2018 IEEE International Conference on Robotics and Automation (ICRA)
May 21-25, 2018, Brisbane, Australia

A robust method to predict temporal aspects of actions by observation

Emmanouil Hourdakis and Panos Trahanias

Abstract— **The ability to predict the duration of an activity can enable a robot to plan its behaviors ahead, interact seamlessly with other humans, by coordinating its actions, and allocate effort and resources to tasks that are time-constrained or critical. Despite its usefulness, models that examine the temporal properties of an activity remain relatively unexplored.**

In the current paper we present, to the best of our knowledge, the first method that can estimate temporal properties of an activity by observation. We evaluate it on three use-cases *(i)* wiping a table, *(ii)* chopping vegetables and *(iii)* cleaning the floor, using ground truth data from real demonstrations, and show that it can make predictions with high accuracy and little training. In addition, we investigate different methods to approximate the progress of each task, and demonstrate how a model can generalize, by reusing part of it in different activities.

I. INTRODUCTION

The timing of actions plays an important role when planning complex task pipelines for robots. For example, utilizing information regarding the duration of its actions, a robot can plan ahead [2], allocate resources to tasks that require more effort to be completed [17], or maximize its availability, by coordinating with a human [3] or other robots [4]. Despite the broad applicability of the subject, there haven't appeared in the literature models that can make temporal predictions about an activity only by observation.

Humans rely heavily on temporal predictions when planning their activities [14], even though their ability to make time-estimates is error-prone and biased by their sensorimotor system [1]. Similarly, in robotics, models that can estimate the duration, and other temporal properties, of an activity can prove useful. To handle complex tasks assignments, however, the robot inferences must go beyond single actions (e.g. reach-to-grasp), to the level of coherent activities (e.g. cleaning a table). In this context, in addition to duration, there are several temporal properties that could be useful to a robot, such as:

- How long will it take to finish this activity?
- Which behavior configuration will finish this task as fast as possible?
- I want to clean the table during the 5 minutes of free time that I have. How clean will the table be after that?
- Am I performing the task in an efficient manner, given the spare time that I have?
- I have 3 minutes of spare time. What tasks can I perform during this period?

In the current paper we introduce Generative Time Models (GTMs), a principled formulation to derive models that can make predictions about the duration by observing an activity. GTMs can answer the aforementioned questions with minimal prior experience and training. We apply the concept on three example use-cases, involving tasks with repetitive actions and different complexity: (i) cleaning a table, (ii) chopping vegetables and, (iii) wiping the floor. Using ground truth data from real demonstrations, we show that the method can predict the duration, and other temporal properties of the aforementioned activities with high accuracy. In what follows we present a brief literature review, whereas the methodology is rigorously formulated in section III. Section IV presents experimental evaluation results and section V outlines a detailed discussion. Finally, we present our conclusions and suggestions for future work in section VI.

II. LITERATURE REVIEW

In robotics, time has been well acknowledged as a glue factor for several cognitive skills [5]. Consequently, the timing of robotic actions has been studied in various contexts, including robot-navigation [20], human-robot interaction [3], robot-robot [4] and human-robot teams [4], to name a few. Time is also useful in the context of human robot coordination where studies use the execution time of actions [15], [16] to synchronize two or more agents. Temporal properties in these models are obtained by timing the actions as they are executed by the robot. More recently, researchers have suggested models that can anticipate human actions, and used them for the design of autonomous driving systems [26].

In computer vision, the timing of actions is used for action segmentation [6], since temporal information is a strong discriminative factor for action classification. For example, in [7], [8], the authors use local spatio-temporal features to recognize different actions. In [9], the authors discriminate between discrete and oscillatory motions, by embedding temporal information in their action semantics. In [10], activity recognition is performed by modeling the temporal structure of a task using graphical models, while in [11] using semi-latent topic models.

The works cited above focus on the temporal aspects of a single action. The work presented here is, to the best of our knowledge, the first to predict the duration and other temporal properties of a coherent activity, only by observation. The main contributions of the current paper can be summarized as:

[1]Emmanouil Hourdakis is with the Computational Vision and Robotics laboratory of the Foundation for Research and Technology - Hellas ehourdak@ics.forth.gr

[2]Panos Trahanias is a Professor of Computer Science in the University of Crete and head of the Computational Vision and Robotics laboratory of the Foundation for Research and Technology - Hellas trahania@ics.forth.gr

1) A method for predicting the duration, and other temporal properties, of an activity.
2) Implementation and evaluation of three use-case scenarios in kitchen-related activities.

III. Generative Time Models

Generative Time Models (GTMs) represent a method to derive observation models for real-time estimation of the duration, and other temporal properties, of an unfolding activity. They consist of two observation models: *(1) task progress* and *(2) control*. The first one estimates the progress of the task, i.e. how much of the activity has been completed. The second identifies and records time and task information about the primitive motions that appear during the activity. This information is subsequently used to predict the overall duration of the activity. In the following section we describe the mathematical formulation of GTMs using the 'wipe the table' activity as a running example. In addition, the concept of this work is depicted in the video at the following link: `http://hourdakisemmanouil.com/GTMconcept.mp4`, where the model is used to yield predictions every 100 simulation iterations.

To define a GTM, we assume an activity, whose progress can be observed by a function $O : \mathbf{t} \mapsto \mathbb{R}$, with $O = \{o_1, o_2, ..., o_N\}$ a univariate time-series, uniformly sampled at $T = \{t_1, t_2, .., t_N\}$. For the 'wipe the table' activity, the function O represents the progress of the wiping activity, the percent of the table that has been wiped, i.e. $o_N = 30$ means that the table has been wiped by 30% at a time t_N. In addition, we assume a control process $C : \mathbf{t} \mapsto \mathbb{R}^P$ with $C = \{c_1^p, c_2^p, .., c_N^p\}$, the set of p dimensional vectors corresponding to the state variables that describe the output of the control process, at times T. For the example of wiping the table, the control process C would contain the position of the sponge, with $c_N^2 = \{x, y\}$ the x, y coordinates of the sponge's position. The goal of a GTM, is to approximate a function $f : \mathbf{C}, \mathbf{t} \mapsto \mathbb{R}$, that describes how the control system affects the progress of the task, i.e. $O(t) = f(C, t)$. To accomplish this, a GTM looks for standard motion patterns in C, i.e. primitives, and calculates how each primitive affects the task progress. For the example of wiping the table, f describes how much each primitive wipes the table. Once obtained, f can be used to calculate the duration of the activity, by predicting how the task progress will change in respect to the primitives being used.

In the following section we outline the mathematical underpinnings for the GTM model, including how primitives are extracted by observation from the control process, the information obtained for each primitive, and how the temporal properties of an activity are predicted. The methodology presented pertains to periodic primitive motions, i.e. repetitive behaviors that are densely observed within an activity. As the density of the observed primitives gets lower, additional metrics are required to facilitate robust temporal predictions; the latter is a topic of future work as presented in the Conclusions section.

A. Primitive segmentation

To approximate f we assume that at any point in time C takes one, out of k, standard primitive motions. For the example of wiping the table, where we have repeating oscillatory motions, the primitives are described by their amplitude and period. To obtain the primitives, a GTM segments the signal in C, by looking for local extrema at small Δt intervals and stores their starting t_s and ending t_e times. To evaluate the local extrema it employs the prominence algorithm [18], which makes use of the fact that the derivative of a signal, at a point of a peak has a zero-crossing at the peak maximum.

To identify peak signal positions, the algorithm smooths the signal's first derivative, by convolving it with a Gaussian kernel, and then stores the indices of the zero-crossings on the smoothed derivative. For each index, a prominence value is calculated, which indicates whether there has been a significant change in the motion direction vector. The algorithm returns the n largest peaks whose prominence exceeds a certain threshold value. The output of the segmented signal, for the 'wipe the table' example is shown in Fig. 1:

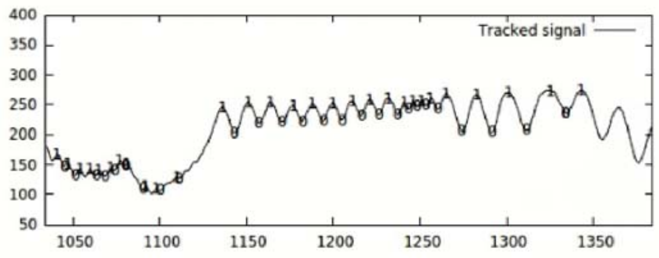

Fig. 1: Segmented signal (distance from reference point) of the sponge's position, obtained by tracking the sponge whilst the human is cleaning the table.

The segmentation process yields a vector of n non-regular intervals $I = \{\{t_{s_1}, t_{e_1}\}, \{t_{s_2}, t_{e_2}\}, .., \{t_{s_n}, t_{e_n}\}\}, I \subseteq T$, which correspond to the times that the segmented motions $B = \{M_{(o1)}, M_{(o2)}, .., M_{(on)}\}$ are observed in $C(t)$. To describe a motion in each interval, the GTM uses a feature vector $M_{(oi)} = \{a_{i1}, a_{i2}, .., a_{iv-1}, d_i\}$, which consists of (i) $v - 1$ global spatio-temporal features, recorded during the time-window $t_i \epsilon \{t_{si}, t_{ei}\}$ of the behavior and, (ii) the behavior's duration $d_i = t_{ei} - t_{si}$. For the example of wiping the table, where we want to discriminate primitives on the basis of their amplitude and period, the features in $M_{(oi)}$ contain each segmented motion's amplitude and period. Using the feature vector $M_{(oi)}$, behaviors in B are classified into K classes by solving the following optimization problem:

$$K = arg \max_{K \subseteq B} \Big(\sum_{i=1}^{k} \sum_{j=1}^{n} \|\mathbf{M_{(i)}} - \mathbf{M_{(j)}}\| \Big) \tag{1}$$

From eq. 1 we obtain the k different cluster centers $K = \{M_{(1)}, M_{(2)}, .., M_{(k)}\}$ that are used as labels for the different primitive actions appearing in the activity. To obtain K we cluster the data in B using k-means. The result is a signal segmentation and labeling module that runs online, whilst tracking the control process. Figure 2 illustrates the output of the labeled primitives, after the end of the training phase. Each primitive is marked with a different color and index.

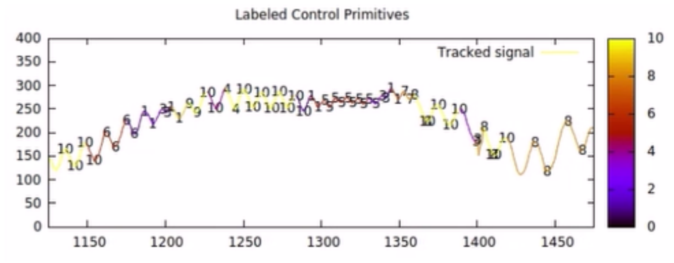

Fig. 2: A tracked signal, segmented and labeled, by the components of the GTM. Different labels correspond to different primitives in K, and are assigned different colors as indicated by the palette on the right.

B. Primitive modeling

After having segmented and labeled the primitives that exist in C, the algorithm calculates three quantities for each primitive: *(1)* the effect that each primitive has on the task progress, *(2)*, its frequency of occurrence within C and (3), its duration.

1) Effect of each primitive: To describe how each primitive affects the activity progress, a GTM assigns a function $f_{M_{(i)}}(t)$ to each $M_{(i)}$, for which it holds:

$$O(t_{pe}) = O(t_{ps}) + \int_{t_{ps}}^{t_{pe}} f_{M_{(i)}}(t)dt, \quad \forall \{t_{ps}, t_{pe}\} \epsilon I^{M_{(i)}} \tag{2}$$

i.e. the change in task progress during the interval $\{t_{ps}, t_{pe}\}$, where primitive $M_{(i)}$ is observed, can be determined by the integral of the function $f_{M_{(i)}}$ for that time period. A GTM approximates the function $f_{M_{(i)}}$ for each primitive using the output of the function $O(t)$, during the intervals $I^{M_{(i)}}$ that the primitive is observed. After training, we obtain the set of $\{M_{(1)}, M_{(2)}, .., M_{(k)}\} \rightarrow \{f_{M_{(1)}}, f_{M_{(2)}}, .., f_{M_{(k)}}\}$, that describe how each primitive changes the task progress.

2) Frequency of occurrence: To estimate the frequency of occurrence,the probability density function p of the primitives in K is obtained using Kernel Density Estimation (KDE). Given a sample set of n values from an identically distributed variable l, the Density estimator p around a point l_0 is:

$$p(l_0) = \frac{1}{n}\sum_{n} G_h\Big(\frac{l - l_0}{dl}\Big) = \frac{1}{nh}\sum_{n} G\Big(\frac{l - l_0}{hdl}\Big) \tag{3}$$

where $G(z, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{z^2}{2\sigma^2}}$ is the Gaussian kernel, and h is the Kernel bandwidth, which acts as a smoothing parameter [5]. Larger h values will result in smooth (and thus high-biased) density distribution.

C. Estimating the temporal properties

Having segmented and described each primitive that is observed, a GTM approximates the activity progress O in future time-steps, using a finite mixture model:

$$O(t) = f(C, t) = \sum_{i=1}^{k} p(i) * \phi_i \tag{4}$$

where $p(i)$ are the mixing components, that satisfy $p(i) \geqslant 0 \forall i \epsilon k$ and $\sum_{i=1}^{k} p(i) = 1$, while $\phi_i = \int_t^{t+t_i} f_{M_{(i)}}(t)dt$ are the local basis functions. A GTM uses eq. 4 to predict future states for the activity progress, i.e. the expected change for the task progress is calculated using the probability of observing the primitive, and the amount that each primitive completes the task. Using equations 3, 4 on can derive useful information about the activity:

1) Task progress caused by a single primitive: Given $p(i)$, the probability density function of K, the expected duration for primitive $M_{(i)}$ is $t_i = p(i) * d_i$, while the expected task change, due to that primitive is:

$$O_{M_i}(t + t_i) = p(i) * \int_t^{t+t_i} f_{M_{(i)}}(t)dt \tag{5}$$

2) Predicting the task progress: Given the probability $p(i) \forall i \epsilon k$ for all primitives one can estimate, using eq. 4, how the task progress will change from t to $t_k = \sum_{i=1}^{k} (p(i) * d_i)$:

$$O(t + t_k) = O(t) + \sum_{i=1}^{k} \left(p(i) * \int_t^{t+d_i} f_{M_{(i)}}(t)dt \right) \tag{6}$$

Eq. 6 provides an estimate of the activity progress forward in time, using the $f_{M_{(i)}}$ as basis functions. Due to eq. 1, these primitives have well defined duration, and therefore are also useful basis functions for making duration predictions.

3) Remaining duration of the activity: Using the pdf $p(i)$, we can estimate how long will require for the activity at time t to finish (i.e. $O(t) \rightarrow 0$):

$$T_{rem} = \frac{O(t)}{\sum_{i=1}^{k} \left(p(i) * \int_t^{t+d_i} f_{M_{(i)}}(t)dt \right)} * \sum_{i=1}^{k} (p(i) * d_i) \tag{7}$$

or working inversely, if a task starts at time $t = 0$, and is performed for a D duration, its state progress will become:

$$O(D) = \frac{D}{\sum_{i=1}^{k} (p(i) * d_i)} * \sum_{i=1}^{k} \left(p(i) * \int_t^{t+d_i} f_{M_{(i)}}(t)dt \right) \tag{8}$$

4) Primitives and task completion: To find the primitive model that will perform the task in the fastest manner:

$$M_{(i)} = arg \max_{i \subseteq K} \frac{\int_t^{t+d_i} f_{M_{(i)}}(t)dt}{d_i} \tag{9}$$

The time required to finish the activity, given a certain primitive model $M_{(i)}$ (i.e. $p(i) = 1$) is:

$$T_i = \frac{O(t)}{\int_t^{t+d_i} f_{M_{(i)}}(t)dt} * d_i \tag{10}$$

The quality of the predictions above depends on how well the set of basis functions, can describe the progress of the activity. There are various information theoretic measures to quantify this, e.g. the squared error loss function $\mathcal{L}$ for eq. 2. In general, if the density of the observed behaviors B on the signal $C(t)$ is high, then one can obtain a good approximation.

IV. Use-case evaluation

In the current section we apply the method to three different household activities for assessment purposes. In all examples, the experimental setup consists of a room, containing a table (Fig. 4) and an AsusXtion RGBD camera mounted at approximately 3m above the floor. A human is asked to perform the activity while the GTM uses the camera input to analyze it. To evaluate the method, we use duration measurements from real demonstrations.

A. *Cleaning a table*

1) Activity Observation: The first use-case involves a human wiping the surface of a table. The observation model, using depth information from the camera, detects the boundaries of the surface and, given the sponge position from the control model, estimates the percent of it that has been wiped.

To detect the table surface, we use the Hessian Normal form $\hat{m}\cdot x = -p$. of the general plane equation $ax+by+cz+d=0$ [21], where $\hat{m} = (\frac{a}{\sqrt{a^2+b^2+c^2}}, \frac{b}{\sqrt{a^2+b^2+c^2}}, \frac{c}{\sqrt{a^2+b^2+c^2}})$ is the unit normal vector. The value $p = \frac{d}{\sqrt{a^2+b^2+c^2}}$ determines the distance of the plane from the origin. 3D points with $p > 0$ lie the half-space determined by the direction of $\hat{m}$. We obtain the 3D point cloud of the points that satisfy the plane equation using Random Sample Consensus (RANSAC) and estimate the boundaries of the point cloud. Since the control model tracks the sponge in 2D, we back project the surface boundaries, using the camera's intrinsic and extrinsic parameters, to estimate the area to be wiped.

Having obtained the task progress, the Activity Observation component uses the output from Control Observation, in order to find the task progress intervals that correspond to the different primitives. For each of the intervals $I^{M_{(i)}} = \{\{t_{s_1}^{M_{(i)}}, t_{e_1}^{M_{(i)}}\}, \{t_{s_2}^{M_{(i)}}, t_{e_2}^{M_{(i)}}\}, .., \{t_{s_n}^{M_{(i)}}, t_{e_n}^{M_{(i)}}\}\}$, that a primitive $M_{(i)}$ is observed in $C(t)$, we sample from the $O(t)$ function in order to approximate $f_{M(i)}$ (Fig. 3). For the current example, we assume that $f_{M_{(i)}}$ is constant. To estimate it, we average the change in task progress over all segments $I^{M_{(i)}}$ in $O(t)$, where behavior $M_{(i)}$ is observed:

$$f_{M_{(i)}}(t) = \frac{\sum_{i=1}^{L}(O(t_e) - O(t_s))}{L} = \mathbb{C}\forall\{t_s, t_e\}\epsilon I^{M_{(i)}} \tag{11}$$

where $I^{M_{(i)}}$ is the time vector containing the starting and ending times in $C(t)$ that behavior $M_{(i)}$ is observed.

Given a task state $O(t)$, the remaining time required for the activity to complete can be approximated using eq. 7. To find the primitive that will wipe the table as fast as possible, we use eq. 9, while the overall time of the task can be found using eq. 7, which for the current example was 0.56min or 33seconds.

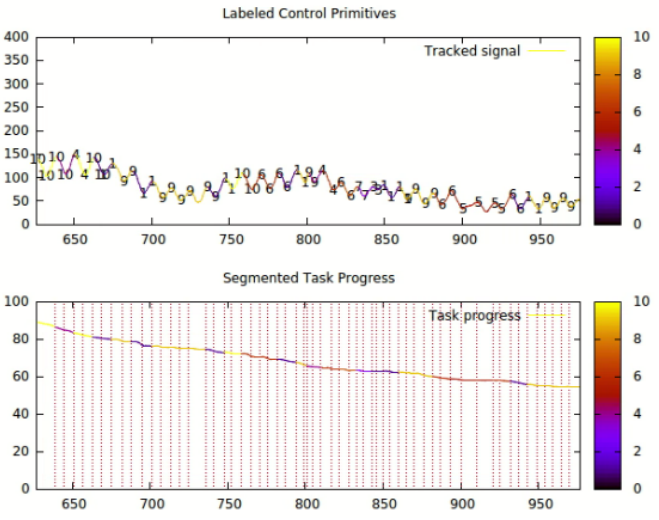

Fig. 3: Segmenting the output of the Activity Observation component (bottom) using the segmentation intervals of the Control Observation component (top). The segmented observation signal in the bottom graph, along with the corresponding labels of the control signal in the top graph, can be used to estimate the function $f_{M_{(i)}}$ of a primitive $M_{(i)}$.

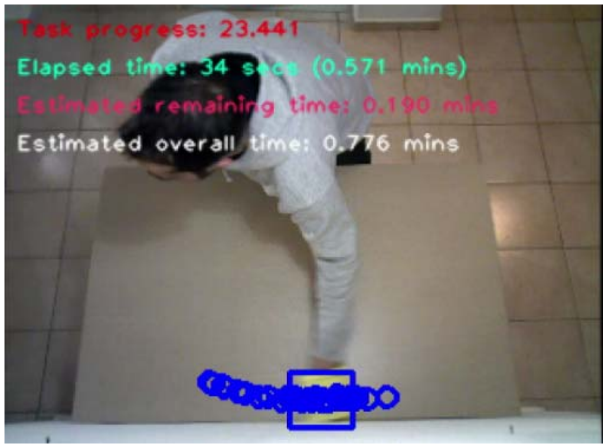
Task progress: 23.441
Elapsed time: 34 secs (0.571 mins)
Estimated remaining time: 0.190 mins
Estimated overall time: 0.776 mins

Fig. 4: Predicting accurately the overall and remaining time for the 'wipe the table' task.

To evaluate the task, we run 100 different trials of the experiment, performed by 3 proficient users. The timing of the task begun as soon as a user picked the sponge, and ended when the table surface was wiped above 95%. This implementation choice was decided during the experimental evaluation of the model, where we observed that setting the progress goal to 100% would introduce prolonged activity times, that do not reflect the real duration of the task. We observed in all cases that, when asked to clean a surface by 100%, users would spent prolonged times trying to wipe/recover all those small regions that have been left

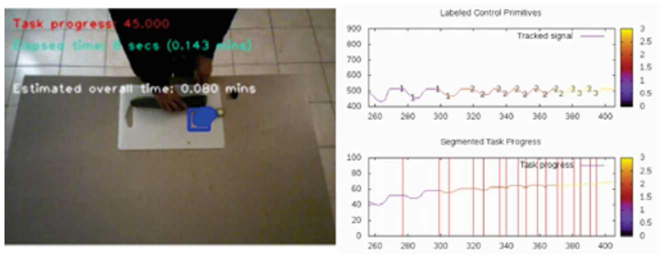

Fig. 5: (left) Snapshot of the 'chopping vegetables' activity. (right) Primitives identified and labeled during the activity.

unclean up to this point. Hence, this remaining 5% is performed with reduced efficiency, and does not reflect the dynamics of the task up to this point. In summary, as our results show (Table I), the GTM was able to predict with high accuracy the duration of the task.

We note that the results presented in this section are dependent on the proficiency of the user performing the task. More experienced users are more consistent, having a smaller repertoire of primitives with lower variance in their durations. This yields more robust predictors. One additional factor that affects performance is the previous knowledge acquired by the GTM, during the training sessions. The results presented here regard GTMs that were trained using data from three proficient users. For a more detailed discussion about the performance and implementation details of the model, please refer to the Discussion, section VA.

TABLE I: Prediction accuracy for the wipe table task, for 100 demonstrations performed by three users. Columns 2-4 show the average values (in seconds), across all trials of a user.

User	Av. Predicted time	Av. Gr. truth	Error(sec)
#1	39.124	33.213	5.911
#2	30.427	35.214	4.787
#3	39.224	44.231	5.007

B. *Chopping vegetables*

As an additional example we develop a GTM for the activity of chopping vegetables. To implement it we reuse the control observation component described in section III and eq. 11 for approximating function $f_{M_{(i)}}$. The goal of the GTM is to identify the different cutting primitives used, and determine which one is the fastest, and how long it will require for someone to chop one (or more) cucumber.

1) Activity observation: For this task, the observation model must track the length of the cucumber that is cut by each primitive. In this case, the Activity Observation component determines the displacement of the tip of the knife l from point v, $d = \|\mathbf{v} - \mathbf{l}\|$ coordinate of the knife's tip, i.e. the displacement of the knife, from the start of the cucumber.

Table II lists the estimated effect on task progress and duration of each primitive. Using eq. 9, we find that the fastest primitive is $M_{(1)}$, which cuts 3.58cm of the cucumber in 0.25 seconds.

The GTM used to estimate the duration of this example consists only of a new observation model. The control observation component was re-used from the previous use-case. This also demonstrates the ability of GTMs to generalize well, since new tasks can be implemented easily. The following table lists the association of each primitive with the task progress (i.e. the $f_{M_{(i)}}$ function).

TABLE II: Temporal information about the primitives identified during the task. Column 2 lists each primitive's duration, while Column 3 how each primitive's changes the task progress.

Primitive	Duration of primitive d_i	Association $f_{M_{(i)}}$
1	0.328911	2.126913
2	0.491832	1.284661
3	0.264343	2.020385

Using the information from Table II, we can now make temporal predictions about the duration of the task. If we have 20 cucumbers of average length 30cm, then using primitive 1, we would require 92,7 seconds to finish the task, or approximately 1.54 minutes.

TABLE III: Prediction accuracy for the cut cucumbers task, for 100 demonstrations performed by three users. Columns 2-4 show the average values (in seconds), across all trials of a user.

User	Av. Predicted time	Av. Gr. truth	Error(sec)
1	4.1983	3.1728	1,0255
2	5.9371	3.8291	2,1026
3	2.8263	3.9182	1,0919

As the results show, the GTM can yield an accurate prediction despite that the duration of the activity being very short (approximately 3 seconds). Again, the results presented are dependent on the profficiency, and other performance factors, of the human executing the activity. For the current task, all three users were considered experienced, since they had to perform the task numerous times during the experimental evaluation of the model. In the future we plan to investigate further how the proficiency of the user evolves with its experience, and how this is reflected in the primitive information we track for GTMs.

C. *Cleaning the floor*

In addition to the previous examples, we consider the use-case of cleaning the floor. For this experiment we use the Control observation component developed in section III, extend the activity observation component developed in section IV-A, and focus on a more elaborate implementation for the $f_{M_{(i)}}$ function. We employ the same setup as the previous experiments, however in this case, we track the tip of a mop, which cleans a stained floor surrounding the table. In addition to swiping the area, this task now involves cleaning the floor.

1) Activity observation: For this use-case we use the observation model developed in section IV-A, which tracks the area of the floor that has been wiped, and add an additional module, in order to detect and quantize the amount of dirt on the floor. To detect the stains, we apply a normalized box filter to smooth the image, and detect Agast features [25] on the output. The final observation model is given by the area that has been wiped, plus the number of Agast features multiplied by a factor.

2) Association function $f_{M_{(i)}}$: In contrast to the previous examples, the progress of the task in this case depends on additional contextual information, e.g. the concentration of the dirt. To approximate how the observation model changes in respect to such irregular quantities (i.e. find the function $f_{M_{(i)}}$), liquid state machines [24] are employed since they are good approximators for dynamics over short spatio-temporal intervals [19].

This ability stems from the fact that the signal is distributed upon the many computational units of the LSM, making it more robust to noise perturbations and oscillations. The leaky property of the membrane potential, along with the dynamic synapse model [22] allow LSMs to maintain the gradient of the error signals for prolonged activation periods over the liquid. More specifically, the signal is accumulated, over many intervals, in the liquid due to: (i) the delay implemented in the synapses and, (ii) the non-linear -leaky- differential equations of the neuron. These two properties make LSMs robust spatio-temporal predictors, as demonstrated by various works (for a comprehensive review, the reader is referred to [23]).

The liquid in the current example is modeled by leaky integrator neurons, which describe the evolution of the membrane potential of the neuron as the following differential equation:

$$\tau_m \frac{\delta V_m}{\delta t} = -(V_m - V_{rest}) + R_m * (I_{syn}(t) + I_{inj} + I_{noise}) \tag{12}$$

where Vm is the membrane voltage, $\tau_m = C_m R_m$ is the membrane time constant, R_m is the membrane resistance, C_m is the resistor capacitance, I_{inj} is a constant current injected to the neuron and I_{noise} a Gaussian random process, with zero mean and a small variance. After the emission of a spike, the membrane potential is reset to its resting value V_{rest}. $I_{syn}(t)$ is the incoming current from the presynaptic neurons, and is calculated according to the following equation: $I_{syn}(t) = \sum w_{ij} N_j(t)$. N_j is the output of the j^{th} presynaptic neuron, t is the current simulation time, while w_{ij} is the weight connecting the presynaptic neuron i and the postsynaptic neuron j.

We connect neurons within the liquid using the dynamic synapse model suggested in [22]. In this model, a synapse's n postsynaptic potential (EPSPn) changes according to $N_t = K * R_n * u_n$, with $u_{n+1} = u_n * exp(\frac{\Delta t}{\tau_f acil} + U(1 - u_n * exp(-\frac{\Delta t}{\tau_f acil})$ and $R_{n+1} = R_n(1 - u_{n+1}) * exp(-\frac{\Delta t}{\tau_r ec)} + 1 - exp(-\frac{\Delta t}{\tau_r ec})$.

To input the signal into the liquid, we transform it into a stochastic pattern of spike firings, using an in-homogeneous Poisson process, which introduces irregularity in the spiking patterns. The formula for the probability density of a spike in an interval Δt is given by $P(spikes|\Delta t) = e^{-r\Delta t * \frac{(r\Delta t)^n}{n!}}$.

To approximate function f_{M_i} we input to the network the output of the Activity Observation at $O(t-1)$ (the number of Agast features and the area that has been wiped) and the feature vector $M_{(i)}$, which is observed at each instant, and use as supervised data the output of the observation model at time $O(t+n)$, i.e. n steps forward. For the simulations, we have used networks with liquids of approximate 4000 neurons. To learn the spatio-temporal signal that is output from the liquid, we solve the non-linear mapping between the liquid states and the objective function using least squares with Cholesky factorization. In the following figure (Fig. 6) we demonstrate how the LSM was able to approximate the task progress changes that will be inflicted by a primitive.

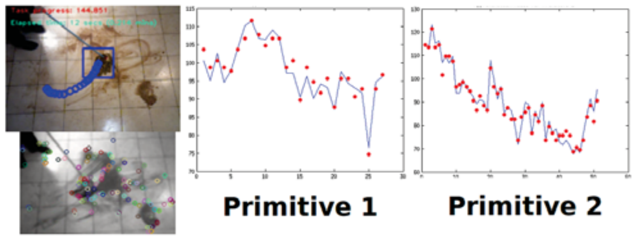

Fig. 6: Learning the approximation function for the cleaning the floor activity. (left) The mop as it is being tracked, and the features detected for the stain (middle) The approximation function learned for primitive 1 and (right) primitive 2.

The output of the network can now be used to yield predictions that take into account more explicit information on how the task progresses. For the current example, cleaning the floor, the LSM enabled us to estimate the effect of a primitive more accurately, even while wiping over the stains. Table IV evaluates the duration estimates of the method, after 30 demonstrations of the task. As the results show, the GTM was able to yield quite accurate predictions.

TABLE IV: Prediction accuracy for the clean the floor task, for 30 demonstrations performed by three users. Columns 2-4 show the average values (in seconds), across all trials of a user.

User	Av. Predicted time	Av. Gr. truth	Error (sec)
1	69.8172	76.2818	6.464
2	62.9177	71.9281	9.010
3	59.2469	68.2934	9,046

V. Discussion

A. Implementation Details

The experimental evaluation showed that the model can yield predictions that have a small degree of error. The quality of the predictions depends on the amount of processing power and data that has been made available to the model. What we have found, is that the GTM predictions are user-dependent, and can improve with more training.

From an implementation point of view, there are some metrics that affect this performance: (i) the number of primitives clustered in each prediction step, (ii) the discretization/sampling rates that will be used when processing the observation signal, and (iii) the user being consistent in his/her performance across different task executions. The first two are task dependent, but can be easily adjusted, according to the complexity of the activity that is being observed. Consequently, we suggest one general guideline: more complex activities require an increased number of clusters and data samples to be processed.

Regarding the third factor that affects performance, *consistency of the user*, it pertains to whether a user is being consistent across trials in its selection of primitives. In essence it is a metric of the user's ability to perform the task efficiently, and evolves with user experience. Inefficient users perform the task arbitrarily, executing behaviors from a broader repertoire, while efficient users are more consistent in the selection of movements. We plan to include relevant information in the model's predictions in future versions of the theoretical framework.

GTMs make their temporal predictions, using the observed primitives as a basis function. Amongst others, each primitive is assigned (i) an expected duration and (ii) an expected effect on the task progress. With this information, GTMs look for robust predictors, over small segments of the activity, i.e. look for behaviors that appear with somewhat consistent duration and expected effect on the task progress. Behaviors that have not been observed yet, will be assigned to the class whose feature vector is closest to the feature vector of the primitive. An approximate number of 20 classes was found sufficient for the complexity of the tasks described in the manuscript. In addition, we note that we re-run the clustering loop in small intervals (every 100 iterations in our implementation). As a result, if there is a high intra-class variability this module will assign new class labels as required.

Finally, we note that the current implementation uses mean-shift tracking in order to observe a scene, and therefore is sensitive to relevant issues such as poor lighting conditions or intense user movements. During runtime, the method can compensate for any failures in the tracking process, however behavior trajectories must be observed regularly.

To test the model's processing capacity against the aforementioned metrics, we integrated the current implementation on a Dell Inspiron M1000, equipped with an NVIDIA GM107GLM [Quadro M1000M] GPU card. The model was run on the CPU's processor, an Intel(R) Core(TM) i5-6300HQ CPU @ 2.30GHz, while the visual tracker, kernel density estimation and Liquid State Machine modules on the GPU. To stress the model, we used 30 labels for clustering and a 1KHz for sampling the activity observation model, a configuration that would facilitate rather complex activities. Results show that the model was able to yield predictions every 10 simulation cycles, which is extremely appropriate for the considered actions.

B. Methods for deriving relevant models

The methodology presented is generic and can be applied to any repetitive task, given that the appropriate activity and control observation components have been implemented. Consequently, the equations presented in section III can be satisfied by a family of methods. The computational tools that will be used must be appropriate for the complexity of the task under investigation. For example, when facing a more elaborate task, one might find out that the segmentation process is not robust to compensate for the motion changes. In these cases, a different control observation model, with a more elaborate segmentation algorithm, should be chosen.

1) Strategies for deriving the activity observation component: We suggest that this component should be designed so that it can deliver accurate task progress estimates in regular intervals. For periodic tasks, the resolution for the activity observation model should be set to a value that matches or is smaller than the periodicity of the observed movements. For the tasks presented, we have used a sampling rate frequency of 1KHz.

2) Strategies for deriving the control observation component: The control observation component should be able to produce dense segmentation intervals of the activity. The density of those segmentation intervals, should be high in order to obtain a large amount of data for the primitive models. In the current implementation, where we focus on oscillatory movements, the density of the motions observed is inherently high, so this problem is confronted by selecting the segmentation intervals based on the motion shifts in the direction vector. In future versions we plan to release metrics of the ability of the two models to describe an activity. This would also help faciliate involving discrete movements.

C. Notes on the methodology

The proposed methodology was formulated in order to allow model reuse. Accordingly, the equations that describe the activity and observation model are kept completely segregated. Hence, those two facets of the activity are treated separately. The first, control observation, observes how the user acts on the activity. The second, activity observation, observes how the activity progresses based on that. What we have found, and demonstrated with the use-cases selected, is that a lot of activities share one or both facets. For example, for the three activities investigated we use the same control observation model, since they all involve repetitive motions. In addition, some share common properties in the way they progress. For example, for the moping activity we only had to introduce, on top of the wipe observation model, a single module to detect 'dirt' features. In the near future, we plan to take advantage of this architecture, in order to introduce additional activity and control observation models.

Inherently, the methodology is modular and employs various learning processes, including an unsupervised method for clustering the primitives, or a supervised method for estimating the activity progress. However, as it is implemented, the model, does not require any data prior to execution, apart from the number of class labels that will be used for the primitives and the sampling interval of the observation model. Supervised data needed are automatically collected and labeled for a primitive, during the learning process.

VI. CONCLUSIONS

The notion of predicting the temporal properties of an activity is novel to robotics, but can prove invaluable. In the current paper we have formulated the concept of Generative Time Models (GTM), i.e. models that can answer questions regarding the temporal properties of a task.

In addition to duration, the methodology computes complementary temporal information for each primitive model. These include the duration of the primitive, its frequency of occurrence and the effect it has on the task progress. These can be used as performance metrics for the user. In the future, we plan to elaborate further on those metrics, investigating the intra- and inter- class variability of the primitive models, in order to provide more elaborate user profiles.

Following the broad categorization of primitives to discrete and periodic, the model pertains to the latter category of motion primitives. It remains also to be applied to the first class of discrete motion primitives. In the near future we plan to publish an extension of the methodology, that will facilitate both types of behaviors, i.e. discrete and periodic, as well as combinations of these in a unified framework. Our aim, is to make the same method and principles apply also to this second class of behaviors, in order to suggest a complete framework under which temporal predictions can be made.

Regarding the robustness of the methodology, in the near future we plan to extend it to include information such as the user's consistency in the model's predictions. In addition, we plan to investigate tasks with irregular dynamics, such as the ones that appear in the end of the wipe the table activity (cf discussion in Section IV), in order to compensate for these durations.

GTMs can make accurate predictions with few training iterations, providing information that is useful to various applications, including Human-Robot interaction, scene perception, robot planning and process modeling, to name a few. The formulation proposed in this paper, with segregated Control and Activity observation components, allows the concept of GTMs to generalize, by reusing Control and Activity observation models across tasks. In the near future we plan to elaborate the concept even further, creating an array of observation and control abstraction models, in order to capture the whole spectrum of common household activities.

ACKNOWLEDGMENT

This work has been partially supported by the FETPROACT-2-2014, EU Horizon 2020 project (grant no. 641100) “Mind and Time - Investigation of the temporal attributes of human-machine synergetic interaction” project (TIMESTORM; www.timestorm.eu), and the FETOPEN-4-2016-2017, EU Horizon 2020 project (grant no. 754423) “Industrial Exploitation and Market Uptake of a Temporal Cognition Toolbox for Commercial Robots” (ENTIMENT; http://www.entiment.eu)

REFERENCES

[1] Zakay, D., and Richard A. B. ”The role of attention in time estimation processes”, Adv. in psychology, v.115, pp. 143-164, 1996.

[2] Holme, Petter, and Jari S., (eds), Temporal networks. Springer, 2013.

[3] Wilcox, Ronald, Stefanos Nikolaidis, and Julie Shah. Optimization of temporal dynamics for adaptive human-robot interaction in assembly manufacturing, Robotics Science and Systems VIII, pp. 441-448, 2012.

[4] Levine, Steven James, and Brian Charles Williams. Concurrent Plan Recognition and Execution for Human-Robot Teams, ICAPS, 2014.

[5] M. Maniadakis, P. Trahanias, Temporal cognition: a key ingredient of intelligent systems, Frontiers in Neurorobotics, v. 5, 2011.

[6] Wang, Heng, et al. Evaluation of local spatio-temporal features for action recognition.” BMVC-British Machine Vision Conference, 2009.

[7] Laptev, Ivan. On space-time interest points, International journal of computer vision, v64:2-3, pp. 107-123, 2005.

[8] Bregonzio, Matteo, Shaogang Gong, and Tao Xiang. ”Recognising action as clouds of space-time interest points.” Computer Vision and Pattern Recognition, CVPR IEEE, 2009.

[9] Eren Erdal, et al. Enriched manipulation action semantics for robot execution of time constrained tasks, Humanoid Robots (Humanoids), IEEE-RAS 16th International Conference, 2016.

[10] Lin, Zhe, Zhuolin Jiang, and Larry S. Davis. Recognizing actions by shape-motion prototype trees, Computer Vision, IEEE, 2009.

[11] Wang, Yang, and Greg Mori. Human action recognition by semilatent topic models, IEEE transactions on pattern analysis and machine intelligence v31:10, pp. 1762-1774, 2009.

[12] Cakmak, Maya, et al. Using spatial and temporal contrast for fluent robot-human hand-overs, Proc. of the 6th international conference on Human-robot interaction. ACM, 2011.

[13] Hoffman, Guy, and Cynthia Breazeal. Effects of anticipatory action on human-robot teamwork efficiency, fluency, and perception of team, ACM/IEEE Int. Conf. on Human-robot interaction, 2007.

[14] Kononowicz, Tadeusz W., Hedderik Van Rijn, and Warren H. Meck. Timing and time perception: A critical review of neural timing signatures before, during, and after the to-be-timed interval, Sensation, perception and attention v.2, pp. 1-35, 2016.

[15] Glasauer, Stefan, et al. ”Interacting in time and space: Investigating human-human and human-robot joint action.” Roman IEEE, 2010.

[16] Liu, Zhi, et al. Adaptive neural control for dual-arm coordination of humanoid robot with unknown nonlinearities in output mechanism, IEEE transactions on cybernetics v45:3, pp. 507-518, 2015.

[17] Nunes, Ernesto, and Maria L. Gini. Multi-Robot Auctions for Allocation of Tasks with Temporal Constraints, AAAI, 2015.

[18] Palshikar, Girish. Simple algorithms for peak detection in time-series, Proc. 1st Int. Conf. Advanced Data Analysis, Business Analytics and Intelligence, 2009.

[19] Maass, W., Natschlager, T. and Markram, H. (2002). Real-time computing without stable states: A new framework for neural computation based on perturbations. Neural computation, 14(11), 2531-2560.

[20] Maniadakis, Michail, et al. Collaboration of heterogeneous agents in time constrained tasks, Humanoid Robots (Humanoids), IEEE-RAS 16th International Conference on. IEEE, 2016.

[21] Gottwald, S. The VNR concise encyclopedia of mathematics. Springer Science and Business Media, 2012.

[22] Dollr, Piotr, et al. Behavior recognition via sparse spatio-temporal features, Visual Surveillance and Performance Evaluation of Tracking and Surveillance, 2nd Joint IEEE Int. Workshop on, 2005.

[23] Maass, Wolfgang. ”Liquid state machines: motivation, theory, and applications.” In Computability in context: computation and logic in the real world, pp. 275-296. 2011.

[24] Lukovsevcius, Mantas, and Herbert Jaeger. Reservoir computing approaches to recurrent neural network training, Computer Science Review v3.3, pp. 127-149, 2009.

[25] Mair, Elmar, et al. Adaptive and generic corner detection based on the accelerated segment test, Eur. Conf. on Computer Vision. Springer, 2010.

[26] Jain, Ashesh, Hema S. Koppula, Bharad Raghavan, and Ashutosh Saxena. Know before you do: Anticipating maneuvers via learning temporal driving models. CORNELL UNIV ITHACA NY DEPT OF COMPUTER SCIENCE, 2015.