POLITECNICO DI MILANO

School of Industrial and Information Engineering

Master of Science in Automation and Control Engineering

POLITECNICO
MILANO 1863

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

Supervisor: Prof. Paolo ROCCO

Co-supervisor: Ing. Riccardo MADERNA

Author:

Maria CILIBERTO ID: 905403

Academic Year: 2019 – 2020

Contents

List of Figures

List of Tables

Abstract

In recent years, the interest in collaborative robotics has been continuously growing. This is due to the fact that humans and robots working together can efficiently complete tasks that are very difficult for either agent to accomplish alone. The robot executes operations that require strength and could be dangerous for the human to perform, while the operator adds flexibility to the task. To collaborate fluidly, robots must recognize humans' intentions and adapt to their actions appropriately. However, from the interaction between robots and humans, a large amount of problems arise, mainly due to the uncertainties caused by the latter. This thesis offers a robust way to estimate the advancement of the current human operation, considering the variability introduced by the human being. Our work accounts for the possibility that the operator performs the operation in different ways and estimates the progress of the current operation taking into account this issue. From the estimate of the advancement, it is then possible to derive the expected duration of the current task. The proposed method compares the ongoing sequence with a set of possible reference ones and this is done with the use of the Open-Ended Dynamic Time Warping. The proposed solution proved to be satisfactory, guaranteeing low errors in estimating the duration of operations and robustness to the uncertainties introduced by the human.

Keywords: Collaborative Robotics, Assembly, Dynamic Time Warping, Temporal Sequences.

Sommario

La robotica collaborativa è un settore in continua espansione. Ciò è dovuto al fatto che operatori umani e robotici che collaborano possono completare efficacemente operazioni che altrimenti sarebbero difficili da eseguire singolarmente. Il robot conduce operazioni che richiedono forza e che potrebbero essere pericolose per l'umano, mentre l'operatore aggiunge flessibilità. Per collaborare in maniera efficiente, è necessario che i robot riconoscano le intenzioni dell'umano per adattarvisi e scegliere di conseguenza il prossimo compito da eseguire. Dall'analisi dell'interazione tra uomo e robot scaturiscono diversi problemi, alcuni dei quali dovuti al fatto che l'essere umano ha un comportamento non sempre predicibile. Questa tesi offre un metodo robusto per stimare l'avanzamento dell'operazione collaborativa che tiene in considerazione l'incertezza introdotta dall'umano. Il nostro algoritmo affronta la possibilità che l'uomo possa eseguire l'operazione in diversi modi e stima l'avanzamento dell'operazione corrente tenendo in considerazione questo aspetto. Il metodo proposto paragona la sequenza corrente dei movimenti dell'operatore a un insieme di sequenze di riferimento possibili e ciò è possibile tramite l'Open-Ended Dynamic Time Warping. La soluzione proposta si è rivelata soddisfacente, garantendo bassi errori nella stima della durata delle operazioni e robustezza alle incertezze introdotte dall'umano.

Parole chiave: Robotica Collaborativa, Assemblaggio, Dynamic Time Warping, Sequenze Temporali.

Chapter 1

Introduction

The project presented in this thesis is part of the research related to industry 4.0. This term refers to the current trend of industrial automation to integrate new production technologies for improving working conditions, increasing performance and quality production of the plants. A role of fundamental importance in this revolution is carried out by collaborative robotics. Robots are no longer used solely to replace the human operator where possible, but also to interact with the latter in order to improve working conditions and performance. Examples are the collaborative assembly operations, where automata are used for the production or to perform repetitive actions that require precision, while the human operator performs the operations that are difficult to automate and increase their flexibility. To exploit to the fullest extent the resources, robot and human being, and to make the best use of the time available to them, the robot should be able to correctly estimate the duration of the ongoing operation performed by the human and to schedule its tasks consequently. This is not an easy task since the operator's behaviour is neither controllable nor repeatable. For example, if the human is performing the same operation several times, he/she will perform it at different paces and will not always exactly repeat the same movements. Moreover, some assembly operations can be performed in different ways, for example, the human could take

an element before another one and still get to the final product. So it is important to find a method to estimate the progress of the human task considering such variability.
In this thesis, a method for monitoring in real-time the operation performed by the human operator, considering the issues previously explained, was developed. The actual progress of the operator is estimated in order to obtain an estimate of the progress of the current operation at each time instant. This is then used to calculate the expected duration of the current operation and therefore the remaining time before its conclusion.

1.1 State of the art

There are several articles in the literature concerning human-robot collaboration([1],[3]). Many challenges arise to achieve effective collaboration: in addition to safety issues([15], [14]), the estimation of the progress of the operation performed by the human is particularly interesting ([2], [11]). Since the human is a source of uncertainty in many researches, the prediction of human activity has been discussed. In [5] a Hidden Markov Model was trained to infer the current human activity as perceived by a mobile robot. In [12] the human activity pattern is modelled with higher-order Markov Chains. [10] used gaze information to monitor the worker's activity and interpret his/her future intention. While [8] and [13] discuss different ways to solve this problem using probabilities.
In order to have real-time monitoring of the task advancement, we started from the work in [6],[9] that estimates the duration of the ongoing operation by monitoring the human's skeletal positions and compares them with a reference sequence to evaluate at which point of the reference the operation is. The expected duration of the operation is then found as a function of the advancement percentage and of the elapsed time at each instant. The main limitation of this algorithm is that it considers just one reference sequence, so when the operator performs a variant of the task it cannot

predict precisely the final duration. This is due to the fact that, since the operation is performed in a different way, the human's movements will be different from the ones of the reference sequence.

1.2 Proposed solution

This thesis aims at providing a robust method to estimate the duration of the operation performed by the human operator in collaborative robotics applications. In particular, our work consists in a real-time monitoring of the ongoing task and is robust to the variability introduced by the poor predictability of the human.
We focused on the possibility that the human could change the order of the operations that compose the task, considering also error variants (for example, the operator forgets to pick an element for the operation or he finds a defective part and has to substitute it). To do so it is necessary to have a set of reference sequences, each one corresponding to a variant of the task execution, which in turn are made up of several segments. Since we wanted to develop a plug-and-play application, we decided to add as reference a temporal sequence that the operator has performed at least once. The algorithm keeps track of the last segment performed and since it knows which are the possible next segments, he monitors the ongoing part of the operation with all of them. Then two situations were considered:

- the human is performing a variant already present in the set of reference sequences
- the ongoing operation is a new variant.

In the first case, the algorithm is able to recognize which variant is the most probable to be performed and evaluates the advancement rate and the expected duration. To give a good estimate of the duration also in the second case, we implemented a function with which the algorithm can perform an early recognition of the new variant. [7] proposes a method

to align partial temporal sequences, but it involves offline computations. Conversely, our method works in real-time, making the algorithm easier to use.

1.3 Chapters organization

The remainder of this thesis is organized as follows:

The second chapter describes the work developed in [6] and [9] which served as a starting point for this thesis.

The third chapter explains in detail the changes made to the previous work. In particular, the creation of a reference template that considers the possible variants of the operation execution is described.

The fourth chapter presents the experimental results obtained by applying the algorithm developed and comparing it with basic algorithms.

The fifth chapter draws the main conclusions about the work presented in this thesis and presents cues for future works.

Chapter 2

Monitoring of a single task

As it has been said in Section 1.1 the starting point of this thesis is based on [6]. Section 2.1 describes a simple method for the estimation of the operation duration. After that,the algorithm from which we started is presented. In particular, Section 2.2 explains the Dynamic Time Warping algorithm. In Section 2.3 a modification of the algorithm called Open-Ended Dynamic Time Warping is described. Section 2.3.1 presents the changes applied to it for monitoring the advancement of a task. Finally, Section 2.4 shows the results obtained in the previous work and its main limitation.

2.1 Past duration average method

The method presented here estimates the duration of the current operation as the average of the duration of the previous ones, according to the equation:

$$\hat{T}_D(T_e) = \mathbb{E}[\tau \in \mathcal{T} | \tau > T_e] \tag{2.1}$$

where $\mathcal{T}$ represents the vector containing all previous operations duration saved in memory and T_e is the time elapsed since the start of the current operation. $\hat{T}_D(T_e)$ is then calculated as the average of the elements of $\mathcal{T}$ with longer duration. Each time the duration of the operation previously

carried out is exceeded by the current one, $\hat{T}_D(T_e)$ is evaluated as the elapsed time T_e. The algorithm just presented works correctly if the past operation and the current one have a similar duration. However, it is possible that the human executes the operation at different speeds or perform some variant of it, so that the operation execution takes more or less time than usual. Therefore, we present a method that links the duration of the current ongoing operation to the estimated progress of it to take this problem into account.

2.2 Dynamic Time Warping

The Dynamic Time Warping (DTW) algorithm allows to measure the similarity between different temporal sequences. In our case, the algorithm receives as input a time sequence and returns as output the measure of similarity with a reference time series. The underlying idea of DTW is to locally deform the time axes of the input signal in order to associate each of its points to one point of the reference sequence. This makes the algorithm robust to variations in time dimension. Let $X = \{x_1...x_i...x_N\}$ be the input sequence and $Y = \{y_1...y_j...y_M\}$ the reference one. Let φ_x and φ_y be two functions that map the indices i of X and j of Y to a common index $k \in \{1, 2...T\}$ where T is the current operation length, such that $\varphi_x(k) = i$ and $\varphi_y(k) = j$. It is possible to combine these two functions in order to obtain a general warping function $\varphi\{\varphi_x(k), \varphi_y(k)\}$ and the cumulative distance between X and Y denoted as $D(X, Y, \varphi)$:

$$D(X, Y, \varphi) = \sum_{k=1}^{T} c(x_{\varphi_x(k)}, y_{\varphi_y(k)}) \tag{2.2}$$

where $c(x, y)$ could be any function that evaluates the distance between X and Y. The aim of DTW is to find φ that minimizes $D(X, Y, \varphi)$. The algorithm builds an $N - by - M$ matrix where each element (i, j) stores the cumulative distance $D(i, j)$ of the optimal warping and it is computed as:

$$D(i,j) = d + s(i,j) \tag{2.3}$$

where s is the Eucledian distance between x_i and y_j and d is:

$$d = min\{D(i-1,j), D(i,j-1), D(i-1,j-1)\} \tag{2.4}$$

The following constraints are taking into account to handle limit cases:

$$\begin{cases} D(0,0) = 0 \\ D(i,0) = \infty \quad \forall i; \\ D(0,j) = \infty \quad \forall j; \end{cases} \tag{2.5}$$

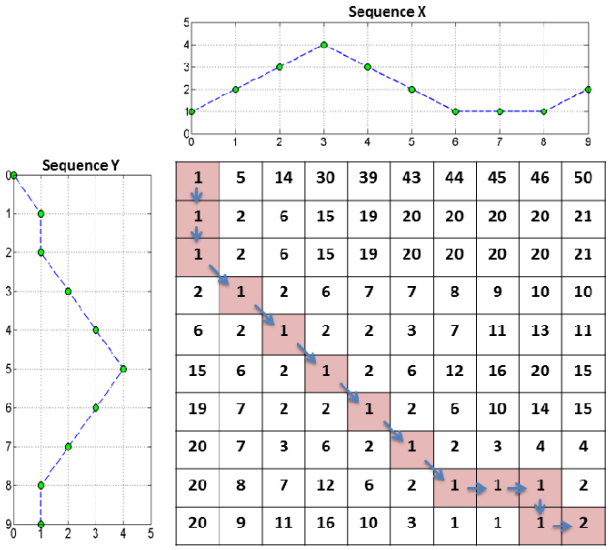


Figure 2.1: Example of DTW Matrix

In figure 2.1 an example of DTW matrix is shown where the red path represent the optimal alignment φ.

2.3 Open-Ended Dynamic Time Warping

The algorithm that has been previously explained, is mainly used for the offline comparison of complete signals. Instead, a modification called

Open-Ended Dynamic Time Warping (OE-DTW)([11]) allows to compare an input sequence that is only representative of a partial execution of the template with the reference sequence. Thanks to this peculiarity, it is possible to apply the OE-DTW to monitor operations in real time. The main intuition consists in applying several times DTW between X and different signals $Y^{(j)}$ obtained by truncating Y to the j-th element $\forall j = [1...M]$ where M is the number of elements of Y. The open-ended similarity is then defined as:

$$D_{OE} = \min_{j=1...M} D_{DTW}(X, Y^{(j)}) = \min_{j=1...M} D(N, j) \tag{2.6}$$

where $D(N, j)$ is the last row of the DTW matrix. This means that DTW can be applied just once in order to retrieve the open-ended similarity, thus allowing an efficient computation.

2.3.1 Task advancement monitoring

In this work we are interested in finding the advancement of the ongoing task, so the information obtained with the OE-DTW are used to find the optimal truncation index j_N^* that indicates at which point of the reference sequence the ongoing task is. From this information the associated percentage of advancement $adv\%$ is evaluated as:

$$adv\% = \frac{100 j_N^*}{M} \tag{2.7}$$

where $j_N^* = \underset{j=1...M}{argmax} D(N, j)$. Furthermore, one can estimate the duration $\hat{T}$ of the ongoing operation at time instant k knowing the elapsed time $T_e(k)$ and the percentage of advancement[1] as:

$$\hat{T}(k) = \frac{T_e(k)}{adv\%(k)} \tag{2.8}$$

[1]We considered the possibility to use a moving window to evaluate the expected duration of the operation to take into account sudden changes in the advancement rate and improve the duration estimate. See Appendix A for further information.

2.3.2 Input signals

Since our aim is to monitor human task advancement in collaborative robotics we have chosen to use the operator's index fingers and wrists Cartesian position as input signals. It is important to separate movements of the left side of the human from the ones of the right side because they are not forced to be synchronous. Therefore, the algorithm is applied twice and the two obtained warping paths are merged together into an average warping path $\bar{\varphi}$ as follows:

$$\begin{cases} \bar{\varphi}(k) = \frac{\varepsilon \cdot \varphi^{sx}(k) + (1/\varepsilon) \cdot \varphi^{dx}(k)}{\varepsilon + 1/\varepsilon} & \varphi^{dx} \geq \varphi^{sx} \\ \bar{\varphi}(k) = \frac{\varepsilon \cdot \varphi^{dx}(k) + (1/\varepsilon) \cdot \varphi^{sx}(k)}{\varepsilon + 1/\varepsilon} & \varphi^{dx} < \varphi^{sx} \end{cases} \tag{2.9}$$

where φ^{sx} and φ^{dx} are the warping paths obtained by applying DTW respectively to the left and right motion, and ε is a weight with an exponential behavior that decreases proportionally to the difference between the two indices $|\varphi^{sx} - \varphi^{dx}|$. Be α a designed parameter used to determine the decreasing speed, then $\varepsilon = \exp^{-\alpha \cdot |\varphi^{sx} - \varphi^{dx}|}$

2.3.3 Handling occlusions

Loss of data during the monitoring of human activity may occur due occlusions of features or tracking errors. Two cases are possible: either only some of the signals are missing and the algorithm works as usual considering only the available dimensions, or all features are unavailable at the same time. The latter case is more critical: the algorithm shown so far would update the cumulative distance $D(i, j)$ searching for the minimum distance d among the neighbors of the point (i, j), but when an occlusion occurs, input samples are discarded, so that the first points before and after the occlusion are considered to be subsequent. This fact causes wrong alignments and leads to low performance of the algorithm.

For this reason the rule shown in 2.4 is modified as follows to take into account the occlusion length in the minimum search:

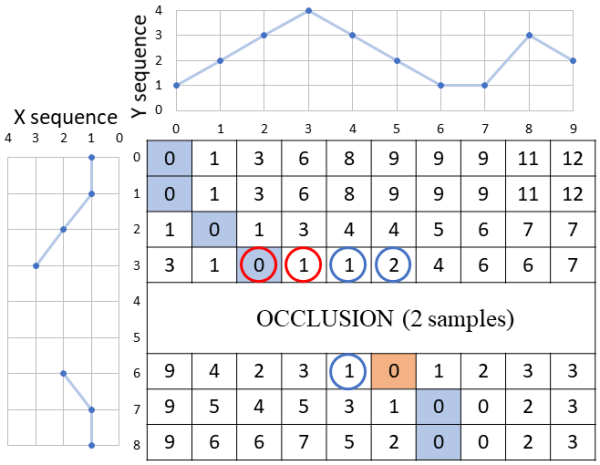


Figure 2.2: DTW Matrix with Occlusion: DTW exploration space is represented with blue circles while the red circles represent the OE-DTW one.

$$d = min\{D(i-1,j), D(i,j-1), D(i-k,j-1)\} \tag{2.10}$$

where $k = 1...L_{occ}$ and L_{occ} is the occlusion length. With this expedient, the algorithm can detect a missing part of the signal and is able to handle the occlusion and retrieve the correct result (figure 2.2).

2.3.4 Selection of the reference

To increase flexibility of the algorithm a learning phase is not required and the first instance of the activity to be monitored is taken as the first reference sequence. Then, the reference sequence is updated when one instance results shorter than the reference one. This is done because the operator becomes familiar with the operation by repeating it several times and also because a shorter execution is more likely to be free from errors and pauses.

2.3.5 Output Filtering

From the modified DTW, we obtain the advancement of the current activity, which can be used to compute the expected duration. However, it is inconvenient to directly use the DTW output in the equation, as it suffers from a couple of problems. Firstly, sections of the activity during which the human is mostly still may cause a temporary stop in the advancement, followed by a jump when motion returns informative. The second problem is the presence of a bad initial transient, when only few data are available and progress estimate is poor. In order to address the first problem, a finite state automata with three states that describe the possible situations has been modeled. The states are:

- N: Normal behavior;
- L: Linear growth;
- S: Saturation.

In the first case (N) the algorithm behaves in the standard way, when a pause is detected we impose a linear growth (L) with speed equal to the average rate of progress of the activity. To set a limit on forced growth, the case of saturation (S) has been devised. The transitions between states (figure 2.3) are triggered by:

a) $N \to L : \varphi(k) = \varphi(k-1)$;

b) $L \to S : \varphi(k) > \varphi(k-1) \vee \breve{\varphi}(k-1) - \varphi(k) > \delta_M$ where δ_M is a limit threshold;

c) $S \to N : \varphi(k) \geq \breve{\varphi}(k-1)$.

where $\varphi(k) = \varphi(k-1)$ and $\varphi(k) > \varphi(k-1)$ respectively represent a pause and a jump in the advancement are due to DTW errors. The additional second condition on the $L \to S$ transition is used to represent a

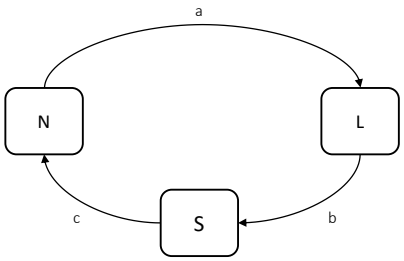


Figure 2.3: Phase transitions: When a pause is detected we impose a linear growth to the advancement, in case there is a real stop in the execution of the operation the growth is saturated. When the operation returns informative we return to the normal behavior state.

real stop in the execution performed by the human. The modified warping path $\breve{\varphi}$ is computed as follows depending on the current state:

$$\begin{aligned} N : \breve{\varphi}(k) = &\quad max\{\varphi(k), \breve{\varphi}(k-1)\} \\ L : \breve{\varphi}(k) = &\quad \tfrac{\breve{\varphi}(k-1)}{k-1} \cdot k \\ S : \breve{\varphi}(k) = &\quad \breve{\varphi}(k-1) \end{aligned}$$

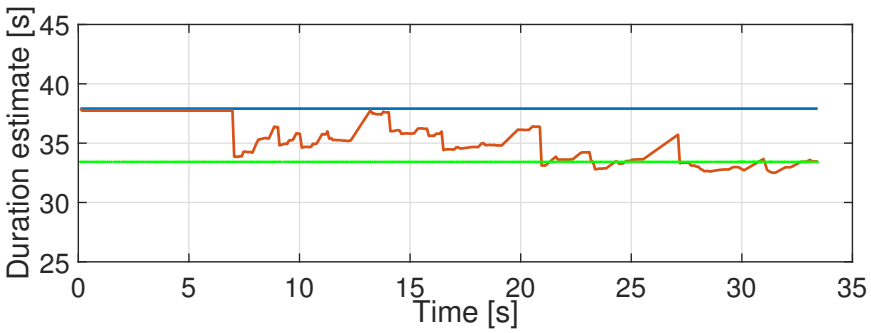
Due to the small number of samples at the beginning of monitoring the initial transient offered poor performance, so the rule described in 2.8 has been modified as follows:

$$\hat{T} = \begin{cases} \mathbb{E}[\tau \in \mathcal{T} | \tau > T_e] & adv\% \leq 20\%; \\ \frac{T_e}{adv\%} & adv\% > 20\%; \end{cases} \tag{2.11}$$

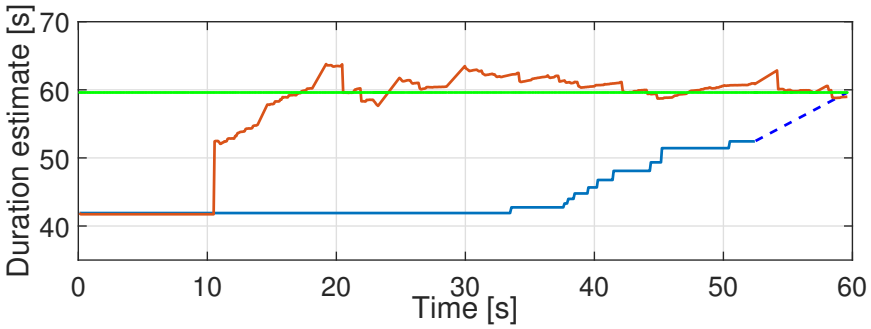
where $\mathbb{E}[\tau \in \mathcal{T} | \tau > T_e]$ is the average duration of the past executions of the operation that are greater than the elapsed time as it has been described in Section 2.1.

2.4 Results

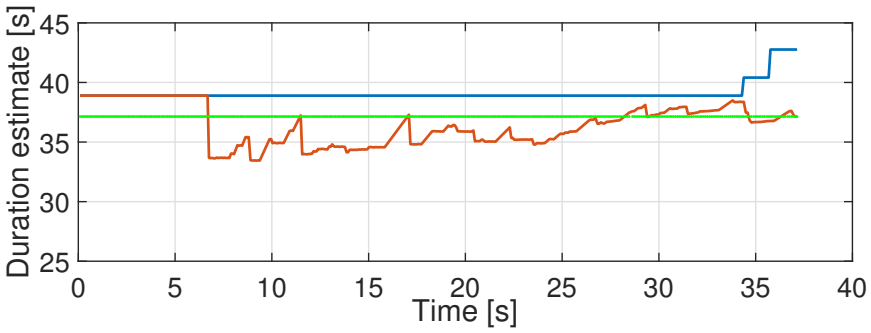
With all previous adjustments, the algorithm works well as it estimates the duration of the operation appropriately even in critical situations. Figure 2.4 shows the comparison between the results obtained with DTW algorithm (the ones in orange), the ones obtained with the basic algorithm explained in Section 2.1 in blue and the actual duration of the operation which is represented in green. In figure 2.4a the graph of the duration estimate of the second implementation of the operation is shown, in this case the basic algorithm does not have enough data to provide a correct estimate. In figure 2.4b the result obtained when monitoring a longer operation with respect to the reference one are presented and it can be seen that the basic algorithm underestimates the final duration. Lastly, when the operation is shorter than the reference (figure 2.4c), the basic algorithm tends to overestimate the final duration since the average of the past ones results higher. A longer instance could be due to a tired operator or to an error in the operation, while a shorter one is possible due to the fact that the human becomes more and more experienced and faster in the execution of the operation. Conversely, DTW algorithm works well in all the three situations.



(a) Second Execution



(b) Execution longer than average



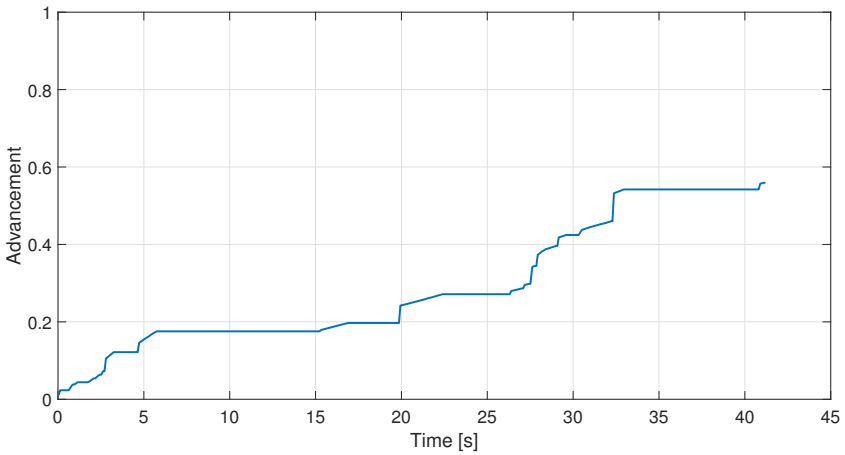
(c) Execution shorter than average

Figure 2.4: Example of results obtained using the DTW-based algorithm

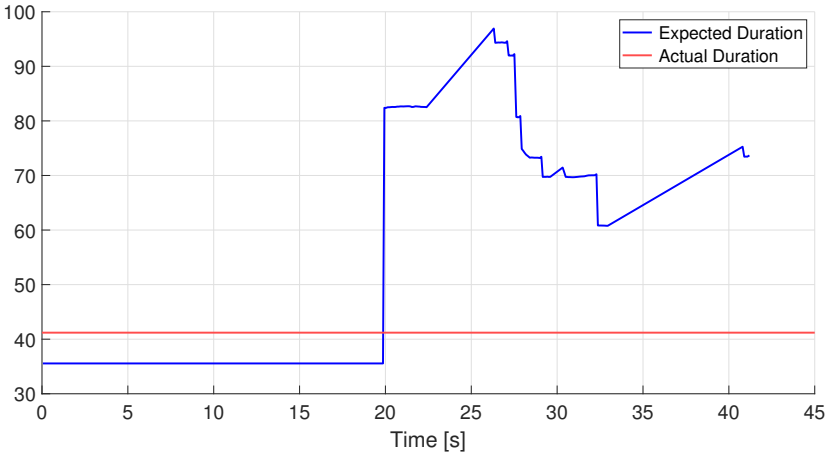
As it has been shown, the proposed algorithm offers good results as long as the operations to be monitored are always performed in the same way. Figure 2.5 shows the results obtained when the human executes a variation of the operation. In particular, the operation consisted in the assembly of a caster wheel. The first time, the operator picks the main body and

insert a screw to allow for wheel mounting, then takes the rubber wheel which is fixed using another screw. The sequence obtained was taken as reference and used in the monitoring of a second instance when the human changed the order of the operations, mounting the rubber wheel first, then inserting the mounting screw. It is apparent that the algorithm offers poor performance in this case. The advancement keeps a constant value for long periods and consequently does not reach 100%. As a consequence the expected duration is greater than the real one. Since the algorithm works comparing the actual position of the human's skeletal points, it is straightforward that when the human changes the order in which he performs the operation, the DTW matrix costs become high and a bad estimation of advancement and expected duration follow.

In general, a human can perform a task in different ways and the execution times among the different variants can change considerably. In collaborative robotics applications it is necessary to know the exact duration of the task in order to better schedule the next operations. So we worked on the algorithm to take into account this issue. Changes applied to give more robust estimates are shown in the next chapter.



(a) Advancement Estimate



(b) Duration Estimate

Figure 2.5: Example of error case using DTW in presence of variants of operation.

Chapter 3

Monitoring of task variants

3.1 Introduction

As it has been shown in Section 2.4, the algorithm from which we started is able to estimate the duration of a task when it is always executed in the same way, while it offers poor performance when the human changes the order in which he performs the operation. Our work focuses on the online recognition of the task variants to be able to compare each instance with the associated template, instead of using one single template for all the instances. In the following, the definition of variant is declared and the explanation of how the template is built and updated is presented. Section 3.4 shows the main intuition behind the recognition of a new variant while Section 3.5 presents the changes applied to the previous work to estimate the expected duration of the task.

3.2 Variant definition

To recognize a variant, we have decided to divide the operation into different segments. The distinction between the various segments consists in the recognition of a feature. Let us define a set of feature F. The main requirements of $f_i \in F$ are that:

- it is known a priori;
- it is possible to detect its occurrence online;
- it must be such that the segment of the operation between certain features is always executed in the same way (i.e. there are no variants of the segment).

The first requirement is due to the fact that, by knowing the features, the algorithm is able to recognize them. The second one is straightforward, since the operation has to be segmented online during the monitoring of the operation itself. The last one is fundamental for the definition of coherent reference sequences. In fact, each template segment will be used to monitor the equivalent section of the following. If the third condition holds, it is possible to apply the algorithm presented in Chapter 2 at the operation segment level. In our work we have decided to use as features the positions of some elements necessary for an assembly operation and they will be better described in Chapter 4 . In particular, we have considered some intermediate workstations and the final one. The latter is used to distinguish the end of the task execution.
Let us define as V the set of all the known variants. A variant $V_x \in V$ is then defined as the ordered set of features that occur during the operation from the starting position to the final one (for example $V_x = (f_i, f_j, f_k)$ where f_k represents the final position).

3.3 Building the template

As a consequence of the definition of variants, the template is no more a single sequence, but a set of different sequences each one representing a segment of the operation. With this expedient, it is possible to compare the input sequence (i.e. the one associated with the operation that the human is doing) with all the possible variants that the human has already done at least once. As a result, the template can now be represented as

a tree where each node contains different data useful for the recognition and updating of the individual segment. Let $\mathcal{T} = (N, A)$ be the template tree where N is a set of nodes and A is the set of arcs. A node $n_i \in N$ is defined as a 5-tuple $n_i = (o_i, d_i, p_i, C_i, e_i)$ where:

o_i is the origin feature

d_i is the destination feature

p_i is the parent

C_i is a set of children nodes

e_i is the number of times the segment has been executed

Origin and destination respectively represent from which feature the segment started and where it finished. The parent must be a node n_j that has as destination the origin of the actual node (i.e. $d_j = o_i$) while children contained in C_i are the ones that have as parent node n_i. Arcs in A connects parents to children, that is $a_{ij} \in A$ if and only if n_i is the parent of node n_j. The number of executions stores how many time that segment has been executed, this will be used to evaluate the probability of taking a variant rather than another (it is better explained in Section 3.4). A variant $V_x = (f_0, ...f_i, f_{i+1}, ...f_N)$, where N is the number of considered features, can now also be defined as an ordered set of nodes $V_x = (n_0, ...n_j, ...n_{N-1})$ where $n_j : o_j = f_i$, $d_j = f_{i+1}$.
Defining as T the set of all the known reference segments, $t_{ij} \in T$ contains data about skeletal points Cartesian positions collected during the execution of the segment that goes from f_i to f_j, that is $t_{ij} = (wl_{ij}, wr_{ij}, fl_{ij}, fr_{ij})$ where $wl_{ij}, wr_{ij}, fl_{ij}, fr_{ij}$ respectively are the temporal sequences that stores data about the left wrist, right wrist, left finger, right finger of the segment that goes from f_i to f_j and they all have the same length $|t_{ij}|$. This latter information is used to update the template (how it is done is explained in subsection 3.3.1). It is possible that in a more complex tree

different branches contain the same segment, so that there exists at least two nodes n_i and n_j, with $i \neq j$, where $o_i = o_j$ and $d_i = d_j$ (i.e. with same origin and destination) and they are all related to the template sequence denoted as t_{ij}.
Let us consider a simple example, where three different features denoted as f_1, f_2, f_3 are used. The first time that the human executes the task, he visits the workstations in numerical order. The graph depicted in figure 3.1 shows the resulting tree. There is a root with a single branch, this one is made of as many nodes as the segments of the task (so three in our case). Each node contains the data related to the part of the task they represent. In particular, the node n_1 has origin $o_1 = 0$ and destination $d_1 = 1$, its parent is the root 0 and its child is node n_2, then the segment associated to n_1 is monitored with data contained in t_{01}. In the same way, the node n_2 has origin $o_2 = 1$ and destination $d_2 = 2$, its parent is node n_1 and its child is leaf 3. The segment associated to n_2 is monitored with data contained in t_{12}. Lastly, node n_3 has origin $o_3 = 2$ and destination $d_3 = 3$, its parent is node n_2 and it has no children, so it is a leaf of the tree. All these nodes have been executed once so they all have one as number of execution. So at this point we have just one variant V_1 of the task, where $V_1 = (n_0, n_1, n_2, n_3)$.

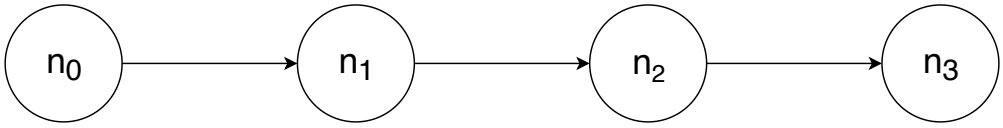


Figure 3.1: Example of template tree: node 0 is the root, while node n_1 contains the data related to the segment 0-1, node n_2 the ones related to 1-2 and leaf 3 the data regarding the part 2-3

It is possible that by mistake, or because the task allows to do so, the operator decides to execute the task visiting f_2 before f_1. As a consequence, a new branch of the tree is created (as depicted in figure 3.2). This new branch has respectively a node n_4 related to $f_0 \rightarrow f_2$, a node n_5 with data

about the part $f_2 \rightarrow f_1$ and a leaf (n_6) related to segment $f_1 \rightarrow f_3$. So we have another variant of the task $V_2 = (n_0, n_2, n_1, n_3)$ In this way, even if the human repeats variant V_2 another time the algorithm will be able to compare it with the correct set of sequences.

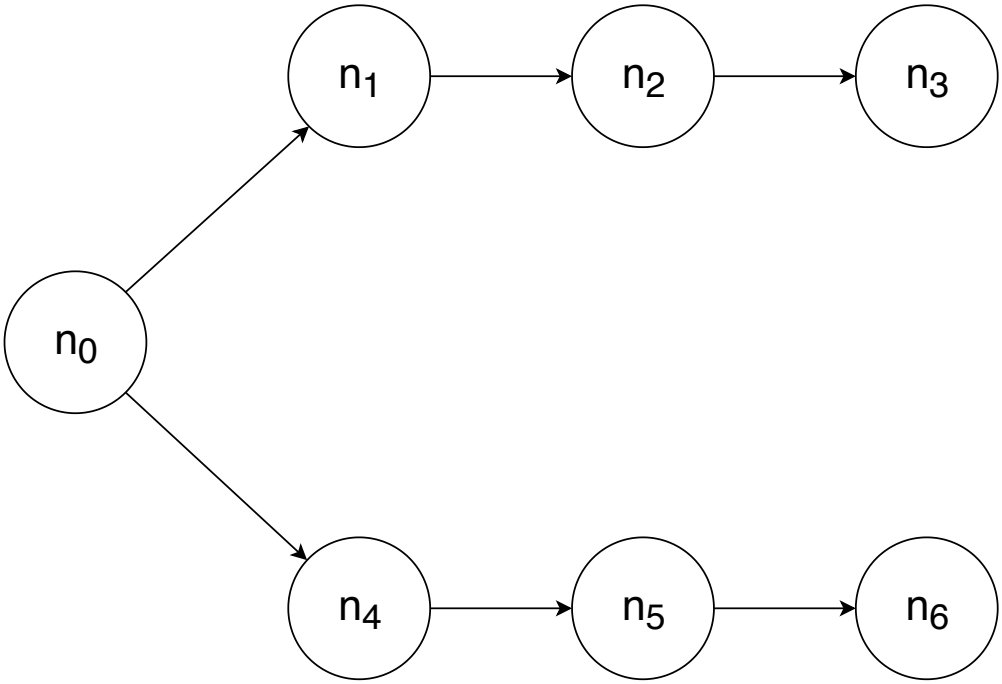


Figure 3.2: Example of template tree with a fork: if the human executes a variant of the task, the template tree is enriched with information about the new variant

3.3.1 Template update

As already discussed in Section 2.3.4 in the previous work, when the final position is reached, a check on the duration of the monitored operation and the reference sequence is performed and the shorter one is taken as template. So data regarding the movements of the skeletal points of the shortest execution are taken as reference sequences. Since the template is now divided into different segments it is possible to update the reference sequences in two moments:

- at the end of each segment
- at the end of the task.

In the first case, when a feature is reached, it is checked if the last instance of the segment lasted less than the reference one. If it is so the reference sequence regarding that segment is updated. When the template tree becomes complex and there are different segments for each branch, this method could be time consuming. As a consequence, the algorithm could miss some of the movements of the operator and it could give a wrong estimate of the next segment. So we have decided to update each sequence at the end of the task execution. When the final position is reached each segment of the last execution of the task is compared with the associated reference sequence. Let us define as $t'_{ij} = (wl'_{ij}, wr'_{ij}, fl'_{ij}, fr'_{ij})$ the data related to the segment just monitored and as $t_{ij} = (wl_{ij}, wr_{ij}, fl_{ij}, fr_{ij})$ the corresponding reference segment in the template set T. If $|t'_{ij}| < |t_{ij}|$ the reference sequence is updated as $t_{ij} = (wl'_{ij}, wr'_{ij}, fl'_{ij}, fr'_{ij})$.

3.4 Recognition of variant

The easiest way to detect a variant is to check if the last reached feature f_i is part of an already existing variant. If the monitored segment already exists in the template tree the algorithm continues the monitoring of the ongoing operation and its advancement and duration evaluation while, if it is not, a new branch of the tree must be created and the algorithm returns as expected duration the average of the past duration of the task. As a consequence, the expected duration of the segment is wrong, due to the fact that segment i is compared with a different segment for all its duration.
In order to have a better estimate of the duration of the segment, we have decided to perform an early recognition of a new variant. The main intuition of the proposed method is the computation of probabilities at

each time instant. Since at each time sample the ongoing operation is compared with the segments in the template tree that have the same origin feature, probabilities are used to decide which among them is the most likely to be the one that the human is performing. Moreover if all of them have a low probability to be the ongoing segment, then the algorithm assumes that the human is performing a new variant.
The number of execution e_i of each segment i is used to evaluate the historical probability P_h of that segment. Let e_i be the number of executions of node i and let us denote as p the parent of node n_i, the historical probability of segment i is evaluated as:

$$P_{h_i} = \frac{e_i}{e_p} \tag{3.1}$$

while the probability of the variant V_x is calculated as:

$$P_{V_x} = \prod_{n_i \in V_x} P_{h_i} \tag{3.2}$$

3.4.1 Updating current probability of the segment

The historical probability is used as a starting point in the evaluation of the current probability. In fact, while monitoring the human activity, a second probability, defined as P_{curr_i} is updated considering the similarity between the ongoing segment and the segment i considered as reference. To evaluate the current probability also the DTW matrices' costs are taken into account since they are the indices of the similarity of the input sequence with respect to the reference one. Since we have two DTW matrices for each segment (one for the right side and one for the left one) we have decided to consider a unique cost $\widetilde{D}_{OE_i}$ for each segment i. It is evaluated as:

$$\widetilde{D}_{OE_i}(k) = min\{D^{sx}_{OE_i}(k), D^{dx}_{OE_i}(k)\} \tag{3.3}$$

where $D^{sx}_{OE_i}(k)$ and $D^{dx}_{OE_i}(k)$ respectively are the costs of DTW matrices corresponding to the best alignment index of the left and right side at instant k, evaluated as in equation 2.6. Furthermore, in order to take into

account the average cost trend instead of the instantaneous evaluation, we have decided to consider a normalized cost D_{N_i} equal to:

$$D_{N_i}(k) = \frac{D'_{N_i}(k) + D''_{N_i}(k)}{2} \tag{3.4}$$

where D'_{N_i} and D''_{N_i} are two costs that differently affects the final cost D_{N_i} and consecutively P_{curr_i}. In particular the first one is calculated as:

$$D'_{N_i} = \widetilde{D}_{OE_i}(k) - \widetilde{D}_{OE_i}(k-1) \tag{3.5}$$

and it detects a change in the movements performed by the operator very fast, this means that if the ongoing segment is a new variant, the costs evaluated with this method increase as soon as possible. As a drawback, the current probability obtained using only this method is very chattering and this would cause a bad estimation of the final duration. On the other hand, the second one is calculated as:

$$D''_{N_i} = \frac{\widetilde{D}_{OE_i}(k)}{k} \tag{3.6}$$

and it is a normalized cost that produces smooth trends of the probabilities and a less chattering expected duration. Unfortunately, it is very slow in the detection of differences in the human's movements, so it could not recognize a new variant if it is very similar to another one present in the template tree. In order to have a less chattering probability and a quite fast method we have decide to combine these two giving the same weight to both.

Now that a unique cost has been defined, it is possible to estimate the current probability of each segment. In particular, we have considered the recursive bayesan classifier (similar to the one presented in [13]), so that the current probabilities of each segment starting from the last reached node are calculated as:

$$P_{curr_i}(k) \propto P_{curr_i}(k-1) \cdot f(D_{N_i}(k)) \tag{3.7}$$

where f is a Gaussian distribution with mean $\mu = 0$ and standard deviation $\sigma = 0.27$. This latter is a design parameter whose value has been

determined as the one that minimized the classification error during test experiments. That is, it is the value that leads to recognize as soon as possible and correctly the segment that corresponds to the one performed by the human. The starting condition is that $P_{curr_i}(0) = P_{h_i}$, then the current probability of segment i is updated as follows:

$$P_{curr_i}(k) = \frac{1}{\sigma\sqrt{2\pi}} \exp^{-\frac{D_{N_i}(k)^2}{2\sigma^2}} \cdot P_{curr_i}(k-1) \tag{3.8}$$

Since the recursive law could return a probability greater than 1, a normalization is performed if the sum of the current probabilities of the variants existing at instant k is greater than 1. It is important to say that the normalization is performed only if the latter condition applies because, otherwise, $1 - \sum_{i:n_i \in V_x} P_{curr_i}$ is the probability that the operator is performing an unknown segment. If at some point the costs become high, thus showing that the current operation is far from those in the template, then the current probabilities of the segments in the template tree are lowered. If their sum is below a certain threshold then the human is executing a new variant of the task, since it is highly improbable that he is doing one of the others present in the template. So the algorithm stops comparing the two sequences and just records input data to store them as new nodes of the template tree. In the meanwhile, the algorithm returns as expected duration the average of the past execution duration.

Example

Going back to the example shown in figure 3.1 when we only have a branch (i.e. variant V_1) as template, we have that $P_{h_i} = 1 \ \forall i \in 1, 2, 3$. Let us assume that the next time that the human executes the task he will visit f_2 before f_1. While executing the segment $f_0 \rightarrow f_2$, the algorithm will compare it with data of the segment $f_0 \rightarrow f_1$, since it is the only one known at this time. It is straightforward that since the operation is different the costs of DTW matrices increase causing a decreasing trend in P_{curr_1}. When the current probability of that variant goes below a certain

limit it becomes evident that the operator is performing a new variant V_2. So we have that at the end of the second execution the template tree has two branches ($V_1 = (n_0, n_1, n_2, n_3)$ and $V_2 = (n_0, n_2, n_1, n_3)$). When the human starts a new execution of the operation we will have that the probability of the two variants will be equal to 50% for both, since the human has performed variant V_1 the first time and V_2 the other time.

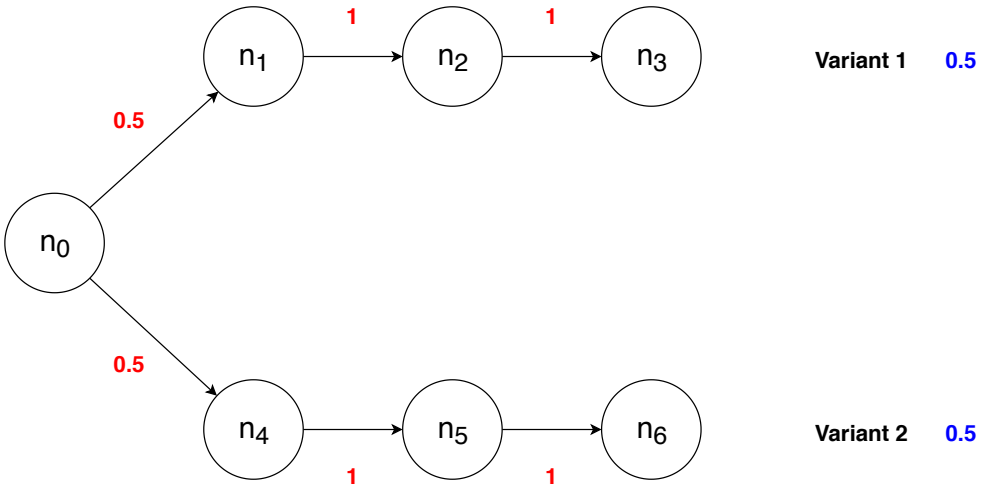


Figure 3.3: Initial probabilities: each segment has its historical probability (the ones represented in red) and the probability of each variant is evaluated as the product of the probabilities of the segments that compose it (the ones in blue)

3.5 Advancement and duration estimate

Since the template sequence is now divided into many segments, we have decided to consider two types of advancement estimate:

- with respect to the segment
- with respect to the whole operation.

In the first one the advancement is estimated as in 2.7, the only difference lays in the fact that the reference sequence is just a part of the operation. The second one is more complex to evaluate, as it has to consider all the possible variants present in the template tree. Let $\bar{n}$ be the node that represent the last completed segment. Let $\bar{V} = \{V_i \in V | \bar{n} \in V_i\}$ be the set of variants that contain node $\bar{n}$ and let $V_i = (V^{pre}, V_i^{post})$: $V^{pre} = (n_0, ...\bar{n})$ and $V_i^{post} = (ci, ...n_N)$ (where n_N is a leaf) be a partition for variants in $\bar{V}$. The DTW algorithm is applied for each children $c_i \in \bar{C}$. Let $L_{prec_i} = \sum_{n_i \in V^{pre}} |t_{jk}|$ be the length of the segments already executed, then the advancement of the variant V_i at instant k is calculated as:

$$adv_{V_i}(k)\% = 100 \cdot \frac{j_N^*(k) + L_{prec_i}}{L_{prec_i} + L_{next_i}} \tag{3.9}$$

where L_{next_i} is an estimate of the length of the remaining part of the operation that is evaluated as the convex combination of the average lengths for all variants that starts from $\bar{n}$, weighted over the historical probabilities of those variants. In particular, L_{next_i} is evaluated as:

$$L_{next_i} = |t_{\bar{o}\bar{d}}| + \sum_{i:V_i^{post} \in \bar{V}} (L_i^V \cdot Prob_i^V) \tag{3.10}$$

where L_i^V and $Prob_i^V$ are evaluated as:

$$\begin{aligned} L_i^V &= \sum_{k:n_k \in V_i^{post}} |t_{o_k,d_k}| \\ P_i^V &= \prod_{k:n_k \in V_i^{post}} P_{h_k} \end{aligned} \tag{3.11}$$

This estimation of the length of the remaining part of the operation minimizes the error between the actual length and the estimated one. Let us define the mean error of the length of the remaining part of the operation considering sub-variant V_i as $e_{m_i} = (L^* - L_i)^2$, the total mean error considering all variants is $e_m = \sum_i p_i (L^* - L_i)^2$. So to minimize it we impose that:

$$\frac{de_m}{dL^*} = 2 \cdot \sum_i p_i (L^* - L_i) = 0 \tag{3.12}$$

and we obtain that $\sum_i p_i \cdot L^* = \sum_i p_i L_i$. So the estimate that minimizes the error is $L^* = \sum_i p_i L_i$ since $\sum_i p_i = 1$.
Let us recall the example of figure 3.3 and consider that the third time the operator executes the operation he repeats the variant V_1 while the fourth time an error variant is executed. The resulting tree is shown in figure 3.4 and shows probabilities and lengths of each segment. When the fifth execution is started the first part of the operation is compared with segments $f_0 \to f_1$ and $f_0 \to f_2$ (i.e. segments $f_0 \to f_1$ and $f_1 \to f_2$). As a consequence, the advancement will be calculated twice: one considers V_1 as the current variant, the other has to consider the convex combination of the other two branches (V_{2_1} and V_{2_2}). Let us call $adv_{V_1}\%$ the advancement estimate with respect to branch 1 and $adv_{V_2}\%$ the advancement estimate that considers the other two branches, they are evaluated as:

$$\begin{aligned} adv_{V_1}\% &= 100 \cdot \frac{j_N^*}{|t_1|+|t_2|+|t_3|} \\ adv_{V_2}\% &= 100 \cdot \frac{j_N^*}{(|t_4|+(|t_5|+|t_6|)\cdot(P_{h_5}\cdot P_{h_6})+(|t_7|+|t_8|)\cdot(P_{h_7}\cdot P_{h_8})} \end{aligned} \tag{3.13}$$

where $|t_x|$ represents the size of the data set contained in node n_x and P_{h_i} is the historical probability of segment i, according to the notations used in 3.3.

In the same way as for the advancement, we have decided to separately evaluate the expected duration of the single segment and the one of the whole operation. Let $Padv_i\%(k)$ be the percentage of the advancement of segment i at time k, the duration $\hat{T}_{partial_i}(k)$ of this latter is estimated as follows:

$$\hat{T}_{partial_i}(k) = \frac{T_e(k)}{Padv_i\%(k)} \tag{3.14}$$

On the other hand, the estimate of the total duration of the operation is estimated considering the presence of different variants in the template tree. Let $\bar{n}$ be the last node reached and $c_i \in \bar{C}$ be one of its child, for each variant $V_i \in \bar{V}$, an expected duration is calculated as:

$$\widehat{T}_{V_i}(k) = \frac{T_e(k)}{adv_{V_i}(k)\%} \tag{3.15}$$

that is the rule described in 2.8 updated considering the percentage of advancement of each variant V_i containing the current node $\bar{n}$. In order to return a unique duration estimate $\hat{T}(k)$ at any instant k, we have decided to collect the expected duration of each segment and multiply them by their current probabilities. In this way, the most probable variant will have more weight on the expected duration, while the less probable will not influence it as much.

$$\hat{T}(k) = \sum_{n_i \in C_i} \widehat{T}_{V_i}(k) \cdot P_{curr_j}(k) \tag{3.16}$$

where $n_i \in \bar{C}$ are the children of the current node $\bar{n}$ and their number corresponds to the number of variants containing $\bar{n}$ which are known at that point of the operation. Besides, $P_{curr_j}(k)$ is the current probability of the reference segment t_{ij} that is part of variant V_i. In order to take into account the possibility that the human is executing an unknown segment the rule in 3.16 is updated as follows:

$$\hat{T}(k) = \sum_{n_i \in C_i} \widehat{T}_{V_i^{post}}(k) \cdot P_{curr_j}(k) + (1 - \sum_{n_i \in C_i} P_{curr_j}(k) \cdot \mathbb{E}[\Delta T]) \tag{3.17}$$

where $\mathbb{E}[\Delta T]$ is the average of the past duration of the task.
With the implemented changes, it is now possible to correctly estimate the duration of the total operation even if the operator executes one of the known variant in the task. In the following Chapter, the results obtained applying the algorithm to a real assembly task will be shown.

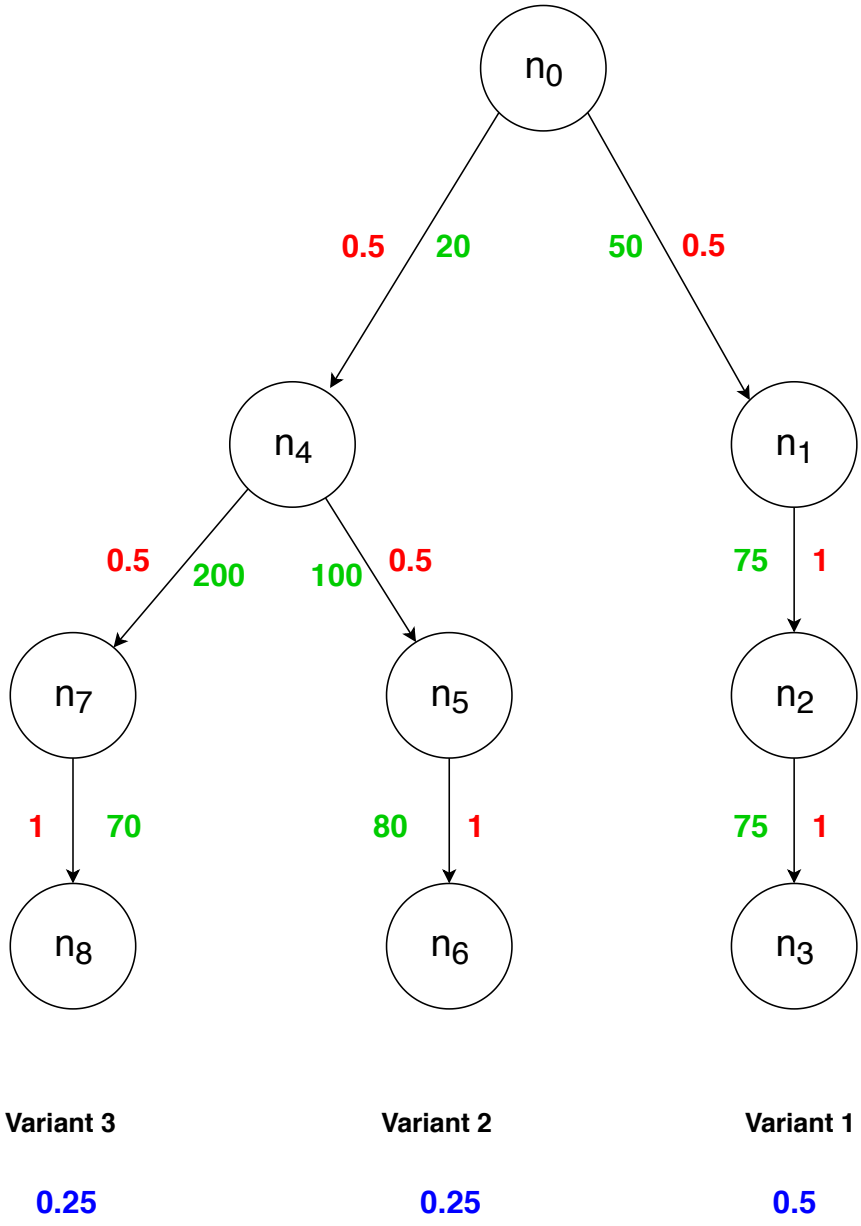


Figure 3.4: Example of tree with forks: n_0 and n_4 have two children each, this leads to a tree that represent three possible variants. In order to estimate the advancement of the ongoing operation the length of every segment of each variant has to be considered. The length of the segments are represented in green, the historical probabilities of the segments are represented in red and the probability of each variant is depicted in blue. The percentage of advancement of each variant are calculated as:
$adv_1(k)\% = 100 \cdot \frac{j^*_N(k)}{50+75+75}$ and
$adv_2(k)\% = 100 \cdot \frac{j^*_N(k)}{(20+[(100+80)\cdot 0.5+(200+70)\cdot 0.5]}$

Chapter 4

Experimental Validation

After discussed in the previous Chapter all the changes applied to the algorithm described in Chapter 2, the experimental results obtained with the new method are presented in this Chapter. In particular, Section 4.1 briefly explains how data were recorded. Section 4.2 describes the experimental set up and gives a description of the segments of the operation. Finally, Section 4.3 is divided in two parts: subsection 4.3.1 illustrates that the algorithm correctly builds the template tree and that it is able to recognize a new variant of the operation. While subsection 4.3.2 compares the performances of our work with the results obtained with other algorithms.

4.1 Monitoring system

As noted in subsection 2.3.2 we have decided to use as input signal of the algorithm the human's skeletal points Cartesian positions. In order to do so we have used a Kinect v2 sensor, but the algorithm presented in this thesis works regardless of the sensor used. The sensor is able to recognize the wrists and the hands, but in assembly operations this is not sufficient, due to the fact that in some part of the operation wrists and hands may not be informative as they are stationary. Another issue arise due to the

fact that the data about the hands obtained through the Kinect are often unstable and inaccurate. In order to find the fingers and acquire data also from them, the human is provided with gloves with a colored identifier on the index finger of each hand. It has been decided to consider the index fingers because they usually are the most informative ones [1].
In order to detect if the human visits the features in real-time, we have used the positions $P_{f_i} = \{P_{x_{f_i}}, P_{y_{f_i}}, P_{z_{f_i}}\}$ of these latter and considered a sphere around it. Considering that $P'_{f_i} = \{P'_{x_{f_i}}, P'_{y_{f_i}}, P'_{z_{f_i}}\}$ is the current position of human's finger, if $\sqrt{(P'_{x_{f_i}} - P_{x_{f_i}})^2 + (P'_{y_{f_i}} - P_{y_{f_i}})^2 + (P'_{z_{f_i}} - P_{z_{f_i}})^2} < \delta$ (where δ is the sphere radius), then the human is visiting feature f_i. This means that if one of the human's hand enters the sphere he is visiting the corresponding feature.

4.2 Experimental set up

The experiment used for testing involves the assembly of two wheels on a pre-arranged base (i.e. the holes where to insert the wheels are not created within the experiment). Figure 4.1a shows the main parts that compose the final product (Figure 4.1b). The work table includes an input zone, an output zone and several boxes containing the pieces necessary for the operation. Figure 4.2 shows the arrangement of the various elements used for the experiments. In particular zone IN and OUT respectively are the input zone (from where the human takes the base before the assemblying operation) and the output zone (where the human puts the final product).

[1] See [6] for further details.

(a) Main body of the wheel structure

(b) Big screw

(c) Wheel

(d) Small screw

(e) Base

(f) Nut

(a) Main components of the product

(b) Final product

Figure 4.1: Product assembled during the experiment

The buffers respectively contain:

1. wheels that will be mounted on the left side of the table
2. big screws used to fix the wheels to the table
3. a zone where to put defective elements
4. wheels that will be mounted on the right side of the table
5. small screws and nuts used to fix the wheel to the main part.
6. the main bodies of the wheel structure.

Since we wanted to test the robustness of the algorithm we have considered a complex tree made of six variants. In order to do so, we defined 5 features:

f_0 corresponds to the starting position of the human

f_1 corresponds to buffer 1

f_2 is buffer 2

f_3 is buffer 3

f_4 corresponds to buffer 4

f_5 is the output zone

These 5 features allows to have many combinations and consequently a big set of segments. Here are summarized the ones we considered:

$f_0 \rightarrow f_1$: the human picks up the base from the input zone, puts it on the assembly zone, takes the main body from buffer 6, puts this last on the up left hole of the base and reaches out for the left wheel in buffer 1 (f_1).

$f_0 \rightarrow f_2$: the human picks up the base from the input zone, puts it on the assembly zone, takes the main body from buffer 6, puts this last on the up left hole of the base and reaches out for the big screw in buffer 2 (f_2).

$f_1 \rightarrow f_2$: the operator takes a small screw from buffer 5 and inserts it in the wheel on the up left hole of the base. Then he takes a nut from buffer 5 and fixes the wheel. The human takes the main body from buffer 6 and puts it on the down right hole of the base. After that, he reaches out for the big screw in buffer 2 (f_2).

$f_1 \rightarrow f_5$: the operator takes a small screw from buffer 5 and inserts it in the wheel on the up left hole of the base. Then he takes a nut from buffer 5 and fixes the wheel. After that, he places the final product in the output zone (f_5).

$f_2 \rightarrow f_1$: the big screw is fixed to the table in the up left hole with the use of a hex key. Then the operator picks a wheel from buffer 1(f_1).

$f_2 \rightarrow f_2$: the human takes a hex key and fixes with the big screw the main body to the table in the up left hole. Then he takes the main body from buffer 6 and puts it on the down right hole of the table. Then he picks another big screw from buffer 2(f_2).

$f_2 \rightarrow f_4$: the human picks a hex key and with it he fixes the main body to the base in the down right hole. Then he picks a wheel from buffer 4(f_4).

$f_2 \rightarrow f_5$: the big screw is fixed to the base in the up left hole then the operator places the finished part on the output buffer(f_5).

$f_3 \rightarrow f_1$: the operator takes a small screw from buffer 5 and insert it in the wheel on the down right hole of the base. Then he takes a nut from buffer 5 and fixes the wheel. After that, he picks a wheel from buffer 1 (f_1).

$f_3 \rightarrow f_2$: the operator takes a small screw from buffer 5 and insert it in the wheel on the down right hole of the table. Then he takes a nut from buffer 5 and fixes the wheel. After that, he picks a screw from buffer 2 (f_4).

$f_4 \rightarrow f_1$: the operator takes a small screw from buffer 5 and inserts it in the wheel on the down right hole of the base. Then he takes a nut from buffer 5 and fixes the wheel. After that, he takes the wheel from buffer 1 (f_1).

$f_4 \rightarrow f_2$: the operator takes a small screw from buffer 5 and inserts it in the wheel on the down right hole of the base. Then he takes a nut from buffer 5 and fixes the wheel. After that, he reaches out for the big screw in buffer 2 (f_2).

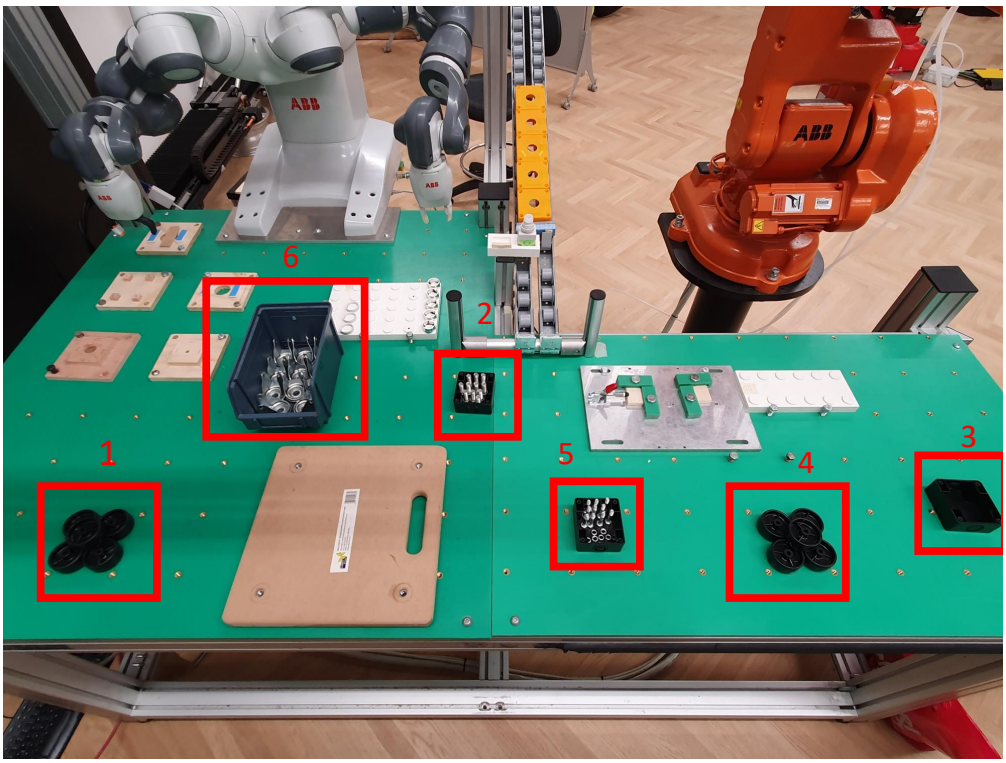
$f_4 \rightarrow f_3$: the operator takes a small screw from buffer 5, he realizes that the screw is defective and brings it to the waste buffer (f_3).

$f_4 \rightarrow f_5$: the operator takes a small screw and a nut from buffer 5, so he fixes the wheel on the down right side of the base and put the product in the output zone(f_5).

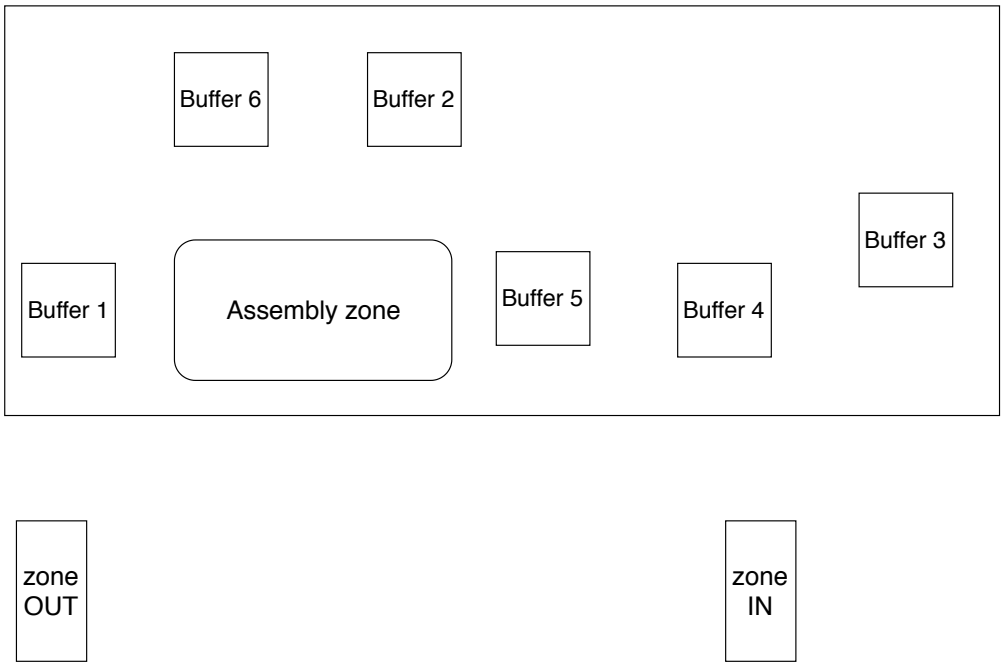
Figure 4.3 shows the tree we built during our experiments. It is made of six branches (i.e. six variants). The considered variants are composed as follows:

- $V_1 = (f_0, f_2, f_1, f_2, f_4, f_5)$
- $V_2 = (f_0, f_2, f_2, f_4, f_3, f_1, f_5)$
- $V_3 = (f_0, f_1, f_2, f_4, f_2, f_5)$
- $V_4 = (f_0, f_2, f_2, f_4, f_5)$
- $V_5 = (f_0, f_1, f_2, f_4, f_3, f_2, f_5)$
- $V_6 = (f_0, f_2, f_2, f_4, f_1, f_5)$

where V_2 and V_5 represent variants where a scrap screw was taken from the operator, so he stops the execution of the task to throw the screw in the waste zone and after that he continues the operation from where he stopped. V_4 corresponds to a variant where the human forgets to fix the wheel on the top right side of the table, so he delivers an incomplete product to the output zone. Finally V_1, V_3 and V_6 are the most significant variants where the operator correctly assemblies the product exchanging the order of features visited.



(a)



(b)

Figure 4.2: Experimental set up

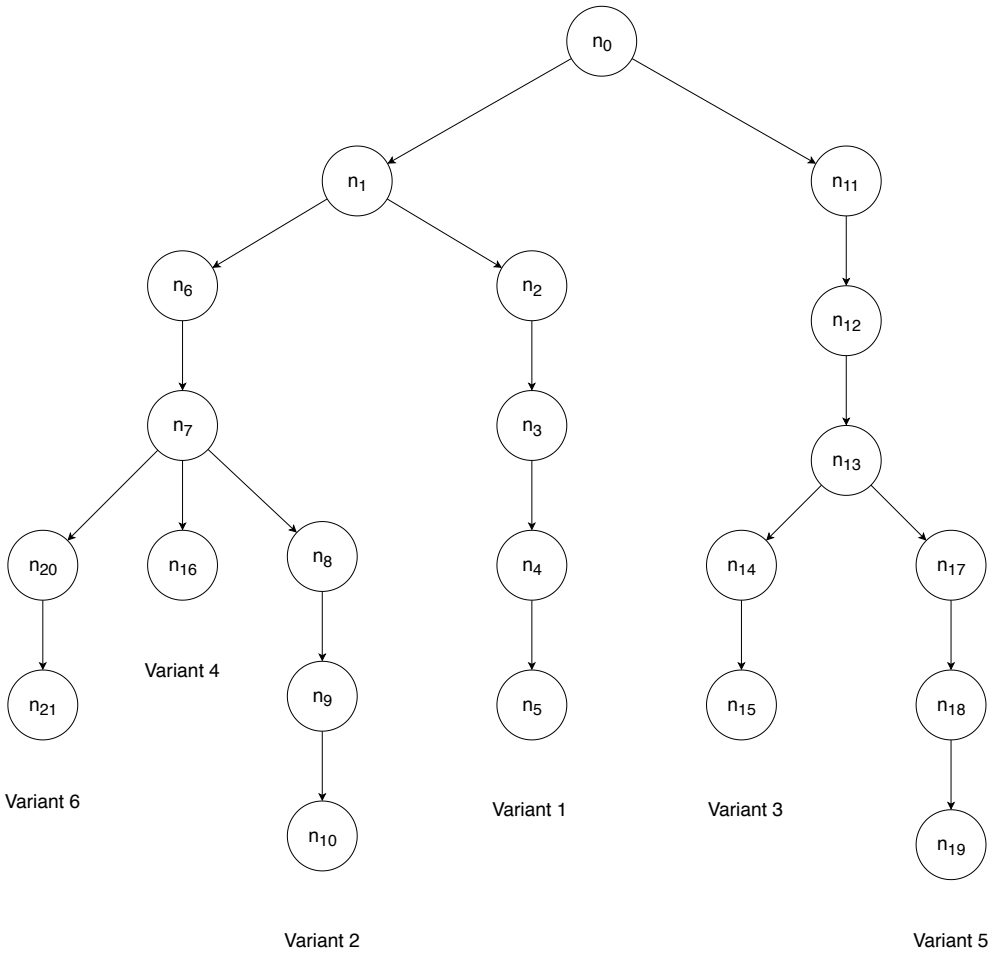


Figure 4.3: Tree built during the experiments. It consists of three plausible task variants ($V_1,\ V_3,\ V_6$) and three error variants ($V_2,\ V_4,\ V_5$)

A segment can be represented by different nodes, it depends on when that segment has been executed in the variant of the task. Table 4.1 summarizes which nodes are related to each segment.

$f_0 \rightarrow f_1$	n_{11}	$f_0 \rightarrow f_2$	n_1
$f_1 \rightarrow f_2$	n_3, n_{12}	$f_1 \rightarrow f_5$	n_{21}
$f_2 \rightarrow f_1$	n_2	$f_2 \rightarrow f_2$	n_6
$f_2 \rightarrow f_4$	n_4, n_7, n_{13}	$f_2 \rightarrow f_5$	n_{15}, n_{19}
$f_3 \rightarrow f_1$	n_9	$f_3 \rightarrow f_2$	n_{18}
$f_4 \rightarrow f_1$	n_{20}	$f_4 \rightarrow f_2$	n_{14}
$f_4 \rightarrow f_3$	n_8, n_{17}	$f_4 \rightarrow f_5$	n_5, n_{16}

Table 4.1: Nodes - segments relationships

Section 4.3.1 discusses the most interesting passages of the tree construction.

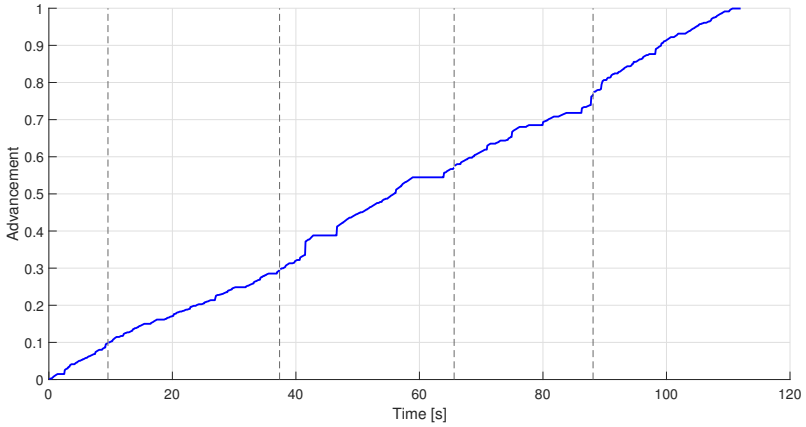
4.3 Obtained results

In this section the most important results obtained during the experiments are shown. In particular, in subsection 4.3.1 the online construction of the template is explained through its most interesting passages (i.e. the recognition of a new variant and the comparison between different segments). Then in subsection 4.3.2 the results obtained with our algorithm are compared with the ones obtained with other ones. In particular, we have considered the algorithm described in Section 2.1 and the one developed in [6]. Since our aim is to correctly estimate the duration of the ongoing operation, the errors between the expected duration and the actual one obtained with the different methods are compared.

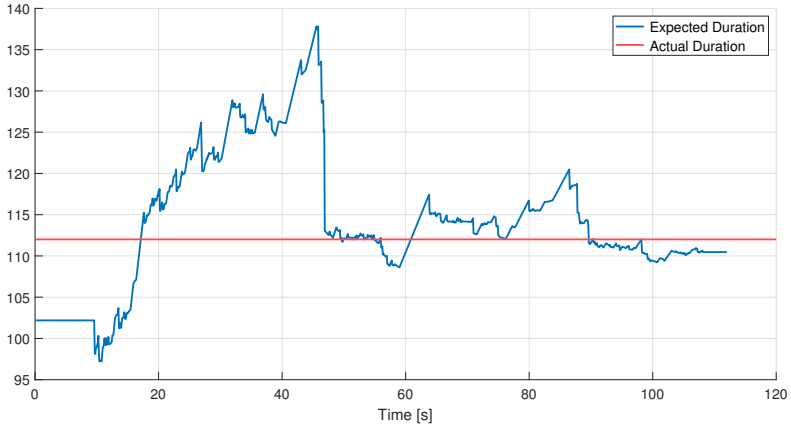
4.3.1 Building of the template tree

During the first execution, the operator assembles the product following the variant V_6. The operator picks the main body of the wheel structure, fixes it on the top left side of the base, then he picks another main body and puts it on the down right side of the base and he fixes the wheel on this position. After that, he takes another wheel to fix it on the top left side of the base. These set of sequences is used to build the first template branch. The second time the operator repeats the variant previously done. The graphs of advancement and expected duration are shown in figure 4.4. The results obtained are good considering that this was just the second execution of the variant. In particular the error between the expected duration (in blue) and the actual one (in red) in the last twenty seconds of the operation is around 1.5 s over a total duration of 112.1 s, which means an error of 1.3% (figure 4.4c).

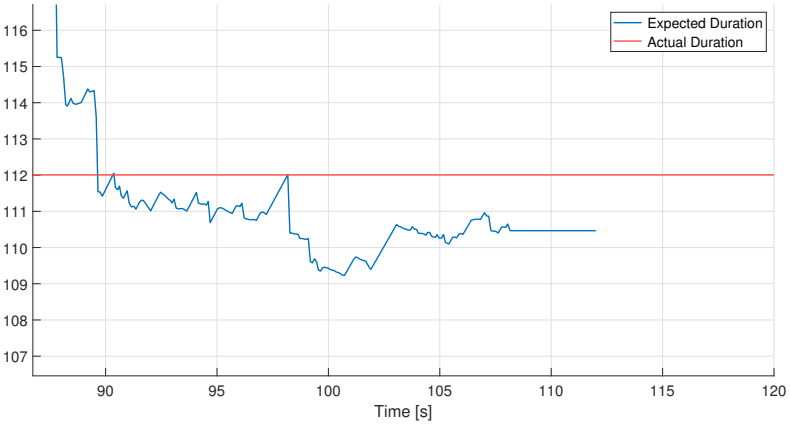
The third time that the human executes the operation he performs variant V_1. The difference between V_1 and V_6 lays in the segment after n_1 (figure 4.5). After having picked the main body from buffer 6 and a big screw from buffer 2 (f_2), in variant V_1 the operator fixes the main body to the base with the screw and then picks a wheel from buffer 1 (f_1). In variant V_6 he picks another main body from buffer 6 and then another big screw from buffer 2 (f_2). So the initial part of segments $f_2 \rightarrow f_2$ and $f_2 \rightarrow f_1$ are the same as the human fixes the main body to the table by inserting the screw. The difference lays in the final part and the proposed algorithm recognizes this difference. Figure 4.6 shows how the current probability behaves while monitoring segment $f_2 \rightarrow f_1$ when it is compared with segment $f_2 \rightarrow f_2$. It starts as 1.0 (i.e. the ongoing operation corresponds for sure to the reference one) and in the final part it rapidly corrects itself going to a very low percentage. The current probabilities are updated continuously in order to detect changes in the human movements and to promptly react to them in the estimation of the duration.



(a)



(b)



(c)

Figure 4.4: Advancement (a) and duration (b) estimate of the second execution of variant V_6 when there is only this latter as possible template. (c) shows the result of the last twenty seconds of the operation.

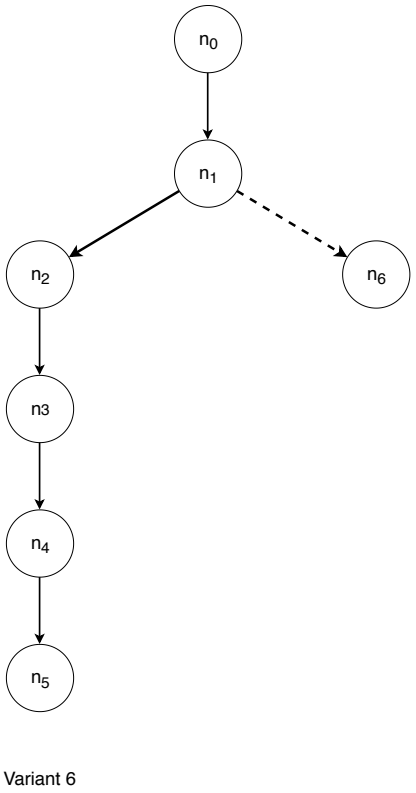


Figure 4.5: While monitoring the third execution of the operation, the template is composed of just one variant. The dashed line indicates that, after having completed the segment related to node n_2, the human is performing an unknown new segment.

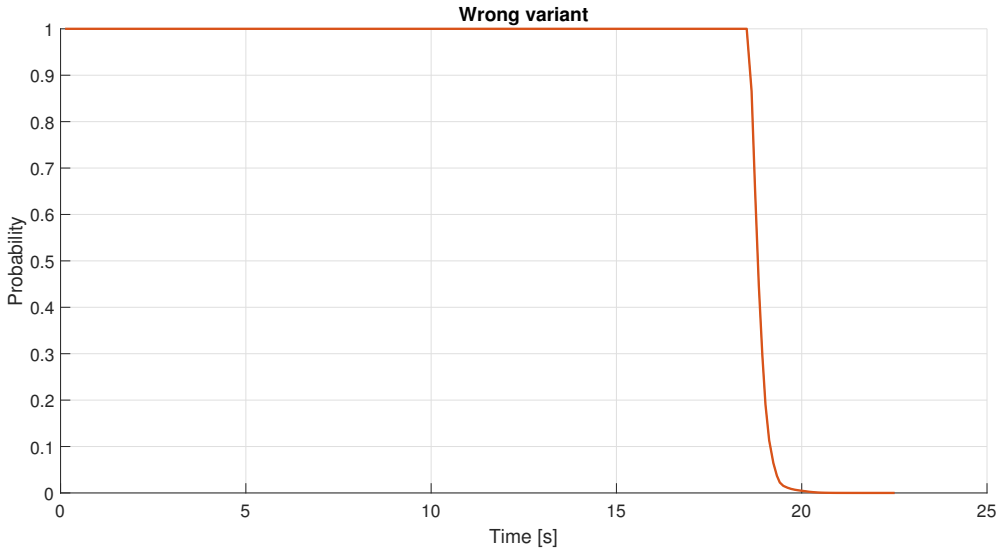


Figure 4.6: The current probability is corrected when changes in the human's movements are detected, in the meanwhile the template tree is enriched with data about the new segment

So now we have that the template tree is made of two variants (as in figure 4.7) and from now on the ongoing operations will be compared with both. We repeated variant V_6 and when monitoring segment $f_2 \rightarrow f_2$ the current probabilities of the two possible segments are very similar, but in the final part the algorithm is able to recognize the correct one (as it is shown in figure 4.8).

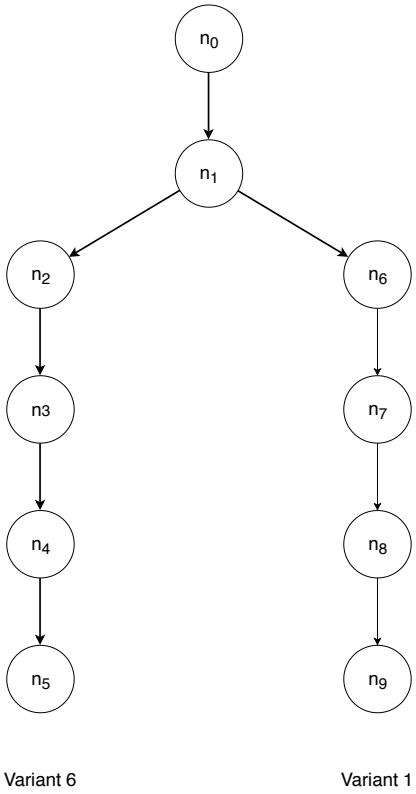


Figure 4.7: Template tree when variant V_1 and V_6 have been executed.

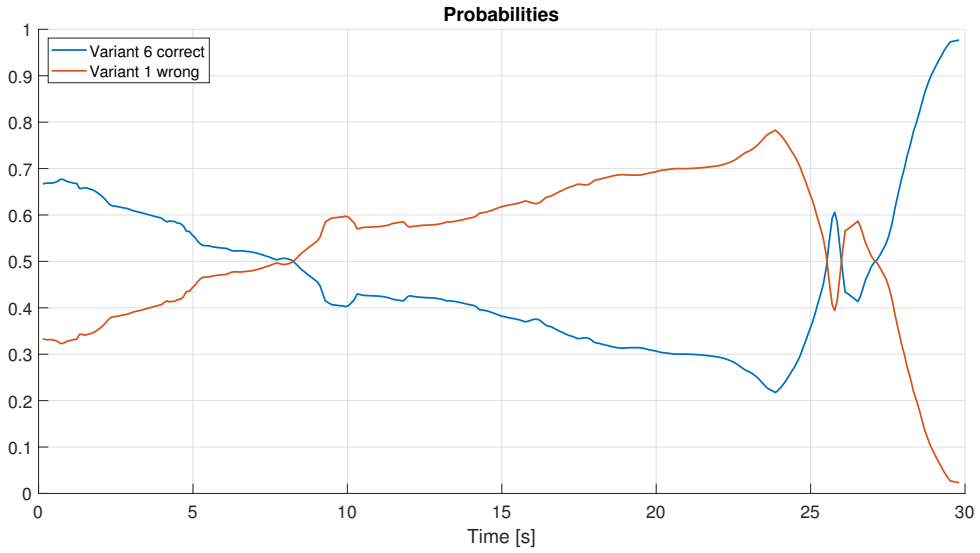


Figure 4.8: Considering the template tree shown in 4.7, after having reached node n_1. At each time instant, the algorithm compares the following sequence with the ones which data are related to nodes n_2 and n_3. The current probabilities of segment $f_2 \rightarrow f_2$ and $f_2 \rightarrow f_1$ are very similar. But in the end the algorithm is able to recognize the correct variant(the one in blue)

After that, the operator executes an error variant (V_4). Since the ongoing operation in shorter with respect to the reference ones (as the operator forgets to complete the product) the algorithm overestimates the duration of the operation. However, it is able to detect the new variant and consequently add it to the template tree. (figure 4.9).

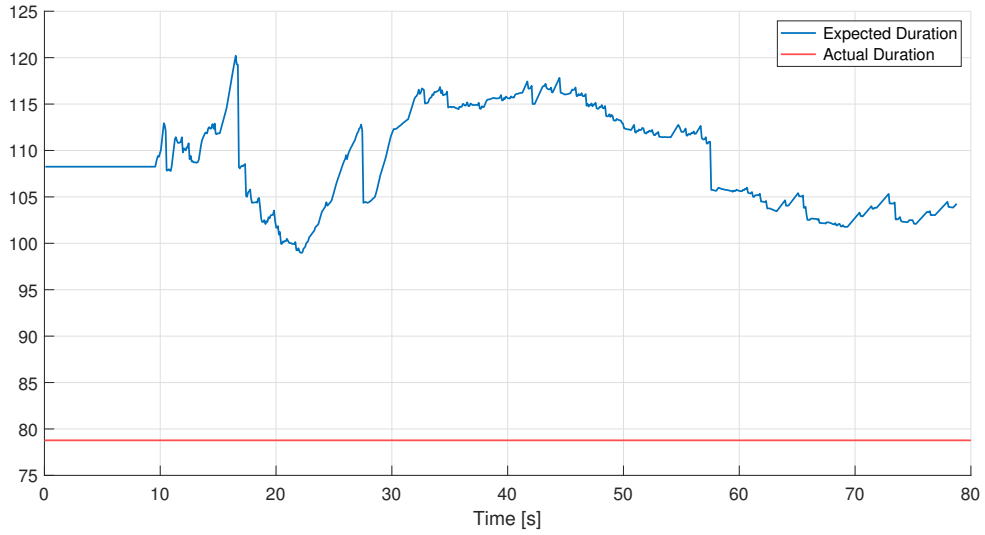


Figure 4.9: During the first monitoring of the error variant that corresponds to a shorter execution of the operation, the algorithm overestimates the duration as it compares the ongoing operation (that is just partial) with sequences of the complete operation

Let us now consider to have a template tree corresponding to the one in figure 4.10 where node n_3 has two children, n_4 and n_{10}. Variant V_2 was executed to verify that even if none of the two sequences ($f_4 \rightarrow f_5$ and $f_4 \rightarrow f_1$) corresponds to the ongoing segment ($f_4 \rightarrow f_3$), the algorithm is able to correct both current probabilities (as it is shown if figure 4.11).

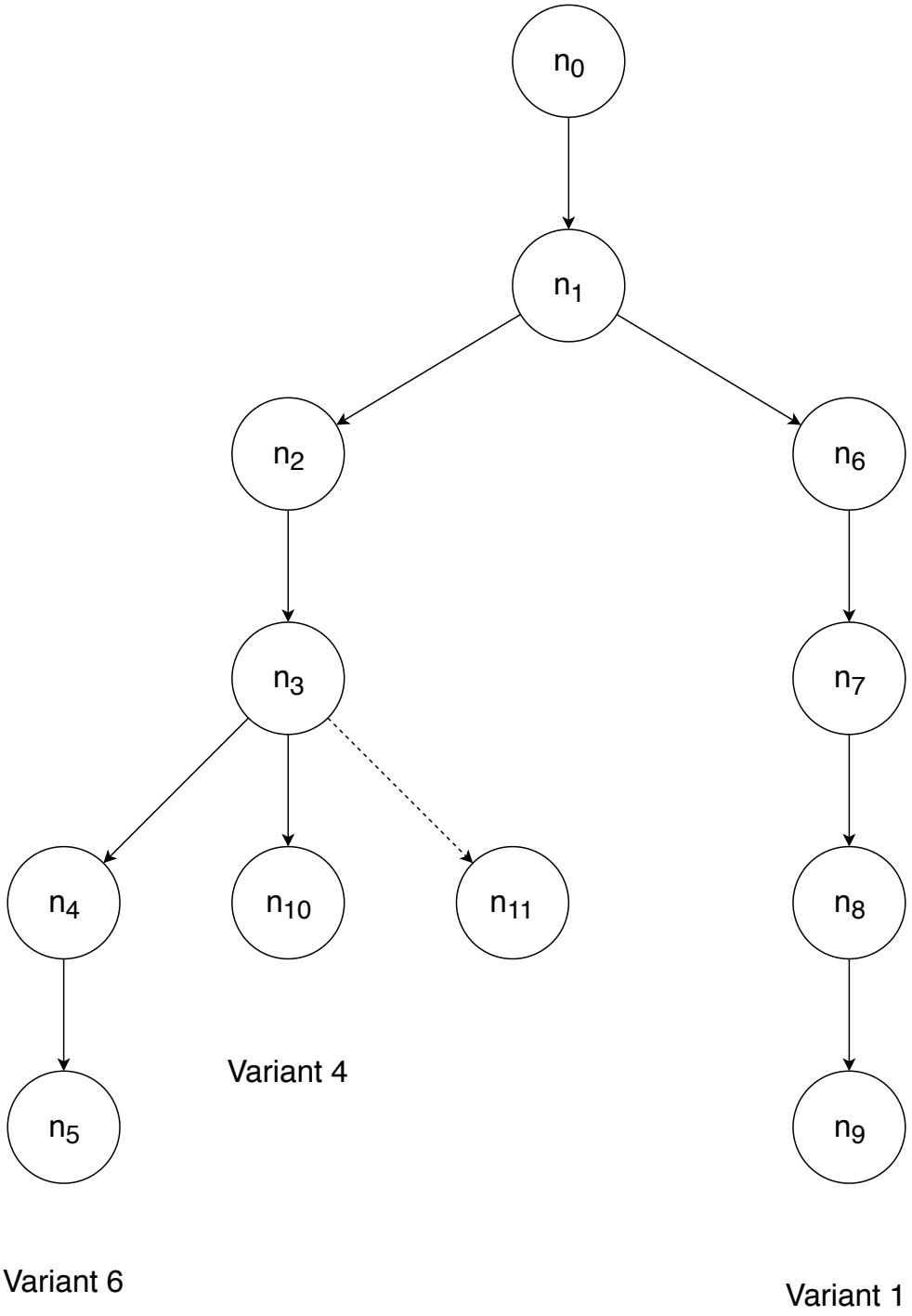


Figure 4.10: Template tree where n_3 has two children. After having reached node n_3 the algorithm compares the ongoing segment with the ones related to n_4 and n_10, but DTW's matrices costs become high at a certain point so the algorithm detects a new segment as children of node n_3.

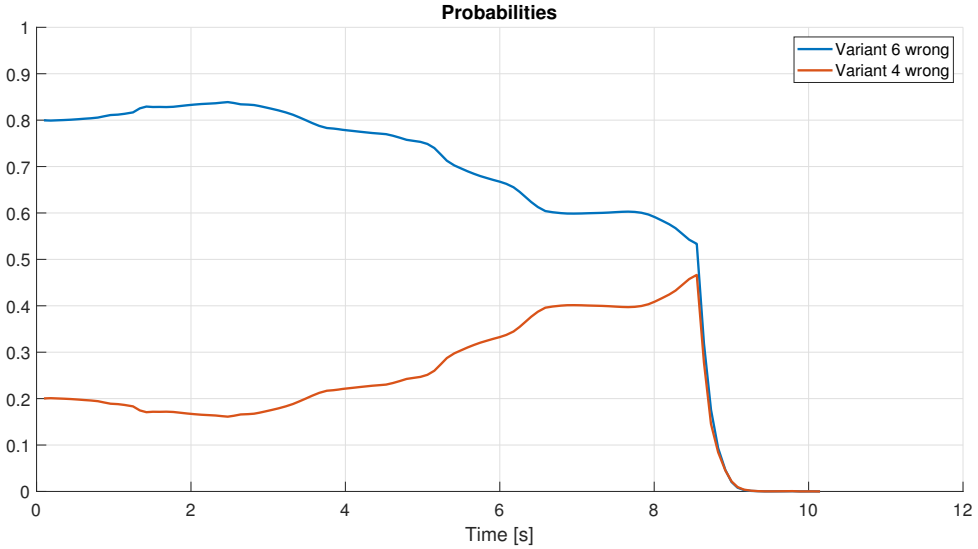


Figure 4.11: During the first monitoring of a third new segment starting from node n_3, the current probabilities of the two existing segments in the template rapidly goes to zero when the algorithm detects a change in the human's movements

After that, we have as template tree the one represented in figure 4.12, which is composed by 4 variants. In particular, the main difference between variant V_6, V_4 and V_5 starts from node n_7. After that node, one branch represents the end of the normal way to execute the task, one leads to the delivery of an incomplete product and the last one considers a sequence where a defective screw was find and thrown in the waste zone.

The trivium is the most interesting part of our tree as it let us understand whether the update of the current probabilities works well even when there are more than two segments. So we repeated the variant V_4 where the correct segment was $f_4 \rightarrow f_5$ and the behavior of the current probabilities is depicted in figure 4.13. In this latter it is visible that the algorithm is able to identify the correct segment even if it starts from an historical probability lower than the one of the others.

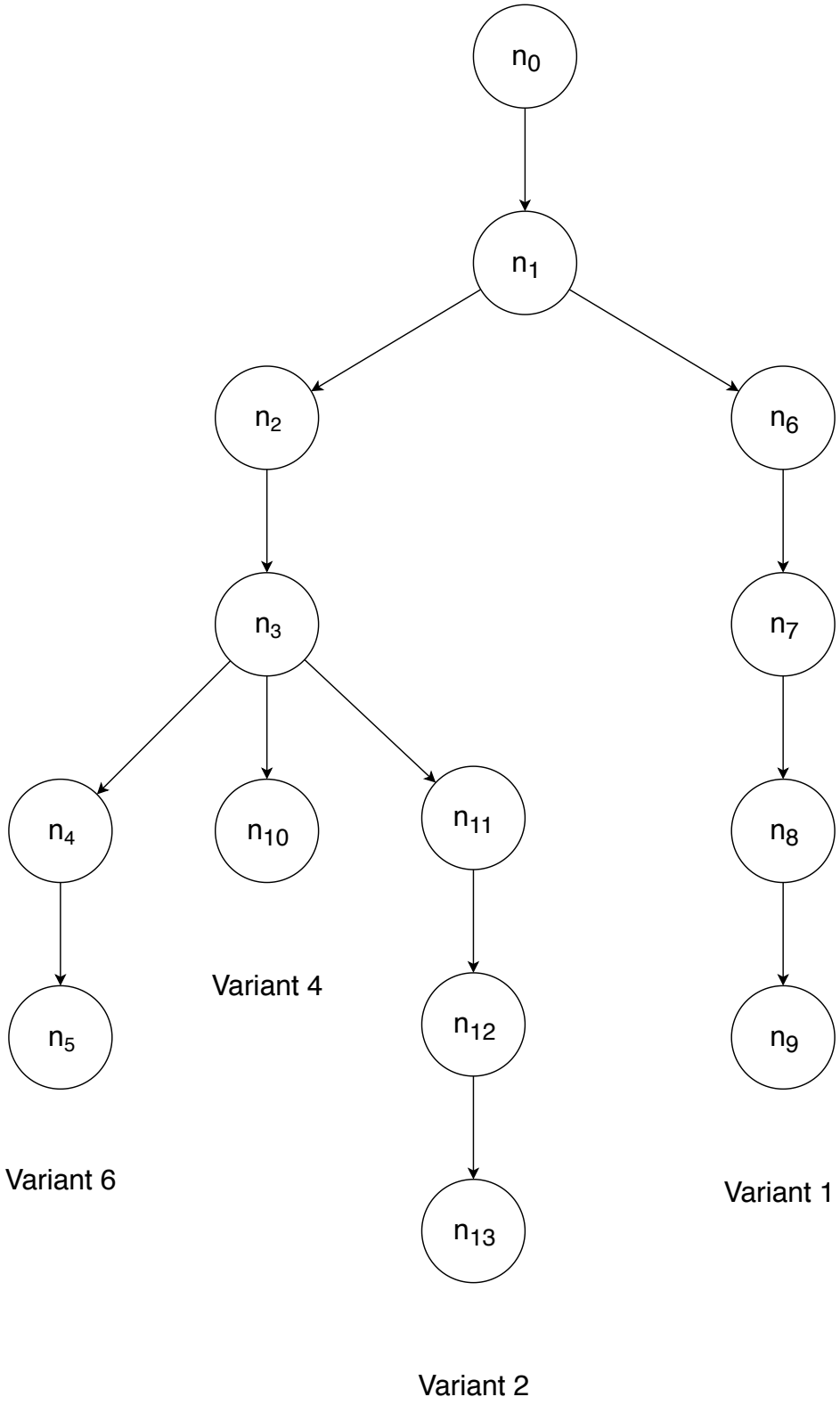


Figure 4.12: Now that the user has performed four variants of the operation task at least once, the template tree in made as shown.

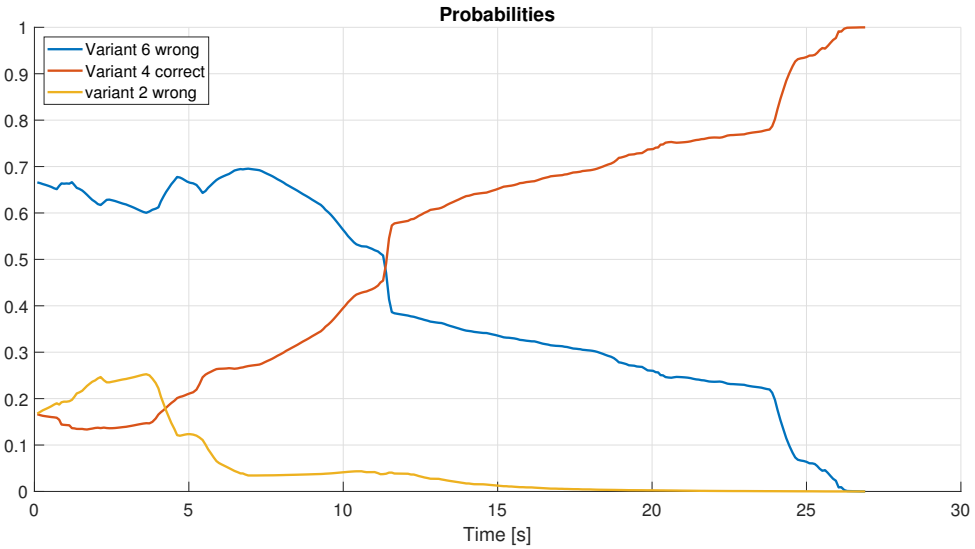


Figure 4.13: Current probabilities of the trivium when the second variant inserted in the template tree is the one being executed. The algorithm is able to identify the correct segment even if it starts from an historical probability lower than the one of the others.

After some instances, variant V_3 and V_5 were executed to complete the template tree. To verify that the algorithm works well with such a complex tree, we performed one last time variant V_6 that was the first one inserted in the template. The obtained results are shown in figure 4.14. It is apparent that although this variant includes a fork (segment $f_2 \rightarrow f_2$ compared also with $f_2 \rightarrow f_1$) and a trivium ($f_4 \rightarrow f_5$, $f_4 \rightarrow f_3$ and $f_4 \rightarrow f_1$) the estimated duration well behaves for almost the entire duration of the operation. In fact, after the initial part that is estimated with the mean of the past duration, the maximum error is around 5s over a total length of 89s.

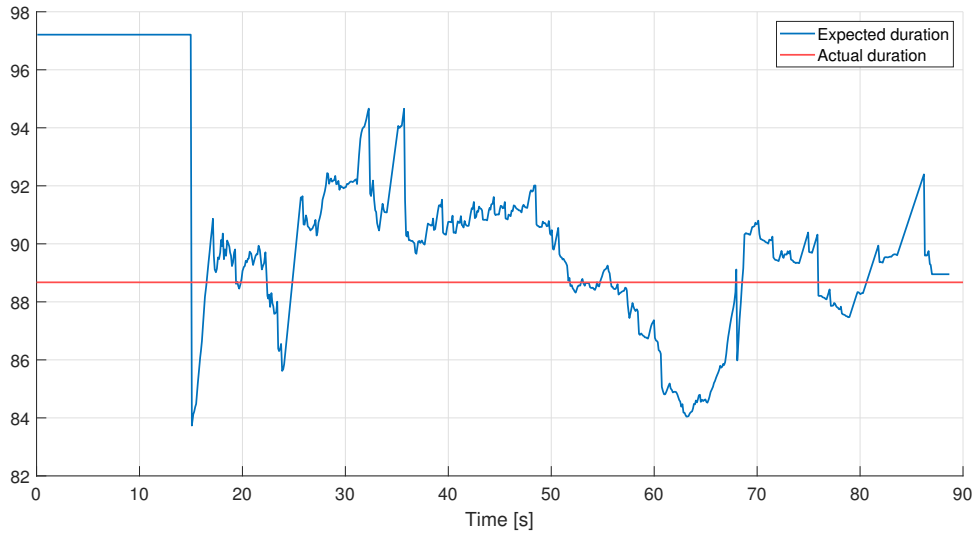


Figure 4.14: In red it is shown the actual duration of the operation while in blue it is represented the expected duration evaluated with our algorithm. It is apparent that, even after the building of the entire template tree, the expected duration gives good results.

4.3.2 Performance evaluation

In order to evaluate the performances of our work, we have decided to compare it with other two methods. The first one is the algorithm developed in the previous work ([6]) while the second one is the one described in Section 2.1 and uses as expected duration the average of the past duration of the operations that are greater than the current elapsed time. The operator has executed the task 139 times and the tree in figure 4.3 was built and updated as described in Chapter 3. We decided to calculate the errors of the three algorithms when they work at full capability and therefore we excluded the data related to the learning phase. Specifically, we considered the last 33 experiments. In particular, the most plausible variants have been executed more times than the error ones as it would happen in a real case (table 4.2 summarizes how many times each variant has been performed). To compare the performances of the different

Variant V_1	12
Variant V_2	3
Variant V_3	6
Variant V_4	2
Variant V_5	3
Variant V_6	7

Table 4.2: Number of execution of each variant considered to evaluate the error

methods we have evaluated the error between the expected duration of each method and the actual duration of the operation at each time instant. Let $\hat{T}_i(k)$ be the expected duration of the operation evaluated with the $i-th$ method and $\bar{T}$ the actual duration of the operation. The error $e_i(k)$ at instant k is computed as:

$$e_i(k) = |\hat{T}_i(k) - \bar{T}| \tag{4.1}$$

Figure 4.15 shows the expected duration evaluated with each method while monitoring an instance of variant V_1 (that was executed the majority of the time). It is apparent that our method performs well as its behavior shows small amplitude oscillations around the actual duration for the majority of the time. The method that uses the past operations duration has an offset, due to the fact that the average is greater than the actual duration of the monitored operation. While the algorithm developed in the previous work fails as it had as reference sequence another variant which is likely to be that of V_4, that is the error variant where the human forget a part of the task and outputs an incomplete product.

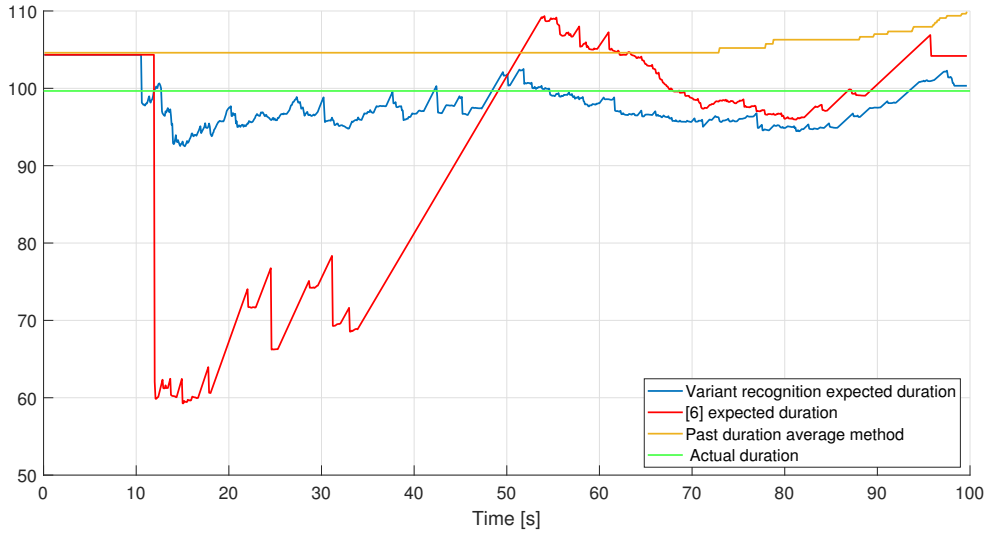


Figure 4.15: The green line represents the actual duration of the operation. Our algorithm (in blue) estimates well as its mean error is around 2s after the initial part. For comparison, the algorithm developed in previous work is represented in red while the method described in 2.1 is represented in yellow

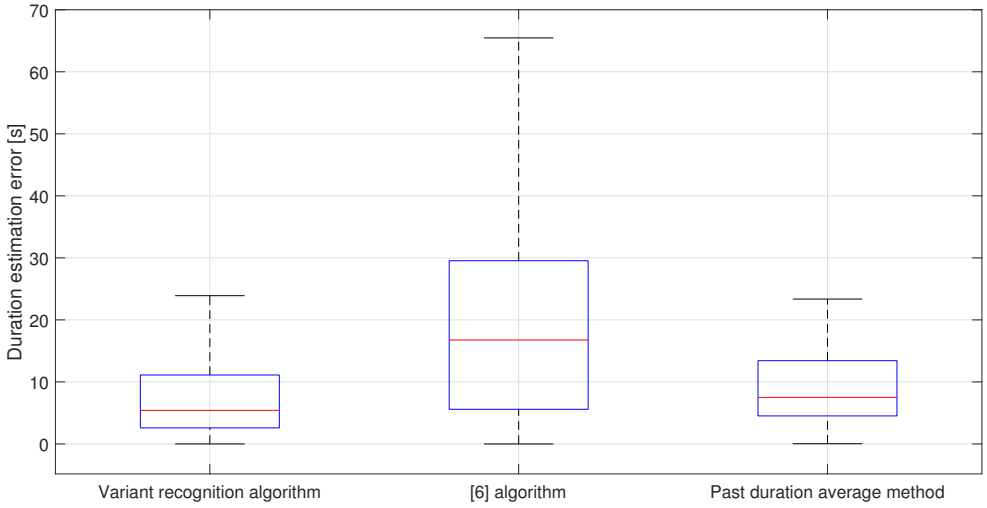


Figure 4.16: Previous work's estimate is not correct in the majority of the situations. The method that uses the average of past duration estimates well the duration of the operation, this is also due to the fact that the most plausible variants have similar duration. Our method is slightly better than the average method.

The boxplot in figure 4.16 shows the aggregated results obtained with the different methods. It is immediate to see that the work developed in the previous work is the worst one as it has a median equal to 16.8s, this is due to the fact that it considers only one sequence as the reference template. Initially, the first executed instance is taken, but eventually, the algorithm updates the template with a shorter one, without considering that the latter could be another variant of the task. Therefore, when the human executes a variant of the task different from the one used as template sequence, its DTW matrices costs arise and a bad advancement and duration estimate follow. The method that uses the average of past duration estimates well the duration of the operation, this is also due to the fact that the most plausible variants have similar duration, so when estimating one of them the average method is good. Our method is slightly better than this one with a smaller median error (our median error

is around 5.4s and the one of the past duration average is around 7.5s) and a similar variance. In particular, our method is able to correct itself quickly when it realizes what variant of the task the operator is performing. As example, figure 4.17 shows the behavior of the expected duration of our algorithm and the one obtained with the method described in 2.1. The monitored operation was the error variant V_4 (that is the one where the operator delivers to the output zone an incomplete product). The average past duration method clearly overestimates the duration of the operation because it is shorter than the other operations. Also our method overestimates the final duration of the operation. This is due to the fact that variant V_4 and variant V_6 are basically the same operation, but with the first one the last part of the operation is missing. So our algorithm considers as correct variant V_6 for the majority of the time (since it is the most probable one between the two, as it is one of the plausible variants of the operation), but then, when the operator performs the last segment of the operation ($f_4 \rightarrow f_5$) the current probability of the correct variant arises and the estimated duration is adjusted to the actual value of the operation duration.

On the other hand, the method proposed tends to produce peaks or valleys in the duration estimates in the first part of the operation and then it corrects as the operation goes on (as in figure 4.18). This is due to the fact that, when calculating the advancement rate according to equation (3.9), the total length of the operation is evaluated as $L_{next_{V_y}} + L_{prec_{V_y}}$, where $L_{prec_{V_y}}$ is the length of the past segments that belong to variant V_y while $L_{next_{V_y}}$ is just an estimate of the following segments length. This estimate (as explained in section 3.5) is calculated as a convex combination of lengths and historical probabilities of the possible next segments, so it could be imprecise as the historical probability of a segment that the human will not execute could be grater than the correct one. As the operation progresses, the number of segments performed increases and unfeasible variants of the operation are excluded, the length of the segments to be

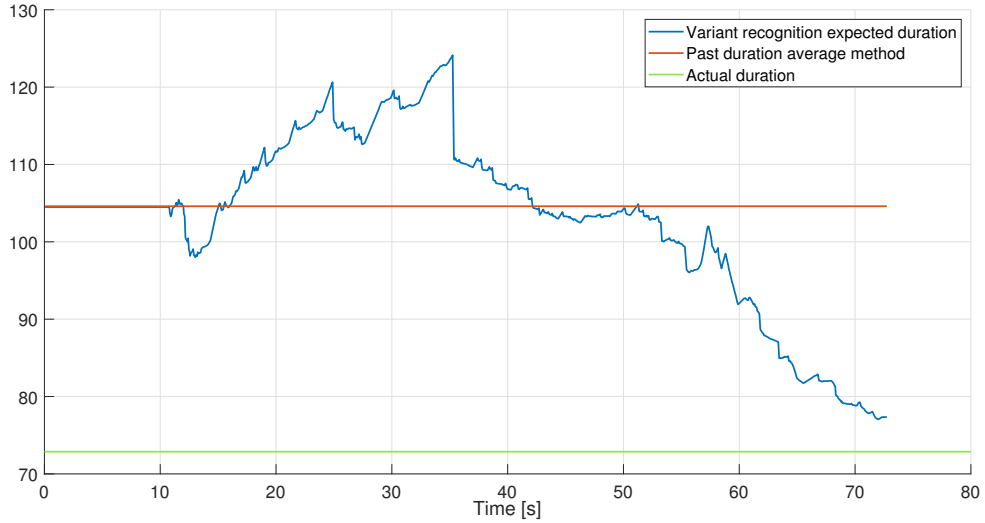


Figure 4.17: Our algorithm (blue) is able to correct itself when it recognizes the correct variant of the operation that the human is performing. While the method that uses past duration (red) keeps on overestimating the final duration. The green line represents the actual duration of the operation.

estimated is reduced and consequently the estimate improves. A final comparison was made to assess the estimation error between the three methods when monitoring the most probable variant (V_1). In order to do so, the algorithm of the previous work was trained with data related to variant V_1 only, while our algorithm and the past duration average method were trained with data related to all the variants present in the template tree. The box plot of the obtained results are shown in figure 4.19. It is possible to see that the past duration average method offers the worst performance in this case (its median error is around 5.2s), while the one of the previous work and our algorithm medians are around 4s on an average duration of 99s. Our method and the one developed in [6] almost offer the same performance. This means that the presence of several variants of the operation that are taken into account by our algorithm does not imply any worsening of the estimate of the main variant of the task. The slightly

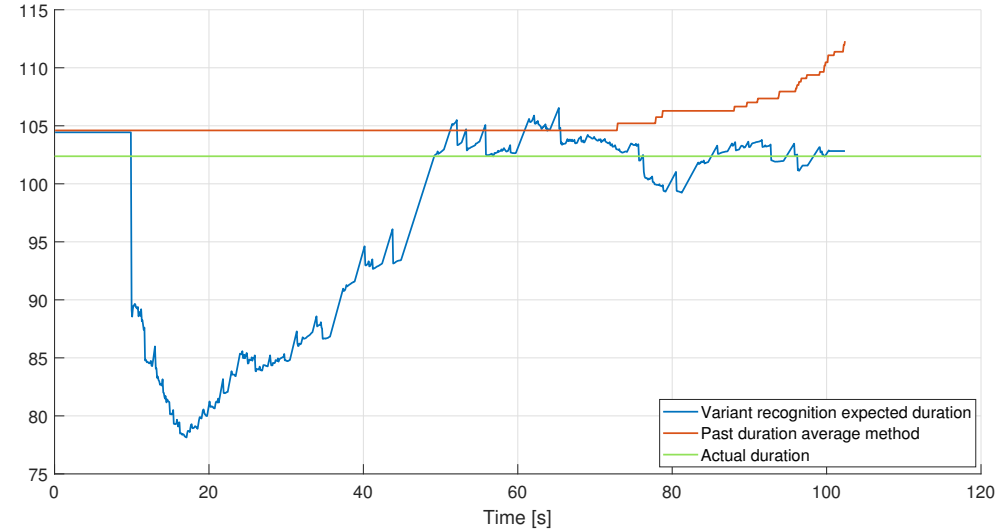


Figure 4.18: Our algorithm (blue) presents a valley around 18s due to the fact that the estimate of the length of the next segments is imprecise. In cases like this, the method that uses the average of past duration has a lower mean error.

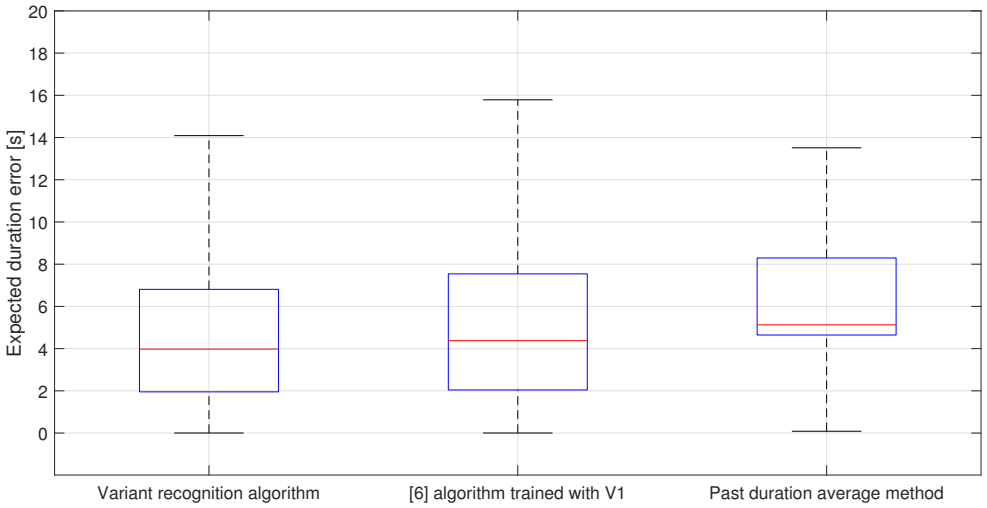


Figure 4.19: Boxplot of the errors obtained with the three methods when monitoring always the same operation variant.

worse performance of [6] may be due to the fact that it has fewer data to learn the best operation template and evaluate its average duration.
In conclusion, it can be said that we have made a significant improvement in the estimate of the operation duration since when only one variant is performed our algorithm offers the same performance as the previous one, while it is better when the human performs a variant of the operation.
Taking into account the obtained results, next chapter draws the main conclusions concerning the work presented in this thesis and explains possible future developments.

Chapter 5

Conclusions and future works

The behaviour of human operators in collaborative robotics applications, such as assembly operations, has been the subject of many studies in recent years, due to the ever-expanding use of collaborative robots within industries. In particular, in assembly operations, the robot should be able to correctly estimate the duration of the ongoing operation performed by the human and to schedule its tasks consequently. However, from the interaction between robots and humans, a large amount of problems arise, mainly due to the uncertainties caused by the latter. The human operator could execute the operations at different paces or in different ways if the task allows to do so. This last issue leads to different variants of the same operation and each one has a different time of execution. In order to better estimate the duration of the human task, the robot should be able to understand which variant is the one that the operator is performing.
In this thesis, a robust method for the real-time monitoring of human task advancement has been proposed. The main idea behind our work is to divide the ongoing operation into segments with the use of intermediate known features. Our algorithm is able to understand which, among currently feasible segments, is the most probable to be executed by the human, and to recognize a new variant (i.e. a sequence of segments that the operator has never done before) when it occurs. This leads to a correct

duration estimate of the ongoing operation also in presence of particular cases, such as error variants with only a partial completion of the product. This is possible by monitoring the movements of the human and by assigning them a cost that leads to penalizing the least likely segment (i.e. the reference sequence that differs most from the operation that the human is performing).
The obtained results are quite promising as the mean prediction error with respect to the average duration of the operation is 7.5%. To test the robustness of the algorithm, experiments were performed using similar segments, so part of the error is due to the fact that this method recognized the correct one after some seconds. On the other hand, the main limitations of the algorithm can be summarized in two points:

- Features known a priori: in this work the features (i.e. elements that allow to divide the operation into different segments) are know a priori. This means that the set of segments is limited to a combination of the known features, while it would be interesting to identify a feature in real-time to find also variants of the operations that are not part of this set. For example, a particular error variant that leads the human to a different workstation from the one that he uses. Moreover, a non optimal choice of the features lead to segments with section that are similar with each other. In turn, this makes it more difficult to distinguish the correct variant being executed.

- The estimate of the length of the next segments: the obtained results show that if the structure of the operation under analysis is described by a template tree with several branches, in the initial part of the operation the algorithm is imprecise since it evaluates the expected duration of the operation considering an estimate of the length remaining part of the operation, that can differ from the actual one.

Future works should investigate how to improve the performance of the algorithm considering these two issues.

Appendix A

Window size for task duration extrapolation

In the previous work the expected duration of the operation at instant k, $\hat{T}(k)$, was estimated as:

$$\hat{T}(k) = \frac{T_e(k)}{adv\%(k)} \tag{A.1}$$

where $T_e(k)$ and $adv\%(k)$ respectively are the elapsed time and the estimated percentage of advancement at instant k. This means that all the data recorded until instant k are considered in the estimation of the expected duration. This makes sense if the trend of the advancement is almost linear (like it happens in figure A.1). Unfortunately, there are cases in which the algorithm causes sudden changes in the slope of the advancement estimate, for this reason, we have tried to consider a smaller window for the duration estimation. In order to do so, the rule in A.1 was modified as:

$$\hat{T}(k) = \frac{T_e(k) - T_e(k-x)}{adv\%(k) - adv\%(k-x)} \tag{A.2}$$

where $k - x = y$ is the window size. We considered as optimal window size the one that causes the minimal error in the estimation of the duration. Let e_m be the mean error of the estimated duration with respect to the

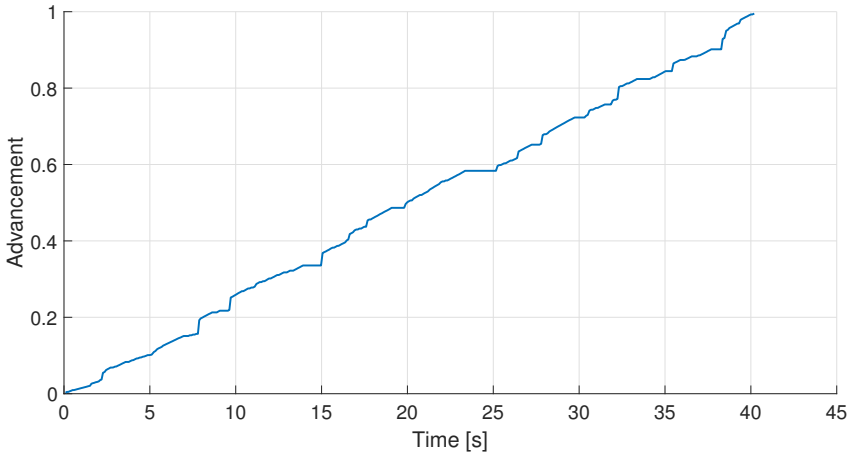


Figure A.1: Example of linear advancement trend

actual duration of the operation T_D, it is evaluated as:

$$e_m = \frac{\sum_{y=0}^{N} \hat{T}(y) - T_D}{N} \tag{A.3}$$

where N is the size of data in the operation. To have a robust estimate, we have considered 70 experiments where the operation monitored consisted in the assembly of a caster wheel. For each experiment, the mean error was calculated following the rule in A.3. Then the average of all the mean errors was evaluated for each y. In figure A.2 it is shown the relation between the mean error and the window size. It is apparent that the error decreases as the size of the window increases. Given the results obtained, we have decided to keep on using all the available data to estimate the duration.

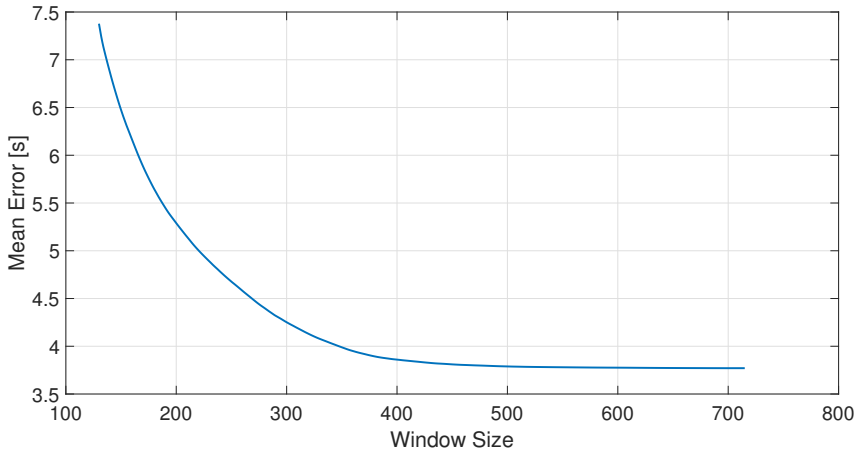


Figure A.2: Mean error sensitivity with respect to window size

Bibliography

[1] Arash Ajoudani, Andrea Maria Zanchettin, Serena Ivaldi, Alin Albu-Schäffer, Kazuhiro Kosuge, and Oussama Khatib. Progress and prospects of the human—robot collaboration. *Auton. Robots*, 42(5):957–975, June 2018.

[2] 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)*, pages 445–450, March 2015.

[3] Patrik Gustavsson, Magnus Holm, Anna Syberfeldt, and Lihui Wang. Human-robot collaboration – towards new metrics for selection of communication technologies. *Procedia CIRP*, 72:123 – 128, 2018. 51st CIRP Conference on Manufacturing Systems.

[4] 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)*, pages 1931–1938, May 2018.

[5] Richard Kelley, Alireza Tavakkoli, Christopher King, Monica Nicolescu, Mircea Nicolescu, and George Bebis. Understanding human intentions via hidden markov models in autonomous mobile robots. In *Proceedings of the 3rd ACM/IEEE International Conference on*

Human Robot Interaction, HRI '08, pages 367–374, New York, NY, USA, 2008. ACM.

[6] Paolo Lanfredini. Stima dell'avanzamento di un'operazione di assemblaggio per applicazioni di robotica collaborativa. Master's thesis, Politecnico di Milano, 2017-2018.

[7] Przemyslaw A. Lasota and Julie A. Shah. Bayesian estimator for partial trajectory alignment. In *2019, Robotics: Science and System (RSS 2019), Freiburg im Breisgau, Germany*, June 2019.

[8] Guilherme Maeda, Marco Ewerton, Gerhard Neumann, Rudolf Lioutikov, and Jan Peters. Phase estimation for fast action recognition and trajectory generation in human–robot collaboration. *The International Journal of Robotics Research*, 36(13-14):1579–1594, 2017.

[9] A. M. Zanchettin P. Rocco R. Maderna, P. Lanfredini. Real-time monitoring of human task advancement. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, November 2019.

[10] K. Sakita, K. Ogawara, S. Murakami, K. Kawamura, and K. Ikeuchi. Flexible cooperation between human and robot by interpreting human intention from gaze information. In *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, volume 1, pages 846–851 vol.1, Sep. 2004.

[11] Paolo Tormene, Toni Giorgino, Silvana Quaglini, and Mario Stefanelli. Matching incomplete time series with dynamic time warping: an algorithm and an application to post-stroke rehabilitation. *Artificial Intelligence in Medicine*, 45(1):11 – 34, 2009.

[12] A. M. Zanchettin, A. Casalino, L. Piroddi, and P. Rocco. Prediction of human activity patterns for human–robot collaborative assembly tasks. *IEEE Transactions on Industrial Informatics*, 15(7):3934–3942, July 2019.

[13] 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 (IROS)*, pages 6595–6600, Sep. 2017.

[14] Andrea Maria Zanchettin, Nicola Ceriani, Paolo Rocco, Hao Ding, and Bjoern Matthias. Safety in human-robot collaborative manufacturing environments: Metrics and control. *IEEE Transactions on Automation Science and Engineering*, 04 2015.

[15] M. Zinn, O. Khatib, B. Roth, and J. K. Salisbury. Playing it safe [human-friendly robots]. *IEEE Robotics Automation Magazine*, 11(2):12–21, June 2004.