A Discrete Hidden Markov Models Recognition Module for Temporal Series: Application to Real-Time 3D Hand Gestures.

Yannick Dennemont, Guillaume Bouyer, Samir Otmane, Malik Mallem

A Discrete Hidden Markov Models Recognition Module for Temporal Series: Application to Real-Time 3D Hand Gestures.

Yannick Dennemont*, Guillaume Bouyer*, Samir Otmane* and Malik Mallem *
* IBISC laboratory, Evry University
e-mail: FirstName.LastName@ibisc.fr

***Abstract*—This work studies, implements and evaluates a gestures recognition module based on discrete Hidden Markov Models. The module is implemented on Matlab and used from Virtools. It can be used with different inputs therefore serves different recognition purposes. We focus on the 3D positions, our devices common information, as inputs for gesture recognition. Experiments are realized with an infra-red tracked flystick. Finally, the recognition rate is more than 90% with a personalized learning base. Otherwise, the results are beyond 70%, for an evaluation of 8 users on a real time mini-game. The rates are basically 80% for simple gestures and 60% for complex ones.**

***Index Terms*—gesture recognition; virtual reality; temporal series classification; 3D position; context-awareness.**

I. Introduction

Gestures are a human communication basis and also used in almost all oral conversations. They have different purposes, patterns, and realizations. Gesture recognition is to acknowledge intended patterns from different realizations which vary both for an individual and between individuals. In the case of 3D interaction, gestures are felt easier and quicker rather than oral commands [Bolt, 1980] so they can enhance the immersion. Moreover this modality offers new interaction possibilities. Besides, by using known gesture metaphors from our daily life, the interaction cognitive load and learning difficulties can be lessen. Finally, real-time is a recognition constraint. Numerous methods have been applied to gestures recognition. Among them, Hidden Markov Models (HMMs) have been chosen as the module basis processes. They allow good recognition rates in real-time (compared to particular filters) and are naturally robust to time variation and incomplete measurements (compared to neural network). Module general uses are introduced in the section 2, before refocusing on gestures. Then the sections 3 and 4 present the recognition principle and its implementation. Afterwards, the sections 5 and 6 detail offline and online performances evaluation. Conclusion and perspectives are presented in the section 7.

II. Implementation ideas

Interaction with an application needs sensors to retrieve information from the users. The devices uses can be straightforward (e.g. mouse displacements) or can need processing to extract patterns from the data (e.g. gestures). This work focus on providing a module to recognize those patterns. Besides, different sensors can be used for each applications. Therewith the module implementation tends to be generic in order to be used by various applications and sensors (flystick, glove, haptic device, etc.). Practically, the module only needs a (time) series of (sampled) data. The sensors data are pre-processed (with at least a quantification) before the recognition itself. This leads to several opportunities:

- several sensors can be used, fused together or each specialized on specific recognitions;
- any available sensing devices can easily be used;
- Different entries lead to different recognition purposes.

The user's hand 3D positions are our only inputs in this paper. Indeed, they are usually available data and common to all our interaction devices. Thus, the module provides gestures recognition as patterns are searched in hand trajectory samples. However, assume other vectors with data like distance to the closest object etc. The same module now classifies situations (e.g. the current task: selection, manipulation or navigation) and more broadly can acquire contextual information on an activity recognition layer (Fig. 1). Reciprocally context can improve the recognition (e.g. different recognition models can be applied depending on the task). This is more generally part of the context-awareness problematic [Dennemont et al., 2012].

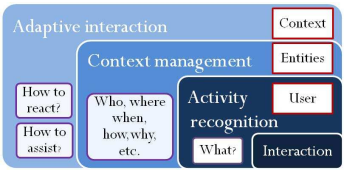

Fig. 1. Recognition (e.g. gesture) as a layer toward context-awareness

III. Recognition Principles

A. Hidden Markov Models

HMMs are successful mathematical structures for temporal or spatiotemporal series recognition. They have a Dynamic Time Warping natural capacity and are usually robust to measurements variations. Their good performances for speech recognition [Mitra and Acharya, 2007][R.Rabiner, 1989] has led to their application to signs language recognition [Starner and Pentland, 1995]. HMMs variations are also implemented, e.g. Input Ouput HMMs [Justa and Marcela, 2009]

where online measurements affect the model parameters, Coupled HMM [Brand et al., 96] coupling two HMMs, Semantic Network Model [Rajko et al., 2007],etc. They can be used combined to other models, e.g. Markov Model and Self Organizing Map [Caridakis et al., 2010]. The recognition rates vary in those works from 80% to 100% [le Martin, 2000][Mitra and Acharya, 2007]. A time-depending process can be represented by a Markov Model if the current state probability only depends on a finite number of precedent states, the model order. This is illustrated by the directed link between states. The links types define the model architecture Fig. 2. Architecture defines the progression inside the model and is crucial for good performances. A left-to-right architecture embeds an idea of succession (e.g. different steps in a gesture) and has usually better results with fewer states [Nianjun et al., 2003]. HMMs add a layer (Fig. 3). The current model state is no more directly observable. Instead, each state has observable symbols emission probabilities.

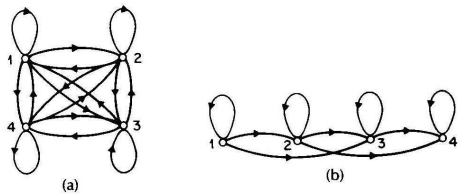

Fig. 2. Architecture examples [R.Rabiner, 1989]: a) fully ergodique with 4 states b) a 2nd order left-to-right with 4 states

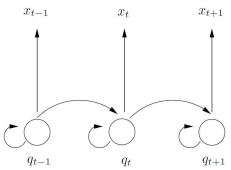

Fig. 3. HMM example [Justa and Marcela, 2009] with a 1st order left-to-right architecture : states q_i can be estimated through the observations x_i

Practically, HMMs are constituted of:

- Initial state probabilities;
- States with transition probabilities;
- Symbols emission probabilities for each state.

HMMs utilization has different issues:

- Evaluation: symbols sequence generation probability with a model (Baum-Welch algorithm).
- Learning: choose the model parameters to maximize sequences generation probability (Baum-Welch used here).
- Decryption: find the most probable states sequence that explain a symbols sequence (Viterbi algorithm).

B. How to acknowledge a gesture?

Recognition processes different symbols sequences' lengths (corresponding to different gestures lengths) from the same symbols flux (Fig 4). Each HMM should maximize (after learning) the likelihood of a sequence containing its gesture. The best likelihood corresponds to the best gesture candidate. Finally this likelihood is compared to a threshold. Indeed, a gesture presence in the flux is not known. Thus this can be assumed when gesture likelihood is greater than the noise likelihood. To obtain good results, the threshold is adaptive and partly managed by a HMM, trained to recognize noise.

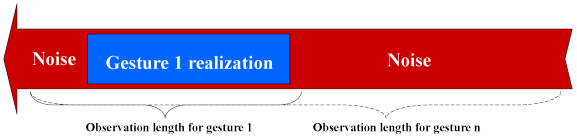

Fig. 4. Symbols flux after sensors pre-processing

C. Recognition algorithm

The module implements on Matlab the algorithm illustrated Fig.5. It allows quick prototyping and parameterization. The different process parts are detailed in the next section. After a successful recognition, a wait is applied. It gives times to users in order to return to a rest position, without detecting this needed movement.

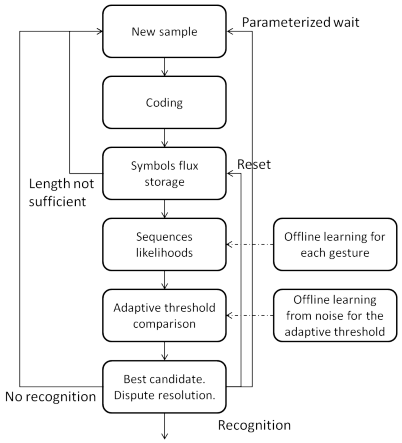

Fig. 5. Implemented algorithm

IV. Recognition module construction

A. Emitted symbol definition

Symbols must be built from sensors measurements. The information coding choice, including quantification, reduces the symbols alphabet and should increase the symbols efficiency. It also defines the set of gestures that can be recognised. For this set, the focus is the "trajectory pattern" and may allow some deformations or velocity differences, but no global orientation change (e.g. to distinguish left and right). This coding extracts movements directions information (Fig. 6). Our sensors supply the 3D positions. The first coding step is to get the velocity. Next the velocity orientation in two plans is quantified (1). Finally those 2D orientations are aggregated into one single symbol (2). This coding improves efficiency since the focus is now the movements. Using directly the position would have introduced rarely used symbols. Besides, symbols are now also independent of the initial position and scene scale. Finally, the recognition can also easily be limited to the movement 2D projection, by using directly $Symbol = Symbol_{xy}$ (1) [Caridakis et al., 2010], for intended planar gestures.

$$Symbol_{xy} = \frac{\lfloor \arctan(\frac{Velocity_y}{Velocity_x}) \rfloor}{\frac{2 * \Pi}{N_{xy}}} \quad (1)$$

$Symbole_{xy}$ and $Symbole_{yz}$ are combined to code the 3D direction in one symbol (2):

$$Symbol = (Symbol_{yz} - 1) \times N_{xy} + Symbol_{xy} \quad (2)$$

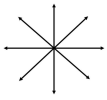

Fig. 6. Considered plan $[xy]$ direction orientations for $N_{xy} = 8$

A last "rest" symbol is added for no movement or a movement too small to be significant. Therefore an alphabet of $N_{xy} \times N_{yz} + 1$ symbols is constituted. Afterwards, $N_{xy} = N_{yz}$ is referred as the directions number.

By changing the coding function, other gestures sets can be defined. For example, coding the angle difference or the cross product between two successive sampled velocities respectively help to define rectilinear or circular gestures (without an exhaustive training base). The module allows each set of gestures to define its own coding function which also permits those different interpretations from the same physical captor.

B. States and architecture definition

States and architectures can be defined independently of the sensors. They have a large impact on the performance. Our first offline tests show that the 1st order left-to-right architecture for the gesture and fully ergodic for the noise give the better result. We obtain a performance rate of more than 90%. The same rate magnitude is obtained with two left-to-right architectures. But it becomes less stable depending on other parameter (states numbers etc.). However, the rate is 30% lower with two fully ergodic architecture. In fact, the first architecture combination is natural as a sequential order for gestures is assumed but no hypothesis is made for the noise. Our entire following tests (in the sections 5 and 6) are based on this combination. The states number is acquired by offline tests in the section 5. Best results are obtained when using 5 to 10 states.

C. Emission and transition probabilities learning

The models probabilities are obtained with the Baum-Welch algorithm and depend on the training sets. All training sets have been created by the same individual. Mixing individuals can improve the overall performance. However this hypothesis allows us to observe the models adaptations between individuals. Three examples sets have been made. The first two are constituted of 10 natural realizations of each gesture. The first one includes separated gesture used as the training set. The second is a single flux recording for offline experiments (section 5). This separation intends to get the best intrinsic model parameters and not an over learning. The last set contains 20 realizations of each gesture, which have now been carefully performed to fit ideal gestures idea. This results on lesser (offline) performances for the training set creator but better (online) performances with several users (section 6).

D. Threshold definition

The threshold (noted $Thld$) is the limit upon which gestures recognition is acknowledged. It cannot be a simple constant since sequence likelihoods diminish as their length (noted L) increase. Indeed each added symbol has a probability inferior or equal to 1. A first possibility is to use a uniform probability repartition to calculate the threshold likelihood of a sequence (noted $UniT$). Another possibility with better results is to use a specially trained HMM [Lee and Kim, 1999] (noted $HMMT$). This solution permits to include unwanted gestures with the noise. Adding and tuning a constant gain usually permit to double the recognition rate (3).

$$Thld(L) = HMMT(L) + gain \quad (3)$$

Two improvements can be done (4). First the gain value can be automatically defined to allow the best recognition for the gesture set (noted $AutoG$). Assume $MinMaxScore$ the least of all maximum scores (i.e. gesture likelihood minus the threshold likelihood, with no initial gain) for each gesture in a training set. To be recognized, a gesture must get a positive score:

- if $MinMaxScore < 0$, set the gain to: $1.1 \times MinMaxScore$. It allows the least probable gesture to become recognizable.
- if $MinMaxScore > 0$, set the gain to: $0.9 \times MinMaxScore$. It allows the least probable gesture to be just recognizable.

The second improvement is to use the uniform threshold to reduce the calculus cost. To avoid HMM threshold decryption for each gesture, we constitute K gesture classes (e.g. short and long gestures). Then we decrypt the HMM threshold only for K sequences; which lengths are their class training set's lengths mean. The lengths difference is compensated with the less costly uniformed gain. Adding the two improvements (4) gives a second expression for the threshold.

$$Thld(L) =_{sgn(L-K)} UniT(L) + HMMT(K) + AutoG \quad (4)$$

E. Gesture dispute management

Although the gain is adaptive, some gestures may just be incompatible. Indeed, it is the case when the module needs to recognize two nested gestures: e.g. the "Right" gesture (Fig. 7) is included in the "Z" gesture (Fig. 9). In this situation, the shorter gesture will always be the only one recognized. Those disputes depend on the gestures alphabet and must be declared (as a list of the names in dispute in a text file) so as to be managed by the module . The module then introduced a delay (the average lengths difference) before concluding recognition.

V. Offline Experimentation

A. Gestures alphabet

The entire tests and evaluation use a 6 gestures alphabet. It includes 4 simple gestures (”Up”, ”Left”, ”Down” and ”Right” Fig.7) and 2 complex gestures (”Left loop” Fig.8 and ”Z” Fig.9) with a dispute possibility.

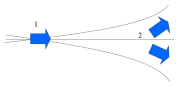

Fig. 7. ”Right”: a simple gesture with execution variations

Fig. 8. ”Left loop”: a complex gesture

Fig. 9. ”Z” in dispute with ”Right”

B. Offline experiments:parameters effects

Several tests are performed. A first series of offline tests confirm that the intuitive architecture combination basically supplied best performances (section 4.B). Then, a second series of offline tests (Fig. 10, Fig. 11) use one whole recording of 10 executions for each gesture. This symbols flux simulates a real situation as movements between gestures have been kept. The performance rate (5) includes false and missed recognition errors. It is expressed in percent and can be negative.

$$Performance = 1 - \frac{Total\ Errors\ Number}{Performed\ Gestures\ Number} \quad (5)$$

Tests are depicted on Fig.10 and Fig.11 as variations of the best early reference configuration which uses:

- 8 directions in each plan for quantification,
- 5 states for each HMMs,
- Each observation sequence length equals their maximum training set examples length.
- Threshold's gain as a parameter (3).

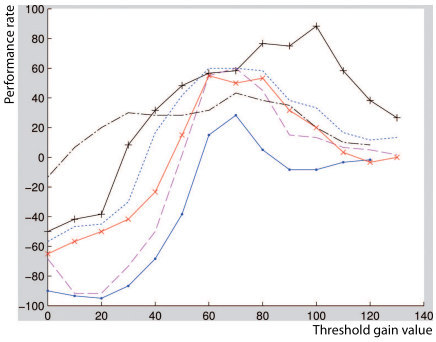

Fig. 10. Parameters effects on performance:
× reference configuration
− · − No rest symbol
· 3D→ 2D
+ 8→ 16 directions
−− Maximal length → average length
· · · Complex gestures: 5 states →10 states

Though the alphabet gestures are intended to be in a plan, Fig. 10 shows that reducing the recognition from 3D to 2D data drops the performances. Besides, the added ”rest” symbol is crucial to process efficaciously the continuous symbols flux. Indeed, it modifies the gesture perception and mostly permits to better acknowledge rests between gestures. Then, performances rise above 90% when increasing the number of directions until 32, Fig.10. Increasing the states number boosts performances (up to 10 states) and tends to broaden the curve around its maximum, thus to enhance performance stability. The performance augmentation is still obtained when the states number raise is restricted to longer gestures (natural since complex gestures are basically twice longer that simple gestures). Finally, the maximal length for observation achieves better results, but it depends on the (homogeneity of the) learning base (section 6). As a result, a median directions number between 16 and 32 provides better results. With too few directions, gestures are not well perceived. However, the performance decreases with too many directions. Indeed, the quantification step restrains the perceived variations and thus can help the recognition. Besides, too many directions can lead to have too specific examples in our small learning base. To sum up, the evaluation basis configuration is a 1st order left-to-right architecture for gestures' HMMs and a fully ergodic architecture for the threshold's HMM. Symbols are coded with 32 directions (in each plan) and use the full 3D data and the ”rest” symbol. Simple gestures' HMMs use 5 states whereas complex gestures' HMMS use 10 states.

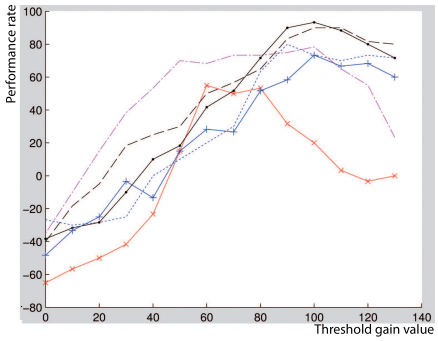

Fig. 11. Best offline configurations:
× reference configuration
· · · 8→ 32 directions
· 8→ 32 directions & Complex gestures: 5 states →10 states
−− 8→ 32 directions & 5 states →10 states
− · − 8→ 32 directions & Complex gestures: 5 states →15 states & Simple gestures: 5 states →10 states
+ 8→ 32 directions & complex gestures 5 states →10 states & maximal length → average length

VI. Online evaluation

A. Application presentation

The real-time evaluation is made on a Virtools mini-game scenario with a tracked flystick (Fig.12). A game evaluation has two advantages. First, it permits a real time test of the module with shared calculus capacities. Second, it also limits the tests tediousness that could lead to users' attention drop. The scenario uses the previously introduced alphabet (section 5.A).The scenario goal is to perform the good gesture to destroy orbs moving towards the user. Orbs can be on a cardinal point or in the screen center. The user should perform

a translation gesture toward an orb situated on a cardinal point to succeed. In case of a central orb, one of the complex gestures needs to be performed depending on the orbs' color ("Z" for blue orbs and "Left Loop" for red ones). Orbs generation is random and their average time on screen is 4s.

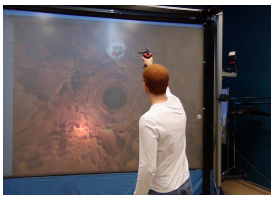
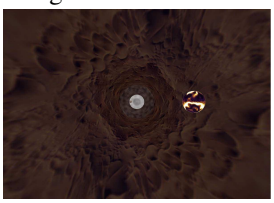

Fig. 12. Real-time evaluation scenario

B. Recognition frequency modification

The recognition module slows down in this evaluation and it induces performances decrease. To better reach real-time, the recognition check frequency and /or the data sampling frequency can be reduced. Initially, both are set to 50HZ. A supplementary test shows that offline performance drops of about 20% when the sampling frequency is divided by two. However, results stay basically the same with recognition check for each new sample, every 2 samples, or every 5 samples. But this can add a recognition delay up to 100ms.

C. Experimental setup

Users only know the game rules. They have not been informed of a delay for dispute gestures, the wait after a successful recognition etc. The results are obtained for 10 users which act to destroy 30 orbs at each try. Each try is a variation of the off-line best configuration (section 5):

- 50Hz theoretical recognition frequency. It induces slow-downs, reduced by using the average observation length.
- 10Hz recognition with the maximal observation length (offline best choice).
- 10Hz recognition with the average observation length and the ideal gestures learning base (section 4.C).

Each configuration is tested with 3 gain values, relative to the offline best value:

- A gain 33% inferior
- The offline best gain
- A gain 33% superior

Finally a new measurement is introduced (6). It focuses on users' success, i.e. performing good gestures in time. A test with a good success rate can get a poor performance rate.

$$Success = \frac{Good\ Detected\ Gestures\ Number}{Situations\ Number} \quad (6)$$

D. Results

The user who made the learning bases reaches performance and success rates of more than 90%.

The average successes fit a normal distribution (Lilliefors test result: $0.052 < 0.121$ for a 5% threshold). Each parameter effect is confirmed with an ANOVA test:

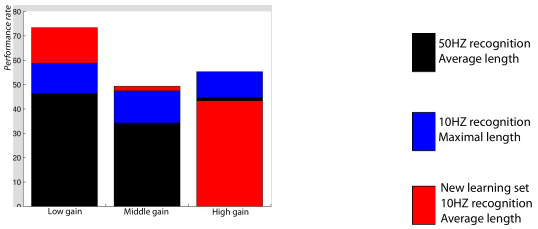

Fig. 13. Real-time scenario average performances (5)

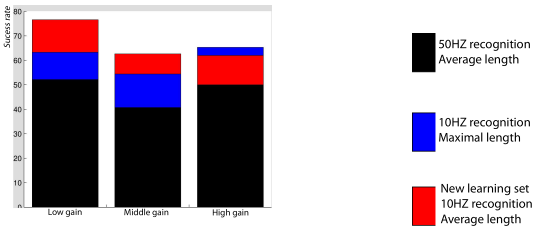

Fig. 14. Real-time scenario average successes (6)

- configuration ($P = 1.26 \times 10^{-2} F = 5.49$);
- gain ($P = 3.74 \times 10^{-5} F = 1.77$);
- gesture ($P = 7.71 \times 10^{-4} F = 6.76$).

When the user factor is kept, success not longer fit a normal distribution (Lilliefors test result: $0.181 > 0.0468$ for a 5% threshold). Assuming the robustness to this hypothesis violation, effects are confirmed with an ANOVA test:

- configuration ($P = 2.1 \times 10^{-3} F = 9.7$);
- gain ($P = 3.6 \times 10^{-3} F = 5.8$);
- gesture ($P = 7 \times 10^{-6} F = 6.7$);
- user ($P = 1.9 \times 10^{-11} F = 15$).

Parameters couples without the gain are interdependent:

- configuration/gesture ($P = 2.3 \times 10^{-3} F = 2.8$);
- configuration/user ($P = 2.5 \times 10^{-5} F = 2.1$);
- gesture/user ($P = 3.9 \times 10^{-10} F = 3.8$).

The parameters values offering the best result are sum up Fig. 15. The first configuration, theoretically with a 50Hz recognition frequency, has poor result around 50% due to slowdowns (Fig 13 and Fig. 14). When reducing the calculus frequency, the results rise of a little more than 10%. The new learning base (with fewer variations in the intended gestures) always reaches the best result but for the highest gain value. It increases the success rate of 15%. Thus, somehow letting the HMMs manage new encountered variations from other users is more efficient that trying to learn too much variations, at least from a single individual. The best configuration has now an admissible success rate at 75% and a performance rate slightly smaller. Average results are usually better with a smaller gain value than the one found offline. Indeed, it seems adequate to reduce the threshold when applying to other users a model trained by an individual. Results sometimes increase with the gain value. In fact, users tend to slowly adapt their natural gestures when they encounter several rejections. This users learning of which gestures are better recognized can explain some of the differences between offline and online tests.

Parameters	Best value	Reasons
Dimension	3D	Gestures are not fully planar
Rest symbol	On	Noise filter. Add to gestures dynamic and separations
Directions	32 in each plan	Details without over-fitting
Observation length	Average gesture length	Better for a heterogeneous learning base
States number	5 for short gestures, 10 for long gestures	Boost and stabilise the performances (in fact using $\simeq 5$ states per second of gesture)
Gain	33% less than the optimal off-line gain	help with the inter-individuals differences
Sampling Rate	50Hz	A good capture of the time-line execution
Recognition Rate	10 Hz	Ease the calculus strain
Learning base	Ideal gestures tries	Help with the inter-individuals differences

Fig. 15. Best parameters for the average performances

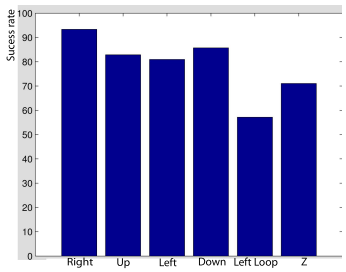

Fig. 16. Average successes by gestures for the low threshold gain and the new learning set

Simple gestures have a success rate greater than 80%. Complex gestures have lesser success around 60% (Fig. 16). Besides, "Z" gesture is better recognized than "Left loop". This can be partially explained because "Z" is a symbolic gesture known by all users. It is less subjective than the "Loop" idea. "Left Loop" is the only gesture to have a better success (70%) for the middle gain value. Probably since users can learn the gesture on their first try series and then improve on the second before the excessive threshold value.

VII. Conclusion

This work presents a recognition module applied to gesture. The module is based on discrete HMMs. 3D positions are the only sensors data used. A personalized training set allows performances above 90%. By using this training for other users, the real-time evaluation reaches admissible performance (up to 90% for simple gestures and 70% for complex ones). It validates both the real-time and a real situation issues for virtual reality. Indeed the users' displacements in front of the screen do not add false detection. All errors are due to intended gestures that are wrongly interpreted. The velocity direction coding reduces the speed and scale effect on gestures patterns. But for very slow or quick gestures, the results drop. The difficult part is that users also tend to accelerate when they have false rejections (more energetic gestures feel easier to be recognized). To lessen this drawback, several modifications can be done with their own disadvantages. The first is to change the coding by considering the full velocity vector (it adds training and observation lengths issues) or no symbols repetition (it adds more gestures confusions possibilities). Another solution is to manage the speed as contextual information (to apply different models or observation lengths for different speed magnitude). Besides the recognition would benefit from a specific threshold gain for each gesture. Finally, the recognition module only needs examples bases in order to be used. It can automatically adjust the HMMs (with a reference configuration where states number is proportional between bounds to the examples' lengths) and threshold models (the relative gap between automatic and optimal offline gains is usually less than 10%). For better results, the recognition module is easy to parametrize (architectures, states, quantifications, gain etc.) with an independent configuration text file. The coding can be personalized beyond the quantification as it can depend on the available sensors, their uses and the set of gestures wanted. This module can be used for any temporal series recognition and be a tool to supply contextual information without a semantic model. Yet, our first interest in this paper is its application to gesture recognition from 3D positions, which achieve good performances: between 70% for a shared training set and up to more than 90% for a personalized one.

References

[Bolt, 1980] Bolt, R. A. (1980). "put-that-there": Voice and gesture at the graphics interface.

[Brand et al., 96] Brand, M., Oliver, N., and Pentland, A. (96). Coupled hidden markov models for complex action recognition. Technical report, Massachusetts Institute of Technology Vision, and Modeling Group.

[Caridakis et al., 2010] Caridakis, G., Karpouzis, K., Drosopoulos, A., and Kollias, S. (2010). SOMM: Self organizing markov map for gesture recognition. *Pattern Recognition Letters*, 31.

[Dennemont et al., 2012] Dennemont, Y., Bouyer, G., Otmane, S., and Mallem, M. (2012). 3d interaction assistance through context-awareness: A Semantic Reasoning Engine for Classic Virtual Environment. In *Simulation and interaction in intelligent environments, VISIGRAPP 2012 special session*, pages 562–567.

[Justa and Marcela, 2009] Justa, A. and Marcela, S. (2009). A comparative study of two state-of-the-art sequence processing techniques for hand gesture recognition. *Elsevier*.

[le Martin, 2000] le Martin, J. (2000). *Reconnaissance de gestes en vision par ordinateur*. PhD thesis, INPG.

[Lee and Kim, 1999] Lee, H.-K. and Kim, J. H. (1999). An hmm-based threshold model approach for gesture recognition. *IEEE Transactions on pattern analysis and machine intelligence*, 21(10).

[Mitra and Acharya, 2007] Mitra, S. and Acharya, T. (2007). Recognition: A survey. *IEEE Transactions on system, man and cybernetics*, 37(3).

[Nianjun et al., 2003] Nianjun, L., C., L. B., and J., K. P. (2003). Evaluation of hmm training algorithms for letter hand gesture recognition. In *IEEE International Symposium on Signal Processing and Information Technology*.

[Rajko et al., 2007] Rajko, S., Qian, G., Ingalls, T., and James, J. (2007). Real-time gesture recognition with minimal training requirements and on-line learning. In *Computer Vision and Pattern Recognition (CVPR 2007)*.

[R.Rabiner, 1989] R.Rabiner, L. (1989). A tutorial on hidden markov model and selected application in speech recognition. In *IEEE Procedeeding*.

[Starner and Pentland, 1995] Starner, T. and Pentland, A. (1995). Real-time american sign language recognition from video using hidden markov model. In *ISCV*.