

“Those Look Similar!” Issues in Automating Gesture Design Advice

A. Chris Long, Jr.¹

Human Computer Interaction Institute
Carnegie Mellon University
Pittsburgh, PA 15213
chrisl@cs.cmu.edu
<http://www.cs.cmu.edu/~chrisl>

James A. Landay and Lawrence A. Rowe

EECS Department
University of California at Berkeley
Berkeley, CA 94720-1776
{landay,rowe}@cs.berkeley.edu
<http://www.cs.berkeley.edu/~{landay,rowe}>

ABSTRACT

Today, state-of-the-art user interfaces often include new interaction technologies, such as speech recognition, computer vision, or gesture recognition. Unfortunately, these technologies are difficult for most interface designers to incorporate into their interfaces, and traditional tools do not help designers with these technologies. One such technology is pen gestures, which are valuable as a powerful pen-based interaction technique, but are difficult to design well. We developed an interface design tool that uses unsolicited advice to help designers of pen-based user interfaces create pen gestures. Specifically, the tool warns designers when their gestures will be perceived to be similar and advises designers how to make their gestures less similar. We believe that the issues we encountered while designing an interface for advice and implementing this advice will reappear in design tools for other novel input technologies, such as hand and body gestures.

Keywords

Pen-based user interfaces, gestures, similarity, perception

1. INTRODUCTION

Today, state-of-the-art user interfaces often include new interaction technologies, such as speech recognition, computer vision, or pen gesture recognition.² Although the creators of these technologies are able to build applications using them, most interface designers are not familiar with these new technologies, and so have difficulty incorporating them into their interfaces. Unfortunately, design tools for traditional graphical user interfaces provide little assistance to designers who want to use these new technologies.

One type of assistance that a design tool could provide is to give unsolicited advice to designers about how to create and improve their interface. Based on our study of gesture design [3], we built a gesture design tool, *quill*, that advises designers on how to improve their gestures [2]. Adding advice to *quill* raised interesting implementation and interface issues. We believe these issues are common to design tools that offer advice.

1. Based on work done at U.C. Berkeley.
2. A gesture is a command issued with a pen (or other stylus).

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

PUI 2001 Orlando, FL USA

Copyright 2001 ACM 1-58113-448-7-11/14/01 ...\$5.00.

We were motivated to work on gesture interaction because gestures are fast, commonly used, and often easier to remember than textual commands [6]. Also, gestures are useful on displays ranging from the very small to the very large [7, 8]. However, users of current systems sometimes find gestures hard to remember and their gestures are often misrecognized [3].

People may find gestures difficult to learn and remember in part because of their perceptual similarity or dissimilarity. We contend that similar operations with a clear spatial mapping, such as scroll up and scroll down, should be assigned similar gestures. Conversely, gestures for similar abstract operations, such as cut and paste, may easily be confused if they are visually similar.

We have also found that it is difficult to design gestures that will be recognized well by the computer [3]. Naive gesture designers often created gestures that the computer viewed as similar and thus were difficult for the computer to recognize.

To help designers improve their gestures, *quill* continuously analyzes gestures for both of these types of problems. When *quill* predicts that two gestures will be perceived to be similar by people or confused by the computer, it warns the designer and provides advice about how to fix the problem.

To detect when people will perceive gestures as similar, *quill* includes an experimentally-derived model of human-perceived gesture similarity [2, 5].

This paper gives an overview of the similarity experiments, followed by a description of *quill*. Next, it discusses challenges of giving advice to users. The paper concludes with future work and a summary.

2. GESTURE SIMILARITY EXPERIMENTS

To better understand the principles people use to judge gesture similarity, we asked people to judge the similarity of gestures in three experiments. ([5] describes the first two; [2] describes all three.) In the first two, 42 participants judged relative similarity of gestures by choosing the most different gesture from many groups of three gestures. In the third experiment, 266 participants judged the similarity of pairs of gestures on an absolute scale.

We analyzed the similarity judgements from these experiments using multi-dimensional scaling (MDS),³ linear regression, and logistic regression. Our algorithm for predicting similarity combines a relative similarity model, which correlates 0.56 with reported similarity, with an absolute similarity model, which predicts whether a gesture pair will be perceived as similar or not. It has an accuracy of 87.7% overall, 99.8% for not similar gestures, and 22.4% for similar ones.⁴

We developed this model so *quill* could warn designers about gestures that people are likely to perceive as very similar. For this

3. MDS is a common analysis technique used in psychology for analyzing similarity data [1].

purpose, it is very beneficial that our model generates very few false positives (100% – 99.8% = 0.2%). Its large miss rate (100% – 22.4% = 77.6%) is disappointing, but we believe for our application that is a much better direction in which to err.

3. QUILL

quill helps designers of pen-based user interfaces create and improve gestures. It is designed to allow designers with no expertise in gesture recognition or in psychology to create gestures that are easy for the computer to recognize and that people will not easily confuse with one another.

Designers train the *quill* gesture recognizer, which uses the algorithm by Rubine [9], by drawing ten to fifteen example gestures for each type of gesture, or *gesture class*. Three gesture classes and their example gestures are shown in Figure 1. For applications with many gesture classes, the designer may organize classes into *gesture groups*. For example, gestures might be grouped into edit operations and file operations. The gesture classes and groups form the *gesture set*. Once training gestures have been entered, the designer may test the recognition of gestures by drawing them (see Figure 2).

A study of the gesture design process revealed that people had difficulty discovering and fixing problems with their gestures [3]. *quill* attempts to remedy this by providing *active feedback*—that is, unsolicited advice about problems and how to fix them.

quill uses the similarity metrics described above to predict whether people will perceive gestures as similar or not. While the designer enters gesture classes and gestures, *quill* analyzes the gestures in the background. When the designer pauses for a few seconds, *quill* offers unsolicited warnings to the designer about gesture classes that people may perceive to be similar. The warnings appear as text messages in the log at the bottom of the window, and as warning icons next to the relevant objects in the tree and desktop views of the gesture set. (Figure 3 shows an example of a recognition problem.) The object may not be visible at the time the advice is given. For example, a gesture class may be in a group whose tree view is not expanded. However, we do not want to hide warnings,

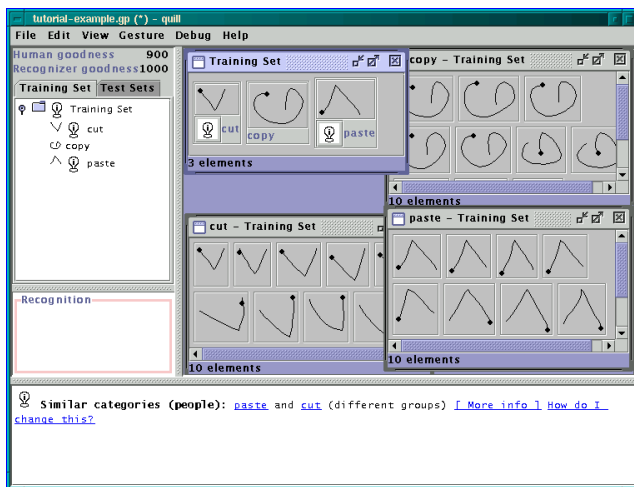


Figure 1. *quill* example with three categories: cut, copy, and paste. The windows show the whole set and some training gestures for each class.

4. The overall correctness is not the simple average of the two cases, because 4511 gesture pairs were rated not very similar and 940 were rated similar.

Recognized gesture in green

Recognized gesture highlighted

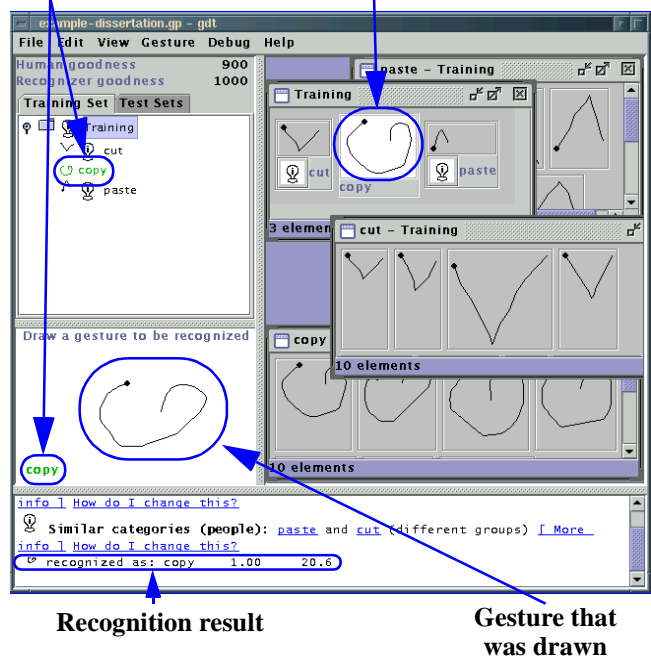


Figure 2. *quill* recognizes a gesture and shows the result in several different ways.

so a warning icon is propagated up the gesture hierarchy (i.e., the gestures, gesture classes, groups, and gesture set) to all its ancestors. If an object has more than one warning, the icon for the most severe warning is used.

The designer may follow hyperlinks in the log to obtain more information about the problem and advice about how to make the gestures less similar.

quill also looks for potential problems with recognition, such as gesture classes that the recognizer will see as similar, and misrecognized training gestures (see Figure 3). As with human similarity problems, *quill* provides advice about how to fix the problems. We have considered automatic or semi-automatic methods of morphing gestures to fix problems [2], but have not implemented them.

4. ADVICE CHALLENGES

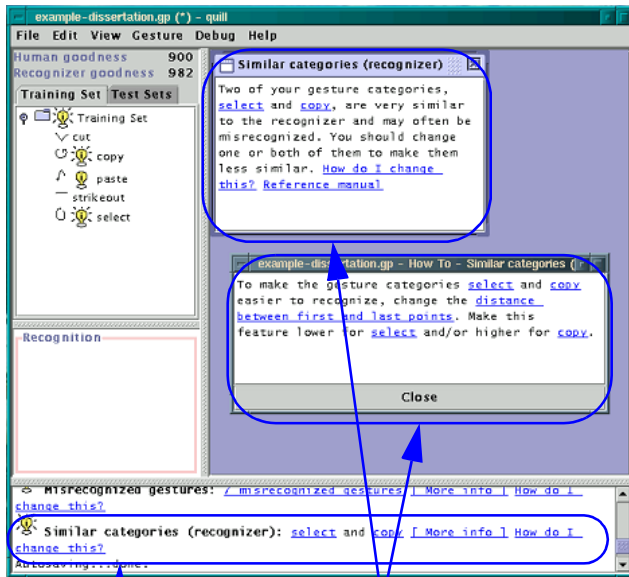
We encountered many interesting issues related to advice in *quill*, some related to the interface and others to the implementation. We also found issues related to the similarity metrics upon which some of the advice was based. Our observations here are based on pilot studies with members of our research group using a low-fi prototype and *quill*, on a usability evaluation of *quill* with 10 professional interface designers, and on our own intuition from implementing the system. We collected both quantitative (i.e., performance) and qualitative data in the evaluation.

4.1 User Interface Challenges

This subsection discusses the user interface challenges raised by adding advice to *quill*. We found challenges in three areas: advice timing, how much advice to display initially, and advice content.

4.1.1 Advice timing

We considered forcing users to ask for feedback in *quill* so that it would not interrupt them before they were finished entering



1. Warning 2. Additional information

Figure 3. Example *quill* warning and advice. First, *quill* shows an unsolicited warning to the designer. Then the designer clicks the hyperlinks to get more information.

gestures. However, in our study with *gdt*, an earlier gesture design tool [3], most participants did not call up the information that could have helped them improve their gestures. Therefore we decided that *quill* should show unsolicited advice—that is, be active in the design process.

Unfortunately, this approach raised the difficult question of when to show the advice. It is desirable to show advice as soon as possible so that users know what action caused the advice. In pilot studies with *quill*, we found that people ignored advice that occurred later because they thought it was irrelevant to what they were doing at the time the advice appeared.

However, giving advice as soon as it is available may interrupt or distract the user. This problem is especially bad for brainstorming or other creative activities, because it disrupts the flow of ideas.

Another problem with displaying advice too early is that the advice may become stale as the user works. In *quill*, for example, adding new example gestures or gesture classes may cause gesture classes to become more similar (according to human similarity or recognizer metrics) and thus possibly generate warnings. However, adding more objects may also cause existing gesture classes to become more different, thus possibly invalidating some warnings.

For *quill*, we decided to delay recognizer-oriented advice, because new gesture classes can cause old classes that were too similar to become dissimilar⁵, and we wanted to minimize advice that would be incorrect. We delay recognizer advice until the designer tests a gesture, because it is likely that the designer is finished entering new classes, at least temporarily, and is trying to evaluate the gestures. *quill* displays human-oriented advice as soon as possible,

5. The recognizer places gesture classes in a high-dimensional weighted Euclidean space. When new classes are added or existing ones modified, any of them may move in the space and change the similarity relationships.

because two gesture classes that are determined to be similar to people will not become dissimilar later.

The decision of when to offer advice would be easier in a more structured, workflow-oriented application, because then the application could show feedback when it is unlikely to be intrusive, such as after design and before testing. We are considering redesigning *quill* to be more workflow-oriented.

A problem related to when to show advice is when to remove advice. *quill* displays icons next to gesture objects that have active advice so the designer does not forget about the advice. Advice should be deactivated and its icon removed when the user directs it to be ignored, or when the system determines that the problem has been “fixed.” Unfortunately, it is not always clear how to determine what “fixed” means. One strategy is when a warning is created, remember the state of the relevant gesture objects and when the state changes more than some amount, remove the warning. A problem with this strategy is deciding how much change is enough. Another strategy is to evaluate the whole gesture set whenever it changes to determine what advice is still valid.

Initially, *quill* evaluated the entire gesture set whenever any part of it changed, and displayed all advice that currently applied. We soon realized that this design was unworkable, however, because we found in pilot testing that it deluged the user with the same advice over and over. Currently, *quill* analyzes the entire gesture set every time a change is made (after a short delay). Advice that did not apply the last time the analysis was done is displayed in the log. Advice is considered expired if it is no longer relevant, and in that case it no longer causes a warning icon to be displayed. However, notices are never removed from the log, because designers may want to go back and look at them later.

4.1.2 How much, how soon

Another issue we found was how much advice to display initially. In general, this depends on whether the users will be novices or experts. We did not want to unduly burden expert users by bombarding them with a lot of text for every piece of advice, but we wanted also to prevent novices from getting lost. *quill* shows a concise message initially, but provides hyperlinks to more verbose information (see Figure 3).

4.1.3 What advice

Finally, the actual content of the advice is important, as with any documentation. The tables and graph of *gdt* were too terse and technical, so for advice in *quill* we wrote English prose supplemented with drawings. The *quill* user study showed progress in this respect, but with more room for improvement.

4.2 Implementation Challenges

While implementing advice in *quill*, we encountered several issues that may apply to other applications that use advice.

4.2.1 Background analysis

Long-running operations such as complex analysis, database access, and network communication are becoming more common in end-user applications and these tasks are often run in background threads. For example, the analyses in *quill* are run in the background to avoid freezing the interface. We evaluated a number of approaches to background analysis to maximize user freedom and minimize user confusion. A summary of the options we considered are:

1. Lock all user actions during advice computation. This option would be fairly easy to implement, and simple for the user to understand, but the delay would annoy the user.

2. Disable all user actions that would change any state during advice computation. The ability to examine the gesture set would be beneficial, but the inability to make changes at some times but not others might confuse users.
3. Allow any action, but if a change happens that affects analysis, cancel analysis. This option would allow freedom for users, but they may not realize that some actions will cancel analysis. Also, it raises the question of whether canceled analyses should automatically restart.
4. Allow all actions. This option would be easy to implement, and would allow users complete freedom. However, it may result in advice that is not consistent with the gestures, which would likely confuse users.
5. Disable user actions that would change state in use by current analysis. This option would be efficient in that it allows all non-conflicting operations to occur. However, it would be difficult to implement and would almost certainly confuse users because different actions on different gesture objects would be enabled and disabled as different analyses were performed.

Analyses in *quill* may be started automatically by the system or manually by the user. We opted for strategy #2 for user-initiated analyses because we thought disabling some actions in response to a user action would not be confusing. For system-initiated analyses, we decided to use #3, and automatically restart canceled analyses. This option allows users to act freely without having to be concerned about or even know about the background analyses. We also provide feedback to the designer during analysis by putting a question mark after the feedback numbers to indicate that they may not be valid.

In our study with *quill*, this approach seemed to work well. Most designers in the study discovered that the computer updated its advice if they were idle for a few seconds. Two or three designers appeared not to discover this; possibly the feedback that analysis is occurring should be less subtle, at least for novice users.

4.2.2 Advice for hierarchies

Many interfaces structure complex information with hierarchies, such as the gesture set, group, class, and gesture in *quill*. In *quill*, all *notices* (i.e., pieces of advice) that apply to an object are stored in a list property of the object. Ancestors show a warning icon if any of their descendents have notices (see §3), so parent objects in the hierarchy need to know about *inherited notices* (notices of their descendents)⁶ as well as *intrinsic notices* (notices about themselves). This requirement raised the issue of how to store inherited notices, which may apply to other interfaces that display hierarchies.

The question is what information each parent (i.e., gesture class, group, or set) should store about their inherited notices. Since notices nearly always refer to multiple gesture objects, whereas gesture objects will have multiple notices less often, we decided to keep a list of all descendents that have one or more notices.

4.3 Similarity Metrics Challenges

In our formal evaluation of *quill*, we found that professional designers had difficulty using the advice about human similarity. The single biggest problem was that the similarity predictions were sometimes wrong. The models *quill* uses to predict human-perceived similarity are not perfect, and participants rightly disagreed with it at times. The model seemed especially prone to overestimate similarity when a gesture was a letter or contained a

letter. This flaw is probably due to the model being based entirely on non-letter data. We hypothesize that a small difference in a non-letter shape might be perceived by people as a large difference in a letter.⁷ Ideally, *quill* would have separate models for letter and non-letter shapes, and would apply them appropriately.

5. FUTURE WORK AND CONCLUSIONS

We have many ideas for ways to extend *quill*. A few of them are:

- To support a database of known gestures (e.g., copyedit marks) *quill* can compare new gestures against,
- To collect more data about human-perceived similarity to improve the similarity models, especially for letters, and
- To (partially) automate the repair of human similarity and recognition problems by morphing gestures.

For more future directions, see [2].

Our approach of measuring perceived similarity, creating models, and providing model-based advice may be useful for other gesture techniques. Some collaborative and context-aware systems monitor users' movements (most commonly with cameras) and allow interactions based on gestures with their hands and other body parts. As with gestures in pen-based interfaces, a designer of these types of gestures would benefit from knowing when their gestures would be perceived as similar by people, and from advice about how to make their gestures recognized better by the system. Designers of hand gestures would probably also want the system to compare their gestures with a known library, such as American Sign Language, and warn of similar gestures.

We believe the lessons we learned about how to design and build an interface that gives good advice at the right time will be helpful for authors of other design tools that offer advice to the designer.

6. REFERENCES

- [1] Kruskal, J. B. and Wish, M. *Multidimensional Scaling*. Number 07-011 in Quantitative applications in the social sciences. Sage Publications, Beverly Hills, CA, 1978.
- [2] Long, Jr., A. C. *Quill: a Gesture Design Tool for Pen-based User Interfaces*. PhD dissertation, University of California at Berkeley, Berkeley, CA, Dec. 2001. Available at <http://guir.berkeley.edu/pubs/quill/dissertation.pdf>.
- [3] Long, Jr., A. C., Landay, J. A., and Rowe, L. A. Implications for a gesture design tool. *CHI 1999, ACM Conference on Human Factors in Computing Systems, CHI Letters*, 1(1), 40–47.
- [4] Long, Jr., A. C., Landay, J. A., and Rowe, L. A. PDA and gesture use in practice: Insights for designers of pen-based user interfaces. Technical Report UCB//CSD-97-976, U.C. Berkeley, 1997. Available at <http://guir.berkeley.edu/pubs/quill/tech-report-142.html>.
- [5] Long, Jr., A. C., Landay, J. A., Rowe, L. A., and Michiels, J. Visual similarity of pen gestures. *CHI 2000, ACM Conference on Human Factors in Computing Systems, CHI Letters*, 2(1), 360–367.
- [6] Morrel-Samuels, P. Clarifying the distinction between lexical and gestural commands. *International Journal of Man-Machine Studies*, 1990. 32, 581–590.
- [7] Pedersen, E., McCall, K. and Moran, T. Tivoli: An Electronic Whiteboard for Informal Workgroup Meetings. In *Human Factors in Computing Systems (SIGCHI Proceedings)*, 391–398. ACM SIGCHI, Addison Wesley, April 1993.

6. Normally in a hierarchy, children inherit from parents. However, notices are inherited by parents from their children.

7. The Stroop effect, for example, shows that shapes with semantic meaning are cognitively processed differently from meaningless shapes [10].

[8] Pier, K. and Landay, J. A. Issues for location-independent interfaces. Tech. Rep. ISTL92-4, Xerox Palo Alto Research Center, Dec 1992.

[9] Rubine, D. Specifying gestures by example. In Computer Graphics (SIGGRAPH), pages 329–337. ACM SIGGRAPH, Addison Wesley, July 1991.

[10] Stroop, J. R. Studies of interference in serial verbal reactions. *Journal of Experimental Psychology*, 1935. 12, 643–662.