A Declarative and Classifier Gesture Recognition Method for Creating an Effective Feedback and Feedforward System

Alessandro Carcangiu

Deptment of Electrical and Electronic Engineering,
University of Cagliari, Cagliari 09123 Italy

Abstract. For recognizing gestures in an interactive application, we could apply different approaches, which offer many advantages and disadvantages. On the one hand, Machine Learning techniques guarantee a high accuracy and robustness to noise, but they do support the creation of effective feedback and feed-forward systems since they do not provide intermediate information. On the other hand, we can find in the literature declarative and compositional methods, which allow gesture sub-parts identification at the cost of a lower recognition rate. Therefore, one of the most common problems in gesture interface development is how to set the trade-off between a high precision in recognition and the support for user's guidance. With this mind, the main goal of my Ph.D. is finding a way for filling the gap between these two approaches, bridging the gap between Machine Learning and declarative and compositional approaches.

1 Introduction

In recent years, gesture interaction gained importance in many interactive settings, such as e.g., houses, offices and hospitals. Nowadays, smartphones, tablets, laptop and desktop computers are equipped with gesture input devices, which complement more standard techniques like pointing and typing. In general, we define an interactive gesture as a body movement that recognized through motion sensing input device. The tracking capabilities influence different aspect in the interaction design. For instance different devices track different body parts: fingertips (touchscreens), hands and fingers (*Leap Motion* [12], *Intel RealSense* [7]), arms or legs (*Kinect* [10]).

Despite their spread, window toolkits provide little support for managing gestures in interface development: they allow either to define high-level events or to track synchronously the raw data coming from the device. This increases the development complexity and, in many cases, decreases the usability of the resulting interface. A gesture is not an atomic event like a mouse-click: it usually requires different movements and may last for many seconds, which is a perceivable timespan for a user. During that time, s/he user needs guidance in completing the movement and s/he should be informed about the system interpretation of the input. For these reasons, developers need to provide users

with advices on gesture performance during the interaction: i) which movements were recognized by the application (defined as feedback) and ii) the possible interface state(s) reachable after the user finishes the current gesture (defined as feedforward) [20].

While introducing feedback and feedforward in gesture interfaces has positive effects on the user experience (UX) [20], developers have technical difficulties in building such systems. On the one hand, the interface needs a high recognition accuracy. This problem was addressed through different machine learning methods, like Hidden Markov Models (HMM), Dynamic Time Warping (DTW), Time-Delay Neural Networks (TDNN) and Finite-State Machines (FTM) [11,14,5], support vector machines [11]. All methods demonstrated a high recognition accuracy and they are suitable for recognizing complex gestures. However, they do not support composition, they need many examples in the training phase and they usually provide a single class label for the whole gesture, which is represented as an atomic event even if the time dimension is internally taken into account by the recognition approach. On the other hand, we can find in the literature different methods for describing the temporal evolution of a gesture through composition, for instance *GestIT* [18,19] and *Proton++* [9,8]. They declaratively define a gesture through a set of composed sub-parts, allowing their identification during the performance. The drawback in these methods is the accuracy: the sub-part identification relies on heuristics which not are as precise as the classification approaches. With this mind, we have proposed *DEICTIC*[2], which integrates the declarative and compositional gesture description model *GestIT* [18,19] with HMMs and uses HMMs in order to recognize basic gesture segments (or primitives) instead of whole gestures.

2 Related Work

In this section we will briefly summarise the state of the art for the approaches we are trying to bridge.

2.1 Machine Learning-based Approaches

Computer Vision-based approaches have just tried to identify gesture sub-parts in order to: i) reduce the size of the training dataset and ii) improve their performances. Chen et al. [4] define primitives 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 [21], Yang et al. identified primitives by 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. A gesture is labelled and defined with sequences of primitives. This representation at first may be useful for building UIs. Unfortunately, the automatic identification of primitives usually lead to a set of basic gestures that is difficult to understand

for designers, and this decreases their applicability for creating feedback and feed-forward systems.

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.

2.2 Compositional and Declarative-based Approaches

As mentioned earlier, declarative approaches allow splitting a gesture into several sub-components and there are different compositional approaches based on heuristic gesture recognition. An example could be *GestIT* [18,19] and Proton++ [9,8]. The latter, is a multi-touch framework, which allows describing declaratively custom gestures, separating the temporal sequencing of the events from the related code to the user interface behaviour. It also allows developers to describe custom gestures in a declarative way, through regular expressions and using the operators of concatenation, alternation and Kleene's star. Developers could define a regular expression by a triplet which is composed by an event type, a touch identifier and the interface item hit by the touch.

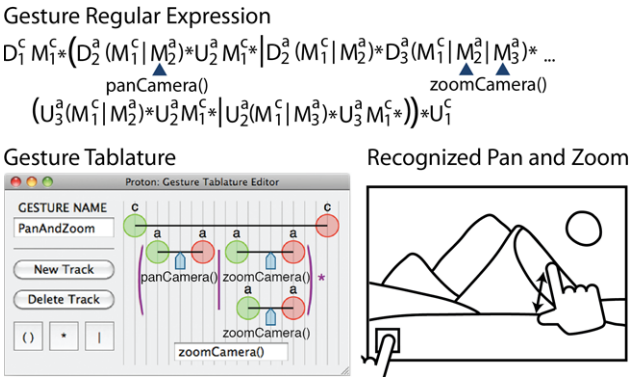

Fig. 1. The figure shows the editor tool for Proton [9,8] and an example of a generated expression which describes a gesture

In *GestIT* [18,19] gestures are modelled through expressions that define their temporal evolution. The expressions are obtained by using two main elements: i) ground term or primitive, which describes the smallest part of a gesture and in general is associated to a fundamental movement which may considered atomic by users, ii) and composite terms, which represent a set of operators allowing to link ground terms between them or other composite terms, in order to define more complex gesture. To describe the set of operators consider two gesture g and h (either ground or composite terms): $g*$ is the continue iteration of g; $g \gg h$

defines the sequence that connects g with h, firstly the user performs g then h; $g \parallel h$ defines that g and h are performed in parallel simultaneously; $g[\,]h$ is the choice between either g or h; $g[> h$ disables the iteration of g by performing h; $g \,|=|\, h$ the connected gesture can be performed in any order (e.g. first doing g and then h or vice versa).

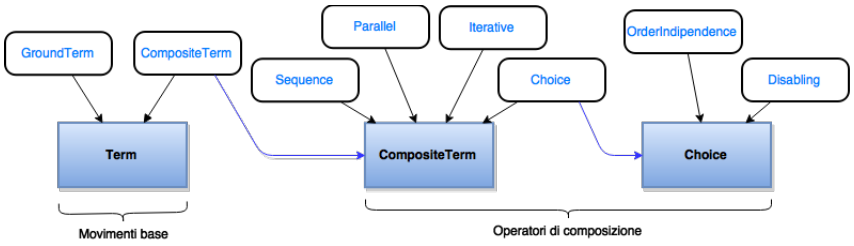

Fig. 2. Term operators hierarchy from *GestIT* [18,19].

Another example is Midas [15] that introduced a rule-based approach for multitouch gestures. These rules work on different features, for example the 2D positions, the finger tracking state or its speed and consists of two components: a prerequisite part and an action part.

2.3 Feedback and Feedforward

As we discussed in previous sections, feedback and feedforward systems were developed to help user during the interaction, in particular for inexperienced users who interface with specialised machines which have a low affordance or in situations where the users lack familiarity with the interface. These systems help users in different ways, for example showing what will be the result of a particular input or explaining how to complete a certain operation.

In gesture interfaces feedback is used to show which portion of the gesture has been recognized and what the system understood from the user's movements. These suggestions can help the users to understand what movements they are doing and, eventually, to repair both recognition or execution errors. Feedforward helps users during the interaction differently i.e., showing the effects on the UI of the possible ways for concluding the current gesture, thus guiding the user towards the desired effect. An effective visualization of such guidance has been described by Bau and Mackay [1], through a dynamic guide that combines feedback and feedforward to help users to learn, execute and remember gesture sets. Their approach shows a graphic representation of the gesture part that has been correctly recognized and, at the same time, a representation for all possible paths that may be followed for completing a correct interaction.

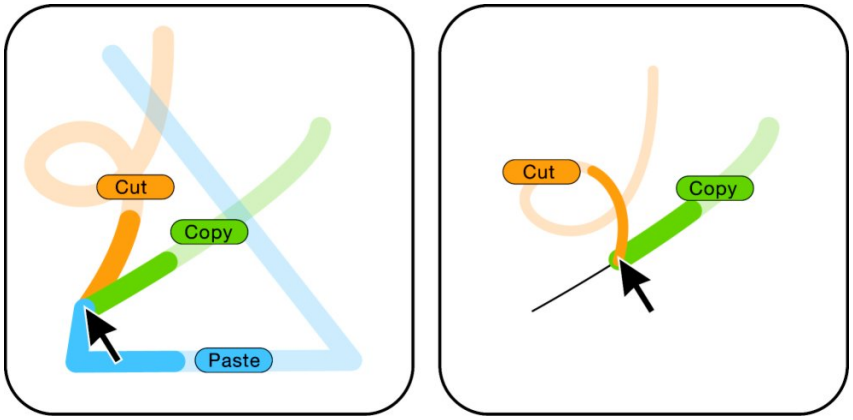

Fig. 3. In this image we can see an example of feedback and feed-forward system proposed by *OctoPocus* [1].

LightGuide [17] proposes a similar approach in a 3D setting, showing feedback and feedforward directly on the body parts which user should move to perform a gesture. This approach using a set of projector and Kinects to show the suggestions.

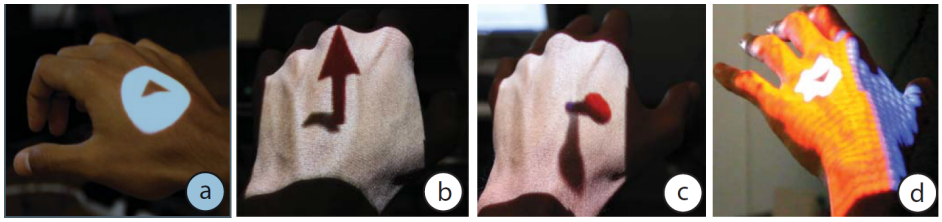

Fig. 4. This figure shows an overview of the range of 3D cues that *LightGuide* [17] creates to help guide a users movement. In (a), a user is shown a 2D arrow with a circle that moves in the horizontal plane, (b) shows a 3D arrow, (c) a 3D path where blue indicates the movement trajectory and (d) uses positive and negative spatial coloring with an arrow on the users hand to indicate depth.

Schwarz et al. [16] proposed another model which is a general architecture with the goal of proving support for continuous feedback about uncertainty. Their work is based on prior work in modelling uncertainty using Monte Carlo and tracks multiple interfaces; indeed, this architecture shows all these possible new system states for each sequence of input that the user may have intended. It works reducing the number of possible interfaces (according to the user input), combining and showing these in a single interface. Summarising, the model shows to user which will be the new program state basing on user input and the most

likely states whether there are some similar actions. In addition, when the user completes the task, the architecture allows user to return to the previous state or to select another state.

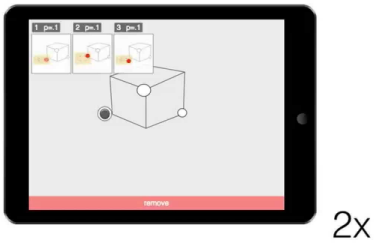

Fig. 5. An example of feedback system for a tablet app proposed by Schwarz et al. [16]

3 DEICTIC

As we said in previous section, definitively *DEICTIC* we combined the declarative and compositional gesture description model *GestIT* [18,19] with HMMs in order to integrates the declarative advantages with the accuracy and the robustness to input offered by classifiers. DEICTIC, similarly to GestIT, defines a gesture by using ground and composite terms. In our approach, a primitive movement is recognized by a single left-to-right trained HMM while the composite operators are obtained by combining the HMMs of their operands. The result of composition is a new HMM which provides information during the recognition and recognizes the gesture according to the temporal semantics. The two follow paragraphs will explain how DEICTC uses these components.

3.1 Ground Terms.

A basic movement in DEICTIC is described by a single HMM which uses a left-to-right (or Bakis [6]) topology. For defining a "basic" HMM, we must define his number of states and train the HMM for learning the probability distributions of both transitions and observations from a set of correct performance samples.

In DEICTIC we defined three 2D primitives:

1. **Point** which defines the starting position of the stroke. It is defined specifying the x and y coordinates and its notations is $P(x, y)$;
2. **Line** defines a linear movement of a specified offset in the x and y axes starting from the current position. It could be used to describe a vertical, horizontal or diagonal movement and we represent a line with the notation $L(\Delta x, \Delta y)$;

3. **Arc** specifies a quarter of a circle starting from the current position and finishing at the specified offset. The arc could follow a clockwise or counter-clockwise direction and we represent it with the notation $A_{\circlearrowright}(\Delta x, \Delta y)$ (clockwise direction) or $A_{\circlearrowleft}(\Delta x, \Delta y)$ (counter-clockwise direction).

3.2 Composite Terms

DEICTIC allows defining complex gesture by using five GestIT composite terms: *iteration*, *sequence*, *choice*, *disabling* and *parallel* (we do not use *order independence* because it could be derived from the other operators). These composite terms are generated in DEICTIC starting from the HMM of their operands automatically without requiring a new training step for the generated HMM. We explain in the following list how the composition is accomplished for each operator.

- **Sequence**. Consider a generic stroke g, its subcomponents and their HMMs. The HMM for recognizing g is obtained by connecting the end state of its first subcomponent with the start state of the second component and so on. The number of states of the new HMM is made up of the sum of the numbers of states of each model.
- **Iterative** The HMM for recognizing the iteration of the generic stroke g^* is obtained, starting from the basic HMM, by adding a transition from all states that are connected with the ending state to all states that are connected with the start state. In this way, we obtain a loop in the topology without changes in the probability distributions. Obviously, the number of states of the composed HMM will be the same of the basic HMM.
- **Choice**. Consider now the generic strokes g. Now, suppose the possibility to describe g through two different primitive sequences: g' and g''. The choice between the two descriptions is obtained putting the original HMMs in two separate recognition lines, without any transition between their states. The only contact points are the starting and the final states of the new HMM. The HMM obtained, once again, has a number of states which growns linearly.
- **Disabling**. Consider the case of two generic strokes g and h and the disabling operator. In DEICTIC the disabling operator for example $g^*[> h$, and its HMM are obtained in two steps: first, we add a transition between the starting states of each ground term in g^* stroke to the starting state of h; second, a link is added from the ending state of g^* to the starting state of h. This is necessary for supporting the case when the user performs the first gesture entirely before starting h. For maintaining the outgoing probability property, we split the original transitions likelihood among all involved arcs.
- **Parallel**. Finally, consider the possibility to do in parallel two different strokes, g and h that are totally independent in their transitions. The composite HMM for $g \parallel h$ will contain a state for each pair ($state_g$,$state_h$), that are linked if only the transition is valid in both starting models. For example, if we are in the state (g_1, h_1), we can: i) go forward on h, e.g. g_1, h_2; ii) go forward on g, g_2, h_1 or iii) go forward on both, g_2, h_2). The hmm obtained

has a number of states equal to $n * m$, where n is the number of states of Z and m the number of states of $\square$. Since the two gestures are independent the transition probabilities of the composite HMM are given from the product of probabilities of each gesture.

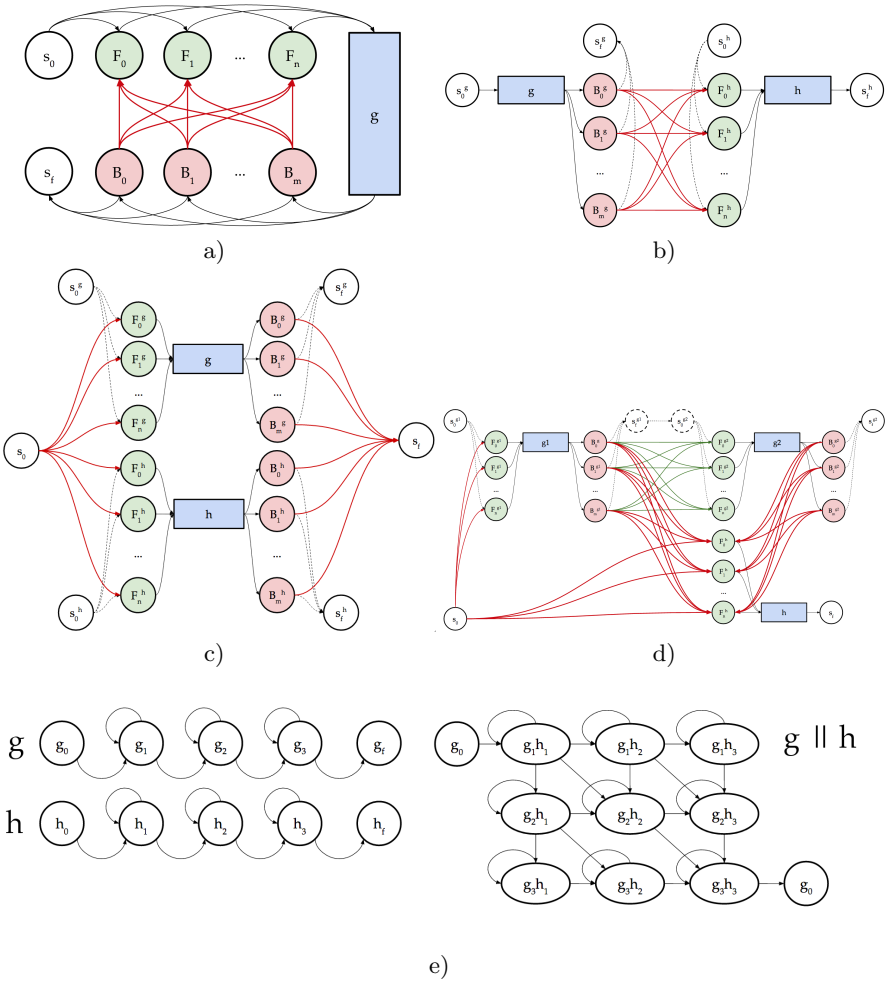

Fig. 6. Composite HMM topologies: a) iterative, b) sequence, c) choice, d) disabling, e) parallel. We denote with g and h the gestures HMMs to be composed.

3.3 Experiments and Results

In the test phase, our main goal is to prove the efficiency and the efficacy of our approach recognizing gestures. First of all, we would prove if DEICTIC is usable

in a feedback and feed-forward systems. Second, we would done a comparison between DEICTIC and machine learning-based approach about accuracy rate. For doing it, we choose ad-hoc HMM that is a hmm trained with the whole gesture.
Taking this into consideration, we tested DEICTIC with a dataset created by ourself and just used on our previous work about DEICTIC [3]. It is made up by 10 gestures strokes with 60 samples for each one. They are performed by 14 different people and recorded through *Leap Motion* device.
In the preliminary test we used a subset of these gesture. The hmms created with DEICTIC are trained only if they are used for recognizing basic movement; conversely, the ad-hoc HMMs are trained with the whole gesture and we evaluated the recognition performance using the leave-one-out technique. The 1 shows their description model and the recognition rate results obtained with DEICTIC and ad-hoc HMMs.

Gesture	**Model**	**DEICTIC**	**HMM**
$\leftarrow$	$Point(0,0) \gg Line(-1,0)$	100%	100%
$\rightarrow$	$Point(0,0) \gg Line(1,0)$	100%	100%
V	$Point(2,2) \gg Line(1,-2) \gg Line(1,2)$	100%	100%
$\bigwedge$	$Point(0,0) \gg Line(1,2) \gg Line(1,-2)$	100%	100%
$\leftarrow$	$Point(0,0) \gg Line(2,2) \gg Line(0,-2) \gg Line(-4,2)$	98.34%	98.34%
$\triangle$	$Point(0,2) \gg Line(-1,-2) \gg Line(2,0) \gg Line(-1,2)$	100%	100%
$\square$	$Point(0,1) \gg Line(0,-1) \gg Line(2,0) \gg Line(0,1) \gg Line(-1,0)$	98.34%	100%

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

4 Conclusion and Future Works

Summarising, DEICTIC is much accurate than the declarative methods and allows automatic generation of composed HMM. In addition, taking advantage of HMM, DEICTIC predicts the continuation of a gesture and permits sub-part identification helping developers to create feedback and feed-forward systems. Last but not least, the HMMs obtained from composition do not require additional training with respect to the basic HMMs; it should be noted that, for a particular gesture, the time asked to train and compose its basic HMMs is much less than is required to define and train its ad-hoc HMM. However our approach has some limits. First of all it does not support on-line recognition e has not the same accuracy of ad-hoc HMM. Another problem is the number of

states of composed HMMs: it tends to increase or linearly in sequence, choice and disabling case or quadratically in parallel case.
In order to improve DEICTIC, we have thought to substitute HMMs trying other machine learning-based method. We have looked for an approach which manages sequential data, supports on-line recognition and preserves a high precision rate. With this mind, we have started to study neural networks, in particular the Time Delay Neural Network, or *TDNN*. Now we do not know whether this method could be useful for us and we think this doctoral consortium is the best opportunity to obtain advices and suggestions.

At the same time, we are working on: i) defining an extensive set of information which reports all the point that gesture recognizer methods must expose in order to build effective feedback and feed-forward systems; ii) creating a set of guidelines for helping developers to select the appropriate classification technique according to the information needed by the feedback and feed-forward system at hand; iii) using DEICTIC in a real-time scenarios and testing its features building a feedback and feed-forward systems.

5 University Doctoral Program Context

I am Alessandro Carcangiu, a Phd student at the department of electrical and electronic engineering of the University of Cagliari. My tutors are Giorgio Fumera, Fabio Roli (from the same department) and Lucio Davide Spano (from the Dep. of Mathematics and Computer Science). In October 2015 I have started my Phd and I planed to defend my thesis in October 2018.

References

1. Bau, O., Mackay, W.E.: Octopocus: a dynamic guide for learning gesture-based command sets. In: Proceedings of the 21st annual ACM symposium on User interface software and technology. pp. 37–46. ACM (2008)
2. Carcangiu, A.: Gesture recognition through declarative and classifier approach. In: Proceedings of the 22nd International Conference on Intelligent User Interfaces Companion. pp. 185–188. ACM (2017)
3. Carcangiu, A., Spano, L.D., Fumera, G., Roli, F.: Gesture modelling and recognition by integrating declarative models and pattern recognition algorithms. In: Proceedings of the 22nd International Conference on Intelligent User Interfaces Companion. pp. 185–188. ACM (2017)
4. 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)
5. Cheng, H., Yang, L., Liu, Z.: Survey on 3d hand gesture recognition. IEEE Trans. Circuits Syst. Video Techn. 26(9), 1659–1673 (2016)
6. Elliott, R.J., Aggoun, L., Moore, J.B.: Hidden Markov models: estimation and control, vol. 29. Springer Science & Business Media (2008)
7. Intel: Intel realse. `http://www.intel.com/content/www/us/en/architecture-and-technology/realsense-overview.html`

8. 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, California, USA (2012)
9. 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, Texas, USA (2012)
10. Microsoft: Kinect. `http://www.xbox.com/en-CA/en-EN/xbox-one/accessories/kinect`
11. Mitra, S., Acharya, T.: Gesture recognition: A survey. IEEE Trans. Systems, Man, and Cybernetics, Part C 37(3), 311–324 (2007)
12. Motion, L.: Leap motion. `https://www.leapmotion.com/`
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. Rautaray, S.S., Agrawal, A.: Vision based hand gesture recognition for human computer interaction: a survey. Artif. Intell. Rev. 43(1), 1–54 (2015)
15. Scholliers, C., Hoste, L., Signer, B., De Meuter, W.: Midas: a declarative multitouch interaction framework. In: Proceedings of TEI 2011. pp. 49–56. TEI '11, ACM, New York, NY, USA (2011)
16. Schwarz, J., Mankoff, J., Hudson, S.E.: An architecture for generating interactive feedback in probabilistic user interfaces. In: Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems. pp. 2545–2554. ACM (2015)
17. Sodhi, R., Benko, H., Wilson, A.: Lightguide: projected visualizations for hand movement guidance. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. pp. 179–188. ACM (2012)
18. Spano, L.D., Cisternino, A., Paternò, F.: A Compositional Model for Gesture Definition. In: Proceedings of HCSE 2012. pp. 34–52. Springer (2012)
19. 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)
20. 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)
21. Yang, Y., Saleemi, I., Shah, M.: Discovering motion primitives for unsupervised grouping and one-shot learning of human actions, gestures, and expressions. IEEE transactions on pattern analysis and machine intelligence 35(7), 1635–1648 (2013)