then we defined one HMM for the tested workflows (1 workflow-HMM) in a hierarchical fashion. The states of the workflow-HMM were defined by the detected segments, the emitted observations were the probabilities for each task as provided by the task-HMMs, the transition matrix was given by the transition probabilities between the tasks, and the prior was given by the tasks' priors (same as the ones used for the ESN+PF). Then we run the Viterbi algorithm to find the sequence of tasks for the workflow-HMM. This scheme will be hereafter referred to as HMM+HMM. The performance seems to be inferior to the ESN+PF and the errors stem from the Markovian assumption, the errors in the segmentation (the segmenation accuracy was measured as 14.1 ± 10.3 frames $mean \pm std.$) and the inaccuracies of the observation model.

4.2 The TUM kitchen dataset case

Another workflow application is presented by the TUM kitchen dataset [24] (also publicly available). It contains several instances of a table-setting workflow performed by different subjects, involving the manipulation of objects and the environment. For our purposes we have used the views camera 0 to recognize the following sequential tasks (where different permutations are allowed): (1) Taking a tray and putting it on the table. (2) Taking a napkin and putting it on the table (3) Opening a drawer, taking a fork and putting it on the table (4) Opening a drawer (the same as in 3), taking a knife and putting it on the table (5) Opening a drawer (the same as in 3, 4), taking a spoon and putting it on the table. (6) Opening a shelf, taking out a plate and putting it on the table (7) Opening a shelf, taking out a cup and putting it on the table. The most common task sequences were 1-2-3-4-5-6-7, 1-2-4-5-6-3-7 and 1-2-3-7-4-5-6.

We have used workflows/episodes with IDs: 0_0, 0_1, 0_3, 0_4, 0_6, 0_7, 0_8, 0_9, 0_10, 0_11, 1_0, 1_1, 1_2, 1_3, 1_4, 1_6, 1_7 in a 4-fold cross validation scheme. The ground truth was based on the annotation provided in the dataset. As soon an object was arranged on the table and the subject started heading away from it, we marked that point as the end of the current segment and the beginning of a new one. We also extracted similar features as in the previous application.

The results obtained for the ESN, ESN+PF and HMM+HMM methods (Fig. 3), were similar to the previous application. We note that tasks 4, 5 and 6 bear a great resemblance since they consist in opening the same drawer, picking a similar object (fork, knife, or spoon) and placing it on the table, thus it is quite difficult for a classifier to differentiate among them; that is confirmed by the misclassification rates among tasks 4, 5, 6.

5 Conclusions

We proposed an online framework for behavior recognition in workflows in real-time. We verified the effectively Markovian behavior of the ESN and we showed how to mitigate it by using a priori information, which can be embedded in a set of hypotheses (particles). The a priori information about the tasks sequence gave better results than the conventional ESN. It also gave better results compared to making an explicit Markovian assumption like by using a hierarchy of HMMs.

References

1. Kosmopoulos, D., Chatzis, S.: Robust visual behavior recognition. Signal Processing Magazine, IEEE **27** (2010) 34 –45
2. Veres, G.V., Grabner, H., Middleton, L., Gool, L.J.V.: Automatic workflow monitoring in industrial environments. In: Computer Vision - ACCV 2010, Queenstown, New Zealand, November 8-12, 2010, Part I. (2010) 200–213
3. Padoy, N., Mateus, D., Weinland, D., Berger, M.O., Navab, N.: Workflow Monitoring based on 3D Motion Features. In: Workshop on Video-Oriented Object and Event Classification in Conj. with ICCV 2009, Kyoto, IEEE (2009) 585–592
4. Behera, A., Cohn, A.G., Hogg, D.: Workflow activity monitoring using dynamics of pair-wise qualitative spatial relations. Volume 7131 of LNCS., Springer (2012)
5. Lalos, C., Voulodimos, A., Doulamis, A., Varvarigou, T.: Efficient tracking using a robust motion estimation technique. Mult Tools and Applications (2012) 1–16

6. Jaeger, H.: The echo state approach to analysing and training recurrent neural networks. Technical Report GMD 148, German National Research Center for Information Technology (2001)
7. Gallicchio, C., Micheli, A.: Architectural and markovian factors of echo state networks. Neural Networks **24** (2011) 440 – 456
8. Rabiner, L.R.: A tutorial on hidden Markov models and selected applications in speech recognition. Proceedings of the IEEE **77** (1989) 257–286
9. Eickeler, S., Kosmala, A., Rigoll, G.: Hidden markov model based continuous online gesture recognition. In: ICPR. (1998) 1206–1208
10. Lv, F., Nevatia, R.: Recognition and segmentation of 3-d human action using hmm and multi-class adaboost. In: ECCV06. (2006) IV: 359–372
11. Hoai, M., Lan, Z.Z., De la Torre, F.: Joint segmentation and classification of human actions in video. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). (2011)
12. Fine, S., Singer, Y., Tishby, N.: The hierarchical hidden markov model: Analysis and applications. Machine Learning **32** (1998) 41–62
13. Oliver, N., Garg, A., Horvitz, E.: Layered representations for learning and inferring office activity from multiple sensory channels. Comput. Vis. Image Underst. **96** (2004) 163–180
14. Xiaoling, X., Layuan, L.: Real time analysis of situation events for intelligent surveillance. In: Computational Intelligence and Design, 2008. ISCID '08. International Symposium on. Volume 2. (2008) 122 –125
15. Kosmopoulos, D.I., Doulamis, N.D., Voulodimos, A.S.: Bayesian filter based behavior recognition in workflows allowing for user feedback. Computer Vision and Image Understanding **3** (2012) 422–434
16. Zhang, D., Ning, X., Liu, X.: Smc method for online prediction in hidden markov models. Kybernetes **38** (2009) 1819–1827
17. Fei, H.: A hybrid hmm/particle filter framework for non-rigid hand motion recognition. In: ICASSP. Volume 5. (2004) V – 889–92 vol.5
18. Jaeger, H., Maass, W., Principe, J.: Special issue on echo state networks and liquid state machines. Neural Networks **20** (2007) 287 – 289
19. Skowronski, M.D., Harris, J.G.: Automatic speech recognition using a predictive echo state network classifier. Neural Networks **20** (2007) 414–423
20. K., G., Venayagamoorthy: Online design of an echo state network based wide area monitor for a multimachine power system. Neural Networks **20** (2007) 404 – 413 Echo State Networks and Liquid State Machines.
21. Voulodimos, A., Kosmopoulos, D., Veres, G., Grabner, H., Van Gool, L., Varvarigou, T.: 2011 special issue: Online classification of visual tasks for industrial workflow monitoring. Neural Netw. **24** (2011) 852–860
22. Arnaud, D., Simon, G., Christophe, A.: On sequential monte carlo sampling methods for bayesian filtering. Statistics and Computing **10** (2000) 197–208
23. Voulodimos, A., et al: A dataset for workflow recognition in industrial scenes. In: IEEE Int. Conference on Image Processing. (2011) 3310–3313
24. Tenorth, M., Bandouch, J., Beetz, M.: The TUM Kitchen Data Set of Everyday Manipulation Activities for Motion Tracking and Action Recognition. In: THEMIS Workshop, In conj. with ICCV. (2009)