2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 25-29, 2020, Las Vegas, NV, USA (Virtual)

Robust real-time monitoring of human task advancement for collaborative robotics applications

Riccardo Maderna[1], Maria Ciliberto[1], Andrea Maria Zanchettin[1] and Paolo Rocco[1]

Abstract— **A crucial problem in human-robot collaboration is to achieve seamless coordination among the agents. Robots have to adapt to human behaviour, which is highly uncertain. In fact, humans can perform each task in many ways and with different speeds, occasional errors and short pauses. This paper offers a robust method to monitor the advancement of the current human activity in real-time in order to predict its duration. The algorithm learns online templates of new variants of the task and uses them as references for a Dynamic Time Warping-based algorithm. The proposed strategy has been tested within a realistic assembly task. Results show its ability to give accurate predictions also in case of peculiar variants, such as those associated with errors.**

I. Introduction

In recent years, human-robot collaboration has been recognized as a paradigm to increase performance, quality, and flexibility of production in complex manufacturing environments. The concurrent presence of the collaborating agents in a shared workspace allows exploiting their strengths, such as the operator's cognitive skills and the robot's accuracy [1]. A crucial problem to achieve an effective collaboration is the seamless coordination among the agents. Robots must be aware at all times of what humans are doing and adapt to their behaviour, which is highly uncertain.

The problem of monitoring human activity has been addressed at different levels [2], such as the recognition of the current task [3]–[5] and the prediction of future ones [6]–[8]. Moreover, monitoring the evolution of the current human activity and predicting its duration allows correctly allocating tasks and reacting to unexpected changes [9]–[11]. This is a tough challenge, since the human is neither fully controllable nor repeatable: even when the operator is instructed on the task to perform, it is impossible to control its execution. At each repetition, he/she will complete the same activity with different speeds and movements. Besides, many operations can be performed in multiple ways, following different sequences of actions. The possibility of execution errors and pauses must be also taken into account. This work focuses on estimating the progress of the current human task considering such sources of variability.

As far as previous works are concerned, [12] presents a strategy to evaluate the advancement of repetitive activities, which requires the observation of task-specific features to learn motion primitives and their effect on the overall progress. [13] proposes a method that combines optimization, supervised learning, and unsupervised learning to build a Bayesian model for partial temporal sequences alignment. Pauses in task execution are explicitly taken into account. In [14] the worker is supported during assembly tasks with information on the progress and the correctness of operations. Instead, [15] detects errors in human activity by monitoring object manipulations. Abnormal behaviours are defined beforehand by domain experts using first-order logic. The method developed in [16] is robust against occlusions. It leverages a probabilistic representation of motion primitives learned from demonstration to align observations to the best fitting model. The problem of parsing complex tasks is tackled in [17], where trained Bayesian networks can also handle operation variants. The one being executed is retrieved in real-time by detecting specific primitive actions. Though unrelated to human monitoring, [18] presents a local alignment of vehicle trajectories that works offline based on Dynamic Time Warping (DTW). Vehicles can change roads following a path that is unknown a priori. Thus, trajectories do not align entirely with any of the known references.

This paper offers a robust real-time method to monitor the progress of the ongoing human activity and predict its duration. It builds on the work presented in [19], which exploits a modified version of the DTW. The human's hands movements are tracked and aligned to a reference template of the activity. Then, the expected duration is extrapolated from the average pace. Unlike other works, our approach does not require any offline training, but learns online from previous repetitions of the same activity. Besides, it handles lack of data due to occlusions or tracking errors.

However, [19] fails in the presence of task variants, as it considers only one template per activity. This paper extends the work to address this issue without the need to define all feasible variants in advance, but requiring only minimal prior knowledge of the task to monitor. It exploits a richer template, able to efficiently encode the entire structure of the task with all its known variants. Previously unseen variants are automatically recognized at run-time and added to the activity template. As an example, we focused on assembly operations, where more than one assembly sequence is feasible to obtain the same product. Also, the possibility to execute error variants has been considered (e.g. the operator delivers an incomplete product).

In the remainder of the paper, Section II summarizes the DTW-based algorithm developed in [19]. Section III presents the main contributions of the paper, i.e. the management of task variants and errors. Section IV describes the experi-

This work was supported by the Smart4CPPS project funded by Regione Lombardia (Accordi per la Ricerca e l'Innovazione, Grant ID 236789).

[1] The authors are with Politecnico di Milano, Dipartimento di Elettronica, Informazione e Bioingegneria, Piazza L. Da Vinci 32, 20133, Milano, Italy (e-mail: name.surname@polimi.it).

mental set-up used to test the proposed method in a realistic assembly task. Results are then discussed in Section V.

II. Progress-based estimate of task duration

In most scenarios, especially in an industrial setting, the human is expected to repeat a given activity multiple times, i.e. a finite set of primitive actions needed to fulfil a task. In case of low variability, one could predict the duration of the ongoing human activity using data collected from past repetitions of the same task. Let $\mathcal{T}$ be the set of past durations and T_e the elapsed time from the start of the current task, then an estimate $\widehat{T}$ of its duration is the conditional expectation of past durations that are longer than T_e:

$$\widehat{T}(T_e) = \mathbb{E}_{\mathcal{T}}[\tau \mid \tau > T_e] \tag{1}$$

However, the human may work at various speeds and perform the task in different ways, which may take more or less time than usual. Thus, data from past executions are insufficient for accurate predictions, but additional information on the present activity is needed. Given a real-time estimate of the advancement $adv(T_e)$ of the task, one can extrapolate a better prediction from the average rate of progress.

In the following, the method proposed in [19] to estimate the advancement and the duration of the ongoing activity is summarized before extending the work in Section III.

The algorithm receives as input a multivariate time sequence describing the human movements, namely the Cartesian positions of the operator's index fingers and wrists. A modified version of the DTW algorithm compares the input to a reference template of the activity, which is learnt online from past repetitions of the same task. The method is robust against nonlinear variations in the time dimension, as well as against lack of data due to tracking errors or occlusions.

Let $X = (x_1, \dots x_{|X|})$ be the input sequence that describes the partial execution of the ongoing activity and $Y = (y_1, \dots y_{|Y|})$ the reference template sequence. The algorithm builds a $|X|$-by-$|Y|$ matrix where each element stores the cumulative distance $D(i,j)$ of the optimal warping for the subsequences $X^{(i)} = (x_1, \dots x_i)$ and $Y^{(j)} = (y_1, \dots y_j)$:

$$\begin{gathered} D(i,j) = \delta + \|x_i - y_j\| \\ \delta = \min\{D(i-1,j), D(i,j-1), D(i-k,j-1)\} \end{gathered} \tag{2}$$

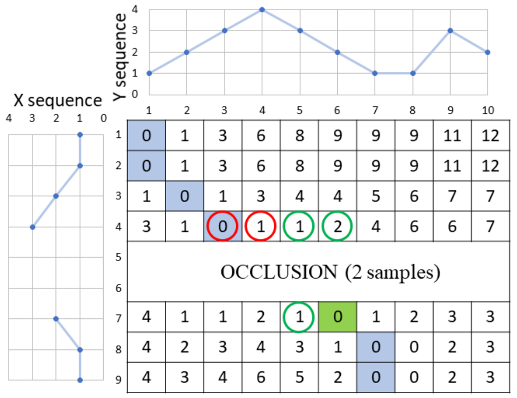
where $k = 1, \dots L_{occ}$ is added to handle occlusions (with L_{occ} the number of missing samples), and constraints are included to handle the limit cases $D(0,j)$ and $D(i,0)$.

The warping path $\phi(i)$ is a non-decreasing function that associates each point of X with the index of optimal truncation j_i^* for the reference sequence Y, so that $Y^{(j_i^*)}$ best matches the subsequence $X^{(i)}$. That is, for $i = 1, \dots |X|$:

$$\phi(i) = \max\{j_i^*, \phi(i-1)\} \qquad j_i^* = \underset{j=1,\dots|Y|}{\arg\max}\; D(i,j)$$

Fig. 1 shows an example of a DTW matrix and warping path in the presence of an occlusion.

Since the relative motion of the hands is not bound to be synchronous during human activity, the DTW algorithm is applied once for each hand. Two warping paths $\phi^{dx}(i)$ and $\phi^{sx}(i)$ are obtained, which are merged weighting more the one that estimates the highest advancement. The obtained path is then modified to increase robustness during low information sections of the task. When a pause in the progress is detected, we impose a linear growth of the warping path with speed equal to the average rate of progress of the activity. A saturation on forced growth is in place to account for real stops in the execution. One can refer to [19] for more detail.



	1	2	3	4	5	6	7	8	9	10
1	0	1	3	6	8	9	9	9	11	12
2	0	1	3	6	8	9	9	9	11	12
3	1	0	1	3	4	4	5	6	7	7
4	3	1	0	1	1	2	4	6	6	7
5	OCCLUSION (2 samples)									
6										
7	4	1	1	2	1	0	1	2	3	3
8	4	2	3	4	3	1	0	0	2	3
9	4	3	4	6	5	2	0	0	2	3

Fig. 1: Example of DTW Matrix. The warping path is highlighted in blue. Circles indicate the elements considered to compute $D(7,6)$ according to (2), the red ones mark the extension that accounts for the occlusion length.

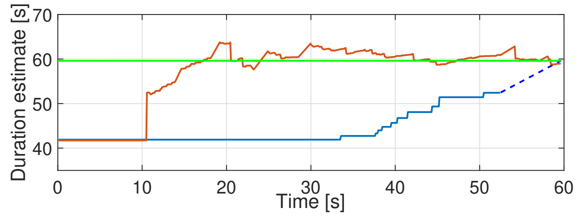


Fig. 2: Results obtained using (1) (blue) and the DTW-based algorithm from [19] (red). Actual task duration in green.

Then, the advancement of the current task at time T_e is:

$$adv(T_e) = \frac{\breve{\phi}(|X|)}{|Y|} \tag{3}$$

where $\breve{\phi}$ is the modified warping path returned by the algorithm. Finally, the expected duration is estimated as:

$$\widehat{T}(T_e) = \begin{cases} \mathbb{E}_{\mathcal{T}}[\tau \mid \tau > T_e] & adv(T_e) \leq \alpha \\ \frac{T_e}{adv(T_e)} & adv(T_e) > \alpha \end{cases} \tag{4}$$

where (1) is used to neglect the initial transient, when little information on the ongoing execution is available, until the advancement exceeds the threshold α.

Fig. 2 compares results obtained using (1) and the DTW-based algorithm in case of a task that lasted longer than average. The first method underestimates its duration based on past data, while DTW exploits information on the reduced pace of the current activity for more accurate predictions.

III. Monitoring of task variants

In general, the human can perform the same task in multiple ways and the execution times among the different variants can change considerably. In this case, the algorithm

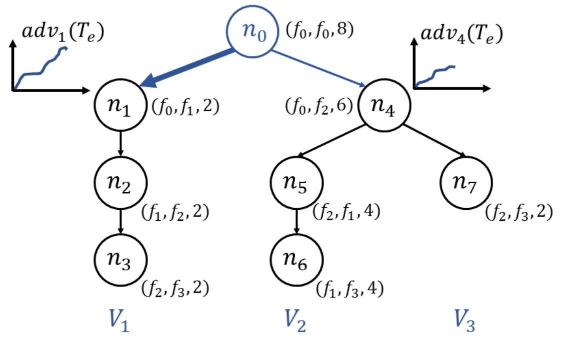


Fig. 3: Example of template tree of a task with three variants. Label for the i-th node is (o_i, d_i, r_i). After n_0, two DTWs run concurrently, one for each branch. The width of the blue arrows is proportional to their current probability.

presented in Section II offers poor performance, as it is not possible to determine a unique template that is consistent with the operator's movements for all variants.

Instead of handling each variant separately, i.e. having one reference for each variant and running multiple instances of the DTW-based algorithm concurrently, a richer template must be considered, able to efficiently describe the structure of the task with all its known variants. Besides significantly reducing the computational load to preserve real-time performance, this allows improving the estimate of task progress, recognizing variants and learning the template. However, a method to recognize task variants in real-time and to compare each execution with the correct reference is needed.

With this aim, the task is segmented online relying on the automatic detection of a set of features $F = \{f_1, \dots f_{|F|}\}$. A variant $V_i \in V$ of the task is then defined as the ordered sequence of features that occur from its start to its end. Features identify specific events during human activity (i.e. the picking of a specific part or tool during assembly operations) and are defined a priori based on general knowledge of the task. The definition of the features should allow capturing all possible behaviours of the operator and applying the algorithm presented in Section II at the segment level, i.e. the prototypical execution of each segment can be described by a single temporal sequence. Even so, there is no need to explicitly know and define all variants beforehand, as the template is built at run-time, adding previously unseen variants and learning better references for the known ones.

In the following, Section III-A describes how to build and update the template, Section III-B addresses the online recognition of variants, and Section III-C deals with the estimate of the advancement and the duration of the task.

A. Reference template structure

From the definition of variant, each execution of the task is described by a sequence of segments, which are delimited by the detection of two features. The set T collects all known reference sequences t_{ij} of $|t_{ij}|$ samples that describe the human's motion along the segment that goes from f_i to f_j.

Conventionally, the start of the activity is associated with a fictitious feature f_0 common to all variants. Then, variants differ from each other after a possible common initial part, defined by the same sequence of features, during which the operator performs the actions in the same way. As a result, the structure of the task can be represented as a tree, with nodes associated with segments and branches marking the presence of variants at that point of the execution.

More formally, let $\Theta = (N, A)$ be the template tree of the task, with N the set of nodes and A the set of arcs. A node $n_i \in N$ is defined as a 5-tuple $n_i = (o_i, d_i, p_i, C_i, r_i)$ where o_i and d_i are the origin and destination features of the segment, respectively (for the root node $o_0 = d_0 = f_0$). Each node is associated with the reference template t_{o_i,d_i} that describes the execution of the segment from o_i to d_i. For complex tasks, more than one node with the same origin and destination may exist, i.e. $\exists i \neq j : o_i = o_j \land d_i = d_j$, which refers to the same common template. p_i is the parent node, which has the destination equal to the child's origin. C_i is the set of children that collects all nodes whose parent is n_i. Arcs in A link parents to children, i.e. $a_{ij} \in A \Leftrightarrow n_i = p_j$. Finally, r_i is the number of times the segment has been already executed during past repetitions of the activity.

Given the tree, a variant $V_x = (f_0, \dots f_N)$ is equivalently defined as the ordered set of N-1 nodes connecting the root to a leaf, and referring to the segments that compose the task.

When the task to be monitored is performed for the first time, the template tree is empty. In fact, the proposed method does not rely on an offline training phase but learns online from past repetitions of the same activity. This increases applicability in dynamic environments, such as present-day manufacturing, which is characterized by fast changes in production and, consequently, in human activities. As a result, the first execution of the task cannot be monitored but is taken as the first reference. Each time the occurrence of a feature is detected, i.e. a segment of the task is completed, a new node is added to the tree and the temporal sequence that describe the human's movements is taken as the reference template for such segment in the set T.

During subsequent repetitions of the same task, the algorithm starts to monitor the first segment. To do so, it runs one instance of the DTW-based algorithm for each child of the root node. In this way, the ongoing human operation is compared to all the known variants that depart from the root node. The algorithm understands which is the most probable variant being performed and evaluates the advancement and the expected duration of the task accordingly. When the next feature is detected, marking the end of the current segment, the algorithm proceeds to monitor the next segment.

Then, new instances of DTW are run for each child n_i of the current node to monitor the next part of the task. The input is the sequence of the human hand positions collected from the start of the segment to the current time, which is compared to the reference sequence $t_{o_i,d_i} \in T$. Conversely, if none of the known children corresponds to the performed segment, i.e. the human is following a new variant, a new branch is added to the template tree. From this point, the progress of the activity cannot be estimated, but the template is updated to include the new variant in the task structure.

Any time a segment is completed, if the temporal sequence that describes its last execution is shorter than the current template in T, the former is taken as the new reference for such segment. The underlying idea is that a shorter activity is more likely to be free from pauses and errors. Moreover, improved operator training also leads to faster executions.

Fig. 3 shows an example of template tree for a task with three variants after eight repetitions of the activity. The set of reference sequences is $T = \{t_{01}, t_{02}, t_{12}, t_{13}, t_{21}, t_{23}\}$, noting that nodes n_3 and n_7 refer to the same segment.

B. Early recognition of task variants

During the monitoring of a task, the exact segment being executed is unknown. While the origin feature is known, the destination is determined only after the completion of such segment. However, early information on which variant is the one being performed is crucial to estimate the progress of the task. This is true both when there is a fork in the tree, as well as when the current node has a single child, since the human might always follow a new variant.

The proposed method for the early recognition of task variants computes a probability value for each of the branches departing from the current node $\bar{n}$, related to the last completed segment. To do so, it exploits a similarity measure that derives from the cumulative distances provided by the DTW applied at the segment level. For each $n_i \in \overline{C}$, two DTW matrices are present, to monitor the movements of the right and the left hand, respectively. A unique cost $\overline{D}_i$ is taken as:

$$\overline{D}_i(k) = \min\{D_i^{sx}(k), D_i^{dx}(k)\}$$
$$D_i^{sx}(k) = \min_j \; D_i^{sx}(k,j) \qquad D_i^{dx}(k) = \min_j \; D_i^{dx}(k,j)$$

where k is the number of samples in the input sequence, and $D_i^{sx}(k,j)$ and $D_i^{dx}(k,j)$ are the last rows of the two DTW matrices computed according to (2). Then, the similarity measure $\widetilde{D}_i(k)$ is given by the convex combination of two different costs with respect to a design parameter $\gamma \in [0,1]$:

$$\widetilde{D}_i(k) = \gamma D_i'(k) + (1-\gamma) D_i''(k)$$
$$D_i'(k) = \frac{\overline{D}_i(k)}{k} \qquad D_i''(k) = \overline{D}_i(k) - \overline{D}_i(k-1)$$

In particular, D_i' is the cumulative distance normalized over the input length, which is a global measurement of the similarity between the input and reference sequences and evolves smoothly during the evolution of the task. Instead, D_i'' considers the local rate of growth of the DTW cost and is more reactive in detecting changes in the operator's movements with respect to the reference template.

Given the similarity measure, the current probability P_i of each branch is computed using a recursive Bayesian classifier (similar to the one presented in [5]). The prior probability of each child n_i of $\bar{n}$ is based on historical data and given by:

$$P_i(0) = \frac{r_i}{\bar{r}}$$

with $\bar{r}$ the past executions of $\bar{n}$. The value is updated with each new input sample according to the following recursive rule:

$$\widetilde{P}_i(k) \propto P_i(k-1) \cdot f(\widetilde{D}_i(k)) \tag{5}$$

where f is the probability density function of a Gaussian distribution with zero mean and standard deviation $\sigma = 0.27$, whose value has been determined as the one minimizing the classification error during preliminary experiments. As the cost $\widetilde{D}_i(k)$ increases, i.e. the sequences are more dissimilar, the probability tends to decrease.

The sum of all probabilities returned by (5) might be greater than 1. Thus, a normalization is introduced as:

$$P_i(k) = \frac{\widetilde{P}_i(k)}{\max\{1, \sum_i \widetilde{P}_i(k)\}}$$

Notice that values are normalized only when their sum is greater than 1. By doing so, $1 - \sum_i P_i$ is the probability that the operator is performing an unknown variant of the task.

C. Estimate of task advancement and expected duration

Since the DTW-based algorithm is applied at the segment level, (3) would return the advancement of the current segment instead of the one of the whole task. As variants might differ in length, the advancement of the task depends on which variant is considered as the one being executed. Still, we aim to find a single value that best describes the overall progress of the activity and leads to the most accurate prediction of its duration. To do so, all the variants that are feasible at the current time instant must be considered, as we cannot know the future human behaviour.

Let $\mathcal{V}_i = \{V_x \in V | n_i \in V_x\}$ be the set of variants that contain $n_i \in \overline{C}$ and consider a partition $V_x = (V^{pre}, V_x^{post})$: $V^{pre} = (n_0, \dots \bar{n})$, $V_i^{post} = (n_i, \dots)$. Note that V^{pre} is common among all $V_x \in \mathcal{V}_i$. Then, the advancement of the task assuming the i-th branch that departs from $\bar{n}$ to be the correct one is computed as:

$$adv_i(T_e) = \frac{L^{pre} + \breve{\phi}_i(k)}{L^{pre} + L_i^{post}} \qquad L^{pre} = \sum_{n_i \in V^{pre}} |t_{o_i, d_i}| \tag{6}$$

with k the number of available input samples at time T_e, L^{pre} the length of the fraction of the task template that has been already executed, and L_i^{post} an estimate of the length of the remaining part of the task template to be performed, considering all variants belonging to the i-th branch $\mathcal{V}_i$.

Let l_{V_x} be the length of the part of the reference template of V_x that remains to be performed and π_{V_x} the probability to be the variant performed by the human based on past data:

$$l_{V_x} = \sum_{n_j \in V_x^{post}} |t_{o_j, d_j}| \qquad \pi_{V_x} = \prod_{n_j \in V_x^{post}} \frac{r_j}{r_{p_j}}$$

Then, L_i^{post} is computed as the value that minimizes the average error with respect to the length L^* of the true variant:

$$L_i^{post} = \arg\min_{L^*} \sum_{V_x \in \mathcal{V}_i} \pi_{V_x} (L^* - l_{V_x})^2 = \sum_{V_x \in \mathcal{V}_i} \pi_{V_x} l_{V_x}$$

Referring again to Fig. 3, when a new repetition of the activity starts, the current node is set to $\bar{n} = n_0$. Since two branches are present ($\mathcal{V}_1 = \{V_1\}$ and $\mathcal{V}_4 = \{V_2, V_3\}$), two instances of the DTW-based algorithm monitor the ongoing task with templates t_{01} and t_{02} associated with $\mathcal{V}_1$ and $\mathcal{V}_4$, respectively. Since no segment has been already completed $L^{pre} = 0$. Besides, $L_1^{post} = l_{V_1}$ and $L_4^{post} = 0.67 l_{V_2} + 0.33 l_{V_3}$, being for instance $l_{V_1} = |t_{01}||t_{12}||t_{23}|$.

Assuming that the i-th branch is the one being executed, one can predict the duration $\widehat{T}_i$ of the activity from (4) using the task progress computed with (6). Then, the final prediction of the duration of the task is obtained by weighting the values obtained for each branch according to their current probability to be the one performed by the human:

$$\widehat{T}(T_e) = \sum_{n_i \in \overline{C}} P_i(k)\, \widehat{T}_i(T_e) + \left(1 - \sum_{n_i \in \overline{C}} P_i(k)\right) \overline{T}(T_e)$$

The last term accounts for the possibility that the operator is following an unknown variant, with $\overline{T}(T_e)$ the average duration of the task computed from (1). As the algorithm becomes more certain about the variant being performed, the influence of unlikely branches vanishes and the prediction converges to the value related to the correct variant.

In case a new node is added to the template at the end of a segment, i.e. the human is following a new variant, the remaining part of the activity cannot be monitored. Instead, the next segments are added to the tree and $\overline{T}(T_e)$ is returned as the best duration estimate for the ongoing task.

IV. Experimental Set-up

The proposed method has been tested on a realistic assembly task. During the experiments, the movements of the human have been tracked using a Microsoft Kinect v2. The sensor directly provides the Cartesian positions of the wrists, while finger data are extracted from images with the help of coloured markers placed on the operator's gloves [19].

The task consisted in the partial assembly of a wheeled base, during which only two wheels were mounted (Fig. 4). The product parts were stored in boxes on the worktable as shown in Fig. 4: wheels for the left (1) and right (2) casters, screws to fix the casters to the base (3), swivel forks (4), and screws and nuts to fix the wheel to the fork (5). Besides, zones are present to store defective parts (6), the bases, and the completed products (not shown in the picture).

For the purpose of task segmentation, features have been defined as the operator reaching specific locations in the workspace, defined as spheres, with one index finger. In addition to f_0 that marks the start of the activity, f_1 is associated with location 1 as shown in Fig. 4, f_2 with location 2, f_3 with location 3, f_4 with location 6, and f_5 with the output zone, which marks the end of the task.

Out of all variants that can be described with the defined features, we have considered six, which lead to the template tree in Fig. 5. The variants differ in the assembly sequence used to complete the product. In V_1 the human mounts the left fork on the base, completes the caster fixing the wheel to the fork, mounts the right fork, and finally the right wheel. In V_3 operations are inverted: the human first assembles the left caster (fork plus wheel), fixes it to the base, then moves to the right caster which is mounted in the same way. During V_6 the operator fixes both forks, then assembles the wheels.

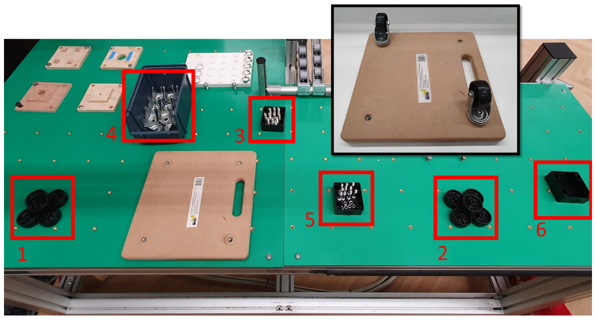
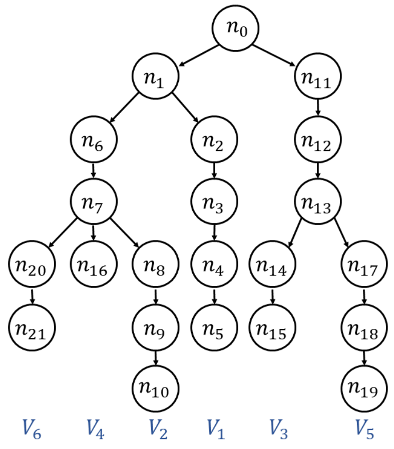


Fig. 4: Picture of the workspace and of the final product.



(o_i, d_i)	n_i
(f_0, f_1)	n_{11}
(f_0, f_3)	n_1
(f_1, f_3)	n_3, n_{12}
(f_1, f_5)	n_{21}
(f_2, f_1)	n_{20}
(f_2, f_3)	n_{14}
(f_2, f_4)	n_8, n_{17}
(f_2, f_5)	n_5, n_{16}
(f_3, f_1)	n_2
(f_3, f_2)	n_4, n_7, n_{13}
(f_3, f_3)	n_6
(f_3, f_5)	n_{15}, n_{19}
(f_4, f_1)	n_9
(f_4, f_3)	n_{18}

Fig. 5: Template tree of the assembly task performed during experiments. Note that many nodes link to the same segment.

Variants that describe error cases are also present. During V_2 and V_5 the operator finds a defective screw while fixing the right wheel. Thus, he/she has to stop and throw the screw in the waste zone before continuing. In V_4 the human forgets to mount the left wheel and delivers an incomplete product.

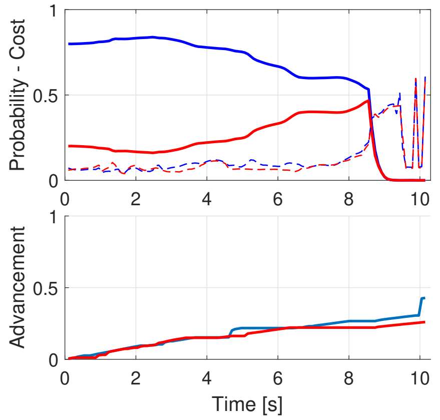
V. Experimental Results

The experimental campaign aimed to verify the variant recognition mechanism and the update of the template tree, and to assess the performance of the proposed algorithm in terms of prediction errors of the activity duration.

A. Variant recognition

The process of building the template from scratch has been repeated three times, assembling each time 15 products following a random sequence of variants. Overall, the algorithm always recognizes the correct variant as soon as the actions of the human differ from the known templates.

To exemplify its behaviour, we can focus on the three-way branch (formed by V_2, V_4 and V_6) that stems from node n_7 in Fig. 5, as it represents the most critical point for variant recognition. Let us consider the case when V_6 and V_4 are already known, i.e. are included in the template tree, and V_2 is performed for the first time by the operator (Fig. 6). The first segments (f_0, f_3), (f_3, f_3), (f_3, f_2) are common among the three variants. Thus, the algorithm is not aware of the new variant and the task is monitored as if it was V_4 or V_6.



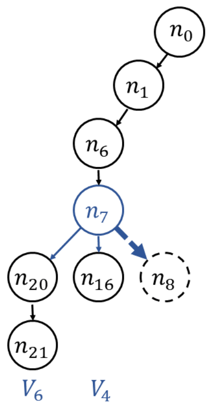


Fig. 6: Recognition of variant V_2, known V_4 and V_6. Both children of n_7 are deemed as wrong (≈ 8.5 s) and node n_8 is added to the tree. Graphs show P_i (top, solid), $\widetilde{D}_i$ (top, dashed) and adv_i (bottom) for segments (f_2, f_1) (red) and (f_2, f_5) (blue).

After node n_7, a fork is already present: one instance of the DTW-based algorithm runs for each of the two branches and their probabilities are updated with each new sample.

At first, the algorithm cannot decide which is the correct variant, since the initial part of segments (f_2, f_1), (f_2, f_5) and (f_2, f_4) is common, as the operator mounts the first wheel. Then, either he/she fixes the second (V_6), or delivers an incomplete product (V_4), or finds a defective screw (V_2). As soon as the human movements differ from both known templates (≈ 8.5 s after the start of the current segment), the algorithm recognizes the presence of a new variant: the costs of both DTWs increase and probabilities drop to zero. It can be also noticed that the advancement of the wrong variants remains below 0.5 at the end of the segment.

Fig. 7 shows a full example of one execution of V_4 when the variant is already known. V_4 represents an error variant that is rarely performed and lasts less than average. Initially, the expected duration is overestimated, as the prediction favours more usual variants, like V_6. As information become available, the prediction converges to the actual task duration. Notice that the correct branch is recognized even if it has the lowest prior probability.

B. *Performance evaluation*

To evaluate performance, the proposed *Multi-variant* (MV) algorithm is compared to other two methods. The first one is the *Single-Variant* (SV) algorithm developed in [19], which describes the prototypical execution of the task with a single template sequence. The second is an ideal method, referred to as the *Prophet algorithm* (PA), which has perfect knowledge of the variant being performed. It behaves as an SV algorithm trained only on past executions of the variant to monitor.

Results come from 33 repetitions of the assembly task. Respectively, the six variants have been repeated 12, 3, 6, 2, 3, and 7 times, reflecting the fact that error ones are infrequent in a real case. To neglect the learning transient, data from previous experiments have been used as a training set to build the template. The average duration of the task has been 105.0 s, from a minimum of 72.9 s to a maximum of 131.0 s, thus showing high variability. The performance index is the average error in the prediction of the duration, normalized over the actual duration of the task T^*:

$$e_m = \frac{1}{T^*}\int_0^{T^*} \frac{|\hat{T}(\tau) - T^*|}{T^*} d\tau$$

Fig. 8 shows examples of the results obtained by the three methods following different task variants. MV consistently gives more accurate predictions than SV, while PA attains the best results. SV fails as the only template sequence it considers is compared to the input regardless of the variant being performed. Since the algorithm takes the shortest past execution of the activity as the reference, this is likely to be one of V_4, i.e. when an incomplete product is delivered.

Table I reports the errors obtained for the different methods. While SV attains a median error (q_{50}) of 17.99% of the task duration, MV gives accurate predictions, with a median error of 4.76%, which means less than 5 s for the average activity. Besides, the prediction becomes more precise as the task progresses, with the error that reduces to 2.01% if only the last 20 s of each execution are considered. The error dispersion is also significantly reduced, with the interquartile range that halves from 6.98% to 3.04%.

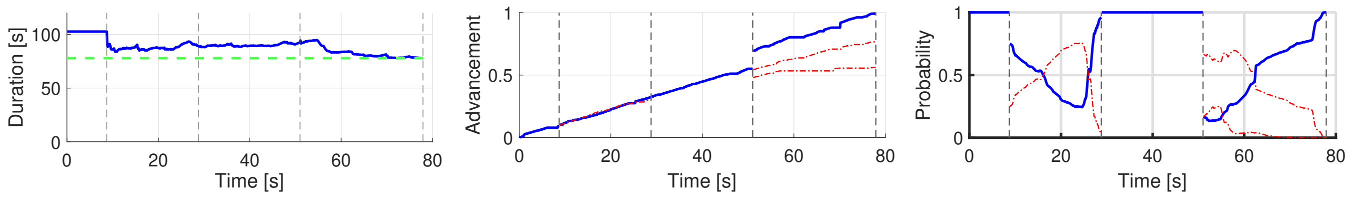
The improvement mainly comes from the fact that the availability of more data reduces the uncertainty of unknown quantities, such as the rate of advancement of the task (from which the expected duration is extrapolated) or the estimate of L_{post_i} used in (6). Also, L^{pre}, whose value is certain, grows with the number of completed segments, while L_{post_i} lowers becoming less important. This kind of uncertainty may produce peaks or valleys in the duration estimate in the first part of the task, similar to the one shown in Fig. 8a.

Fig. 8b reports the results of one execution of V_2, which is an infrequent variant and lasts longer than average. The presence of the more probable variant V_6 in the same part of the template tree leads to an underestimation of the duration of the ongoing task. When the operator performs segment (f_2, f_5), the probability of the correct variant raises and the prediction converges to the actual operation duration.

From Table I we can also notice that MV and PA attain the same performance, with the median of the error difference smaller than $\pm 0.5\%$. This is a major result, which means that the presence of several variants of the task does not worsen the estimate of its duration. An example is shown in Fig. 8c for one instance of V_6. Despite the monitoring of the task requires the evaluation of three forks in the template, one of which composed of three branches, the predictions of MV and PA can be mostly superimposed. Note that the slightly better performance attained by MV in the last 20 s may be due to the more data it has available for template learning.

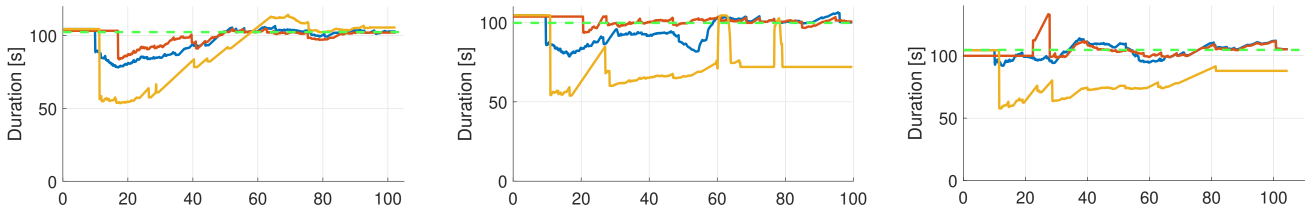
VI. Conclusion

The paper presented a robust method for the real-time monitoring of human task advancement, able to handle the variability of human behaviour and the presence of different variants of the same activity. To recognize the one being performed by the operator, the algorithm exploits real-time



(a) Expected (blue) and actual (green) duration. (b) adv of correct (blue) and wrong (red) segments. (c) P_i of correct (blue) and wrong (red) segments.

Fig. 7: Results for one execution of V_4: as the correct variant is recognized, the duration estimate converges to the real one.



(a) V_1 (b) V_2 (c) V_6

Fig. 8: Results of $\widehat{T}(T_e)$ for different variants obtained with MV (blue), PA (red) and SV (yellow). Actual duration in green.

e_m [%]	Total			Last 20 s		
	q_{25}	q_{50}	q_{75}	q_{25}	q_{50}	q_{75}
SV	5.22	17.99	30.57	3.13	11.35	22.35
MV	2.19	4.76	9.17	0.75	2.01	3.79
PA	2.07	4.46	8.22	1.09	2.85	5.76
MV–PA	-2.68	0.32	3.29	-2.38	-0.34	0.35

TABLE I: Average prediction errors (25-50-75 percentiles) for the complete execution (1^{st} column) and the last 20s (2^{nd} column).

segmentation of the task and a recursive Bayesian classifier, which also allows detecting previously unseen variants. The complete structure of the task in encoded in a template tree that is learnt online from past repetitions of the same activity, thus not requiring a prior training phase.

Results show that the proposed method gives accurate estimates of the task duration also in case of errors, such as the delivery of an incomplete product. Besides, the presence of multiple variants does not worsen performance.

Future works will investigate how to improve task segmentation. The definition of features a priori limits the number of task segments that can be handled by the algorithm. Instead, an automatic definition of features would also allow optimizing segmentation to minimize the similarity among different segments, improving variant recognition.

References

[1] A. Ajoudani, A. M. Zanchettin, S. Ivaldi, A. Albu-Schäffer, K. Kosuge, and O. Khatib, "Progress and prospects of the human—robot collaboration," *Auton. Robots*, vol. 42, no. 5, pp. 957–975, Jun. 2018.

[2] B. Hartmann, *Human worker activity recognition in industrial environments*. KIT Scientific Publishing, 2011.

[3] R. Kelley, A. Tavakkoli, C. King, M. Nicolescu, M. Nicolescu, and G. Bebis, "Understanding human intentions via hidden markov models in autonomous mobile robots," in *Proceedings of the 3rd ACM/IEEE Int. Conference on Human Robot Interaction*, 2008, pp. 367–374.

[4] C. Pérez-D'Arpino and J. A. Shah, "Fast target prediction of human reaching motion for cooperative human-robot manipulation tasks using time series classification," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, May 2015, pp. 6175–6182.

[5] A. M. Zanchettin and P. Rocco, "Probabilistic inference of human arm reaching target for effective human-robot collaboration," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2017, pp. 6595–6600.

[6] A. M. Zanchettin, A. Casalino, L. Piroddi, and P. Rocco, "Prediction of human activity patterns for human–robot collaborative assembly tasks," *IEEE Transaction on Industrial Informatics*, vol. 15, no. 7, pp. 3934–3942, 2019.

[7] H. S. Koppula and A. Saxena, "Anticipating human activities using object affordances for reactive robotic response," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 38, no. 1, pp. 14–29, 2016.

[8] K. Li and Y. Fu, "Prediction of human activity by discovering temporal sequence patterns," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, no. 8, pp. 1644–1657, Aug 2014.

[9] V. V. Unhelkar, P. A. Lasota, Q. Tyroller, R. Buhai, L. Marceau, B. Deml, and J. A. Shah, "Human-aware robotic assistant for collaborative assembly: Integrating human motion prediction with planning in time," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 2394–2401, July 2018.

[10] D. Riedelbauch and D. Henrich, "Exploiting a human-aware world model for dynamic task allocation in flexible human-robot teams," in *2019 Intern. Conf. on Robotics and Automation*, 2019, pp. 6511–6517.

[11] A. Casalino, A. M. Zanchettin, L. Piroddi, and P. Rocco, "Optimal scheduling of human-robot collaborative assembly operations with time petri nets," *IEEE Transactions on Automation Science and Engineering*, pp. 1–15, 2019.

[12] E. Hourdakis and P. Trahanias, "A robust method to predict temporal aspects of actions by observation," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, May 2018, pp. 1931–1938.

[13] P. A. Lasota and J. A. Shah, "Bayesian estimator for partial trajectory alignment," in *2019, Robotics: Science and System (RSS)*, Jun. 2019.

[14] S. Kaczmarek, S. Hogreve, and K. Tracht, "Progress monitoring and gesture control in manual assembly systems using 3d-image sensors," *Procedia CIRP*, vol. 37, pp. 1 – 6, 2015, cIRPe 2015 - Understanding the life cycle implications of manufacturing.

[15] G. Civitarese and C. Bettini, "Monitoring objects manipulations to detect abnormal behaviors," in *2017 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*, March 2017, pp. 388–393.

[16] G. Maeda, M. Ewerton, G. Neumann, R. Lioutikov, and J. Peters, "Phase estimation for fast action recognition and trajectory generation in human–robot collaboration," *The International Journal of Robotics Research*, vol. 36, no. 13-14, pp. 1579–1594, 2017.

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

[18] C. D. Chitraranjan, A. S. Perera, and A. M. Denton, "Tracking vehicle trajectories by local dynamic time warping of mobile phone signal strengths and its potential in travel-time estimation," *2015 IEEE International Conference on Pervasive Computing and Communication Workshops (PerCom Workshops)*, pp. 445–450, March 2015.

[19] R. Maderna, P. Lanfredini, A. M. Zanchettin, and P. Rocco, "Real-time monitoring of human task advancement," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2019.