SimpleFlow: Enhancing Gestural Interaction with Gesture Prediction, Abbreviation and Autocompletion

Mike Bennett, Kevin Mccarthy, Sile O'modhrain, Barry Smyth

SimpleFlow: Enhancing Gestural Interaction With Gesture Prediction, Abbreviation And Autocompletion

Mike Bennett[1,2], Kevin McCarthy[2], Sile O'Modhrain[3], and Barry Smyth[2]

[1]SCIEN, Department Of Psychology, Stanford University
[2]School of Computer Science, University College Dublin, Ireland
[3]Sonic Arts Research Centre, Queens University, Belfast, UK
mikemb@stanford.edu

Abstract. Gestural interfaces are now a familiar mode of user interaction and gestural input is an important part of the way that users can interact with such interfaces. However, entering gestures accurately and efficiently can be challenging. In this paper we present two styles of visual gesture autocompletion for 2D predictive gesture entry. Both styles enable users to abbreviate gestures. We experimentally evaluate and compare both styles of visual autocompletion against each other and against non-predictive gesture entry. The best performing visual autocompletion is referred to as SimpleFlow. Our findings establish that users of SimpleFlow take significant advantage of gesture autocompletion by entering partial gestures rather than whole gestures. Compared to non-predictive gesture entry, users enter partial gestures that are 41% shorter than the complete gestures, while simultaneously improving the accuracy (+13%, from 68% to 81%) and speed (+10%) of their gesture input. The results provide insights into why SimpleFlow leads to significantly enhanced performance, while showing how predictive gestures with simple visual autocompletion impacts upon the gesture abbreviation, accuracy, speed and cognitive load of 2D predictive gesture entry.

1 Introduction

Gestural interfaces are now a familiar mode of user interaction and gestural input is an important part of the way that users can interact with such interfaces. Gestural input accommodates a style of interaction that goes beyond the point-and-click of traditional WIMP-based interfaces and allows users to interact in a more intuitive and efficient [28, 31, 9, 16]. For example, simple swiping motions (whether mouse pointer, trackpad, or direct screen-contact) can be used as more intuitive navigation controls, circumventing the precision that is demanded by more traditional point-and-click interactions [33]. A range of more sophisticated gestures, such as a two-fingered 'pinch' for zooming, are now an increasingly familiar part of gesture-based interaction dictionaries [13, 15].

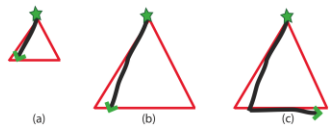

Fig. 1. Example of SimpleFlow. Shown is the visual feedback displayed when entering a triangle gesture. Star is starting point of gesture entry. Black line is the gesture entered so far by the user - (a) is time point 1, (b) time point 2, (c) time point 3. Red line is the real-time predicted gesture.

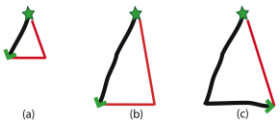

Fig. 2. Example of Scale Free Dynamic Paths shown when entering a triangle gesture. Star, black line & red line have the same meaning as in Figure 1.

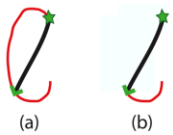

Fig. 3. Example of erroneous gesture predictions that could be shown when entering a triangle gesture. (a) SimpleFlow, (b) SF-Path.

Ultimately, gesture-based interaction promises to provide users with a more intuitive and richer interaction vocabulary, offering greater interaction bandwidth for lower input effort [33, 11]. However, this is not always the case and certainly the quest for more expressive gestures can lead to suboptimal gestures that are difficult for users to produce reliably, leading to interaction failures and frustration. One way to help users as they produce gestures is to provide autocompletion suggestions as the gesture unfolds [4, 30, 5, 13]. If this can be done efficiently and accurately then the user will benefit in a number of ways. For example, by accepting a gesture suggestion the user can complete their gesture early and so gestural input can be made more efficient [27, 11, 6]. Moreover, by encouraging early gesture completion it removes the risk of errors that might have occurred if the gesture had to be manually completed. At the same time there may be additional costs for the user, especially if they find the gesture completions to be difficult to accommodate into the work-flow; if completion suggestions are inaccurate, for example, then users will quickly become frustrated by such inconvenient interruptions.

In this paper we describe two approaches to gesture autocompletion (we refer to these as *predictive gestures*), both of which present the user with visual feedback as a gesture is entered (Figure 1 & 2). Both also provide the user with an option to autocomplete the target gesture early. In addition, we describe the results of a detailed user evaluation, in which the predictive gestures are compared to manual (non-predictive) gesture input, focusing on the gesture shortening, accuracy, speed and cognitive load characteristics of these techniques. In particular we demonstrate and explain how predictive gesture techniques can lead users to significantly abbreviate gestures while improving the speed and accuracy of their gesture input; along with

users stating a clear preference for the predictive gesture input over non-predictive entry.

The contributions of this work are:

1. Based on the visual autocompletion styles in this paper, our results establish that users will shorten predictive gestures by a significant amount. For predictive text entry it is well established that users will significantly shorten words during text entry. Previous to this it had not been established whether users of predictive gesture entry systems will enter shorter gestures, nor had it been established by how much they will shorten the gestures by.
2. Does allowing gesture abbreviation and showing visual autocompletion during gesture entry increase the cognitive demands on users? Unexpectedly, we find that the cognitive load does not increase for the autocompletion styles.
3. We introduce SimpleFlow, an effective style of visual autocompletion, which subtly but for users importantly differs from the simplest form of visual feedback (Scale Free Dynamic Paths).
4. Are there significant trade-offs between predictive versus non-predictive gesture entry and the speed and accuracy with which gestures are entered and recognised? We establish interactions and tradeoffs between gesture autocompletion, gesture abbreviation, input speed and input accuracy for SimpleFlow and Scale Free Dynamic Paths.

2 Related Work

Entering gestures accurately and quickly can be challenging [3, 32, 26]. Some of the challenges arise because of the need for algorithms that accurately recognise gestures [32, 25]. Other challenges include questions about what interface designs provide beneficial visual feedback before, during and after gesture entry [5, 3].

Visual Feedback For Predictions

Various styles of visual feedback have been proposed for gesture entry [13, 31, 5, 18, 2, 30, 33, 22]. Often the feedback styles are for enhancing pre- and post-gesture entry. Research on pre-gesture feedback aims to help users know and remember what set of gestures are available, while post-gesture feedback helps users understand whether they successfully entered a gesture, and if not what went wrong during gesture entry.

ShadowGuides [13], OctoPocus [5], Fluid Sketches [4] and others [18, 24, 2] are examples of interaction techniques which provide real-time visual feedback during gesture entry. These aim to help users learn the gestures better and help them understand how well they are entering the gestures. Other forms of real-time feedback help the user understand what the results of the gesture will be, such as telling them what actions will be performed, e.g. show the user where a dragged icon will end up [6], or what action will be performed [5].

Bau et al. [5] propose a very useful classification framework for the various styles of visual feedback. Within their classification framework SimpleFlow is a dynamic

guide, with feedback that has a continuous update rate, a real recognition value, a gesture filter and a gesture representation.

Applying Gestures
Recently there has been a surge in applications of gestures without visual feedback and in more novel contexts, such as using human skin as a gesture input surface [16], or moving your hands freely in the air as a form of bimanual gesture entry [15].

Other styles of gesture input which are less obviously gesture prediction include Drag-and-Pop / Drag-and-Pick [6], Push-and-Pop [11], Escape [33], Dasher [31] and marking menus [23]. In those examples the effects of users' physical actions are enhanced, so they can perform shorter and less actions than normally required for the tasks, which can be thought of as forms of gesture prediction and abbreviation.

Modeling Gesture Interaction
Models of human performance when following trajectories have also been investigated [1], and could prove useful for modeling gesture input [10, 20], as models and metrics of human performance have proven useful for predictive text input [27, 29]. Related to modeling gestures is research around measuring and modeling the complexity of gestures [26], which could be applied to quantifying the predictability of gestures.

3 Visual Autocompletion & Predictive Gestures

In this section we present[1] two styles of predictive visual feedback, which are shown during gesture entry. Both styles show visual predictions of the target gesture a user is inputting. In this way they allow the autocompletion of gestures and thus encourage quicker and more efficient gesture input. We also outline the non-predictive visual feedback commonly found in many gesture entry applications [12, 8, 14], which we experimentally compare the predictive visual feedback against.

3.1 Visual Autocompletion

We implemented two styles of visual autocompletion for predictive gestures. Both styles are closely related and subtly but importantly differ in how they provide the real-time visual feedback. The first style of visual autocompletion is called *SimpleFlow*, as shown in Figure 1, and the second style is called *Scale Free Dynamic Paths* (SF-Path), as shown in Figure 2.

The purpose of SimpleFlow and SF-Path is to keep the user informed during gesture input, such that they know as early as possible which of the trained gestures (examples in Figure 4) matches their input.

Both styles are designed to enhance user performance during gesture entry, by improving the speed and accuracy with which gestures are entered. Both are also de-

[1] For a video demonstration of the visual autocompletion styles please view the video accompanying this paper.

signed to enable users to abbreviate gestures. The ability to enter abbreviated gestures resembles predictive text entry systems - where users can enter complete words by only typing the first few letters of the words [27, 29].

Figures 1 & 2 show examples of what users see on-screen while using SimpleFlow and SF-Path. The stars and arrow-heads in these figures are for illustrative purposes only; they indicate the start and current end points of the gesture being entered. Gesture entry starts when the user presses the mouse button and begins moving the mouse. When entering predictive gestures, users are simultaneously shown two forms of visual feedback. First, they see the gesture path they have entered so far, i.e. the gesture ink. Secondly, they see the gesture prediction drawn in red. The gesture prediction is automatically scaled to match the shape and scale of the gesture ink. If a user chooses to stop entering the gesture at any time (by releasing the mouse button), then the currently predicted gesture is entered - as though the user drew the full gesture themselves.

Unlike the OctoPocus [5] and ShadowGuides [13] systems, SimpleFlow and SF-Path only show one gesture prediction at a time. If another prediction is more suitable, then the existing prediction is instantly switched out for the new prediction. Only one gesture prediction is shown because participants in our initial pilot studies voiced strong concerns about the high level of visual complexity that occurs when showing multiple simultaneous SimpleFlow predictions.

Another key reason for not showing multiple gesture predictions is we strongly suspect that the style of visual design used to show multiple gestures has a very significant impact upon user performance and preferences. The reason for this suspicion is due to research findings in psychology and perception around visual search, i.e. subjects search for a target stimulus (gesture) amongst distractor stimuli (predictions).

3.2 Standard Non-Predictive Visual Feedback

The non-predictive style of visual feedback evaluated in this work is *Standard Feedback*. Standard Feedback is included because it is the typical gesture input mechanism and visual feedback used in many real-world gesture UIs, e.g. StrokeIt [12] for Windows, wayV [8] for Linux, FireGestures [14] for Firefox.

Standard Feedback does not give predictive visual feedback to users. When entering a gesture only the gesture ink is drawn on-screen. The gesture ink shows users the shape of the gesture they have entered, but no feedback about predictions are provided. Standard Feedback does not provide pre- or post-gesture information about whether the gestures are entered correctly or incorrectly.

For our experiment, during the course of Standard Feedback gesture input, gesture prediction does still take place but the predictions are not shown to the participants. This is done to ensure that the algorithm for recognising gestures is the same across all forms of predictive and non-predictive feedback. In this paper, Standard Feedback serves as a control for comparing and evaluating SimpleFlow and SF-Path.

3.3 SimpleFlow

SimpleFlow provides real-time visual feedback during gesture entry and always suggests a complete gesture. As soon as a user begins entering a gesture they start to receive continuously updated gesture predictions. A gesture prediction is drawn in red underneath the gesture ink (Figure 1). The predicted gesture is automatically scaled so it underlays the gesture ink and visually appears to continue the path of the gesture ink. Often the gesture ink and predicted gesture do not perfectly align on top of each other.

Figure 1 shows a full walk-through of SimpleFlow in action. In Figure 1(a), the user has started to enter their gesture, in this case, a triangle shaped gesture (Figure 4, Gesture 1). SimpleFlow suggests and displays a fully scaled triangle underneath the gesture ink. As the user continues to extend the edge of their input gesture (as shown in Figure 1(b)) the SimpleFlow feedback scales up to accommodate the change. Figure 1(c) depicts the final stage of entering the triangle. The user draws the base of the triangle gesture, and decides that the full gesture prediction provided by SimpleFlow satisfies their needs. Now they cease their gesture input, safe in the knowledge that the correct complete gesture has been recognised.

SimpleFlow is scale and position invariant, however there is a limitation on the minimum size of the gesture predictions. If there was no minimum size limitation, the predicted gestures could be too small to see. For more details on the gesture prediction algorithm see the Appendix.

3.4 Scale Free Dynamic Paths

Scale Free Dynamic Paths (SF-Path) is like SimpleFlow. As with SimpleFlow, SF-Path is scale and position independent, and has an imposed limitation on the minimum size of the gesture predictions. The critical difference between SF-Path and SimpleFlow is how much of the predicted gesture is shown as visual feedback. As the results show, this visually subtle and small difference between SF-Path and SimpleFlow has a very significant impact on user performance and preferences.

Figure 2 shows the visual feedback provided while entering the triangle gesture with SF-Path. As with SimpleFlow the gesture ink is black, and the predicted gesture is shown in red. Unlike SimpleFlow, the complete gesture prediction is not shown on-screen by SF-Path. Instead the prediction and gesture ink are merged to form a single combined gesture, so that the gesture prediction looks to be a continuation of the gesture ink.

SF-Path is less visually complex than SimpleFlow, as there are no red lines underlying the gesture ink (Figure 2). When the gesture predictions are wrong the visual complexity of SimpleFlow is beneficial. For example, Figure 3 shows a side by side example of the differences between SimpleFlow and SF-Path when a visual prediction is wrong. In this example a user is trying to enter a triangle gesture. The prediction algorithm is mistakenly suggesting the gesture C. In Figure 3(a), with SimpleFlow, it is clear that the prediction algorithm is mistakenly suggesting the C gesture. Unfortunately, with SF-Path (Figure 3(b)) it is not at all clear which predicted gesture is merged with the gesture ink to form the combined gesture.

SF-Path resembles the visual feedback provided by OctoPocus [5] - if OctoPocus is altered to show one gesture suggestion at a time, its gesture suggestions are made scale and aspect invariant, and gesture abbreviation is allowed.

4 Experiment

Our experiment tests whether there are significant user performance and preference differences between predictive gestures and non-predictive gestures. It also establishes whether there are significant differences between the two styles of predictive visual autocompletion (SF-Path and SimpleFlow).

4.1 Hypothesis

For each of these hypotheses we are interested in understanding whether they do or do not hold true between the three styles of visual feedback and autocompletion, i.e. Standard Feedback, SF-Path and SimpleFlow. For example, are SF-Path gestures faster than SimpleFlow gestures? We hypothesise that:

- **H1 Shorten:** Predictive gestures enable users to reduce gestures to a shorter form, by entering abbreviated gestures rather than full gestures. Like predictive text entry systems.
- **H2 Accuracy:** Predictive gestures with visual autocompletion improves the accuracy with which users enter gestures.
- **H3 Speed:** Inputting predictive gestures with visual autocompletion is slower.
- **H4 Cognitive Load:** Cognitive load is higher for predictive gestures. Visual autocompletion places higher cognitive demands on users during gesture entry.

For H1 Shorten we expect that the continuous predictive visual feedback provided during the course of gesture entry will enable users to significantly shorten the gestures they enter. If true, users will enter abbreviated short partial gestures rather than whole gestures. Of particular interest is how much do they shorten the gestures by, a small amount or large amount?

Accuracy is an important feature of gesture input. Providing effective real-time gesture feedback should improve users' ability to enter gestures correctly. Whether SimpleFlow and SF-Path are effective forms of visual feedback is established with the H2 hypothesis.

The speed with which gestures are entered is important. If gesture entry takes too long it could be distracting and interfere with workflows. With the H3 hypothesis we expect a speed / accuracy tradeoff, where speed is how long it takes to enter a gesture. Predictive gesture entry may be slower because users spend time evaluating and adapting their gesture input based on the real-time feedback.

Hypothesis 4 establishes whether predictive visual autocompletion requires more cognitive resources than no predictive visual feedback. Ideally, well-designed forms of visual feedback should not increase cognitive load during gesture entry.

4.2 Design

The experiment task was to input a series of gestures correctly. A gesture was deemed correctly entered when the gesture inputted by the participant matched the target gesture they had been instructed to enter. Whether a gesture matched was evaluated by the gesture recognition algorithm.

A within-subjects experiment design was used. The experiment was 3x1, where the independent variable was the style of visual gesture autocompletion and the dependent variable was gesture input accuracy. The three levels of the independent variable were Standard Feedback, SF-Path and SimpleFlow. While the two levels of the dependent variable were *Correct* or *Incorrect*, corresponding to whether the entered gesture correctly or incorrectly matched the target gesture.

All participants were presented with three randomly ordered blocks of randomly ordered gestures. Each block was presented once and corresponded to a level of the independent variable, i.e. Standard Feedback, SF-Path, or SimpleFlow. Within each block each gesture was repeated three times, distributed randomly within a block. At the start of each block participants practiced inputting three gestures. The visual autocompletion drawn during practice gestures was based on the level of the independent variable.

Participants completed the experiment in a single session. Each participant completed a short post-hoc questionnaire, where they provided their rank preferences for the visual autocompletion styles.

Participants

Eighteen volunteer participants took part, three of which did the pilot experiments and the remaining fifteen completed the main experiment. Of those fifteen 10 were male and 5 female. Participants' average age was 26.5 years, with a standard deviation of 3.8 years. All participants naturally used their right hand to control the mouse. The fifteen participants entered 2295 gestures, of which 135 were practice gestures.

Materials

Figure 4 shows the sixteen gestures used for the experiment. The gestures are from the work on the $1 Recognizer [32]. We picked those gestures as they were independently created and have been used in other gesture experiments [32]. Using an independently created set of gestures helps avoid introducing an experiment bias, as the experiment did not run with a set of gestures specifically designed for the experiment task.

The gesture recognition algorithm was kept the same between the three levels. What differed between the levels was whether visual feedback was shown, and what kind of visual feedback it was. Details about the gesture recognition algorithm are provided in the Appendix.

Procedure

The experiment took place in a private room, free from external stimuli. A PC running Windows XP was used to run the experiment, and the same mouse was used by all participants. The PC monitor was equipped with a Tobii T60 eyetracker, which

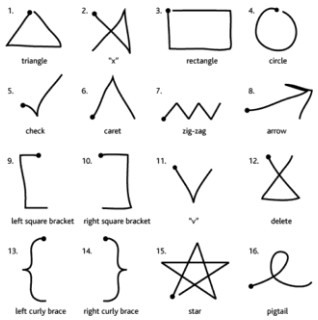

Fig. 4. Unistroke gestures used in the experiment, from [32]. Black points are the starting point for drawing the gestures.

was used to capture pupil dilation at 60Hz. The eyetracker data was used to calculate cognitive load (Section 5.4). Pixels on the monitor were square.

Participants were first calibrated on the Tobii eye tracker, with Tobii's standard calibration and eye tracking software. Then each participant was given time to practice inputting gestures using the mouse, during which no predictions were displayed. The participants then practiced the stimulus-response gesture input task three times, with gestures from Figure 4. At the start of each block participants also practiced inputting gestures, during which they saw the visual autocompletion specific to the block.

The gesture input stimulus-response task consisted of presenting a target gesture to the participants, which was displayed centered on-screen for two seconds. During the two seconds the mouse pointer was not shown on-screen. Text above the target gesture instructed participants to look at the gesture. After the two seconds expired the screen was blanked, a text message appeared informing participants to draw the gesture, and the mouse pointer appeared centered on-screen. The mouse pointer was centered on-screen to prevent a carryover effect occurring between tasks, which could arise if the mouse pointer location was carried between gesture tasks.

Participants then inputted a gesture by pressing and holding the left mouse button down and moving the mouse. When the participant released the mouse button, the gesture was considered complete.

Statistical Techniques

To analyse hypothesis H1 and H3 we use Balanced Repeated Measures Within-Subjects ANOVAs with Bonferroni pairwise comparisons. For H4 we first apply a standard log transform to the results, as tests of normality indicated the results had a non-normal distribution until transformed. Then we apply the same analysis techniques as for H1 and H3.

For hypothesis H2 we use Repeated Measures Logistic Regression (RMLR) with Bonferroni pairwise comparisons. RMLR is used because gesture accuracy is a categorical binary variable, i.e. gestures are either entered *Correctly* or *Incorrectly*.

A Bonferroni test compares each level with every other level, and establishes whether performance between the levels is significantly different. Balancing is performed by selecting a subset of results randomly distributed in the results. For example, if there are 600 results for Standard Feedback (Standard), 550 for SF-Path and 500 for SimpleFlow, then 500 results are randomly selected from each of the levels and pairwise comparisons are performed on those results.

Rather than reporting every p value, we report the meaningful p values, i.e. the highest significant p values out of each three way pairwise comparison. There are nine p values generated per set of results (Standard vs SF-Path, Standard vs SimpleFlow, SF-Path vs SimpleFlow) * (All, Correct, Incorrect gestures).

Accepting Gesture Predictions

For the blocks where participants saw predictive visual autocompletion (SF-Path and SimpleFlow), they were informed they could accept a gesture prediction without having to enter the full gesture. Accepting a gesture prediction was achieved by stopping gesture entry. Participants stopped gesture entry by releasing the mouse button. For example, in Figure 1 a participant could stop entering a gesture at time point (a) if they wanted to accept the prediction and input a triangle gesture.

5 Results

Overall, users of SimpleFlow automatically shorten gestures by 41%, while simultaneously improving the accuracy (+13%, from 68% to 81%) and speed (+10%) of gesture input - and this is achieved with no significant increase in cognitive load. Results from the post-hoc questionnaire indicate that participants prefer SimpleFlow over SF-Path and Standard Feedback.

A total of 135 training gestures and 21 error gestures were removed from the results, leaving 2139 gestures available for analysis, with 714 gestures in Standard Feedback, 713 in SF-Path and 712 in SimpleFlow conditions. Gestures were classified as errors where participants accidentally clicked the mouse button and did not move the mouse.

5.1 Hypothesis: H1 Shorten

Users can and do take advantage of the gesture predictions. Participants enter 39%+ shorter gestures with SimpleFlow and SF-Path, hypothesis H1 is true. SimpleFlow and SF-Path successfully enable users to enter abbreviated short partial gestures rather than whole gestures. This implies that the predictive gesture algorithm makes accurate predictions early and continuously during the course of gesture entry.

Abbreviated Gestures

To establish whether participants abbreviate gestures, we analyse the pixel path length of entered gestures. Gesture path length is a measure of how many pixels are travelled when entering a gesture. Gesture path length is significantly ($p < 0.037$) shorter for

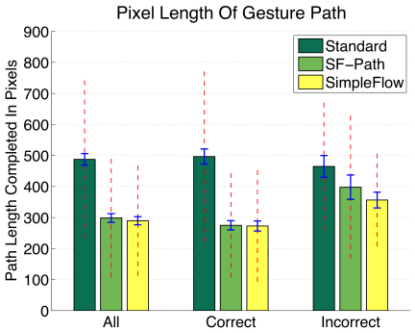

Fig. 5. Pixel path length of entered gesture. Error bars are 95% Wald confidence interval, dashed lines standard deviation

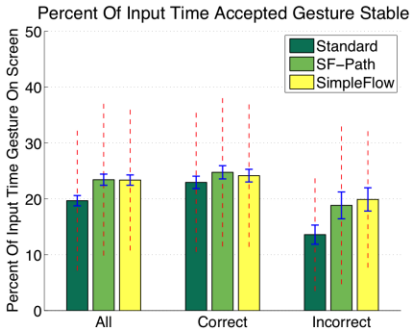

Fig. 6. How long was the accepted gesture stable on screen, as percentage of gesture input time.

All, Correct and Incorrect SF-Path (39%, All) and SimpleFlow (41%, All) gestures, compared to No Feedback (Figure 5). No significant difference exists between SF-Path and SimpleFlow. H1 is true, based on the gesture pixel path length.

A shorter gesture pixel path length does not definitively establish that participants abbreviate gestures. Shorter path lengths could mean participants enter smaller gestures. To rule out this possibility, we measure and analyse the entered path length divided by the predicted path length. We refer to this measure as the Partial Gesture Ratio (PGR). The lower the PGR the better. A PGR < 1 means the entered gesture is shorter than the predicted gesture, PGR > 1 means the entered gesture is longer than the predicted gesture, and PGR = 1 means the gestures are the same length.

Based on the PGR we find that All, Correct and Incorrect gestures entered with SF-Path and SimpleFlow are significantly ($p < 0.002$) shorter than Standard Feedback (Table 1). This further confirms that participants do abbreviate gestures.

Early And Continuous Predictions

Do participants enter partial gestures because the prediction algorithm generates accurate predictions early and continuously during gesture entry? We analyse whether the gesture predictions stabilise early and stop changing. Early stabilisation is measured by dividing the time when gesture predictions stop changing by the time taken to input the gesture.

Predictions do stabilise early and stop changing during gesture entry (Figure 6). For All gestures Standard Feedback is stable for 39% of gesture input time, while SF-Path and SimpleFlow are stable for 47% of input time. All and Incorrect gestures are stable significantly ($p < 0.001$) longer for SF-Path and SimpleFlow. There are no significant differences for Correct gestures.

Number Of Predictions

How often the predictions change before stabilisation could effect whether users enter partial gestures. Ideally a prediction does not change too often, as too many changes

Table 1. Path length of entered gestures, as PGR.

	All	Correct	Incorrect
Standard	1.014 (0.288)	0.994 (0.057)	1.125 (0.535)
SF-Path	0.733 (0.395)	0.668 (0.314)	0.901 (0.542)
SimpleFlow	0.697 (0.323)	0.685 (0.297)	0.791 (0.393)

Table 2. Number of times gesture prediction changed.

	All	Correct	Incorrect
Standard	6.51 (3.73)	6.01 (3.69)	7.74 (3.50)
SF-Path	5.84 (4.30)	5.28 (4.02)	7.82 (4.93)
SimpleFlow	5.48 (3.88)	5.13 (3.71)	6.94 (3.87)

could make it harder for a user to judge which prediction is emerging as the winning prediction.

SF-Path and SimpleFlow generate 5.84 and 5.48 prediction changes for All gestures (5.28 and 5.13 for Correct), which are significantly less gesture prediction changes, for All ($p < 0.003$) and Correct ($p < 0.017$) gestures. No significant differences occur for Incorrect gestures. Shown in Table 2 are the mean number of times the gesture predictions changed.

5.2 Hypothesis: H2 Accuracy

By more than 10% both SimpleFlow and SF-Path significantly ($p < 0.003$) improve gesture entry accuracy, compared to Standard Feedback. Hypothesis H2 is true, predictive gestures with visual autocompletion does improve the accuracy with which users enter gestures.

As shown in Figure 7, for Standard Feedback participants correctly entered gestures 68.1% (SD 15.8%) of the time. The gesture accuracy increased to 78.6% (SD 10.9%) and 81% (SD 12.2%) for SF-Path and SimpleFlow respectively. Performance is not significantly ($p < 0.05$) different between SF-Path and SimpleFlow. In Figure 7 the dot is the mean accuracy, the error bars are the 95% Wald confidence interval and the dashed lines the standard deviation.

5.3 Hypothesis: H3 Speed

Unexpectedly, SimpleFlow is more than quarter of a second faster (+10%, $p < 0.012$) for entering gestures than Standard Feedback and SF-Path. Entering gestures with SF-Path is as fast as with Standard Feedback (Figure 8). Hypothesis 3 is rejected, as visual autocompletion does not slow gesture input.

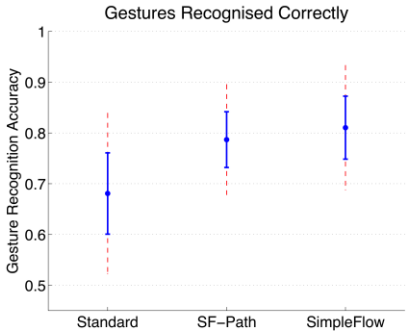

Fig. 7. Predictive gestures with visual autocompletion improves performance when entering gestures.

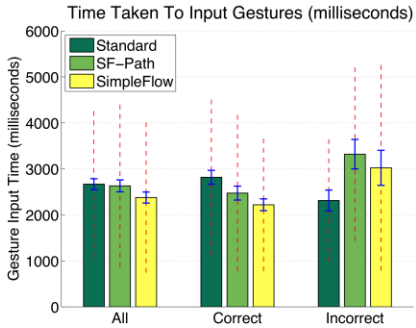

Fig. 8. Length of time taken to input gestures varies based on visual autocompletion.

Time Taken To Enter Gestures

Focusing on the results for Correctly entered gestures - SF-Path is 344 milliseconds faster ($p < 0.006$) than Standard Feedback, and SimpleFlow is faster ($p < 0.038$) than both (Figure 8). H2 is rejected again.

When gestures are entered Incorrectly, then Standard Feedback is faster ($p < 0.009$) than both SF-Path and SimpleFlow. For Incorrect gestures H3 is true. No significant time differences exist between SF-Path and SimpleFlow for Incorrect gestures.

Rate Of Gesture Input

Is SimpleFlow the fastest input style because participants' rate of input is higher? When entering gestures how fast do users move the on-screen mouse pointer? Rate of input is calculated as the average number of pixels travelled by the mouse pointer per millisecond during gesture input.

Standard Feedback leads to a significantly ($p < 0.0001$) higher rate of gesture input than both SF-Path and SimpleFlow; for All, Correct and Incorrect gestures (Figure 9). There are no significant differences between SF-Path and SimpleFlow.

This confirms that the speed improvement for SimpleFlow is not due to an improved rate of input. The speed improvement occurs because participants enter abbreviated gestures, which means it takes less time to enter the partial gestures.

5.4 Hypothesis: H4 Cognitive Load

Does the visual autocompletion place higher cognitive demands on users, than no visual feedback? Or does the visual feedback even decrease the cognitive load? Surprisingly H4 is rejected, SF-Path and SimpleFlow do not lead to increased cognitive load. There are no significant differences ($p < 0.05$) in cognitive load between Standard Feedback, SF-Path and SimpleFlow (Figure 10). This result applies to All, Correct and Incorrect gestures.

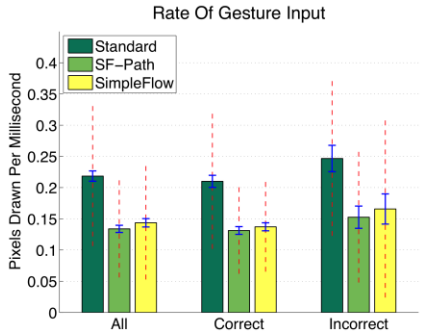

Fig. 9. Rate of input.

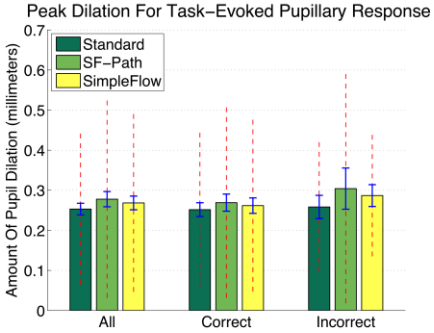

Fig. 10. TEPR Peak Dilation measures of Cognitive Load.

We measured cognitive load by measuring Task-Evoked Pupillary Response (TEPR), which is known to be a reliable indicator of cognitive load [7, 19, 21]. Task-Evoked Pupillary Response is a measure of how much the pupil dilates over time as a function of task hardness. The larger the change in pupil dilation the larger the cognitive load.

As recommended in the literature we removed blink artifacts, applied a low-pass filter to the dilation measures, used a 500ms baseline and used TEPR Peak Dilation to calculate cognitive load [7, 21]. The baseline was captured each time participants entered a gesture, right before they began entering the gesture. Measuring TEPR with a non-contact eyetracker enables us to measure real-time non-subjective quantifications of cognitive load, as an alternative to subjective work load assessments such as NASA TLX [17].

5.5 User Preferences

Participants preferred SimpleFlow 2.2 times more often than SF-Path, and 2.75 times more often than Standard Feedback (Figure 11). In the post-hoc questionnaire participants ranked Standard Feedback, SF-Path and SimpleFlow on a scale from 1 to 3, where 1 is most preferred and 3 least preferred. They could assign equal ranks.

Figure 11 shows the number times each rank was assigned to Standard Feedback, SF-Path and SimpleFlow. Participants ranked SimpleFlow 1st eleven times, SF-Path 1st five times, and Standard Feedback 1st four times. SF-Path is most often ranked 2nd, and Standard Feedback is most often ranked 3rd.

6 Discussion & Limitations

Our design aim for SimpleFlow, was to keep the complexity of the visual prediction feedback as simple as possible, while improving user performance. A related goal was to enhance gestural interaction without significantly changing it; in an effort to avoid requiring users to spend considerable amounts of time skilling up on a new interaction

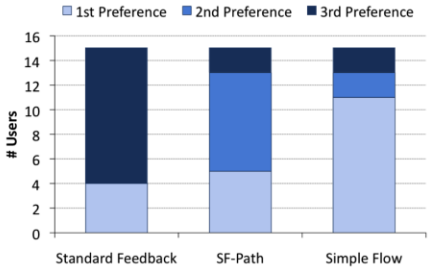

Fig. 11. User preferences as indicated in post-study questionnaire.

technique. Two further key aims were to enable users to take advantage of the gesture predictions without forcing them to use the predictions, and enable users to enter shorter gestures, rather than having to input full gestures. Looking at the results we find that SimpleFlow performs best out of all three forms of visual feedback, while also achieving our goals and aims. During the experiment users had few practice opportunities with the predictive gestures, yet they performed significantly better with predictive gestures versus non-predictive, while also preferring the predictions over no predictions.

Of particular interest we found that SimpleFlow is faster and more preferred to SF-Path, which is the same style of visual feedback as OctoPocus (when one gesture suggestion is shown at a time and the suggestions are scale and aspect invariant). This is interesting because the visual difference between SimpleFlow's visual feedback and SF-Path's visual feedback is very small, even though it has a critical impact. SimpleFlow shows a whole gesture suggestion in conjunction with the partial gesture a user has entered, while SF-Path completes the gesture for a user (Figure 3). When we followed up with users and informally asked them why they preferred SimpleFlow over SF-Path, the general consensus was that showing whole gestures clearly tells the users exactly which predicted gesture is getting matched with their partial gesture. While when partial gestures are completed in SF-Path, it is not always clear to the user why a gesture is completed the way it is, especially when the gesture prediction is wrong.

An interesting limitation of this work, which strongly suggests worthwhile future directions, is that we treat the users' performance and the performance of the gesture prediction algorithm as a combined system. This leaves open questions about the relationship between a user's performance, and what would happen if a better or worse gesture prediction algorithm is used? For example, if a perfect gesture recognition algorithm is used would we find that users enter even shorter gestures? If so, what is the absolute lowest bound on how much users will abbreviate gestures by? Does that interact with the number of available gestures? Related questions include what, if any, are the effects of other performance characteristics of the gesture prediction algorithm, e.g. stability of predictions. Ultimately, what are the desirable characteristics of gesture prediction algorithms? Another interesting question arises around the set of gestures we ran the experiment with, as the range of gesture shapes could impact on user performance. For example, in Figure 4 gesture 2, 11 and 12 all share a common

initial stroke (downward diagonal left stroke) - what would happen if a set of gestures is used that is optimised to minimize the shared starting strokes?

There are many other possibilities for future research, including extending SimpleFlow to handle multi-stroke gestures, creating UI techniques that enable users to interactively refuse and cancel a gesture suggestion, establishing what styles of pre- and post-gesture feedback is beneficial with visual predictions, and enabling users to quickly select from a set of gesture suggestions without having to complete the gestures. Finally, understanding more about why and how users decide to shorten predictive gestures would be very useful. Understanding this would help us further enhance and influence gesture abbreviation, i.e. enable users to enter even shorter gestures more quickly.

7 Conclusions

We found that users of predictive gestures with SimpleFlow and SF-Path visual feedback will significantly shorten gestures (like predictive text entry systems), with no significant increase in cognitive load. Further, SimpleFlow successfully enhances users' gesture entry, both speeding it up and improving the accuracy; along with users preferring SimpleFlow the most. The visual feedback provided by SimpleFlow is visually simple and minimal, and could easily be added to existing gesture entry systems without requiring significant changes to them.

References

1. J. Accot and S. Zhai. Beyond Fitts' Law: Models for trajectory-based HCI tasks. In *Proc. CHI '97*, pages 295–302.
2. P. Agar and K. Novins. Polygon recognition in sketch-based interfaces with immediate and continuous feedback. In *Proc. GRAPHITE '03*, pages 147–150.
3. C. Appert and S. Zhai. Using strokes as command shortcuts: Cognitive benefits and toolkit support. In *Proc. CHI '09*, pages 2289–2298.
4. J. Arvo and K. Novins. Fluid sketches: Continuous recognition and morphing of simple hand-drawn shapes. In *Proc. UIST '00*, pages 73–80.
5. O. Bau and W. E. Mackay. Octopocus: A dynamic guide for learning gesture-based command sets. In *Proc. UIST '08*, pages 37–46.
6. P. Baudisch, E. Cutrell, D. Robbins, M. Czerwinski, P. Tandler, B. Bederson, and A. Zierlinger. Drag-and-Pop and Drag-and-Pick: Techniques for accessing remote screen content on touch- and pen-operated systems. In *Proc. Interact '03*, pages 57–64.
7. J. Beatty and B. Lucero-Wagoner. *The Pupillary System*, chapter 6, pages 142–161. Handbook of Psychophysiology. Cambridge University Press, 2nd edition, 2000.
8. M. Bennett. wayv: Gestures for Linux. page www.stressbunny.com/wayv, 2011.
9. X. Cao and R. Balakrishnan. Evaluation of an on-line adaptive gesture interface with command prediction. In *Proc. GI '05*, pages 187–194.
10. X. Cao and S. Zhai. Modeling human performance of pen stroke gestures. In *Proc. CHI '07*, pages 1495–1504.
11. M. Collomb, M. Hascoet, P. Baudisch, and B. Lee. Improving drag-and-drop on wall-size displays. In *Proc GI '05*, pages 25–32.

12. J. Doozan. Strokeit gestures for Windows. page www.tcbmi.com/strokeit, 2011.
13. D.Freeman, H.Benko, M.R.Morris,and D.Wigdor.Shadowguides:Visualizations for in-situ learning of multi-touch and whole-hand gestures. In *Proc. Tabletop '09*, pages 183–190.
14. Gomita. Firegestures: Gesture control for Firefox web browser. page www.xuldev.org/firegestures, 2011.
15. S. Gustafson, D. Bierwirth, and P. Baudisch. Imaginary interfaces: Spatial interaction with empty hands and without visual feedback. In *Proc. UIST '10*.
16. C. Harrison, D. Tan, and D. Morris. Skinput: Appropriating the body as an input surface. In *Proc. CHI '10*, pages 453–462.
17. S. G. Hart and L. E. Staveland. *Development of NASA-TLX (Task Load Index): Results of empirical and theoretical research*, pages 239–250. Human Mental Workload. North Holland Press, 1988.
18. T. Igarashi, S. Matsuoka, S. Kawachiya, and H. Tanaka. Interactive beautification: A technique for rapid geometric design. In *Proc. UIST '97*, pages 105–114.
19. S.T.Iqbal,X.S.Zheng,and B.P.Bailey.Task-evoked pupillary response to mental workload in Human-Computer Interaction. In *Proc. CHI '04* Extended Abstracts, pages 1477–1480.
20. P. Isokoski. Model for unistroke writing time. In *CHI '01*, pages 357–364.
21. J. Klingner, R. Kumar, and P. Hanrahan. Measuring the task-evoked pupillary response with a remote eye tracker. In *Proc. ETRA '08*, pages 69–72.
22. P. O. Kristensson and S. Zhai. Command strokes with and without preview: Using pen gestures on keyboard for command selection. In *CHI '07*, pages 1137–1146.
23. G. Kurtenbach and B. William. The limits of expert performance using hierarchic marking menus. In *Proc. CHI '93*, pages 482–487.
24. J. Li, X. Zhang, X. Ao, and G. Dai. Sketch recognition with continuous feedback based on incremental intention extraction. In *Proc. IUI '05*, pages 145–150.
25. Y. Li. Protractor: A fast and accurate gesture recognizer. In *Proc. CHI '10*, pages 2169–2172.
26. A. C. Long, Jr., J. A. Landay, L. A. Rowe, and J. Michiels. Visual similarity of pen gestures. In *Proc. CHI '00*, pages 360–367.
27. I. S. MacKenzie and R. W. Soukoreff. Text entry for mobile computing: Models and methods, theory and practice. *Journal of Human-Computer Interaction*, 17:147–198, 2002. 28. D. Rubine. Specifying gestures by example. *SIGGRAPH Computer Graphics*, 25(4):329–337, 1991.
29. R.W.Soukoreff and I.S.MacKenzie. Metrics for text entry research: An evaluation of msd and kspc, and a new unified error metric. In *Proc. CHI '03*, pages 113–120.
30. P. Tandler, P. T, and T. Prante. Using incremental gesture recognition to provide immediate feedback while drawing pen gestures. In *Proc. UIST '01*, pages 18–25.
31. D. J. Ward, A. F. Blackwell, and D. J. MacKay. Dasher - a gesture–driven data entry interface for mobile computing. In *Proc. UIST '00*, pages 129–138.
32. J. O. Wobbrock, A. D. Wilson, and Y. Li. Gestures without libraries, toolkits or training: A $1 recognizer for user interface prototypes. In *Proc. UIST '07*, pages 159–168.
33. K. Yatani, K. Partridge, M. Bern, and M. W. Newman. Escape: A target selection technique using visually-cued gestures. In *Proc. CHI '08*, pages 285–294.

Appendix: Gesture Prediction Algorithm

For experimental reproducibility this appendix outlines the gesture prediction algorithm (Figure 12), which is composed of a gesture concatenation algorithm and a gesture recognition algorithm. When a user enters part of a gesture, the gesture concatenation algorithm generates a

complete gesture based on the partially entered gesture. Then the complete gesture is sent to a gesture recognition algorithm, to check whether it matches any of the trained gestures. The best matching gesture is used as the gesture prediction.

Gesture Concatenation Algorithm The gesture concatenation algorithm is easy to implement, though the computational efficiency of it is open to improvement. The algorithm assumes that gestures start from the same points, like the $1 Recognizer [32]. Unlike OctoPocus [5] and ShadowGuides [13] however, the algorithm is scale independent. Like $1 Recognizer and OctoPocus, and unlike ShadowGuides, our algorithm is for single stroke continuous gestures. The algorithm also handles different gesture aspect ratios, like Protractor [25], but like OctoPocus and ShadowGuides it is not rotational invariant to gesture orientations. Below are the seven steps in the algorithm (see Figure 12):

1. During gesture entry, continuously capture the path of the incoming gesture, generating a partial gesture P (Figure 12(a)).
2. Measure the width w, height h and calculate the length l of the partial gesture P (Figure 12(b)).
3. Scale the trained gestures (Figure 12(c)) so they share the same width w and height h as the partial gesture P. We refer to this set of scaled gestures as SC (Figure 12(d)).
4. For each scaled gesture in SC (Figure 12(d)), remove a path length l from the start of each gesture. This generates a set of cropped gestures CG (Figure 12(e)).
5. Merge the partial gesture P with every cropped gesture in CG. This generates a new set of gestures PG, as shown in Figure 12(f). Each gesture in PG is a gesture prediction. The merging should be done such that the end of P is merged with the start of each cropped gesture in CG.
6. If the gesture recognition algorithm is scale dependent, then the gestures in PG may need to be rescaled to match the scale of the gestures used to train the gesture recognition algorithm.
7. To find the winning gesture prediction send each gesture prediction in PG to the gesture recognition algorithm. The best scoring gesture in PG is the gesture prediction.

We imposed a constraint on the above algorithm, such that the predicted gesture cannot be scaled below a specific size. Without that constraint gestures could be scaled to only a few pixels wide or high, which would make the gestures visually indistinguishable.

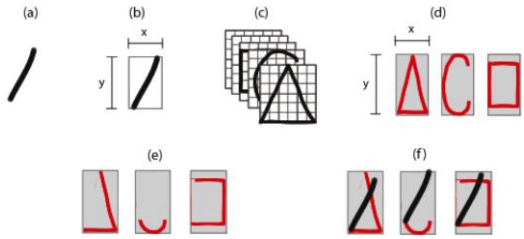

Fig. 12. Gesture Concatenation: (a) Current state of the user gesture is sampled; (b) height, width & length of the sample is measured; (c) the training gesture templates; (d) scale the training templates to match the partial gesture; (e) remove the sampled gesture from each of the scaled templates; (f) a is merged with each of e to produce new templates.

Gesture Recognition Algorithm The gesture recognition algorithm used is a scale and aspect invariant template matching algorithm. The code for which is based on wayV's gesture recognition algorithm [8]. wayV's algorithm accounts for bounding box issues that can arise where the input starts off purely vertical or horizontal.