Gesture Modelling and Recognition by Integrating Declarative Models and Pattern Recognition Algorithms

Alessandro Carcangiu[1(✉)], Lucio Davide Spano[2], Giorgio Fumera[1], and Fabio Roli[1]

[1] Department of Electrical and Electronic Engineering, University of Cagliari, 09123 Cagliari, Italy
{alessandro.carcangiu,fumera,roli}@diee.unica.it
[2] Department of Mathematics and Computer Science, University of Cagliari, 09124 Cagliari, Italy
davide.spano@unica.it
http://pralab.diee.unica.it, http://people.unica.it/davidespano/

Abstract. Gesture recognition approaches based on computer vision and machine learning mainly focus on recognition accuracy and robustness. Research on user interface development focuses instead on the orthogonal problem of providing guidance for performing and discovering interactive gestures, through compositional approaches that provide information on gesture sub-parts. We make a first step toward combining the advantages of both approaches. We introduce DEICTIC, a compositional and declarative gesture description model which uses basic Hidden Markov Models (HMMs) to recognize meaningful pre-defined primitives (gesture sub-parts), and uses a composition of basic HMMs to recognize complex gestures. Preliminary empirical results show that DEICTIC exhibits a similar recognition performance as "monolithic" HMMs used in state-of-the-art vision-based approaches, retaining at the same time the advantages of declarative approaches.

Keywords: Gesture recognition · Hidden Markov Models · Compositional · Declarative

1 Introduction

Gesture recognition is a long-standing research topic in the computer vision field, with many applications to Human-Computer Interaction (HCI) [16]. Vision-based approaches to gesture recognition can be categorized into appearance- and 3D model- (or tracking-) based [2,17]. In particular, the recognition of dynamic gestures (as opposed to static ones, that do not include a temporal dimension) has been addressed using techniques that explicitly consider the temporal dimension, like Hidden Markov Models (HMM), Dynamic Time Warping (DTW), Time-Delay Neural Networks (TDNN) and Finite-State Machines

S. Battiato et al. (Eds.): ICIAP 2017, Part I, LNCS 10484, pp. 84–95, 2017.
https://doi.org/10.1007/978-3-319-68560-1_8

(FTM) [2,12,17], as well as traditional supervised classification algorithms like support vector machines (although they are more suited to static gestures [12]).

Vision-based gesture recognition poses a number of challenges, like coping with a large variety of gestures, and achieving invariance to lighting conditions, viewpoint changes, cluttered background and gesture speed; usually, a trade-off between accuracy, performance and usefulness has to be found, based on criteria like real-time processing capability and scalability [2,17]. Beside the above issues, under the viewpoint of user interface (UI) development it is very important to address the orthogonal problem of *usability*, which is related to the *meaning* of interactive gestures for users [15]: indeed, not all gestures that can be recognized by a machine have a meaning for the human counterpart. In particular, contrary to WIMP (Windows, Icons, Menus, Pointer device) interfaces, gestures are rarely self-revealing, and thus a guidance system for discovering what commands are available and how to trigger them can definitely improve their usability [3]. This implies that the underlying recognition system should be able to provide (through a graphical interface) *feedback* and *feedforward* information [22], i.e., information on which portion of a gesture has been completed, and on its potential completion, which may be more than one.

The two goals of an accurate/effective recognition and a usable gestural interface can be conflicting. Vision-based approaches usually provide a class label to a whole gesture pattern recognized in an input sequence, which is viewed as an atomic event even when the time dimension is internally taken into account (e.g., using HMMs). However, from the user point of view the performance of a gesture cannot be reduced to a single event, since it spans over a perceivable amount of time. On the other hand, compositional and/or declarative approaches have been proposed for modelling gestures, which explicitly take into account the subdivision of a gesture into meaningful sub-parts; however, to recognize sub-parts they rely on heuristic techniques that exhibit a lower effectiveness and robustness with respect to "monolithic" vision-based approaches. We survey the relevant literature on both approaches in Sect. 2.

In this paper we make a first step towards filling the gap between vision-based and compositional/declarative approaches. We start from the declarative and compositional gesture description model GestIT [20,21] (see Sect. 2.2), that solves the intermediate feedback problem and provides a superset of composition operators described in other approaches. We integrate GestIT with HMMs, using HMMs to recognize basic gesture segments ("primitives") instead of whole gestures (Sect. 3). We show that the resulting method, called DEICTIC (DEclaratIve and ComposiTional Input Classifier), is capable of recognizing complex gestures made up of several primitives, rigorously defined according to GestIT operator semantics. Preliminary empirical results (Sect. 4) provide evidence that DEICTIC exhibits a recognition performance comparable to that of standard HMM classifiers, while retaining the advantages of declarative approaches.

2 Related Work

In this section we overview first the vision-based approaches most relevant to our work, i.e., the ones which subdivide gestures into sub-parts, and then the main approaches based on declarative models, including GestIT.

2.1 Vision-Based Gesture Recognition Approaches

Vision-based methods that identify a set of sub-parts (or primitives) common to different gestures have already been proposed, either for increasing the recognition rate or to reduce the training set size in learning-based approaches. Primitives can be broadly defined as a set of distinguishable patterns from which either a whole movement or a part of it can be reconstructed. Different, specific definitions of "primitive" have been considered in the literature: they may represent basic movements (e.g., raising a leg, moving an arm to the left), static poses, or characteristic patterns of low-level signals like the Fast Fourier Transform. In the following we give representative examples for each interpretation of the primitive concept.

In [23] primitives are identified using a bottom-up clustering approach aimed at reducing the training set size and at improving the organisation of unlabeled datasets for speeding up its processing. Gestures are then labelled with sequences of primitives, which is close to a representation useful also for building UIs. However, since primitives are identified automatically, they are difficult to understand for designers while creating feedback and feed-forward systems. In [1] primitives are defined in a context-grammar established in advance using a top down approach, which is more suitable to UI designers; however, grammars were not created taking into account the gesture meaning from the user perspective.

In [13] primitives are used together a three-level HMM classifier architecture for recognizing (i) the primitives, (ii) their composition and (iii) the pose or gesture. However, also in this case unsupervised learning was used for defining both primitives and their composition, which is not suitable for building UIs.

A set of primitives that better suits the understanding by designers includes 3D properties of the movement trajectory. For instance, in [14] primitives identified in a 2D video are used for classifying 3D movements. Here the primitives are functions on the 2D features that represent the user's state. A representation more linked to geometric features on the 3D space for identifying primitives was proposed in [5]; however, it requires the understanding of the underlying mathematical representation, which is not feasible for UI designers.

To our knowledge, the vision-based approach most similar to ours is the one of [9]. It decomposes gestures into application-specific "primitive strokes", and uses a distinct HMM for modelling each stroke; each gesture is then modelled by a composite HMM obtained by concatenating the corresponding stroke models. This technique is valid for describing stroke sequences. Our approach is able to define more complex composite gestures, including alternative (choice) and parallel definitions. In addition, we do not use a re-training step for avoiding a degradation in recognition performance.

2.2 Gesture Recognition: The Declarative Modelling Approach

Declarative approaches allow splitting a gesture into several sub-components. There are different compositional approaches based on heuristic gesture recognition. For instance, Kammer et al. [7] introduced GeForMT, a language for formalizing multitouch gesture for filling the gap between the high-level complex gestures and the low-level device events. GeForMT uses an Extended Backus-Naur form grammar, with five basic movements (move, point, hold, line, circle and semicircle), which are composed through parallel and sequence operators.

A rule-based approach for multitouch gestures has been introduced in Midas [18]. The rules work on different features, for example the 2D positions, the speed and the finger tracking state and consists of two components: a prerequisite part and an action part. The first defines the input fact pattern to be recognized while the second the UI behaviour. Mudra [6] is a follow-up research from the same group extending Midas for multimodal interfaces. It unifies the input stream coming from different devices, exploiting different modalities. Designers define both the low-level handling events and the high-levels rules, combining them into a single software architecture. Khandkar et al. proposed GDL [8] (Gesture Description Language), a domain-specific language designed to streamline the process of defining gestures. GDL separates the gesture recognition code from the definition of UI behaviour. This work defines three components: the gesture name, the code for the gesture validation and a return type. The last component represents the data notified with a callback to the application logic.

More structured and expressive declarative methods are Proton++ [11] and GestIT [20,21]. Proton++ separates the temporal sequencing of the event from the code which describes the UI behaviour. It also allows developers to declaratively describe custom gestures through regular expressions, using the operators of concatenation, alternation and Kleene's star. A regular expression if defined by a triplet: (i) the event type, (ii) the touch identifier, (iii) the interface item hit by the touch; An improved version of the framework [10] included means for calculating a set of attributes that may be associated to an expression literal.

In GestIT [20,21], gestures are modelled through expressions that define their temporal evolution, obtained by composing two main elements: ground and composite terms. A ground term is the smallest block for defining a gesture: it describes an atomic event which cannot be further decomposed. In general, it is associated to a value change of a *feature*, such as the pixel coordinates of a touch on the screen or the position and rotation of a skeleton joint. Composite terms are used for defining more complex gestures through a set of operators. We will use them in the rest of this work, since they are a superset of those included in Proton++ [21]. Considering two gestures g and h (either ground or composite terms): $g*$ is the iteration of g; $g \gg h$ is the sequence that connects g with h; $g \parallel h$ defines that g and h are performed in parallel; $g[\,]h$ is the choice between either g or h; $g[> h$ disables the iteration of g by performing h. Due to space limits, for further details about GestIT we refer the reader to [20,21]. Its main drawback is the heuristic recognition approach for ground terms, which do not guarantee a good recognition accuracy. We try to solve this problem in this work.

3 Combining the Two Approaches with DEICTIC

In defining DEICTIC, our main goal is to make a first step toward filling the gap between machine learning approaches and declarative description methods, combining the advantages for UI developers in providing gesture sub-parts notification (which is a feature of GestIT), together with the high recognition accuracy and the robustness to input noise offered by learning-based recognition approaches like HMMs. We focus in particular on stroke gestures, which may be segmented into sequences of basic components (e.g., points, lines or arcs).

To describe a gesture, DEICTIC uses the approach of GestIT [20,21]: ground terms are defined first, then they are combined through temporal operators for describing more complex ones. The key feature of DEICTIC is that each ground term is recognised using a distinct, "basic" HMM, trained on a set of examples; the same, basic HMMs are then used in different gesture recognizers, in the same way a line following a given direction may be used in more than one trajectory. In particular, more complex strokes are described through the combination of basic HMMs into a composite one, whose topology is defined according to the semantics of the GestIT composition operators (see below). This allows DEICTIC to provide information on the recognition of each single operand inside the composition. Moreover, contrary to a similar approach like the one of [9] (see Sect. 2.1), the composite HMM of DEICTIC does not require additional training with respect to the basic HMMs, providing additional temporal relationships between primitives besides the sequence. In the following we explain how to create basic HMMs for ground terms, and the proposed algorithms to define the topology of composite HMMs.

Ground Terms. For basic HMMs we use the left-to-right (or Bakis) topology, which is the most commonly used one for recognizing simple gestures like lines or arcs. It requires to specify the number of states, whereas the probability distributions of both transitions and observations may be learned from a dataset. We point out that in DEICTIC training data are needed only for ground terms.

Iterative Operator. Given a HMM trained to recognize a gesture g, the iterative operator allows recognizing the same gesture an indefinite number of times. Assuming that the starting state of g is s_0 and the ending state is g_f, the HMM for g^* is defined by adding a transition from all states in the backward star of s_f (represented in red in Fig. 1(a) to all states in the forward of s_0 (in green in Fig. 1(a). This creates a loop in the topology, while no changes are made to the probability distributions.

Sequence Operator. For recognizing a sequence of gestures in a specified order, we use the sequence operator. Given two HMMs trained to recognize respectively gesture g and h, an HMM that recognises the sequence $g \gg h$ is obtained by connecting the backward star of the ending state in g with the forward star of starting state in h. Such operation is depicted in Fig. 1(b). It guarantees that $g \gg h$ has only one starting and one ending state. Since the two HMMs may use a different set of features, the observations of the composed one are obtained by the union of all features considered by both g and h. In other words, the

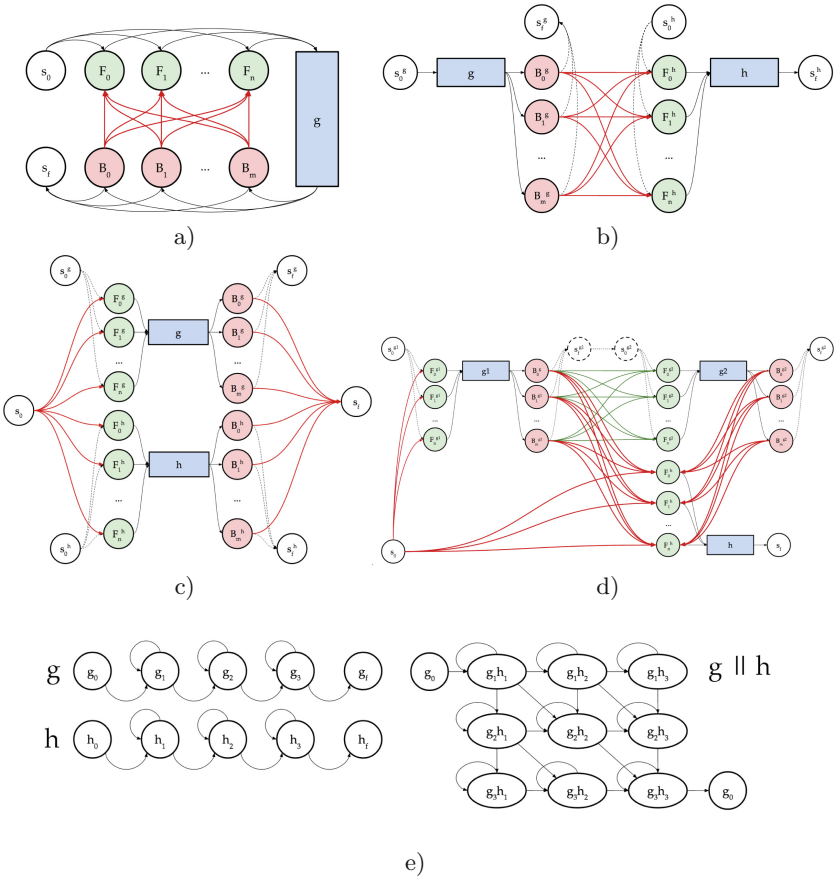


Fig. 1. Composite HMM topologies: (a) iterative, (b) sequence, (c) choice, (d) disabling, (e) parallel. We denote with g and h the gestures HMMs to be composed. (Color figure online)

composite model must specify an emission probability for each feature and for each state.

Choice Operator. A choice between two gestures, denoted by $g[]h$, allows recognizing either g or h. We obtain the corresponding HMM starting from HMMs trained to recognize g and h, and putting them in two separate recognition lines. No transition between the states of g and h is added. The composite HMM has one starting and one ending state. They are linked respectively with the forward star of the original starting state and the backward star of the original ending state in both g and h. All these transitions are equally likely. The choice topology is shown in Fig. 1(c).

Disabling Operator. In defines a gesture that stops the recognition of another one. In general, we use it for stopping an iteration loop. Given two gestures g and

h, the HMM that composes them through the disabling operator is obtained by inserting a set of transitions that represent a short-cut from each ground term contained in g to the starting state in h. Each link models the possibility for the user to stop the execution of g at any time performing h: before starting each one of the ground terms contained in g the model contains a possible transition to h that blocks its recognition. However, since a HMM has a single starting and a single final state, the real HMM topology is more complicated: one needs to consider also the forward star of the starting state and the backward star of the ending state in each ground term contained in g. The schema is shown in Fig. 1(d). In order to maintain the sum of the probabilities towards each state in the HMM equal to 1, we split the original transition likelihood among all involved arcs (both the old and the new ones). We apply the same completion procedure for the observation probability vector we used for the sequence operator.

Parallel Operator. The parallel operator defines the simultaneous performance of two or more independent gestures. Give two gestures g and h, the composite HMM for $g \parallel h$ has a state for each pair (s_g, s_h) where s_g is a state in g and s_h is a state in h. The new HMM represents all the possible combinations of states in g and h. Therefore, we add a transition between two states in the parallel HMM, only if the transition is valid both in g and in h. More precisely, given two states in the parallel HMM (s_i^g, s_j^h) and (s_x^g, s_y^h), we add a transition between them only if the transition from s_i^g to s_x^g exists in g and the transition from $s_i^h j$ to s_y^h exists in h. The observable values of $g \parallel h$ are the concatenation of those in g and h, and are independent from each other.

We finally point out again that, using the proposed approach, composite HMMs need not to be re-trained.

4 Experiments and Preliminary Results

We implemented the above composition algorithms using the Python Pomegranate HMM library [19]. For our first experiments on DEICTIC, we used a data set containing 60 repetitions of 10 stroke gestures, performed by 14 different people. The input sequences consist of the position of the tip of the user's dominant hand forefinger, tracked using a Leap Motion sensor. Our dateset contains the following gestures: swipe left ←, swipe right →, V, caret ⋀, left square bracket [, right square bracket], X, delete ⌫, triangle △ and rectangle □.

We used *only one* ground term for describing such gestures, a left-to-right movement on the horizontal axis. In order to recognize such ground term, we created a Bakis HMM with six states, whose observation vector is composed by two normal distributions, one for the x and one for y coordinate of the finger/hand position. We then collected 14 training examples (separated from the gesture dataset) which, after a normalization and resampling step, were used for estimating the parameters of the ground term HMM. We then "cloned" such HMM and applied geometric transformations to the x and y distributions in the observation vector, such as scaling, translation and rotation, in order to represent different segments in a normalised 2D plane. In order to define the

expressions for the considered gesture set, we used a cardinal direction notation and the x and y coordinates in brackets for representing the starting point (in the list below we always consider the origin, but in the gesture definitions they are positioned between 0 and 1 in both axes). They represent a geometrically transformed version of the same ground term HMM:

- $e(0,0)$, the original term, without any transformation
- $n(0,0)$, the original term rotated 90°
- $s(0,0)$, the original term rotated −90°
- $w(0,0)$, the original term rotated 180°
- $ne(0,0)$, the original term rotated 45°
- $se(0,0)$, the original term rotated −45°
- $nw(0,0)$, the original term rotated 135°
- $sw(0,0)$, the original term rotated −135°
- $nw60(0,0)$, the original term rotated 120°
- $sw60(0,0)$, the original term rotated −120°.

Table 1 shows the modelling expression for all gestures in our dataset. The third column shows the recognition rate for the DEICTIC HMMs, directly fed with samples in the dataset, without additional training besides the original ground term. For comparison, we also trained an ad-hoc HMM for each gesture type defined using the Bakis topology, and performed a 10-fold cross validation for each gesture sample of the same type. We consider these results as an upper limit for DEICTIC, since optimizing the HMM on real samples allows to better adapt the transition probabilities and emission distributions.

The recognition rates of DEICTIC and of the ad hoc HMM are reported in Table 1. The confusion matrix for DEICTIC is shown in Table 2. In summary, the recognition rates are similar, which provides a first evidence that DEICTIC does not significantly degrade the recognition performance with respect to the state-of-the-art HMM classification approach. We had only two errors on our dataset. In the first one, the classifier confused a delete gesture with a triangle. The sample had a small cross (see sample in Table 2), thus it was really similar to a triangle. In the second error (a rectangle confused again with a triangle), the sample had some initial noise that resembled a triangle.

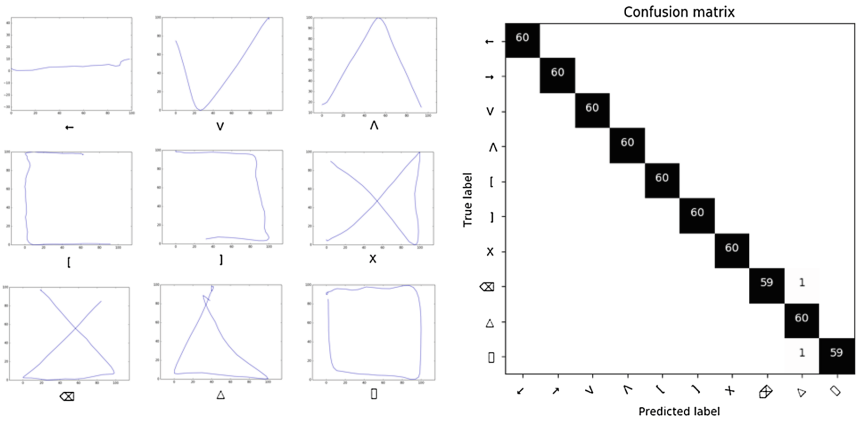
In contrast, DEICTIC exhibits several advantages, in particular considering the development of gestural UIs. First, DEICTIC was able to recognize new gestures, significantly different from the samples included in the training set of each ground term. This is important for UI designers, who would be able to create gesture recognizers exploiting existing components, as they already do with UI widgets.

Second, DEICTIC allows the reconstruction of the most likely sequence of ground terms associated to a particular gestural input, using the Viterbi algorithm [4]. Indeed, by construction, each state in a composite HMM is associated to a single ground term, for each considered stroke (except for parallel gestures, that consider more than one stroke). Such information is not trivial when gestures are composed in choice or parallel, since the designer would have the

Table 1. Recognition rate comparison between HMM defined through DEICTIC and trained ad-hoc (HMM column).

Gesture	Model	DEICTIC	HMM
←	$w(1,0)$	100%	100%
→	$w(0,0)$	100%	100%
V	$se(0,1) \gg se(0.5,0)$	100%	100%
⋀	$ne(0,0) \gg se(0.5,1)$	100%	100%
[	$w(1,1) \gg s(0,1) \gg e(0,0)$	100%	100%
]	$e(0,0) \gg n(1,0) \gg w(1,1)$	100%	100%
X	$ne(0,0) \gg s(1,1) \gg nw(1,0)$	100%	100%
⌫	$se(0,1) \gg w(1,0) \gg nw(0,0)$	98.34%	98.34%
△	$sw60(0.5,1) \gg e(0,0) \gg nw60(1,0)$	100%	100%
□	$s(0,1) \gg e(0,0) \gg n(1,0) \gg w(1,1)$	98.34%	100%

Table 2. Samples gestures included in the dataset and confusion matrix for the recognition using DEICTIC.



possibility to associate different feedback and feed-forward reactions to different ground terms. Such level of granularity is not supported by ad-hoc trained HMMs.

To further test all composition operators, we also considered a set of synthetic sequences produced as follows. First, we randomly grouped the gestures in a set of 5 pairs; then, for each pair we created a set of 14 sequences that should be recognized by the composition of two gestures, using the sequence, disabling and parallel operator. For creating the sequence samples, we simply concatenated those of the first gesture with those of the second one. For creating the disabling samples, we supposed to perform iteratively the first gesture, which should be blocked by the second one. Therefore, we randomly repeated the samples for the first gestures a random number of times between 3 and 5,

concatenating the result with a sample of the second gesture. For the parallel operator, we juxtaposed the samples of both gestures, randomly shifting up or down the rows of the second gesture and filling the blanks with random values. The latter operation guarantees that the gestures may start at different times. We then built one composite HMM for the sequence with the disabling operator, and one for the parallel operator. Similarly to the previous experiment, beside using DEICTIC we also trained an ad-hoc HMM for each gesture category, and evaluated the recognition performance using the leave-one-out technique. Table 3 shows that also in this case our compositional approach did not introduce a sensible degradation of the recognition rate.

Table 3. Recognition rates for syntethic sequences.

	DEICTIC			**HMM**		
Gesture	$\gg$	$[>$	$\parallel$	$\gg$	$[>$	$\parallel$
$\bigwedge$, ⌫	100%	100%	92.86%	92.86%	100%	92.86%
$\leftarrow$, □	100%	100%	100%	100%	100%	92.86%
$\rightarrow$, [	100%	100%	100%	100%	100%	100%
] , △	84.62%	84.62%	100%	100%	100%	100%
V, X	100%	100%	100%	100%	100%	100%

5 Conclusions

We proposed DEICTIC, a declarative and compositional description model for interactive gestures, based on the composition of a set of basic gesture sub-parts (ground terms, or primitives) through a set of operators. We use HMMs, a state-of-the-art technique in vision-based approaches, to recognize ground terms; we then combine such "basic" HMMs into composite HMMs, according to the operators, to describe and recognize complex gestures, retaining at the same time the inspection capabilities on gesture sub-parts needed for providing feedback and feed-forward in user interfaces. The main contribution of our work is the definition of algorithms for defining the composite HMM topology according to the composition semantics of complex gestures, without requiring additional training for the resulting HMM with respect to the basic ones. Preliminary empirical evidence shows that our approach is a promising direction toward filling the gap between the higher recognition accuracy and robustness achieved by vision- and learning-based approaches, and the capability of providing information on meaningful gesture sub-parts exhibited by compositional approaches, which is very useful for user interface design. The main limitation of DEICTIC to be addressed in future work is that the number of states of the composite HMM grows linearly (for the sequence, disabling and choice operators) or quadratically (for the parallel operator) with respect to basic HMMs.

Acknowledgements. This work has been supported by the project D3P2, Sardinia Regional Government (CUP code F72F16002830002) with the funds of Regional Law 7/07, year 2015, "Capitale Umano ad Alta Qualificazione".

References

1. Chen, Q., Georganas, N.D., Petriu, E.M.: Real-time vision-based hand gesture recognition using haar-like features. In: Proceedings of IMTC 2007, pp. 1–6. IEEE (2007)
2. Cheng, H., Yang, L., Liu, Z.: Survey on 3D hand gesture recognition. IEEE Trans. Circuits Syst. Video Technol. **26**(9), 1659–1673 (2016)
3. Delamare, W., Coutrix, C., Nigay, L.: Designing guiding systems for gesture-based interaction. In: Proceedings of EICS 2015, pp. 44–53. ACM (2015)
4. Forney, G.D.: The Viterbi algorithm. Proc. IEEE **61**(3), 268–278 (1973)
5. Holte, M.B., Moeslund, T.B.: View invariant gesture recognition using 3D motion primitives. In: Proceedings of ICASSP 2008, pp. 797–800. IEEE (2008)
6. Hoste, L., Dumas, B., Signer, B.: Mudra: a unified multimodal interaction framework. In: Proceedings of ICMI 2011, pp. 97–104. ACM, New York (2011)
7. Kammer, D., Wojdziak, J., Keck, M., Groh, R., Taranko, S.: Towards a formalization of multi-touch gestures. In: Proceedings of ITS 2010, pp. 49–58. ACM, New York (2010)
8. Khandkar, S.H., Maurer, F.: A domain specific language to define gestures for multi-touch applications. In: Proceedings of DSM 2010, pp. 2:1–2:6. ACM, New York (2010)
9. Kim, I., Chien, S.: Analysis of 3D hand trajectory gestures using stroke-based composite hidden Markov models. Appl. Intell. **15**(2), 131–143 (2001)
10. Kin, K., Hartmann, B., DeRose, T., Agrawala, M.: Proton++: a customizable declarative multitouch framework. In: Proceedings of UIST 2012, pp. 477–486. ACM Press, Berkeley (2012)
11. Kin, K., Hartmann, B., DeRose, T., Agrawala, M.: Proton: multitouch gestures as regular expressions. In: Proceedings of CHI 2012, pp. 2885–2894. ACM Press, Austin (2012)
12. Mitra, S., Acharya, T.: Gesture recognition: a survey. IEEE Trans. Syst. Man Cybern. Part C **37**(3), 311–324 (2007)
13. Natarajan, P., Nevatia, R.: Online, real-time tracking and recognition of human actions. In: Proceedings of WMVC 2008, pp. 1–8. IEEE (2008)
14. Natarajan, P., Singh, V.K., Nevatia, R.: Learning 3D action models from a few 2D videos for view invariant action recognition. In: Proceedings of CVPR 2010, pp. 2006–2013. IEEE (2010)
15. Norman, D.A.: Natural user interfaces are not natural. Interactions **17**(3), 6–10 (2010)
16. Pavlovic, V., Sharma, R., Huang, T.S.: Visual interpretation of hand gestures for human-computer interaction: a review. IEEE Trans. Pattern Anal. Mach. Intell. **19**(7), 677–695 (1997)
17. Rautaray, S.S., Agrawal, A.: Vision based hand gesture recognition for human computer interaction: a survey. Artif. Intell. Rev. **43**(1), 1–54 (2015)
18. Scholliers, C., Hoste, L., Signer, B., De Meuter, W.: Midas: a declarative multitouch interaction framework. In: Proceedings of TEI 2011, pp. 49–56. ACM, New York (2011)

19. Schreiber, J.: Pomegranate (2016). https://github.com/jmschrei/pomegranate
20. Spano, L.D., Cisternino, A., Paternò, F.: A compositional model for gesture definition. In: Winckler, M., Forbrig, P., Bernhaupt, R. (eds.) HCSE 2012. LNCS, vol. 7623, pp. 34–52. Springer, Heidelberg (2012). doi:10.1007/978-3-642-34347-6_3
21. Spano, L.D., Cisternino, A., Paternò, F., Fenu, G.: GestIT: a declarative and compositional framework for multiplatform gesture definition. In: Proceedings of EICS 2013, pp. 187–196. ACM (2013)
22. Vermeulen, J., Luyten, K., van den Hoven, E., Coninx, K.: Crossing the bridge over Norman's gulf of execution: revealing feedforward's true identity. In: Proceedings CHI 2013, pp. 1931–1940. ACM (2013)
23. Yang, Y., Saleemi, I., Shah, M.: Discovering motion primitives for unsupervised grouping and one-shot learning of human actions, gestures, and expressions. IEEE Trans. Pattern Anal. Mach. Intell. **35**(7), 1635–1648 (2013)