Improving Recognizers' Performance by Leveraging the Continuity of Gesture Sequence

Leave Authors Anonymous
for Submission
City, Country
e-mail address

Leave Authors Anonymous
for Submission
City, Country
e-mail address

Leave Authors Anonymous
for Submission
City, Country
e-mail address

ABSTRACT

There is an increased opportunity for using gestures made by pens, fingers, arms, or other path-making instruments to implement more intuitive and effective interfaces. However, the range of applying gesture interaction paradigm is limited by its poor recognition performance and high memory burden on users. In this paper, we propose an algorithm framework that using a dynamic Bayesian network (DBN) integrated with a partially observable markov decision process (POMDP) to improve the performance of gesture recognition by leveraging the continuity of gesture sequence during the interaction process. In a experiment, we evaluated the improvements brought by our approach for recognizers in a pen-based interface. The results show that the framework enhances the recognition accuracy and reduces memory requirement for users by allowing them to conduct equally complex interactions with a smaller gesture set.

ACM Classification Keywords

H.5.m. Information Interfaces and Presentation (e.g. HCI): Miscellaneous; See http://acm.org/about/class/1998/ for the full list of ACM classifiers. This section is required.

Author Keywords

Gesture recognition, Continuity of gesture sequence, Dynamic Bayesian network, Partially observable Markov decision process

INTRODUCTION

Gestures-based interface is changing our daily uses of computing devices. With superior recognition technologies, gestures can provide natural and effective ways to express words, symbols, tables, formulas and drawings [12]. With the development of pens, fingers, arms, or other path-making instruments, there is an increased opportunity for using gestures in a wider range of interaction scenarios [19, 7, 14]. Nevertheless, the application of this interaction paradigm is still limited by the poor recognition performance in complex tasks and high memory burden on users in large set of gestures [8].

IUI'19, March 17–20, 2019, Los Angeles, CA, USA

ISBN 123-4567-24-567/08/06…$15.00

DOI: http://dx.doi.org/10.475/123_4

There are many existing gesture recognition techniques such as $1 recognizer [19] and DTW recognizer [5], which are proved to have high accuracy and efficiency in HCI researches. However, there are also significant drawbacks of these methods. First, these methods can hardly distinguish similar gestures since the similarity measures in these techniques are highly depend on gesture template. Second, these methods bring high memory burden on users. During the interaction, the user often confuses the shapes of similar gestures, or forgets the associated command of the gesture especially in a large set of gestures. Further, there are even more misrecognitions were observed in realistic scenarios, because users are often required to perform a series of gestures or finish gestures in limited time.

To address these problems, we proposes an algorithm framework that improves the performance of gesture recognition by leveraging the continuity of gesture sequence during the interaction process. When a user performs a series of gestures, the order of the gestures and potential system responds of each gesture may provide us valuable information to assist the recognition. We use a dynamic Bayesian network (DBN) [4] integrated with a policy of partially observable Markov decision process (POMDP) [16] to estimate the most possible user intention from the current interaction context. We evaluated the improvements brought by the framework in $1 and DTW with a pen-based interface. The results show that the framework enhances the recognition accuracy of the two algorithms by 6.3 % and 11.9 %, respectively. By using a smaller gesture set, the framework achieves superior recognition accuracy from the two algorithms with the complete gesture set by 5.7 % and 13.0 %, respectively.

RELATED WORK

Gesture recognition is one of the hottest topic in artificial intelligence (AI) and pattern matching. Numerous approaches have been introduced, including Hidden Markov Models (HMMs) [15, 3], Nearest-Neighbor (NN) [2] and, recently, deep learning [18]. Other gesture classification approaches, such as Gesture Coder [10], Gesture Studio [11], and Proton [6], provide developers with tools to implement gesture recognition by means of demonstration and declaration. These methods have been used extensively in domains ranging from on-line handwriting recognition to off-line gesture design. However, many of these methods are left wanting for the uses in user interfaces [19]. Some must be trained with large data set, like HMMs and deep learning approaches, others require high program-

ming skills to train, build and debug, making designers hard to understand the very meanings of gestures they designed.

In the contrast, the Euclidean distance computing the similarity among gesture templates provide a more interpretative compare to the above sophisticated methods. These methods are simple, fast and are widely applied in gesture user interfaces. These methods include Rubine [13], $1 [19], DTW [5], $N [1], and $P [17]. $1 [19] is a cheap gesture recognizer and usable almost anywhere in about 100 lines of code with simple 4 steps including resample, rotate, scale-transfer, and find best score. DTW [5] implemented in dynamic programming method is computationally expensive and flexible in matching gesture templates with different length. In addtion, some extensions like SHARK2 [7] becomes one of the most popular gesture technique for text input in the world.

However, the above algorithm only pay attention on how to use the information of current input gesture to improve the recognition accuracy with better training methods or geometric metrics, the user intention which is partially observable and dynamical changing is omitted. During a interaction process, there is a strong logic between each of operations performed by the user. Studies showed that the continuity of gesture sequence can be used to improve the recognition rate by dynamic probability model [4]. In addition, it is possible that intelligent systems could learn strategies from the interactive environment including both of users' behaviors and system actions to enhance the recognizer [20].

HYPOTHESES

Inspired by the findings of previous studies, we had the following hypotheses on improving the performance of existing recognizers in realistic interactive scenarios:

H1:*The performance of recognizers can be improved by introducing the continuity of gesture sequence.*

Continuous gesture interaction process in user interfaces can be viewed as a series of decision-making behaviors of human [4], and a the decision-making process can be enhanced by intelligent systems that support planning under uncertainty [20]. Thus, if we found a appropriate way to infer user intention and plan for a most profitable system action, the interaction efficiency of our interface should be improved.

H2:*The interaction efficiency can be enhanced by using a smaller gesture set.*

Larger gesture set not only have higher chance to contain visually similar gestures [9], but also higher memory burden on users. Thus, we assume that if our framework can achieve equal performance using a smaller gesture set compare to a larger one, then it should reduce the memory burden on users.

ALGORITHM FRAMEWORK

Based on the above hypotheses, we proposed an algorithm framework that infers user's intention by considering three parts of information including input gesture, prior user intention and rewards of possible system actions, to improve the interaction efficiency in gesture user interfaces.

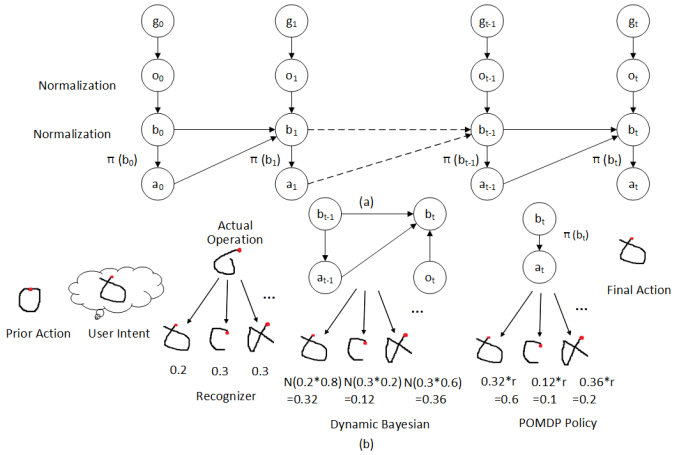


Figure 1. The whole process of the algorithm framework

The whole process of the framework is show in figure 1a. For initialization (time 0), we set the recognition result of a kernel recognizer as the user's intention s_0 as well as the correspond system action a_0. For the following input gestures, our algorithm framework determines the final system action through three steps. Firstly, we use the kernel recognizer to provide the probabilities by comparing the input gesture with the predefined gestures to obtain an observation o_t for the following steps. Secondly, we use the DBN to adjust this probability to the condition probability $b(s_t)$ of the user's intention s_t when observing the last condition probability $b(s_{t-1})$ and last system action a_{t-1}. Finally, we determine the final action a_t of the system to the user's input by maximizing a reward function that considers all possible system responds with the policy of POMDP. Figure 1b shows a simple example of how the framework works. We describe the three key parts of the framework as follow.

Kernel Recognizer

For each input gesture in time t, the framework requires a initial statistical observation o_t for the input gesture. This statistical observation inputs with raw gesture data and outputs with normalized probabilities of how likely the input gesture is one in the predefined gestures. Technically, any gesture recognizer that only rely on the information of current input can be set as the kernel recognizer in our framework. We choose the following two well-known gesture recognizers as the kernel recognizers:

$1 Recognizer

$1 recognizer is a geometric template matcher, it compares the input stroke to gesture templates with the closest gesture distances in a 2-D euclidean space [19], which fix our algorithm framework.

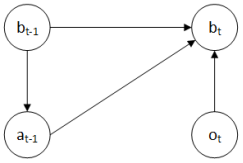
DTW Recognizer

DTW recognizer is a simple and effective method commonly used in speech recognition. The algorithm uses dynamic programming technique to solve the matching problem between two samples with different lengths [5]. We select DTW as it

compares gesture templates by building an adjacency matrix and looking for the shortest path.

Dynamic Bayesian Confidence Inference

The framework provides a stronger prediction for the current user intention by considering users' last behavior. DBN extends Bayesian network by relating states to each other over adjacent time steps. The value of a state at time t can be calculated from the internal regressors and the immediate prior state (time t-1) [4]. We use DBN to estimate the user's intention s_t at time t which depends on prior user intention s_{t-1}, the prior system action a_{t-1} and current observation o_t. Since the user intention is an abstract concept, we use the equation $b(s_t) = p(s_t|o_t, a_{t-1}, b_{t-1})$ to calculate the belief of the user intention $b(s_t)$ at time t. In which the term $p(s_t|s_{t-1}, a_{t-1})$ is a transfer matrix obtained from user data. Figure 2 demonstrates the network and associated formulation in this step.



$$b(s_t) = p(s_t|o_t, a_{t-1}, b_{t-1}) = p(o_t|s_t) \sum_{s_{t-1}} p(s_t|s_{t-1}, a_{t-1}) b(s_{t-1})$$

Figure 2. DBN and the associated formulation at time t

Partially Observable Markov Decision Process

The framework should have the ability of choosing a most profitable system respond among all possible ones. POMDP is a generalization of a Markov decision process MDP. It models the relationship between an agent and its environment. Formally, a POMDP is a 7-tuple (S,A,T,R,Ω ,O,γ) [16], where S is a set of states, A is a set of actions, T is a set of conditional transition probabilities between states, R: S $\times A \rightarrow \mathbb{R}$ is the reward function. Ω is a set of observations, O is a set of conditional observation probabilities, and $\gamma \in [0,1]$ is the discount factor. It is a decision process considering both confidence scores of transfers and rewards. S, T, Ω and O are given by other components of our algorithm framework, the system will make the most profitable operation action based on the reward function.

The solution of above is a DP problem with time complexity is exponentially dependent on the size of the state space, action and observation spaces. we adopt an approximate solution for efficiency consideration. We assign the discount factor γ to 0 leading the algorithm to only regard the immediate maximum reward.

The role of the reward function is to guide the algorithm to find the most profitable system action heuristically. In this paper, the reward function was made as a matrix empirically set through our observations of users' responds. Figure 3 shows the policy of POMDP and the reward function.

EXPERIMENT

We designed an experiment to validate the hypotheses and evaluate our algorithm framework. The experiment was conducted in a pen-based interface for drawing and graph editing using gestures. We designed two blocks in the experiment, to

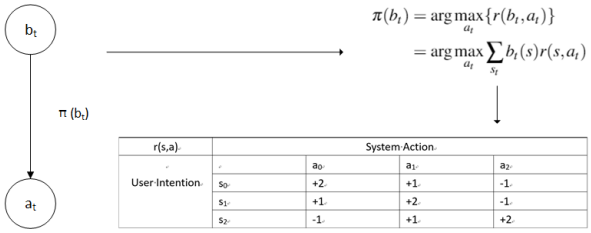


Figure 3. The policy of POMDP and the reward function

test the performance of our approach in more generalizable situation, the first block of the experiment evolved 13 gestures with one-to-one matched system commands similar as the $1 study [19]; to further evaluate our framework in a smaller gesture set that may reduce memory burden of users, the second block of the experiment evolved a smaller (n=9) gesture set but with the same number of commands. Four gestures in the second block linked to 2 system commands. The two gesture sets were designed by considering visual and motor similarities of their matched commands. User inputs were recognized as gesture if they were pressing a button on the pen during when they draw. The experimental interface, the two gesture sets and their associated commands were showed in figure 4.

For each of the two kernel recognizers, we compared recognition results of the following 3 conditions:

Kernel recognizer (**KR**): the results produced by the kernel recognizer (i.e., $1 or DTW) solely.

Framework enhanced recognizer (**FE**): the results produced by recognizer enhanced by our framework.

Framework enhanced recognizer in small gesture set (**FE-S**): the results produced recognizer enhanced by our framework in the small gesture set.

Participants and Apparatus

We recruited 12 participants (6 females and 6 males, with an average age of 26.1) in this study. All of them are right-handed and are daily users of gesture based interfaces.

The experiment was conducted on a CUBE i7 handwritten edition tablet PC which supported electromagnetic touch, and the experimental program was ran on windows 10 operation system. A pen with 112 mm in length and 5 g in weight was used.

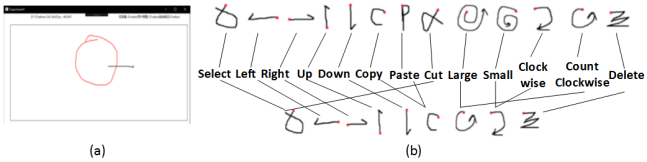


Figure 4. Interface and gesture sets used in the experiments

Procedure

The two blocks of the experiment shared the same task with different gesture sets. In a trail, participants were asked to draw a circle and then conducted a series of editing commands

on the circle. The order of the commands is select, move-left, move-right, move-up, move-down, copy, paste, cut, scale-large, scale-small, rotate-clockwise, rotate-counterclockwise and finally delete the circle. A participant had to repeat the current gesture until the it had been recognized. We recorded the user actual intentions, observations (results of kernel recognizers), the final recognition results, and system actions. Each participant conducted one trail per condition. Thus, a participant had to successfully perform 13 commands $\times$ 2 kernel recognizers $\times$ 3 conditions = 78 gestures in total. Participants were told about the difference between the two gesture sets and were allowed to practice before formal tests.

Note that, to get the results of conditions **FE** and **FE-S**, we need to tune the transfer matrix and the reward function for our framework. To do that, we conducted a calibration test for each of the two kernel recognizers before the formal experiment in each block. The calibration test had the same procedure of the formal one in **KR** condition.

RESULTS

For \$1 (figure 5a), **FE** got the best performance (90.3 %), followed by **FE-S** (89.8 %) and **KR** was the lowest (85.0 %). For DTW (figure 5b), **FE-S** got the best performance (82.9 %), followed by **EF** (82.1 %), and finally **KR** (73.3 %).

We made the following observations from the results. First, the fact that **EF** outperformed **KR** in both of the recognizers indicated that our framework improved recognizers' performance by infering current user intention from the last one, and by introducing the optimal planning of system action, which supported **H1**. Second, our **FE-S** achieved higher performances compared to **KR** in both of the recognizers, which indicated that the framework could reduce memory burden on users by allowing them to conduct complex interactions with a smaller gesture set, which supported **H2**.

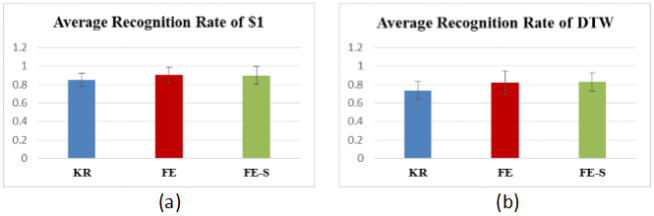


Figure 5. Average recognition accuracy of KR, FE and FE-S.

Figure 6 shows the confusion matrix of all gestures for **EF** recognizers in both \$1 and DTW. We found that some gestures such as move-left, scale-large, rotate-clockwise, delete had lower recognition accuracy. We attribute this as follow. The order of the tested gestures were accidentally divided into 6 groups with stronger logical continuities (i.e., select; move-left, move-right, move-up and move-down; copy, paste and cut; scale-large and scale-small; rotate-clockwise and rotate-counterclockwise; delete). Our framework showed worse performances in the first gesture in each of the 6 group, such as move-left and scale-large, because they can not be inferred from the prior user intention. In the contrast, the second gestures in each of the 6 group, such move-right, cut and scale-small, were well recognized by our algorithm framework.

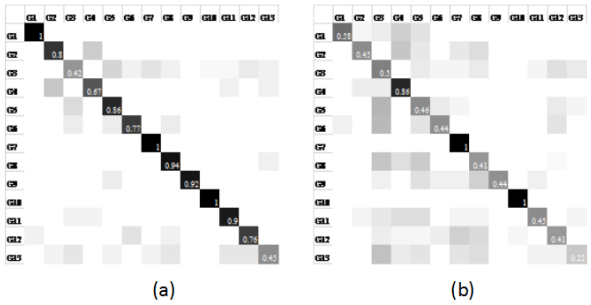


Figure 6. Confusion matrix of FE conditions for (a) \$1 and (b) DTW

Subjective Feedback

We conducted a post-test feedback with a 5-points Likert scale for rating the 3 conditions according to preference, accuracy, fluency, and memory burden. As showed in figure 7. **FE** performed best in preference, accuracy and fluency, and **FE-S** got best score in memory. This results provided additional evidences for supporting **H2**. We received positive responses to our framework, for example: *"I can remember the gesture more easily with the small gesture set" [P1]*; Participants also told us improvements that can be made to our system: *"I feel that some similar gestures such as move-right and move-down, are always be misrecognized"[P5].*

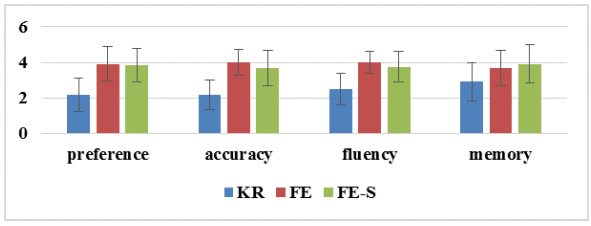


Figure 7. Participant compared algorithms from preference, accuracy, fluency, memory

CONCLUSION

This paper proposed an algorithm framework implemented based on DBN and POMDP to improve recognizers' performance by leveraging the continuity of gesture sequence during the interaction process. The experiment results indicated that the last user intention and optimal planning contribute to improve recognition accuracy. We showed that the proposed framework could be applied in any gesture recognizer that only rely on the information of current input to improve their performance in realistic user interfaces. Further, the framework could reduce memory burden on users by allowing them to conduct complex interactions with a smaller gesture set. In the future, we would like to applied our framework in a wider range of recognizers and make comparisons to other state-of-the-art techniques. We are also interested in testing the feasibility and usability of our approach in a wider range of application scenarios.

REFERENCES

1. Lisa Anthony and Jacob O. Wobbrock. 2010. A Lightweight Multistroke Recognizer for User Interface Prototypes. In *Proceedings of Graphics Interface 2010 (GI '10)*. Canadian Information Processing Society, Toronto, Ont., Canada, Canada, 245–252. http://dl.acm.org/citation.cfm?id=1839214.1839258
2. Lisa Anthony and Jacob O. Wobbrock. 2012. $N-protractor: A Fast and Accurate Multistroke Recognizer. In *Proceedings of Graphics Interface 2012 (GI '12)*. Canadian Information Processing Society, Toronto, Ont., Canada, Canada, 117–120. http://dl.acm.org/citation.cfm?id=2305276.2305296
3. Tim Dittmar, Claudia Krull, and Graham Horton. 2015. A new approach for touch gesture recognition: Conversive Hidden non-Markovian Models. *Journal of Computational Science* 10 (2015), 66 – 76. DOI: http://dx.doi.org/https://doi.org/10.1016/j.jocs.2015.03.002
4. Nir Friedman, Kevin P Murphy, and Stuart J Russell. 1998. Learning the structure of dynamic probabilistic networks. *uncertainty in artificial intelligence* (1998), 139–147.
5. Eamonn J Keogh and Michael J Pazzani. 2001. Derivative Dynamic Time Warping. (2001), 1–11.
6. Kenrick Kin, Björn Hartmann, Tony DeRose, and Maneesh Agrawala. 2012. Proton: Multitouch Gestures As Regular Expressions. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '12)*. ACM, New York, NY, USA, 2885–2894. DOI: http://dx.doi.org/10.1145/2207676.2208694
7. Per Ola Kristensson and Shumin Zhai. 2004. SHARK 2 : a large vocabulary shorthand writing system for pen-based computers. (2004), 43–52.
8. Allan Christian Long. 1998. Improving gestures and interaction techniques for pen-based user interfaces. (1998), 58–59.
9. A Chris Long, James A Landay, Lawrence A Rowe, and Joseph Michiels. 2000. Visual similarity of pen gestures. (2000), 360–367.
10. Hao Lü and Yang Li. 2012. Gesture Coder: A Tool for Programming Multi-touch Gestures by Demonstration. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '12)*. ACM, New York, NY, USA, 2875–2884. DOI: http://dx.doi.org/10.1145/2207676.2208693
11. Hao Lü and Yang Li. 2013. Gesture Studio: Authoring Multi-touch Interactions Through Demonstration and Declaration. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '13)*. ACM, New York, NY, USA, 257–266. DOI: http://dx.doi.org/10.1145/2470654.2470690
12. Andre Meyer. 1995. Pen computing: a technology overview and a vision. *ACM Sigchi Bulletin* 27, 3 (1995), 46–90.
13. Dean Rubine. 1991. Specifying gestures by example. *international conference on computer graphics and interactive techniques* 25, 4 (1991), 329–337.
14. Frode Eika Sandnes, Tek Beng Tan, Anders Johansen, Edvin Sulic, Eirik Vesterhus, and Eirik Rud Iversen. 2012. Making touch-based kiosks accessible to blind users through simple gestures. *Universal Access in The Information Society* 11, 4 (2012), 421–431.
15. Tevfik Metin Sezgin and Randall Davis. 2005. HMM-based Efficient Sketch Recognition. In *Proceedings of the 10th International Conference on Intelligent User Interfaces (IUI '05)*. ACM, New York, NY, USA, 281–283. DOI: http://dx.doi.org/10.1145/1040830.1040899
16. Edward J Sondik. 1978. The Optimal Control of Partially Observable Markov Processes over the Infinite Horizon: Discounted Costs. *Operations Research* 26, 2 (1978), 282–304.
17. Radudaniel Vatavu, Lisa Anthony, and Jacob O Wobbrock. 2012. Gestures as point clouds: a $P recognizer for user interface prototypes. (2012), 273–280.
18. Saiwen Wang, Jie Song, Jaime Lien, Ivan Poupyrev, and Otmar Hilliges. 2016. Interacting with Soli: Exploring Fine-Grained Dynamic Gesture Recognition in the Radio-Frequency Spectrum. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology (UIST '16)*. ACM, New York, NY, USA, 851–860. DOI: http://dx.doi.org/10.1145/2984511.2984565
19. Jacob O Wobbrock, Andrew D Wilson, and Yang Li. 2007. Gestures without libraries, toolkits or training: a $1 recognizer for user interface prototypes. (2007), 159–168.
20. Steve J Young. 2010. Cognitive User Interfaces. *IEEE Signal Processing Magazine* 27, 3 (2010), 128–140.