Pattern recognition using Temporal Fuzzy Automata

Gonzalo Bailador del Pozo[a], Gracián Triviño Barros[b]

[a] *Universidad Politécnica de Madrid*
[b] *European Centre for Soft Computing*

Abstract

In this paper, we propose a syntactic pattern recognition approach based on fuzzy automata, which can cope with the variability of signals of the patterns by defining imprecise models. This approach is called Temporal Fuzzy Automata as it allows the inclusion of time restrictions to model the duration of the different states. The concept of fuzzy state makes it possible to handle ambiguity as the automaton can be in several states at the same time. Another advantage of our approach is the capability to synchronize with the signal, which allows us to avoid the segmentation stage before the recognition process. Furthermore, a learning method based on Dynamic Time Warping is provided that makes it possible to automatically generate models. Finally, to demonstrate the performance and robustness of this approach, we have applied it to the recognition of hand gestures without any kind of signal preprocessing.

Key words: Pattern recognition, Fuzzy models

1. Introduction

Many of the signals we can capture from the real world are composed of characteristic temporal patterns, e.g., the electrocardiogram (ECG) waveform, the accelerations captured at the hip during the human gait cycle, etc. Time influences a signal in two different ways: in the order of the different events that compose this signal and in the duration of these events. It is common for these patterns to present high variability in their amplitude or duration, and sometimes these patterns are corrupted by noise. Therefore a method to recognize these temporal patterns should take into account both aspects of time and be robust against the signal variability and noise.

Connectionist approaches, based on artificial neural networks, have been used widely in temporal pattern recognition, with emphasis on recurrent links to include memory in the network architecture [25]. Even these recurrent neural networks have shown their capability to detect keywords in a continuous speech without segmenting the signal (continuous recognition) [6]. Although we obtained good results in our previous experiences using this approach [2], the parameters (weights and thresholds) that define the behavior of the networks are difficult to interpret by an expert.

Another approach widely used in temporal pattern recognition is Hidden Markov Models (HMMs) [17]. A HMM is able to automatically model the statistical behavior of a temporal pattern using a set of states. Nevertheless, these models are not easy to understand because the obtained states usually do not coincide with the ones that a human expert extracts. During the recognition, a given signal is classified as a pattern using the forward algorithm, which calculates the probability of the HMM to generate this signal. However, this probability can only be used as a relative value to compare with probabilities obtained from HMMs of other classes. Finally, although HMMs are commonly used with isolated signals, they can also be used for continuous recognition by means of comparing the forward probabilities produced by the recognition method for each pattern at every time step [10].

In the case that there is a clear structure in the temporal patterns, syntactic pattern recognition methods can be used. The models produced by these methods are more understandable because they describe patterns as an ordered sequence of different events and each event is meaningful for the expert. One of the most representative methods of this field is the finite automata (FA). This method has been used in many pattern recognition applications like analysis of ECG signals [12], speech recognition [15], etc. In these applications there is always a preliminary preprocessing stage that minimizes the signal variability because FA deal badly with this variability. Furthermore, FA cannot recognize patterns if they present unexpected events that were not modelled previously. In order to solve this problem some authors, e.g. [8], have proposed to use error-correcting automata that accept deletion, addition or substitution of events in the signal. However, this solution implies modifying the automaton, including more states and links, to deal with these errors. Therefore, the resulting automaton is much more complex than the original one.

Another possible solution to tackle the problem of this variability is to include the concept of fuzzy sets in the structure of finite automata. Fuzzy sets, that were introduced by Zadeh in [27], are an extension of classical sets which allow the elements to belong partially to a set with a degree of membership. This feature allows expressing the imprecision of our knowledge of a given set by means of describing the set using smooth boundaries instead of sharp ones. Furthermore, fuzzy set operators can work with elements that belong partially to several sets which makes it possible to deal with the ambiguity present in real world problems. Initial works in fuzzy finite automata (FFA) started using only a fuzzy transition function over a discrete set of states [13, 26]. In FA the transitions between states are possible or not but in FFA these transitions can have intermediate values. One of the first works that introduced the concept of fuzzy state is [24]. The idea consists of allowing the FFA to be in different states at the same time with different membership degrees. Among other early attempts to implement FFA two works stand out: [21] and [18], although their goal is not pattern recognition. These works include the concept of fuzzy state, but it appears as a consequence of the evaluation of some conditions over the inputs. These conditions are defined using some linguistic labels for each state. Therefore, if the input fulfils partially the conditions of several states, the FFA

will be in several states with different degrees at the same time. This feature allows FFA to follow different paths concurrently when the input is ambiguous. In [1], the authors propose fuzzy correcting-error automata that also use the concept of fuzzy state. An interesting point of this work is that the FFA are able to give a matching measure of the pattern. However, this work aims to correct imperfect strings, not to recognize patterns in a signal. Another work based on fuzzy correcting-error automata is presented in [9]. Actually, this approach is not exactly an automaton, because it is a fuzzy controller whose state is a feedback variable to the system. The main problem of this approach is that the conditions over the inputs have to fulfil some restrictions to behave properly. One of the most advanced works using FFA for pattern recognition appeared in [22]. The authors propose to use hierarchical representations of the signal to recognize the temporal patterns of an ECG signal, but each representation in fact is a fuzzy automaton. Recently, FFA based on this hierarchical representation have also been applied in the field of fault detection to detect abnormalities in a DC motor [19]. In both papers, the authors introduce an interesting advantage of using FFA called input synchronization, that is, a fuzzy automaton only starts to follow the signal when this signal coincides with the expected pattern. Therefore, this method can be used for continuous recognition as it avoids the segmentation stage. Nevertheless, both authors do not model explicitly the duration of the different events of the pattern.

In this paper, we propose an implementation of fuzzy automata for pattern recognition that fulfils all the requirements presented above. It can deal with signals with variability in amplitude and duration. It allows the modelling of restrictions over the duration of the states. It is able to synchronize with the signal and when it recognizes a pattern it is able to provide a degree of matching that measures how well the signal matched with the model. Additionally, we also propose an automatic method to generate a model from several instances of the pattern.

The rest of the paper is organized as follows. Section 2 introduces the concept of Temporal Fuzzy Automaton (TFA). In Section 3 we explain how an expert can model a pattern using a TFA. Section 4 shows in detail the process of pattern recognition with the TFA. In Section 5 we present the results of several tests that show the main advantages of our approach. Section 6 explains the machine learning method that we propose to generate a TFA. In Section 7 we show the results obtained when applying our method to the recognition of hand gestures. Finally, in Section 8, we conclude the paper stating our final conclusions.

2. Temporal Fuzzy Automaton

Based on the definition of General Fuzzy Automaton (GFA) presented in [4], we define a Temporal Fuzzy Automaton (TFA) as an eight-tuple:

$$TFA = \{X, Q, D, \delta, F_1, F_2, S, F\}$$

- $X = \{x_1, ..., x_P\}$ are the inputs.
- $Q = \{q_1, ..., q_N\}$ is a finite set of states.
- $D = \{d_1, ..., d_N\}$ is the duration of the states.
- $\delta : Q \times D \times X \times Q \rightarrow [0, 1]$ is the state transition function.
- $F_1 : [0, 1] \times [0, 1] \rightarrow [0, 1]$ is the state membership assignment.
- $F_2 : [0, 1]^* \rightarrow [0, 1]$ is the multi-membership resolution.
- $S \subset Q$ is the set of initial states.
- $F \subset Q$ is the set of final states

In the following paragraphs, we explain each element of this definition in detail, as well as some related conventions and definitions.

Input set X.
This set, which can be composed of several input streams x_j, contains the samples of the signal that the TFA has to recognize.

Convention 1. The values of all the input streams at time t are expressed as x^t and the specific value of the input x_j at time t is x_j^t. Where t is a natural number that identifies each sample of the input.

Set of states Q.
This set contains the different states of the TFA. In contrast to a FA that can be only in one state at every instant, our TFA can be in several states with different degrees. We have adopted the concept of fuzzy state following the definition by Klir of fuzzy set [11] :

Definition 1. A fuzzy state μ_Q defined on a set Q, is a function mapping each element of Q to a unique element of the interval $[0, 1]$.

$$\mu_Q : Q \rightarrow [0, 1] \tag{1}$$

Convention 2. The membership value (mv) of a state q_i at time t will be referred to as $\mu_{q_i}^t \equiv \mu_Q^t(q_i)$.

Definition 2. We consider that a state q_i is active at time t when:

$$\mu_{q_i}^t \geq \theta \tag{2}$$

Where θ is a constant defined by an expert depending on the application.

Duration of states D.
This set provides information about the duration of the different states and it is used for expressing time restrictions in the TFA.

Definition 3. Each element d_i of the set of durations D represents the time that the state q_i has been continuously active. Consequently, the duration of the state q_i at time t can be calculated :

$$d_i^t = \begin{cases} 0 & \mu_{q_i}^t < \theta \\ d_i^{(t-1)} + 1 & \mu_{q_i}^t \geq \theta \end{cases} \tag{3}$$

The state transition function δ.

This function determines the possible transitions of the TFA.

Definition 4. We consider that a transition T_{ij} from the state origin q_i to the destination state q_j is possible if :

$$\exists\, x, d \; such \; that \; \delta(q_i, x, d, q_j) > 0 \tag{4}$$

However, the TFA only follows a transition T_{ij} at time t if this transition is possible and the origin state q_i is active or belongs to the set of initial states S (this last condition allows the TFA to start although no state is active) :

$$\delta(q_i, x^t, d^t, q_j) > 0 \; \wedge \; \left(\mu_{q_i}^t \geq \theta \; \vee \; q_i \in \; S\right) \tag{5}$$

The result of δ can be interpreted as the weight of the transition. In most of the previous papers about fuzzy automata [13], this weight is a constant defined by the designer but in this work this value represents the degree in which some of the conditions of the transition are fulfilled. These conditions are specific for each transition and they are in function of the inputs X and the duration of the origin state d_i. In the next section we will explain how to define these conditions for each transition using linguistic labels.

The state membership assignment function F_1.

This function is used to update the mv of the destination state when the TFA follows a transition T_{ij}. Some works [7] propose to copy the weight of the transition directly to the mv of the destination state (μ_{q_j}), ignoring the mv of the origin state(μ_{q_i}). However, this presents a problem because a transition whose origin is a state with a low mv can produce a destination state with high mv, if its weight is high. In order to solve this problem, the authors of [4] propose to take into account the mv of the origin state using a function F_1 called state membership assignment. This function receives as arguments the mv of the origin state and the weight of the transition respectively and returns the mv of the destination state. Consequently, the equation to obtain the mv of the state q_j after a transition from the state q_i to a state q_j will be :

$$\mu_{q_j}^{t+1} = F_1(\mu_{q_i}^t, \delta(q_i, x^t, d^t, q_j)) \tag{6}$$

Following the indications of the authors, the function F_1 only has to satisfy the idempotence of the elements 0 and 1:

$$F_1(0,0) = 0 \quad , \quad F_1(1,1) = 1 \tag{7}$$

The multi-membership resolution function F_2.

It could happen that several transitions end in the same destination state and then several mv's will correspond to the same state. The problem of multi-membership values was also studied by the authors of [4] and they propose to use another function F_2 called multi-membership resolution. This function provides only one mv to summarize all the mv's of those transitions that end in the same state. Therefore, the update of the mv of a state will be calculated with the following expression:

$$\mu_{q_j}^{t+1} = F_2(v_1, \ldots, v_n) \tag{8}$$

Where n is the number of transitions that end in the state q_j and each v_i is the mv provided by each transition. Following the authors, this function must also fulfil two requirements:

$$F_2(\emptyset) = 0 \tag{9}$$

$$F_2(v_1, \ldots, v_n) = a \quad if \quad \forall\, i\ (v_i = a) \tag{10}$$

Initial states S and final states F.

The set S contains the initial states of the TFA and the set F the final states. The execution of a TFA starts from its initial states, after which it moves between different states according to its state transition function δ until it reaches a final state when the pattern is recognized.

3. Modelling a pattern using a TFA

In this section we are going to explain how an expert can model easily a pattern with a TFA following a simple methodology. During this explanation, we will use a simple example to clarify the procedure. This example consists of a rectangular signal composed of four stages as can be seen in figure 1.

3.1. Defining the states

Before starting to model a pattern we have to decide what inputs X we are going to use to describe it. In our example we only use the amplitude of the signal (x_1). Each state q_i represents a time during which the inputs fulfil some conditions. Although for this example is not necessary, we have defined these conditions using the idea of linguistic interval presented in [14]. When modelling more complex patterns, the linguistic intervals make it possible to use different levels of granularity by combining contiguous membership functions to obtain wider membership functions. A linguistic interval $A_j^{[a,b]}$ is defined over the domain of the input x_j using a set of fuzzy linguistic labels A_j^k (e.g. see figure 2). The two indexes a and b indicate the membership functions where the interval begins and ends respectively. The degree of membership for each value in the interval is calculated using the following expression:

$$\mu_{A_j^{[a,b]}}(x_j) = \sum_{k=a}^{b} \mu_{A_j^k}(x_j) \tag{11}$$

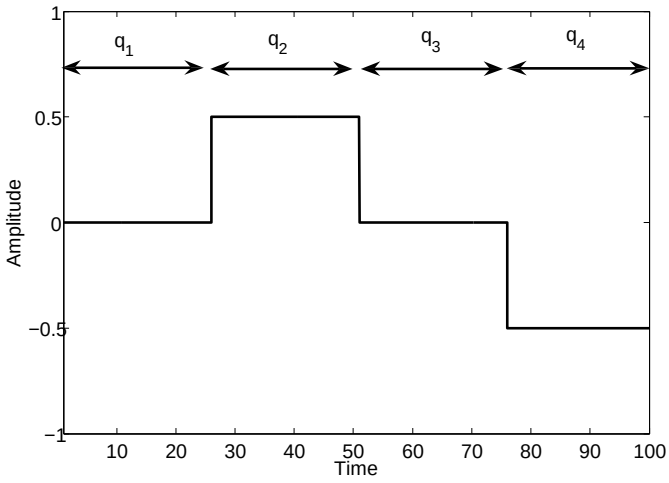

Figure 1: Pattern of a rectangular signal

To define the linguistic labels we use several membership functions that compose a strong fuzzy partition of the input as defined in [20] since this restriction guarantees a membership degree of 1, except for in the boundaries where membership degree decreases. In particular, we propose to use triangular membership functions, not only because of their simplicity, but also because, under some assumptions, they can build fuzzy partitions whose membership functions are uniformly activated [16].

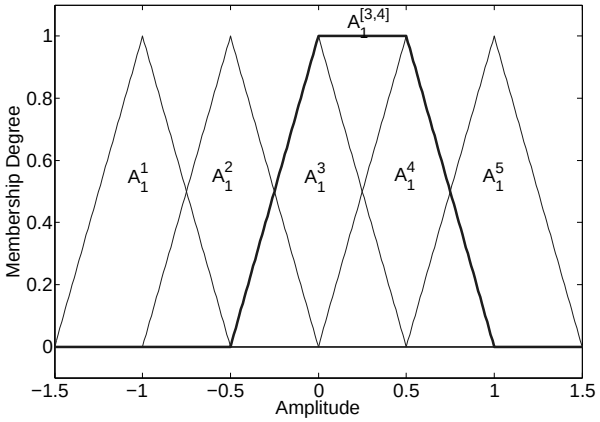

Figure 2: Example of definition of linguistic interval for input x_1

In the general case the conditions of a state ($Cond_{q_i}$) can affect several input streams. In our implementation we have chosen to combine the conditions over all the input streams using the conjunction:

$$Cond_{q_i}(x_1, \cdots, x_P) = \left\{ \mu_{A_1^{[a_1^i, b_1^i]}}(x_1) \wedge \ldots \wedge \mu_{A_P^{\left[a_P^i, b_P^i\right]}}(x_P) \right\}$$

Although the definition of TFA does not constrain its structure, we model the patterns using a left-right structure because the resulting TFAs are more understandable. A pattern is described with several states and a last state, whose only function is to detect the end of the pattern. During the recognition of a pattern the TFA will move from the first state to the last state hence the set S only contains the first state and F only the last state. The TFA that models our example is represented in figure 3. The conditions over the amplitude (x_1) for each state, using the linguistic labels shown in figure 2, are the following:

$$Cond_{q_1}(x_1) = \mu_{A_1^{[3,3]}}(x_1) \text{ , } Cond_{q_2}(x_1) = \mu_{A_1^{[4,4]}}(x_1) \text{ ,}$$
$$Cond_{q_3}(x_1) = \mu_{A_1^{[3,3]}}(x_1) \text{ , } Cond_{q_4}(x_1) = \mu_{A_1^{[2,2]}}(x_1) \text{ , } Cond_{q_5}(x_1) = 1$$

The state q_5 has no conditions because its objective is only to detect the end of the pattern.

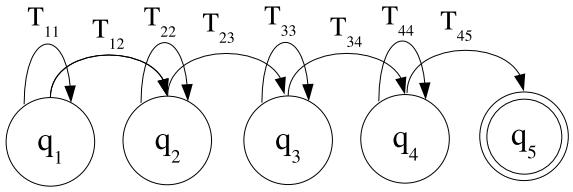

Figure 3: Structure of a pattern using TFA

3.2. Defining the transitions

We have presented how to define the conditions over the inputs for each state, however the evolution of the TFA is only based on the state transition function δ. For this reason, these conditions have to be included in the different transitions of the TFA. Using a left-right structure, the TFA only can move from the state q_i to itself (feedback transition T_{ii}) or to a posterior state q_{i+1} (changing transition T_{ii+1}). All the transitions are defined using two conditions: one over the inputs and the other over the duration of the origin state which makes it possible to include some time restrictions. For any transition T_{ij}, the condition over the inputs is the condition for being in the destination state ($Cond_{q_j}$). However, the time condition is different depending on if it is a feedback transition or a changing transition.

In the case of a feedback transition T_{ii}:

$$Cond_{T_{ii}}(x, d_i) = \{Cond_{q_i}(x) \wedge \mu_{TimeToStay_i}(d_i)\}$$

The time condition is expressed with the linguistic label $TimeToStay_i$ which restricts the maximum time that the TFA can be in the state q_i. The membership function associated to $TimeToStay_i$ is defined over the domain of the duration of the state (d_i). This function starts with the maximum membership value (1) and it decreases when the maximum duration is reached. The goal of this condition is to reject those signals which cause the TFA to remain in a state more time than the expected one. The TFA stays in the same state until the conditions over the inputs are false or the duration of the state is longer than the time defined by the membership function $TimeToStay_i$.

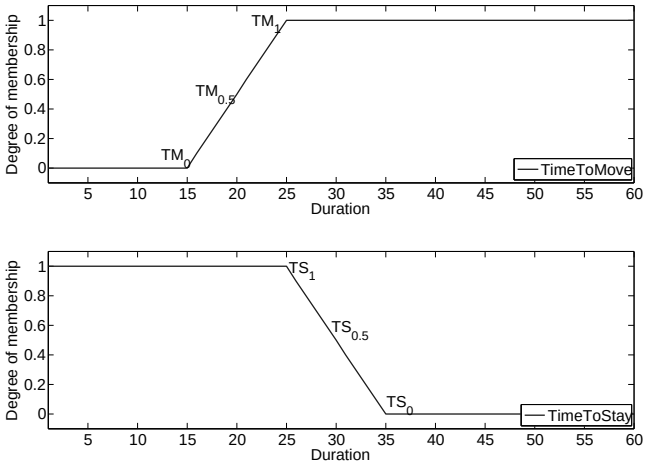

Figure 4: Linguistic labels for the duration of a state

In the case of a changing transition T_{ii+1}:

$$Cond_{T_{ii+1}}(x, d_i) = \left\{Cond_{q_{i+1}}(x) \wedge \mu_{TimeToMove_{ii+1}}(d_i)\right\}$$

The time condition is $TimeToMove_{ii+1}$ which represents the minimum time that the TFA has to be in a state q_i before moving to a state q_{i+1}. Therefore, the membership function associated with the linguistic label $TimeToMove_{ii+1}$ starts with the lowest membership value (0) and it increases when the expected minimum duration is reached. This restriction allows the TFA to move from one state to another when the conditions over the inputs for being in the state q_{i+1} are true and the TFA has been sufficient time in the origin state q_i.

As we said before, the last state is a special case because its goal is only to detect the end of the pattern. This last state has no feedback transition because

the TFA is not supposed to stay in this state. The transitions that end in this state only have conditions over time to ensure that the previous state was active for a while.

$$Cond_{T_{N-1N}}(x, d_{N-1}) = \left\{\mu_{TimeToMove_{N-1N}}(d_{N-1})\right\}$$

All states of the example have the same length (25 samples) and therefore we have used only two membership functions to define the time restrictions for all states. Figure 4 shows both functions, $TimeToMove$ for moving to another state and $TimeToStay$ for staying in the same state. In order to include some imprecision in the duration of the states, the membership function $TimeToMove$ increases gradually from TM_0 to TM_1 and $TimeToStay$ also decreases gradually from TS_1 to TS_0. This imprecision allows the system to be more robust against the variability in the state durations.

4. Recognizing a pattern with a TFA

After modelling a pattern using a TFA, we can use this TFA to detect a pattern in the signal. During this process, we can distinguish three stages:

- Synchronization. The TFA is waiting for the inputs to fulfil the conditions of the initial state.
- Update. The TFA is updated at each time step.
- Ending. The TFA reaches a final state. At this moment we can calculate a measure that indicates the degree of matching of the signal with the expected pattern.

In the next sections each stage will be described in detail.

4.1. Synchronization

Usually the signal that contains the expected pattern is not segmented and includes values prior to the pattern. Therefore, we are interested in synchronizing the TFA with the beginning of the expected pattern. Two problems can affect this synchronization phase. First, the TFA can get synchronized before the expected pattern appears because the signal prior to the pattern has fulfilled the conditions of the states by chance. One could expect that the TFA will not recognize the expected pattern because it will be in a posterior state. However, this problem is solved because the TFA can always enter the initial states even when the TFA is already synchronized (see the conditions to follow a transition as stated in 5). The second problem occurs when the signal before the expected pattern fulfils the condition of the first state. In this case, the duration of this state will be longer than the expected duration. We can solve this problem by modifying the condition $TimeToStay$ of the first state by allowing the TFA to remain in this first state forever.

4.2. Update

For each time step, the TFA updates the mv's of all states. For this task, it uses the state transition function δ to calculate the weight of the possible transitions. This weight is only calculated for those transitions whose origin is an active state at the current time or this origin belongs to the set of initial states S. The value of the weight of the transition T_{ij} is the evaluation of the condition defined for this transition.

$$\delta(q_i, d_i, x, q_j) = Cond_{T_{ij}}(x, d_i)$$

In our approach, we have used the minimum function for the evaluation of the conjunction $\wedge$. The update of the next state is done with the membership assignment function F_1. In this paper we propose a function that introduces a kind of inertia over the mv of the states. The reason for including this inertia is to avoid quick changes in the inputs (due to noise) to produce quick changes in fuzzy state. This inertia makes the TFA more tolerant to noise. Hence our function F_1 is the following:

$$F_1(\mu, \delta) = \alpha\mu + (1 - \alpha)\delta \tag{12}$$

Where α is the inertia factor that defines how much we rely on the weight of the transition versus the mv of the origin state to calculate the mv of the destination state. If α is close to 0, we only take into account the weight of the transition, so the TFA changes the state as soon as the inputs change. But, if α is close to 1, the TFA changes the state slowly and reacts slowly to changes in the input.

After calculating the mv of the states, it is possible that there are several mv's for the same destination state. The multimembership resolution function F_2 is applied to determine the activation of the destination state. In our case we use the maximum function and the next mv for the state q_j can be obtained with this equation:

$$\mu_{q_j}^{t+1} = \max_i[v_i] \tag{13}$$

Where each v_i is the mv produced by each transition ending in state q_j. If there is no transition to a destination state the mv of this state will be set to zero (see first requirement of F_2 stated in 10.). But this behavior is abrupt, because if only one input sample does not match, then the TFA loses the synchronization with the signal. Therefore, we also propose to include inertia to force the system to leave states gradually. In the case that there is no transition to a state, the mv of this state will be updated with its previous mv multiplied by α.

The decision of the value of the α factor is not an easy task and it depends on the application and the noise in the input signal. However, if we want the TFA to be robust to errors with a duration less than N samples, then we can use the following expression to estimate the minimum value for α:

$$1 \cdot \alpha \cdot \alpha \cdots \alpha > \theta \quad \longrightarrow \quad \alpha > \sqrt[N]{\theta} \tag{14}$$

This approximation is obtained by supposing that the TFA starts with the highest mv (1) in a state and no conditions of any transition to this state are

fulfilled. Therefore, the update of the state is only based on its previous mv until it reaches a mv lower than the threshold θ.

After updating the mv of all states, the duration of the states is also updated. The durations of active states are increased while the durations of inactive states are reset.

4.3. Ending

Whether the TFA arrives at a final state without losing the synchronization, this means that the TFA has recognized a pattern and then it could be interesting to measure how well the signal matched with the model. When the TFA is synchronized, the mv of the states represent how well the current input fits with the expected input for the current state at this time instant. In this paper, we propose to use the maximum value of mv's of all states as an instantaneous measure of the degree of matching with the modelled pattern. However, this measure has to be extended along the whole signal, therefore we calculate the average of all the values. The degree of matching (DOM) of a signal S that follows the modelled pattern can be calculated as:

$$DOM(S) = \frac{\sum_{t=t_o}^{t_f} \max_{i=1}^{N} [\mu_{q_i}^t]}{t_f - t_o} \tag{15}$$

Where t_o is the instant of time when the TFA got synchronized and t_f is the moment when the TFA arrives at a final state. In the case that the TFA does not arrive at the final state the degree of matching is zero.

5. Tests

In this section we are going to use the previous simple example to show the different functionalities that our approach possesses. In the following figures, each row represents the mv of a state for each instant of time. White color represents a maximum degree of state activity and black a minimum degree.

5.1. Synchronization

One of the main advantages of using TFA is its ability to synchronize with the inputs. For testing this, we included a noise signal at the beginning of the real pattern. In figure 5 we can observe how the TFA tries to synchronize with the signal but it only achieves this when the real pattern appears. After this synchronization the TFA moves crisply between the different states because it is receiving the exact pattern. During this time the TFA behaves like a finite automaton except in the last state where the mv increases gradually since it is following the time condition $TimeToMove$. It can also be seen that during the part of the signal corresponding to state q_3, the TFA is in the state q_1 and q_3 at the same time. This effect is produced because both states have the same condition and we permit the TFA to enter in initial states ($q_1 \in S$) although the TFA is already synchronized. This shows the ability of the TFA to follow several paths in parallel.

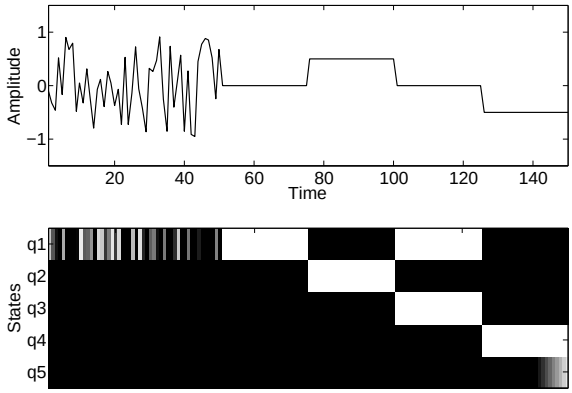

Figure 5: Synchronization of the TFA($\alpha = 0$, $\theta = 0.01$)

5.2. Robustness against time variability

In order to show how the TFA behaves when there is variability in the duration of the states of the pattern, we have increased gradually the period of each state of the pattern from 25 samples to 45 samples. The results of this simulation are shown in figure 6 and it can be observed that the membership degrees of the states decrease although the condition over the input is completely fulfilled. This decrease is caused by the fall in mv calculated by the membership function $TimeToStay$. Note that in the last pattern the TFA loses the synchronization because the duration of the first state is 45 samples and for this duration the membership degree of $TimeToStay$ has fallen to 0 (see figure 4).

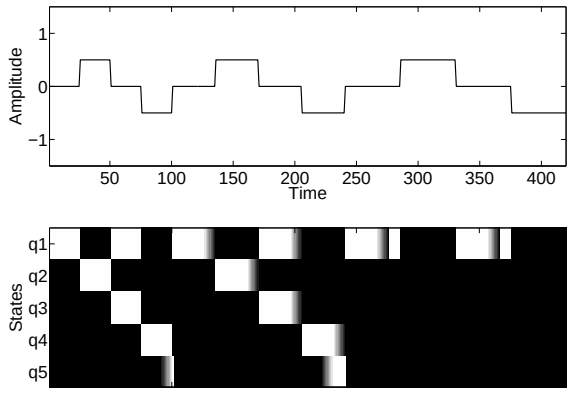

Figure 6: Pattern with increasing period($\alpha = 0$, $\theta = 0.01$)

5.3. Robustness against amplitude variability

In this section we demonstrate the robustness of the TFA against variability in amplitude with two tests. The first test shows that the TFA can deal with variability thanks to the use of fuzzy sets to define the inputs. In figure 7 we can see the mv's of the states for the pattern signal that has been corrupted with uniform noise of range $[-0.25, 0.25]$. Note how the mv of the states get lower and sometimes other states are activated because the input fulfils its conditions and the condition $TimeToMove$ is also true.

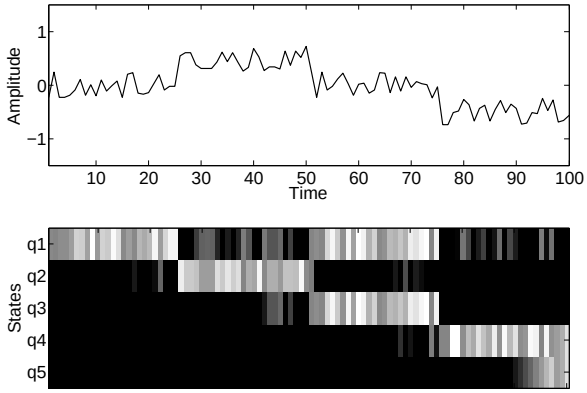

Figure 7: Pattern corrupted with uniform noise $[-0.25, 0.25]$ ($\alpha = 0$, $\theta = 0.01$)

In the first test, the noise was of such magnitude that the signal was always in the support interval of the associated membership function for each state, thus the TFA did not lose track of the pattern. The second test shows how the inertia can help the TFA to maintain the synchronization when there are several consecutive samples that get completely out of the defined membership function. This system is able to deal with this situation by means of the inertia factor. In this test, the trial pattern has some glitches with duration of 2 and 7 samples respectively and the TFA is working with the parameters $\alpha = 0.5$ and $\theta = 0.01$. Figure 8 shows how the TFA can recover from a glitch in the signal with 2 samples of duration but not for a glitch with 7 samples. This is the expected result when considering equation 14 to estimate α factor. For $N = 2$ the expression is true ($0.5 > \sqrt[2]{0.01}$) while for $N = 7$ the expression is false ($0.5 \not> \sqrt[7]{0.01}$). The influence of the inertia factor can also be seen in the transitions between the states which are smoother than before.

5.4. Degree of matching

In order to show how the degree of matching behaves when there is noise in the signal, we have fed the TFA several instances of the pattern corrupted with uniform noise of different ranges using $\alpha = 0$. The following table shows the DOM for each corrupted pattern.

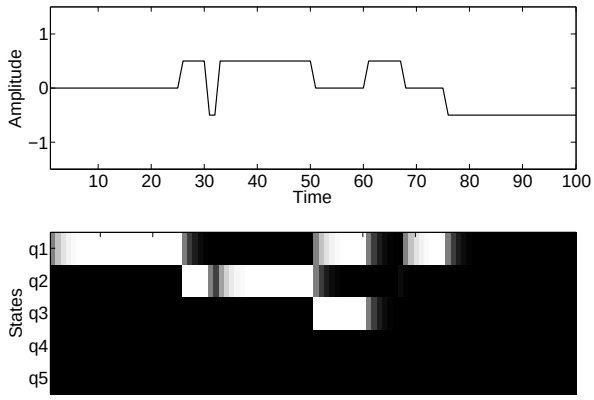

Figure 8: Pattern corrupted with glitches ($\alpha = 0.5$, $\theta = 0.01$)

Range of noise	0	0.05	0.10	0.15	0.20	0.25
Degree of Matching	1.00	0.95	0.91	0.85	0.81	0.75

It can be seen that the wider the range of noise the lower the degree of matching.

6. Learning a TFA from training instances

In this section we explore the possibilities of creating a TFA that models a signal automatically. Using N instances of the pattern to be modelled $T_{1..N}$ as a training set, we generate a TFA that summarizes the common behavior of all. In the case that these training instances only present variability in their amplitude, the general model could be obtained from the maximum and minimum value of the signals for each sample time step. However the training instances usually also present variability in their duration and speed so it is necessary to use more complex techniques.

Dynamic Time Warping (DTW) has been widely used in many fields, such as, speech recognition [3], signature recognition [5], pattern recognition with different time scales [28], etc, and shows a great ability to calculate the similarity between patterns that present variability in the time axis. Instead of only calculating the distance between a reference pattern and an analyzed one for each time step t, this method compresses and expands the time scale of the analyzed sample in order to synchronize both patterns and to minimize the total distance. This means that several time instants of the reference pattern can correspond to only one time instant of the analyzed pattern and, viceversa, one time instant of the reference pattern can correspond to several time instants of the analyzed pattern. DTW provides two results: the minimum total distance between two samples and the warping path that explains how to compress or expand the analyzed sample to produce this minimum value.

6.1. Obtaining a summary using DTW

In this paper we use DTW to obtain a kind of summary of all the training instances in the following way. First of all, the distances between all possible pairs of training instances are calculated. The instance that presents the smallest average distance to all the remaining instances is used as reference to calculate the warping paths. Using these warping paths we rescale all the training instances to extract a summary that has the same length as the reference pattern.

Each element of the obtained summary represents the behavior of all the training instances $T_{1..N}$ at a normalized time step t. It contains the following information:

- For each input stream j, the minimum value $f_j^-(t)$ of the signal for all the training instances $T_{1..N}$.
- For each input stream j, the maximum value $f_j^+(t)$ of the signal for all the training instances $T_{1..N}$.
- For each training instance $T_{1..N}$, a number $W^i(t)$ that indicates how many time steps the signal of the training instance T_i was compressed or extended to fit with the reference pattern at time t.

The sign of the number $W^i(t)$ indicates if the analyzed pattern was compressed or extended; a positive value p means that the analyzed pattern was compressed p times, a negative value $-n$ that the analyzed pattern was extended n times and zero means that there was no need to change the time scale.

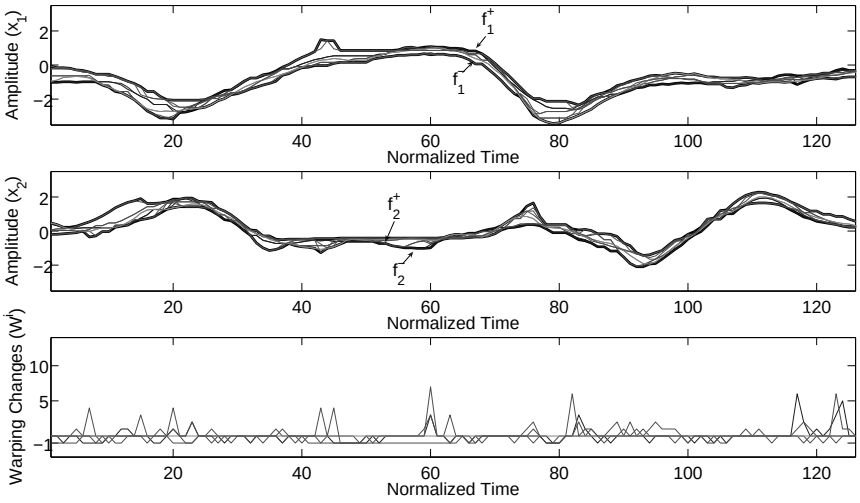

Figure 9: Summary extracted from several training instances

An example of this summary can be seen in figure 9 which represents the summary of five instances $T_{1..5}$ of a pattern described with two components

x_1 and x_2. The two upper graphs represent all the possible values that the training instances took for each normalized time step t. The maximum f_j^+ and minimum f_j^- functions indicate the maximum and minimum values of all the training instances for each input j. In the next subsection, these functions will be the basis for the definition of the linguistic intervals to cover all the possible values. Finally, the lower graph shows how to warp the time for each specific training instance.

6.2. Obtaining the states of the pattern

After obtaining the previous summary, the next step consists of identifying the different states. Each state will represent a time during which all the input streams are stable. Therefore, for all the time samples of this state the conditions over the inputs are equal. As we explained before, the conditions of a state q_i are expressed with a linguistic interval $A_j^{\left[a_j^i, b_j^i\right]}$ for each input stream j. This linguistic interval should cover all the possible values of each input j during the length of state. This means that for every time sample t of the state q_i all the values between the minimum $f_j^-(t)$ and maximum $f_j^+(t)$ should have a degree of membership greater than 0.5 in the membership function associated with the interval $A_j^{\left[a_j^i, b_j^i\right]}$.

However, in order to have a meaningful TFA, two contiguous states have to be distinct from one another; the conditions need to be as distinct as possible. To create a TFA whose consecutive states do not have similar conditions, we obligate to every pair of consecutive states to fulfil a restriction. The conditions over the inputs of two consecutive states (i and $i+1$) have to be disjoint at least for one input j. This means that the linguistic intervals $A_j^{\left[a_j^i, b_j^i\right]}$ and $A_j^{\left[a_j^{i+1}, b_j^{i+1}\right]}$ which define the conditions of the input j for both states have to fulfil the following proposition:

$$(a_j^i > b_j^{i+1}) \vee (b_j^i < a_j^{i+1}) \tag{16}$$

For example, if the interval represented in figure 2 corresponds to the state i, the beginning of the interval of the next state $i+1$ has to be greater than 4 ($a_1^{i+1} > 4$) or its end has to be less than 3 ($b_1^{i+1} < 3$).

The algorithm, that allows the automatic extraction of the different states, scans the summary for consecutive samples with similar conditions. When it finds a sample with conditions different from the previous ones, it generates a new state that summarizes the conditions over the inputs until that sample. Nevertheless, the conditions of this new state must fulfil our restriction of being distinct states. Consequently, if necessary, we adapt these conditions to make this state disjoint with the previous and the next state. We repeat the process until there are no more samples left.

Figure 10 shows the division in different states of the summary presented in figure 9 using this algorithm. It can be seen that there are some values of the signal that are not covered by the conditions of any state. This is caused by applying our restriction to the found states.

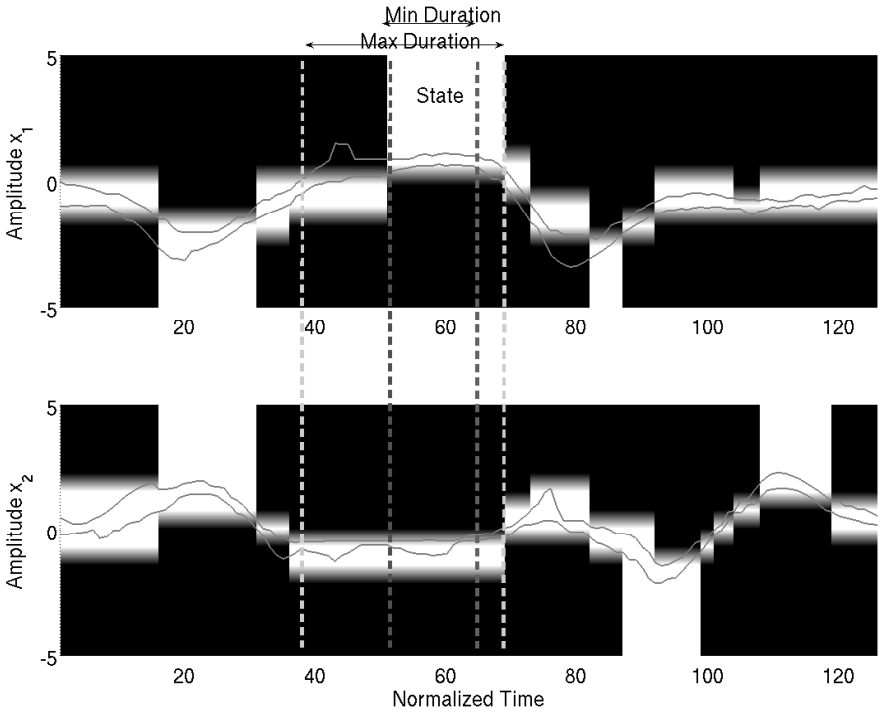

Figure 10: States obtained from previous DTW summary. Each state can be identified because the conditions over the input streams (x_1 and x_2) are the same during the length of the state.

6.3. Creating the transitions of the TFA

As we explained before, the condition over the inputs for a transition is the condition of the destination state. However, the condition over time is different depending on if it is a remaining transition or a changing one. The condition over time for a remaining transition reflects the maximum time that the TFA can stay in a state. It could be said that this time is the duration of the state extracted in the previous subsection but, as we can see in figure 10, it is possible that some values of the signal fulfils partially the conditions of the state before entering it. When calculating this maximum time we should also take this into account. An example of the maximum duration for a particular state is represented in figure 10 with two light dotted lines. In the case of a moving transition, the conditions over the time reflects the minimum time that the TFA has to be in state before moving to the next one. To calculate the minimum duration we should subtract the time in which some values of the signal are part of the linguistic interval of the next state. An example of this minimum duration is represented in figure 10 using two dark dotted lines.

However these periods are expressed in a normalized time and therefore we need to calculate the real durations using the summary. The real duration of a

segment of time that starts at normalized time step t_1 and ends at normalized step t_2 for a specific training instance T_i can be calculated using the following expression:

$$\sum_{t=t_1}^{t_2}(W^i(t)+1) \tag{17}$$

Thus, each training instance will produce a different duration of the segment. In order to incorporate this variability in our TFA, we calculate the maximum and minimum duration for all the training instances and we use these values to define the membership functions that describe the time conditions (see figure 4). In the case of a remaining transition the minimum time value is used to define the point where the membership function starts to decrease (TS_1) and the maximum time value where the function reaches the degree of membership 0.5 ($TS_{0.5}$). In a changing transition, the minimum value indicates when the membership function takes value 0.5 ($TM_{0.5}$) and the maximum value when the membership function has the highest degree of membership (TM_1).

7. Experiment

The purpose of this experiment is to demonstrate the performance of the automatic modelling and the robustness of the TFAs generated. The experiment consists of the recognition of hand gestures.

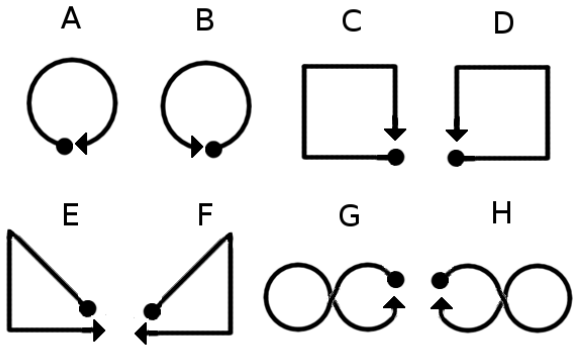

Figure 11: Gestures used to analyze the performance of the method

7.1. Data capturing

The gestures were captured using a triaxial accelerometer. The accelerometer module was connected by Bluetooth to a personal computer which made it possible for the subject to move freely. The data provided by the sensor consists of three acceleration signals: one for each axis (a_x, a_y, a_z). The values are measured in gravity units (g) in the range of [-6g,6g] encoded with 10 bits.

This vector is captured with a sampling rate of 100 Hz, which is fast enough for our purpose since the maximum frequency of hand gestures is about 10 Hz [23]. The set of eight different gestures are represented in figure 11. The beginning and end of each gesture is marked with a circle and an arrow, respectively. In this work, the dataset was recorded using one subject who held the sensor in a vertical orientation with the right hand. Twenty instances of each gesture were recorded in a random sequence, making a total of 160 (8x20) gesture instances. In order to test how our approach behaves in a realistic environment, the person moved around the room during 35 minutes in an unconstrained way while performing the experiments. A continuous sequence of motion and activities was carried out: sitting, standing up, reading books, opening drawers, ... during which the 8 gestures were performed at random instants. Note that these gestures present a high variability in amplitude and duration because they can be done at different speeds and the acceleration values depend on the speed. Although during the recognition stage our method is able to detect patterns without a previous segmentation stage, it is necessary to segment the signal in gestures for the training stage. For this, the subject who performed the gestures was asked to press a wireless button before starting the gesture and to release it after finishing it. Some gestures were bad segmented because sometimes the person forgot to release the button immediately. For this reason some gestures were segmented manually.

7.2. Learning TFAs

For each different gesture, we trained a TFA using several instances as training set. The inputs of these TFAs are the components of acceleration (a_y, a_z). We have not used the a_x component because it does not offer any additional information, since the gestures are bi-dimensional.

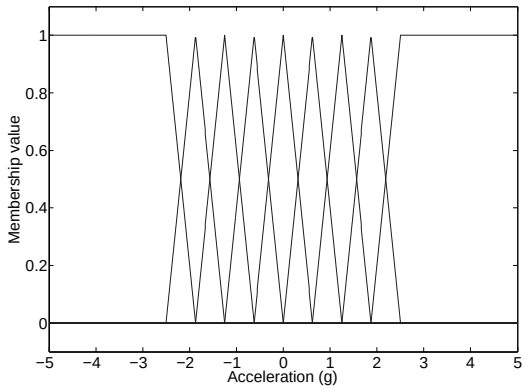

Figure 12: Linguistic labels used for acceleration components (a_y,a_z)

To describe each input, we have used 9 triangular membership functions (see

figure 12) except in the boundaries where we have used trapezoidal ones to cover the domain completely.

For the training phase some segmented instances of each gesture were chosen and the rest were left for testing. In order to reduce the dependence between results and gestures used for training, we have chosen the training instances randomly and repeated this process 20 times. To check the impact of the size of the training set in the performance of the TFAs, we have varied the size of the training set from 2 instances (10% of the available instances) to 10 (50%), giving a total of 180 (20x9) repetitions. For each repetition, we have passed the segmented testing instances to the TFA corresponding to each gesture class and we have counted how many instances were recognized by the TFA. In this test, we have fixed $\theta = 0.01$ and $\alpha = 0.68$, the decision of using this last value will be explained in the next subsection.

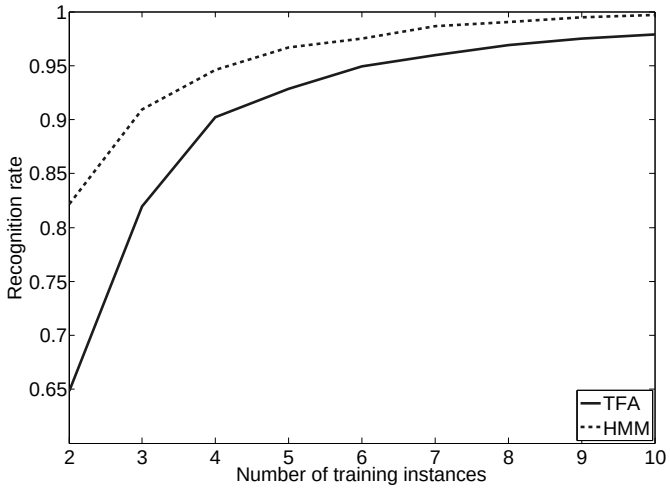

Figure 13: Recognition Rate for testing instances ($\alpha = 0.68, \theta = 0.01$)

In order to compare the results of our method with other existing methods, we have tested Hidden Markov Model (HMM) with the same data. HMM is a typical method used in temporal pattern recognition. However, both methods are conceptually different which makes it difficult to compare. HMM is a classification method, this means it can only decide the class of an instance based on the comparison of the results of the HMMs of each class. But the TFAs are recognizers and indicate if a gesture is recognized or not without any comparison. We present this comparison to give a broad view about the complexity of the problem but we should take into account the differences between both methods. The comparison was done as fair as possible using HMMs similar to our generated TFAs. These models are also left–right, the number of states of the HMM was 20 which is the average length of our TFAs. We used Gaus-

sian HMMs that generate a Gaussian distribution for each input and for each state. In this aspect HMM are less restricted than our TFA since the linguistic intervals produce the discretization of the input.

Figure 13 shows how the average recognition rate for the testing instances for both methods increases when the number of training instances used during learning stage also increases. It can be seen that the results of HMM are better than the TFA ones but we should take into account that classifying a gesture between several classes is easier than deciding if it belongs to any of them. Looking at the results we can say that at least 5 training instances (25% of available instances) are enough to obtain a recognition rate of over 90% using our method. Furthermore, figure 13 also shows that the recognition rate starts to converge after using more than 6 instances. Therefore for this experiment we can conclude that the best number of training instances is 5 or 6. Lastly, we did not include the figure of the recognition rate for the training instances because in all cases, independently of the size of the training set and the method, the recognition rate was close to 100%.

7.3. Analysis of continuous recognition

One of the advantages of our method is input synchronization, this means that a segmentation stage, to look for possible gestures, is only necessary during the training stage. The signal obtained from the sensor is provided sample by sample to all generated TFAs. Furthermore, the signal has not been filtered or normalized in order to check the robustness of our approach. As we explained before, the α parameter makes it possible to control the robustness of the method against noise. We have evaluated different values of α to test how this value affects the number of false negative and false positive gestures.

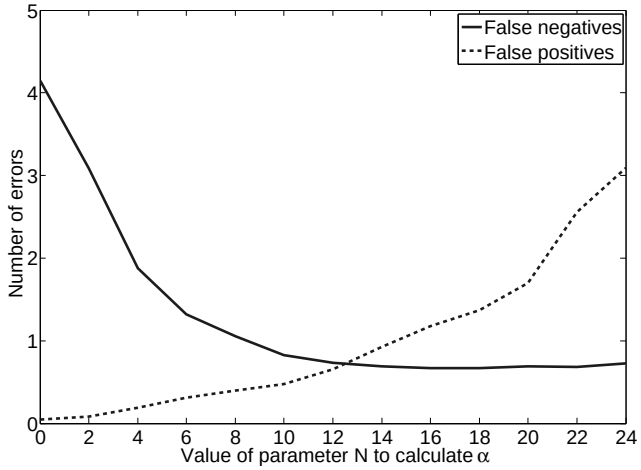

Figure 14: Number of false negatives and false positives with respect to the N parameter

The α value which allows the TFA to accept an error with a duration of N samples can be calculated using the equation 14. For this test, we have obtained the α values substituting N from 0 ($\alpha = 0$) to 24 ($\alpha = 0.83$). We have used 25% of the available instances as training set and we have repeated this process 20 times per gesture class for each tested α. Figure 14 plots α value against the number of false negatives made by the TFA as well as the number of false positives. Among the different values of α we choose the one where both lines cross, which corresponds with $N = 12$ ($\alpha = 0.68$) at this point the number of false negatives is equal to the number of false positives. This value is less than one instance which we consider a good result. On the one hand, one false negative instance is about 5% error. On the other hand, one false positive instance is a really low error, especially when taking into a account that the length of our dataset for continuous recognition is about half an hour and the typical length of a gesture is between 1 to 1.5 seconds.

8. Conclusions

In this paper, we have contributed to the pattern recognition field proposing a new tool called Temporal Fuzzy Automata. The new method uses fuzzy sets for defining the conditions imposed on the inputs. The result is an imprecise but robust model of the temporal pattern of the signal. The main contribution of this model is its capacity to include fuzzy conditions not only in the signal amplitude but also in the description of the signal temporal dimension. The obtained models are understandable and therefore can be checked easily by an expert. Furthermore, we have presented an algorithm that allows the automatic generation of TFAs from some instances of the pattern with a high recognition rate and few false positives.

Although, the recognition rate results of TFA are a little bit worse than the ones obtained with HMM, our method possesses some intrinsic features that HMM only can reach by including extra processing. First, each generated TFA is able to decide by itself when a signal belongs to it. However, in the case of HMM, it is necessary to compare the results of the models of the different classes to determine the class of the signal. Second, TFA does not need to segment the signal during the recognition process because it works in continuous recognition. HMMs usually work with isolated signals though they can also do continuous recognition. Nevertheless, this supposes an extra process to detect peaks in the occurrence probability and this can be considered a posterior segmentation. Third, the TFA rejects those signals that do not belong to any class because they do not fit in any model. HMM is a classification method and cannot reject the signal, it will classify it in one of their classes. Some works, e.g. [10], propose to introduce a new class called "filler model" which represents those signals which do not belong to any pattern. This model is learned with samples of non pattern signals but this is not an easy task because sometimes (like in our experiment) these signals can be as complex as the pattern to be recognized, and a HMM cannot capture the behavior of them all. Finally, our method provides

a measurement that quantifies the degree of matching of the different signals with the model.

References

[1] J. Astrain, González, J. Garitagoitia, Fuzzy automata with ϵ-moves compute fuzzy measures between strings, Fuzzy Sets and Systems 157 (11) (2006) 1550–1559.

[2] G. Bailador, D. Roggen, G. Tröester, G. Triviño, Real time gesture recognition using continuous time recurrent neural networks, in: 2nd International Conference on Body Area Networks, ACM, Florence, Italy, 2007, pp. 1–8.

[3] T. Bin Amin, I. Mahmood, Speech Recognition using Dynamic Time Warping, in: Int. Conf. on Advances in Space Technologies, 2008, IEEE, Islamabad, Pakistan, pp. 74–79.

[4] M. Doostfatemeh, S. C. Kremer, New directions in fuzzy automata, International Journal of Approximate Reasoning 38 (2) (2005) 175–214.

[5] M. Faundez–Zanuy, On–line signature recognition based on VQ–DTW, Pattern Recognition 40 (3) (2007) 981–992.

[6] S. Fernández, A. Graves, J. Schmidhuber, An application of recurrent neural networks to discriminative keyword spotting, in: Int. Conf. Artificial Neural Networks, 2007, pp. 220–229.

[7] C. L. Giles, C. W. Omlin, K. K. Thornber, Equivalence in knowledge representation: automata, recurrent neural networks, and dynamical fuzzy systems, Proceedings of the IEEE 87 (9) (1999) 1623–1640.

[8] K. Y. Huang, Syntactic Pattern Recognition for Seismic Oil Exploration, World Scientific Publishing Company, 2002.

[9] R. Kempf, J. Adamy, Sequential pattern recognition employing recurrent fuzzy systems, Fuzzy Sets and Systems 146 (3) (2004) 451–472.

[10] D. Kim, J. Song, D. Kim, Simultaneous gesture segmentation and recognition based on forward spotting accumulative HMMs, Pattern Recognition 40 (11) (2007) 3012–3026.

[11] G. J. Klir, B. Yuan, Fuzzy Sets and Fuzzy Logic: Theory and Applications, Prentice Hall PTR, 1995.

[12] A. Koski, M. Juhola, M. Meriste, Syntactic recognition of ECG signals by attributed finite automata, Pattern Recognition 28 (12) (1995) 1927–1940.

[13] J. N. Mordeson, D. S. Malik, Fuzzy Automata and Languages: Theory and Applications, Chapman & Hall/CRC, 2002.

[14] J. Moreno-García, J. J. Castro-Sánchez, L. Jiménez, A direct linguistic induction method for systems, Fuzzy Sets and Systems 146 (1) (2004) 79–96.

[15] T. Morimoto, S. Takahashi, Automatic Construction of FSA Language Model for Speech Recognition by FSA DP–Matching, in: O. Castillo L. Xu, S–l Ao (Eds.), Trends in Intelligent Systems and Computer Engineering, Springer, Berlin, 2008, pp. 515–524.

[16] W. Pedrycz, Why triangular membership functions?, Fuzzy Sets and Systems 64 (1994) 21–30.

[17] L. R. Rabiner, A tutorial on hidden markov models and selected applications in speech recognition, Proceedings of the IEEE 77 (2) (1989) 257–286.

[18] L. Reyneri, An introduction to fuzzy state automata, Lecture Notes in Computer Science (1997) 273–283.

[19] G. G. Rigatos, Fault detection and isolation based on fuzzy automata, Information Sciences 179 (12) (2009) 1893–1902.

[20] E. H. Ruspini, A new approach to clustering, Information and Control 15 (1) (1969) 22–32.

[21] F. Steimann, K. P. Adlassnig, Clinical monitoring with fuzzy automata, Fuzzy Sets and Systems 61 (1) (1994) 37–42.

[22] M. B. Tumer, Lee, K. M. Ropella, A syntactic methodology for automatic diagnosis by analysis of continuous time measurements using hierarchical signal representations, IEEE Trans. Systems, Man, and Cybernetics, Part B 33 (6) (2003) 951–965.

[23] C. Verplaetse, Inertial proprioceptive devices: self-motion-sensing toys and tools, IBM Systems Journal 35 (3) (1996) 639–650.

[24] J. Virant, N. Zimic, Fuzzy automata with fuzzy relief, IEEE Trans. Fuzzy Systems 3 (1) (1995) 69–74.

[25] D. Wang, Temporal pattern processing, in: The Handbook of Brain Theory and Neural Networks, MIT Press, Cambridge, MA, USA, 2003, pp. 1163–1167.

[26] W. G. Wee, K. S. Fu, A formulation of fuzzy automata and its application as a model of learning systems, IEEE Trans. Systems Science and Cybernetics 5 (3) (1969) 215–223.

[27] L. A. Zadeh, Fuzzy sets, Information and control 8 (3) (1965) 338–353.

[28] M. Zhou, M. Wong, A segment-wise time warping method for time scaling searching, Information Sciences 173 (1-3) (2005) 227–254.