Investigating Aggregated vs. Sequential Command Recommendation in Graphical User Interfaces

Ben Lafreniere
ben.lafreniere@gmail.com
Reality Labs Research, Meta
Toronto, Ontario, Canada

Zachary J. Davis
zachdavis99@gmail.com
Reality Labs Research, Meta
New York, New York, USA

Michelle Li
michy.li@gmail.com
Reality Labs Research, Meta
Toronto, Ontario, Canada

Junmeng (Andrew) Han
junmenghan@gmail.com
Reality Labs Research, Meta
Toronto, Ontario, Canada

Tovi Grossman
tovi@dgp.toronto.edu
University of Toronto
Toronto, Ontario, Canada

Stephanie Santosa
ssantosa@meta.com
Reality Labs Research, Meta
Toronto, Ontario, Canada

Daniel Wigdor
daniel@dgp.toronto.edu
University of Toronto
Toronto, Ontario, Canada

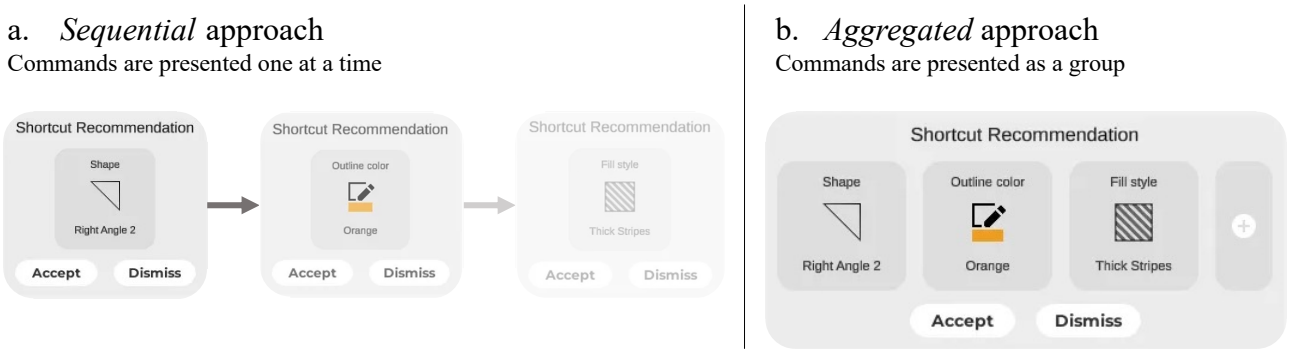

Figure 1: (a) Sequential approach – recommended commands are presented one at a time. Each can be accepted or dismissed by clicking the buttons below the icon, after which the next command appears. Individual commands can be edited via a button that appears when hovering over the command icon. (b) Aggregated approach – all recommended commands are presented at once, and can be accepted or dismissed as a group. Commands can be added via the plus button, and individual commands can be edited or removed via buttons revealed when hovering over the command.

ABSTRACT

Advances in artificial intelligence open the possibility of predicting and recommending sequences of GUI commands to a user. An interesting question raised by this capability is how to present such recommendations to the user – as a *sequential* set of individual command recommendations, or as one *aggregated* recommendation consisting of multiple commands. In this paper we propose an interface for aggregated command recommendation and conduct controlled studies to compare sequential versus aggregated command recommendation across a range of simulated utility conditions. Our results indicate that aggregated command recommendation can improve overall task performance over sequential recommendation, and that this benefit comes from enabling users to rapidly recognize and use high-utility aggregated recommendations. The aggregated command recommendation approach also reduced deliberation time when evaluating and correcting imperfect sets of recommended commands.

GI '25, May 26–29, 2025, Kelowna, BC

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/XXXXXXX.XXXXXXX

CCS CONCEPTS

• **Human-centered computing** → **Human computer interaction (HCI)**; **Empirical studies in HCI**; *Graphical user interfaces*; *Empirical studies in interaction design.*

KEYWORDS

Command Recommendation, Mixed-Initiative Interfaces, Suggestive Interfaces, Automation, Adaptive Interfaces

ACM Reference Format:
Ben Lafreniere, Zachary J. Davis, Michelle Li, Junmeng (Andrew) Han, Tovi Grossman, Stephanie Santosa, and Daniel Wigdor. 2025. Investigating Aggregated vs. Sequential Command Recommendation in Graphical User Interfaces. In *Proceedings of Graphics Interface 2025 (GI '25).* ACM, New York, NY, USA, 13 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

Advances in artificial intelligence (AI) and machine learning promise to automate tasks and unlock new and more helpful forms of assistance. An important question for HCI research, in light of these advances, is how to integrate AI assistance into the user experience of widely-used software, such as desktop applications with graphical user interfaces. One established approach for how AI techniques can support the use of these applications is *command recommendation*, where the system proactively suggests commands to the user [13, 18, 25, 27], based on a model of common workflows in the application or users' workflow histories [4, 31].

Command recommendation is a promising approach to AI assistance in GUI applications because individual commands represent a natural unit of functionality, and existing techniques have been developed to track workflow histories [12] and model and predict command sequences [4, 31]. The ability to predict sequences of commands is particularly compelling because it opens the possibility of AI techniques that assist users by automating higher-level tasks and workflows. However, this raises an interesting question – if a system can predict a sequence of commands to advance a user toward their goal, what is the best interface for presenting these multi-command recommendations to the user? Such an interface needs to support the user in evaluating whether the recommendation is useful, enable the user to make corrections to the commands in the case of imperfect recommendations, and ultimately enable the user to execute the recommended commands.

In this paper we investigate two approaches for presenting recommendations of sequences of commands to a user. In the *sequential* approach the commands are recommended one at a time (Fig. 1a), giving the user an opportunity to accept, edit, or reject each command before the next appears. In contrast, in the *aggregated* approach the commands are recommended as a group (Fig. 1b), allowing the user to evaluate and accept or reject the group as a whole, with the option to make edits to the individual commands that make up the group before doing so. We propose a novel interface design for aggregated command recommendations, and report the results of two studies to investigate how the aggregated vs. sequential approaches perform in terms of task performance and choice to use recommendations versus a static menu alternative. The reported studies also investigate the effects of different levels of simulated recommendation utility (operationalized in this work as how many clicks can be saved by using the recommendations over a static menu alternative), to understand how the quality of recommendations influences use of recommendations and preference for the sequential versus aggregated approach.

The results of a first study demonstrated that, given the choice between a recommender system (sequential or aggregated) and a baseline static menu, participants choose to use the recommendations and realize performance benefits as a result. As compared to the sequential approach, the aggregated interface showed a greater proportion of commands completed through the recommender interface, and also showed a performance advantage in the condition with the highest simulated recommender utility. The aggregated approach also reduced the deliberation time to evaluate and correct a set of recommended commands, as compared to the sequential approach.

Based on the findings from the first study, we refined the two recommender interfaces by adding a visual preview component, and conducted a follow-up study. With the addition of visual previews, the aggregated interface outperformed the sequential interface in conditions with medium and high simulated utility, and showed a trend toward lower task times. A subsequent analysis indicated that much of the benefit of the aggregated approach comes from enabling users to rapidly recognize high-utility recommendations – participants completed trials with perfect recommendations in less than half the time, on average, using the aggregated approach over the sequential approach.

Collectively, our results complement findings from past research on individual command recommendation, and reveal several novel findings related to recommendation of command sequences, namely:

- Aggregated presentation of a recommended set of commands leads to a higher proportion of commands performed through the recommender interface over a fallback static menu, as compared to a sequential presentation;
- Aggregated presentation of a recommended set of commands can reduce deliberation time as compared to sequential presentation of the same set of commands;
- An aggregated command recommendation interface with a well-designed preview mechanism is able to produce performance benefits over a sequential recommendation approach with the same preview mechanism;
- A primary benefit of an aggregated command recommendation approach is that it enables users to rapidly recognize and take advantage of high-utility recommendations.

2 BACKGROUND AND RELATED WORK

This work complements and extends existing work in the areas of proactive suggestions in user interfaces, design guidelines for balancing automation and control, and command and macro recommendations.

2.1 Proactive Suggestions in User Interfaces

HCI research has long sought ways to use intelligent agents or automation to facilitate interactions with graphical user interfaces [17]. An interaction design that has frequently been proposed and investigated for this purpose are *suggestive* or *predictive* interfaces, in which the system makes automatic predictions of the actions the user may want to perform, and presents these actions as suggestions in the interface, which the user can accept or ignore. In the HCI and Graphics research communities, this approach has been investigated for domains such as drawing [33], Computer-Aided Design [25], 3D modeling [18, 24], 3D sculpting [26, 27], data analysis [13, 19], and data visualization [32]. The approach can also be seen in widely-used commercial applications such as Microsoft

PowerPoint.[1] Research on suggestive interfaces has established that the approach can provide multiple benefits, including faster task completion times and promoting learning and discovery of new features [19, 24, 32].

The studies reported in this paper build on this prior work to investigate the design of suggestive interfaces for systems capable of predicting multiple-command sequences, rather than individual commands, as is likely to become increasingly common as AI technologies advance. Specifically, we look at the novel question of whether a system should recommend a sequence of predicted commands sequentially or in aggregate, and the design of user interfaces for this purpose.

2.2 Control vs Automation in User Interfaces

Early work on suggestive interfaces by Igarashi in the domain of 3D drawing suggested that a *purely* suggestive interface is not very practical, and that this approach is best combined with traditional interfaces [18]. This was reinforced by Guo et al. who showed that users often ignored proactive suggestions that were not preceded by an initiating user action (e.g., selecting some items in the user interface) [13]. These findings suggest a need to carefully consider how the presentation and use of suggestions fits into the broader workflow of the user in an application.

How to integrate automation into interactive systems is also a long-standing topic of interest for the HCI community. Foundational principles of how to couple user agency and automation in user interfaces were articulated by Eric Horvitz in his *Principles of Mixed-Initiative User Interfaces* [17], and recently there has been a resurgence of interest in the topic [1, 15, 28]. For example, Amershi's *Design Guidelines for Human-AI Interaction* specifically call out the need to support efficient correction when an AI system is wrong [1]. Another important consideration called out by Jeffrey Heer [15], based on a large body of work in this space, is the need to develop the right *shared representation* that permits both human users and AI systems to contribute to solving problems. For suggestive interfaces, one goal is to present suggestions to users so that they can be easily evaluated. In many of the visual domains in which this approach has been studied, this is accomplished via visual previews [18, 22, 26, 27]. Visual previews have been shown to be particularly effective for supporting users engaged in open-ended tasks in graphical user interfaces [29].

Our work leverages the design guidelines from the existing body of research on Human-AI interaction, with particular attention on providing visual previews and supporting quick user modification when suggesting aggregated groups of commands.

2.3 Command and Macro Recommendations

The graphical user interfaces of desktop applications are typically built around a 'command' metaphor – operations carried out by the user are represented as a series of commands tracked by the system, providing the user with the ability to undo the effects of recent commands. Prior work has advocated for more sophisticated tracking and use of command histories, for example as a way to share or compare workflows [7, 12, 20], to enable the automatic generation of tutorials [5, 11], or to recommend commands and other learning materials [24, 25, 31]. Early work by Kurlander on the Chimera system proposed the capability of turning multi-command sequences from a document history into reusable macros [21], and subsequent work has extensively explored the space of mechanisms for producing and sharing macros [2, 22, 23, 34] and approaches for generalizing or personalizing macros [3, 11, 23, 34]. Perhaps most relevant to our own work is the DiscoverySpace system, which suggests task-level action macros for image-editing tasks [10]. In DiscoverySpace, macros are recorded and shared by the user community. Visual previews of existing macros are displayed, representing a sequence of operations that can be executed in one click. However, with this system, there was no mechanism for subsequent users to edit the sequential actions. A main finding from their study was the desire for more control over the recorded macros and their effects.

The present work is related to this prior work on multi-command macros. However, a major distinction is that macros are typically human created, and shared along with some context about what they do, through which a user can judge their correctness for their purpose. Multi-command recommendations produced by AI are different in that the interface must enable the user to rapidly decide whether to accept or ignore the recommended commands, and permit the user to make adjustments to the commands or their parameters if the recommendation is close but not quite right. Furthermore, prior work has not formally compared sequential and aggregated approaches when suggesting groups of actions, and so the design trade-offs are not currently well understood. The studies presented in this paper are intended to address these gaps.

3 STUDY SYSTEM

3.1 Aggregated and Sequential Command Recommendation Interfaces

We developed a novel interface for presenting aggregated command recommendations (see Fig. 2). The interface represents the recommended set of commands as a series of tiles, with each tile representing an individual command recommendation. Clicking 'Accept' performs all of the commands, clicking 'Dismiss' dismisses the interface without performing any commands (Fig. 2a). Hovering over an individual command tile displays 'Edit' and 'Remove' buttons (Fig. 2b). Selecting 'Remove' deletes the command from the set of aggregated commands. Selecting 'Edit' opens a hierarchical context menu at the location of current command (e.g. the menu would offer alternate colors when editing a command to set a shape's outline color, Fig. 2c). Commands can be added by clicking a plus button to bring up a modal hierarchical menu from which a command can be selected (Fig. 2d).

To study aggregate vs. sequential presentation, we designed a comparable interface for sequential presentation of command recommendations (see Fig. 3a). In the sequential interface only one tile is presented at a time, but otherwise users have the same options of accepting, dismissing, or editing the command recommendation as in the aggregated interface (Fig. 3b). New commands cannot be added to the current recommendation. The next command recommendation is presented only after accepting or dismissing the current command.

[1] https://support.microsoft.com/en-us/office/create-professional-slide-layouts-with-designer-53c77d7b-dc40-45c2-b684-81415eac0617

a. The aggregated interface presents multiple command recommendations as a group of tiles. The group can be accepted or dismissed.

b. Hovering over an individual tile displays the option to edit or remove that command recommendation.

c. Selecting "Edit" opens a menu displaying alternate parameter values.

d. Selecting "Add" enables a user to add a new command.

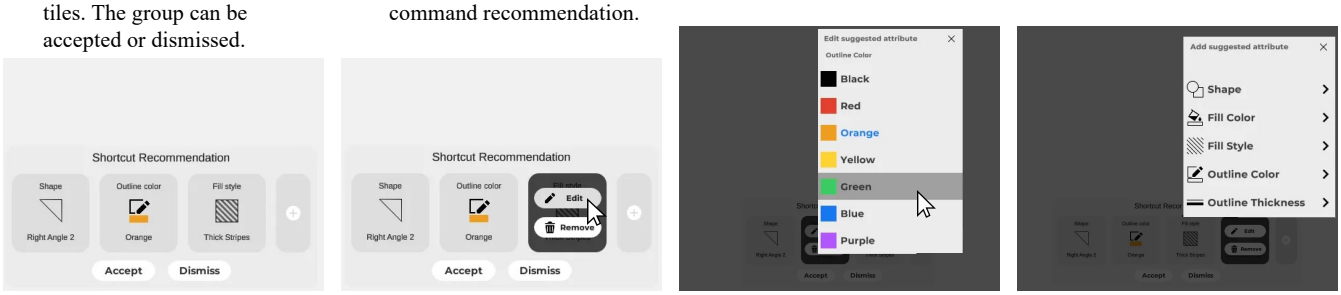

Figure 2: Aggregated interface. Multiple recommendations are presented simultaneously. Participants have the option of editing, removing, or adding new individual commands. Selecting accept executes all commands, dismiss removes all commands.

a. The sequential interface presents recommendations one at a time. Each recommendation can be accepted, dismissed, or edited.

b. Hovering over the tile and selecting "Edit" opens a menu displaying alternate parameter values.

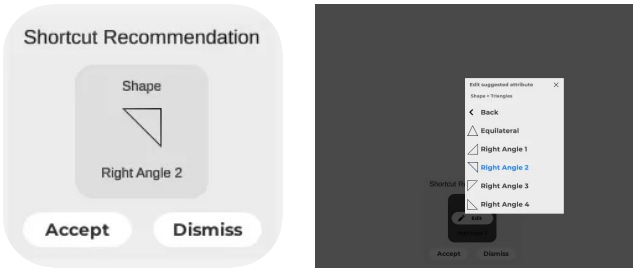

Figure 3: Sequential interface. command recommendations are presented individually, with the option to accept, dismiss, and/or edit the current recommendation. New recommendations are displayed only after accepting or dismissing the current recommendation.

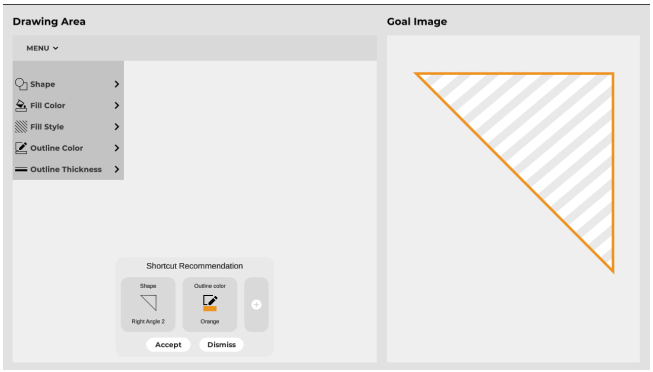

Figure 4: Study system interface. Left: Drawing area. Right: Goal image. The static menu is opened in the top-left of the drawing area, and aggregated "shortcut recommendations" are presented at the bottom center. The recommendations correctly identify the shape and outline color of the goal image, but do not correctly recommend that the image fill style be thick stripes.

3.2 Image Reconstruction Task

An image reconstruction task was developed to allow for controlled study of the aggregated vs. sequential command recommendation interfaces. We chose an image reconstruction task because it allowed us to present the desired end goal to the user as an image, which forced the user to engage in some planning of how and what commands to use to achieve that end goal. An alternative approach of presenting the user with a prescribed list of commands to perform could have undesirable effect of priming participants to memorize or cross-reference these instructions, harming ecological validity.

Fig. 4 shows the study system designed for the image reconstruction task. The video figure included with this paper also includes a demonstration of the study system. At the beginning of the trial a goal image was presented, with five features set per image (shape, fill color, fill style, outline color, and outline thickness). The goal images were generated by starting with a white square with solid fill and black thin outline, and randomly choosing a number of modifications to these default features. For example, Fig. 4 shows a goal image with modified shape (triangle), fill style (thick stripes), and outline color (orange) but the fill color (white) and outline thickness (thin) remain at the default value. A non-default shape was always selected as one of the modifications to the goal image.

Participants were given two methods to execute commands to recreate the goal image. The "Drawing Menu" (top left of the drawing area in Fig. 4) contained all commands and parameters in a static hierarchical menu. See Appendix A for the complete menu hierarchy. The "Shortcut Recommendations" (bottom of drawing area in Fig. 4) were dynamically generated according to a process that differed slightly by experiment. Participants had the option to accept, edit, or dismiss recommendations. Selecting "Edit" on a recommended feature value opened a list of alternate values for that feature (e.g., alternate outline colors). In the case of editing shape values, participants could navigate a hierarchical menu of shapes. When recommendations were aggregated, participants also had the option to add/remove actions to the aggregated recommendations. The order in which recommended commands were presented was randomized, except that shape recommendations (e.g., set shape to circle) were always presented first – we reasoned that participants

would find it confusing to get a recommendation to change the outline or fill of a shape before the shape itself had been set.

3.2.1 Generating Command Recommendations. An important consideration for recommender interfaces is how the user experience changes with the quality of recommendations. In particular, command recommender models may make errors in recommendations, reducing their benefit. In our task we simulate three types of errors a recommendation model could make, and generate recommendations that may include a combination of these errors, to manipulate how many clicks the recommendations can save over using a static menu alternative (i.e., the *utility* of the recommendation). To generate recommendations of varying utility, we started with a perfect aggregated recommendation (i.e., one including all the commands to re-create the target image in one click) and randomly add one of three error types. A "missing action" error removed a command, requiring the participant to either add a new command or use the static menu to execute the missing command. A "wrong parameter action" error selected an incorrect value for a parameter, e.g. choosing a red outline color when the goal image has a green outline. An "unnecessary action" error added a command to set an additional non-default value to the goal image, e.g. adding stripes to the fill style when the goal image has a solid fill. Any error type could be applied to any image feature, except that shape recommendations were never missing (though wrong shapes could be recommended). The above process was repeated to produce recommendations of a desired level of utility.

3.3 Participation Requirements

Participants in Experiment 1 and 2 were recruited on Amazon Mechanical Turk, an online crowdsourcing platform for human intelligence tasks [6]. We only recruited participants who had completed more than 1,000 tasks at a 96% approval rate, and were from an approved list of US States.[2] Repeat participation was not allowed: participants who completed Experiment 1 were excluded from the pool for Experiment 2.

Participants on Mechanical Turk have been shown to occasionally not fully engage with the task [14]. We excluded participants in Experiments 1 and 2 who failed to select the seventh option in an attention check in a post-task debriefing questionnaire ("Could you please select the seventh option on this question, corresponding to a judgment of very high?"). Experimental runs where the participant failed this attention check were re-run with a new participant, with the data for the failed run removed.

4 EXPERIMENT 1

Using the study system described above, we designed an experiment to compare graphical user interfaces with recommendations against those without, and to better understand the differences between sequential and aggregated approaches to sequences of command recommendations. The study was designed to investigate several research questions related to aggregating command recommendations. Firstly, we aim to establish whether aggregating recommendations influences task performance, and whether task performance in the interfaces differ across levels of recommendation quality. We also investigate whether aggregating command recommendations influences use of recommendations when users have a fallback option of a fixed menu. Then we turn to some measures of action-by-action behavior to see whether the interfaces differ in how easy it is for participants to correct errors in recommendation.

4.1 Methods

4.1.1 Participants. We recruited 210 participants (118 male, 91 female, 1 other) on Amazon Mechanical Turk, ranging in age from 21 to 67 (median 36). 31 experimental runs were re-run because the participant failed the attention check, leaving a total of 208 participants after filtering out the two re-run participants who also failed the attention check.[3] Participants were compensated $7.50 for a mean of 23 minutes of participation (SD=13).

4.1.2 Conditions. We tested three approaches to recommendation interfaces. The first "static menu" interface did not provide any recommendations, leaving participants with only the drawing menu to complete the goal image. The "sequential" interface provided recommendations one at a time, and the "aggregated" interface provided all recommendations simultaneously. The static drawing menu was available for use in all conditions.

To determine if task performance or interface use differed across levels of recommendation quality, we simulated recommendation systems at three levels of *utility*. As mentioned in Sec. 3.2.1, we operationalized utility as the number of clicks that could be saved by using the aggregated recommender interface to perform commands instead of the static menu. Participants were assigned to either a low, medium, or high utility condition. To generate a distribution of varying utility, for each trial a utility value was sampled from a normal distribution centered at the participant's target utility (M for low = 3, medium = 6, high = 9, all SD = 3). Degradations to a perfect set of command recommendations (i.e., one that would complete the target image in a single click in the aggregated presentation) were applied until the trial utility was less than or equal to the sampled utility value (see Sec. 3.2.1 for details on degradation types and how recommendations were generated). If the sampled utility value was greater than that of a perfect aggregated recommendation, no degradations were applied.

4.1.3 Study Design. Participants completed 40 trials, with a 15 second break after every ten trials. We employed a between-subjects design with 30 participants assigned to each of seven conditions. One of the seven conditions was the static menu interface, where participants completed the goal images solely through the static drawing menu without being provided with command recommendations. The remaining six conditions were a cross of sequential vs. aggregated interfaces with either low, medium, or high utility recommendations.

To ensure that differences between conditions were not the result of idiosyncratic differences in goal images or recommendations, for each participant using the aggregated interface there was a corresponding participant using the sequential interface that received an

[2] AL, AR, DE, FL, GA, IA, KS, KY, LA, MD, MN, MS, MO, NE, ND, OK, SD, SN, TN, TX, VA, WV

[3] Given the additional time it would take to re-deploy and monitor an additional batch of runs, we opted to stop at 99% participant coverage.

identical sequence of goal images and command recommendations. We achieved this by first generating 40 goal images and aggregated recommendations, and then generating from these a corresponding series of sequential recommendations for the sequential condition.

Using a similar approach, 30 of the 90 participants in the aggregated interface condition (10 for each utility level) were randomly selected to have an additional corresponding participant in a static interface only condition. These participants received the same goal images in the same order as the participant using the aggregated interface, but received no command recommendations.

4.1.4 Study Procedure. Prior to starting the study, participants were asked to provide informed consent and fill out a short demographics questionnaire. Participants were then presented with a step-by-step tutorial that introduced the study system and task, and the interfaces that would be available to them. This tutorial included visual demonstrations of the static menu and each of the features of the recommender interface for the participants' condition. This was followed by the main data collection portion of the experiment, as described above. Finally, the study ended with a post-task debriefing questionnaire.

4.2 Results

To account for participants multitasking or taking unscheduled breaks during trials, we filtered out outlier trials where the trial completion time was greater than three times the interquartile range (25th-75th) from the median for each participant, resulting in 6% of trials being removed.

4.2.1 Task Performance (Clicks). Fig. 5 shows a comparison of the number of clicks performed to complete a trial between the interface conditions. We first compared just the participants who received the same sequence of goal images as the 30 participants in the static interface condition (see the Study Design section above for details on the matching procedure). A one-way ANOVA comparing the number of clicks performed to complete tasks found significant differences between interfaces ($F(85, 2) = 16.75$, $p < 0.001$), with Tukey's HSD posthoc tests revealing that the static interface required more clicks to complete a trial than the aggregated interface (difference $M = 3.76$, $p < 0.001$) and sequential interface (difference $M = 1.63$, $p = 0.041$). Within this group, participants using the aggregated interface completed trials with fewer clicks than participants using the sequential interface (difference $M = 2.11$, $p = 0.005$).

To investigate how performance varied as a function of recommendation utility, we performed an Analysis of Covariance comparing the effect of aggregated and sequential interfaces crossed with the recommender quality on the number of clicks performed to complete a trial. The ANCOVA revealed a significant main effect of interface ($F(169, 1) = 5.16$, $p = 0.024$). A Bonferroni corrected test between interfaces for each recommender quality level showed that this result was driven by differences between the aggregated and sequential interfaces in the high utility condition (difference $M = 2.32$, adjusted $p = 0.003$). There were no significant differences in the low utility (difference $M = -0.20$, adjusted $p = 0.85$) or medium utility (difference $M = 0.34$, adjusted $p = 0.53$) recommender quality conditions.

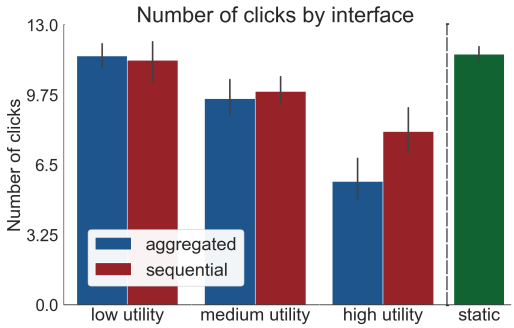

Figure 5: Results from Experiment 1. Participants completed the task in significantly fewer clicks when receiving recommendations (aggregated or sequential) than a static menu alone. Aggregating recommendations was of particular value over sequential presentation for high utility recommendations (i.e., those that most closely matched the goal images).

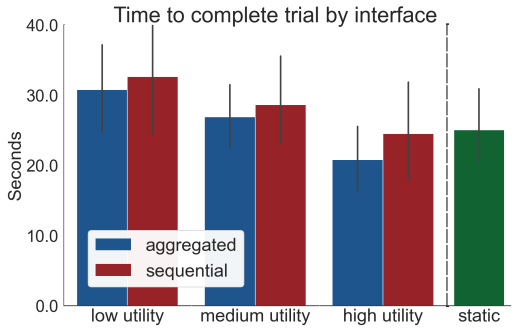

Figure 6: Results from Experiment 1. No significant differences were found in time to complete trials.

4.2.2 Task Performance (Time). Another important measure of task performance is the amount of time participants take to complete a trial. We first compared just the participants who matched the 30 participants in the static menu condition – i.e. the participants using the aggregated and sequential interfaces who received the same recommendations as the 30 participants that used the static menu interface. An Analysis of Covariance did not show significant main effects in the amount of time to complete a trial ($F(85, 2) = 1.22$, $p = 0.302$). There were also no significant differences between participants that used the sequential or aggregated interfaces ($F(169, 1) = 0.71$, $p = 0.399$), see Fig. 6.

4.2.3 Recommendation Use. We hypothesized that aggregating recommendations would influence the way that participants interact with interfaces that include both dynamic recommendations and a static menu. To investigate this question we evaluated the proportion of required actions in each trial that were completed through the static menu as compared to through accepting recommendations (edited or not). We used the last action taken to manipulate each feature to identify which component of the interface (static menu vs recommender interface) was used to complete that action.

When recommendations were aggregated, 58% of actions were completed through the recommendation interface, as compared to 40% for participants using the sequential interface (independent

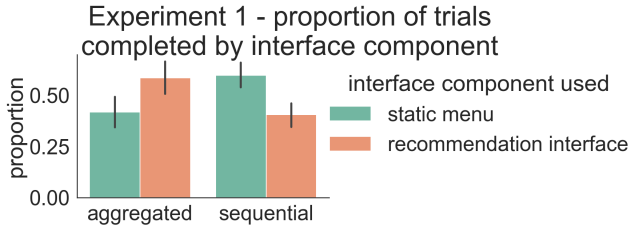

Figure 7: Results from Experiment 1. Participants using the aggregated interface completed a higher proportion of the image using the recommendation interface.

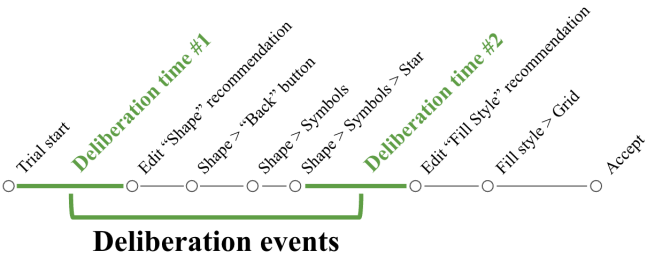

Figure 8: Sample illustration of a time course of participant actions. Deliberation time was operationalized as the amount of time taken before editing a recommendation.

t(175) = 3.61, p < 0.001), see Fig. 7. This may have been the result of participants in the sequential recommendation condition dismissing bad recommendations instead of correcting them. Over 40 trials, participants receiving sequential recommendations dismissed 9 command recommendations on average as opposed to 5.5 individual command recommendations in the aggregated condition (dismissing an aggregated recommendation of three commands was counted as three dismissals), t(175) = -2.32, p = 0.021.

4.2.4 Deliberation Time. One measure of the ease of use of a system is the ability for a user to plan future interactions, also known as "anticipatory planning" [16]. A behavioral signal for anticipatory planning is decreasing deliberation time between subsequent actions, as the participant has already planned out the next action and therefore does not need to pause after executing the current action to determine the next step. We operationalize the deliberation time between actions as the amount of time that a user takes after completing some other action and before opening the edit recommendation menu (see Fig. 8 for illustration of deliberation periods). In the aggregated condition the beginning of a deliberation period may involve edit or removal actions on other command recommendations or use of the static menu. In the sequential condition this may involve acceptance or dismissal of a prior recommendation or use of the static menu.

Fig. 9 shows a steady decrease in deliberation time over subsequent interactions in the aggregated interface conditions that is not present for the sequential interface conditions. This was confirmed by a significant interaction of condition by recommendation edit session on deliberation time in a regression analysis (r(557) = 0.50, p < 0.001).

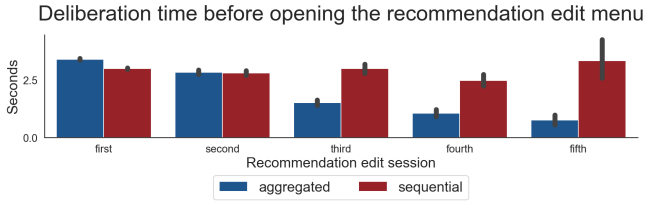

Figure 9: Results from Experiment 1. Deliberation time between subsequent recommendation clicks steadily decreased for aggregated recommendations, but not for sequential recommendations.

4.3 Discussion

4.3.1 Recommendations vs. Static-Menu. Participants were able to complete the image recreation task in fewer clicks when provided with command recommendations than when their only option was to use a static menu. Despite the benefit in saving clicks, we did not observe faster task completion times when command recommendations were available, regardless of recommender interface and utility level. There are several potential explanations for this result. First, it may be that the command recommender interfaces are imposing additional task time, e.g. to assess the presented recommendations, to decide whether to use the recommender interface, or simply by increasing the overall interface complexity, which is outweighing the time savings of fewer clicks. Second, users are familiar and well practiced at using hierarchical menus, whereas the command recommender interfaces are novel, which may have increased the time to use them. Finally, it may be that the command recommender interfaces are providing a benefit for task time in some cases, but it is not apparent in the average trial time data. Further work is needed to investigate these possibilities, but for now our results show that command recommendation can provide a performance benefit by saving interface actions, which suggests these interfaces could be valuable in situations where interface actions are time consuming or effortful to perform.

In addition to the performance benefit, it is encouraging that users chose to use the recommender interface over the static menu for the majority of trials (58%) in the aggregated condition, and a high proportion of trials (40%) in the sequential condition.

4.3.2 Aggregate vs. Sequential Recommendations. When directly comparing the aggregate versus sequential approaches for presenting command recommendations, participants using the aggregated interface completed the task in fewer clicks than when using the sequential interface. As well, a higher proportion of the required actions in the trial tasks were completed through the recommender interface over the static menu when command recommendations were aggregated, which may suggest that aggregating command recommendations can increase engagement with the recommendation interface or that sequential presentation may leave more opportunities for users to switch back to using the static menu.

While we did not find an overall difference in trial completion time between the aggregate and sequential interfaces, participants using the aggregated interface exhibited a decline in deliberation time between subsequent corrections to recommendations. This effect is a hallmark of anticipatory planning, a behavioral pattern

exhibited in tasks such as motor planning where participants who receive information about multiple goals simultaneously are able to plan and execute an action sequence more efficiently than when each goal is presented sequentially [8, 9, 16, 30]. In our task, this suggests that aggregating command recommendations allows participants to identify recommendation errors and use that understanding to plan and efficiently execute corrections. In contrast, those using sequential recommendations must assess each recommendation's quality afresh, eliminating their ability to plan corrections to future, unseen, recommendation errors.

4.3.3 Summary. Results from Experiment 1 suggest that there are benefits to aggregating command recommendations in reducing the number of actions required to complete a complex task, increasing engagement with system recommendations, and providing the ability to plan corrections to recommendation errors. In Experiment 2 we investigate the hypothesis that a visual preview of the system's recommended actions will help participants to more rapidly understand and respond to aggregated command recommendations.

5 EXPERIMENT 2

The results from Experiment 1 suggest that aggregating command recommendations improves anticipatory planning, but there is a possibility that this benefit is counterbalanced by additional time or cognitive effort to assess the group of commands when they are first presented. In Experiment 2 we investigate this possibility by providing participants with a visual preview of the effects the system's command recommendations (see Fig. 10a). We hypothesized that a visual summary would reduce the cognitive effort to assess the group of commands. A decreased assessment cost, complemented with the increased ability to plan, may result in improved overall performance for the aggregated command recommendation interface over the sequential.

Fig. 10 shows the updated study system. Participants in the aggregated interface were given the option to "Accept", "Edit", or "Dismiss" a visual preview (Fig. 10a). The visual preview's "Accept" and "Dismiss" options functioned as in Experiment 1, executing or dismissing all commands included in the recommendation. Selecting "Edit" opened an expanded view that looked similar to the aggregated command recommendation pane from Experiment 1 (Fig. 10b). A visual summary tile was added to the top right of the expanded view to match the demonstration in the visual preview. As in the visual preview, this tile showed a preview of the result if the currently displayed command recommendation(s) were executed.

The sequential interface was unchanged from Experiment 1 except for the addition of a visual summary of the effect that executing the current command would have on the image in the drawing area (see Fig. 11). This was done for consistency between the sequential and aggregated conditions.

5.1 Methods

5.1.1 Participants. We recruited 180 participants on Amazon Mechanical Turk, ranging in age from 20 to 64 (median 34). 43 experimental runs were re-run because the participant failed the attention check, leaving a total of 176 participants after filtering out four of the re-run participants who also failed the attention check.[4] Participants were compensated $7.50 for a mean of 27 minutes of participation (SD=12).

5.1.2 Conditions. As in Experiment 1 participants were assigned to recommendation systems of low, medium, or high utility, corresponding to a mean of 6, 9, or 12 clicks saved when using a perfect aggregated recommendation to complete the image instead of the static menu (all standard deviations = 5). Goal images were generated as in Experiment 1, except that four non-default values were selected (as opposed to three in Experiment 1). We hypothesized that more complex goals would magnify differences between aggregated and sequential command recommendations. Unlike Experiment 1, we did not include a static-menu only condition, because our focus was on investigating the aggregated versus sequential interfaces.

5.1.3 Study Design. A between-subjects experimental design was used crossing aggregated and sequential interfaces by low, medium, and high recommendation system utility. 30 participants in each of the six conditions performed 40 trials, with a 15 second break every 10 trials. As in Experiment 1, every participant receiving aggregated recommendations had a matching participant who received an identical sequence of goal images and sequential recommendations.

5.1.4 Study Procedure. The study procedure was unchanged from Experiment 1, but the pre-study tutorials were modified to include the visual preview features added to the recommender interfaces.

5.2 Results

5% of trials were removed as outliers because the time to completion was greater than three times the interquartile range (25th-75th) from the median for that participant.

5.2.1 Task Performance (Clicks). As in Experiment 1 we analyzed the overall number of clicks required to complete a trial (see Fig. 12). An ANCOVA crossing interface type by recommender quality revealed a significant main effect of interface ($F(166, 1) = 22.56$, $p < 0.001$). Bonferroni corrected t-tests between interfaces for each recommender quality level showed that this result was driven by differences between the aggregated and sequential interfaces in the medium- and high-utility conditions (medium-utility difference $M = 2.28$, adjusted $p < 0.001$; high-utility difference $M = 2.99$, adjusted $p = 0.002$). A significant difference was not found in the low-utility condition (difference $M = 0.70$, adjusted $p = 0.26$).

5.2.2 Task Performance (Time). An Analysis of Covariance crossing interface with recommender quality showed a marginal main effect of interface on the amount of time to complete a trial, with participants using the aggregated recommendations completing trials in 29.8 seconds on average as compared to 33.9 seconds for those using the sequential interface ($F(166, 1) = 3.75$, $p = 0.054$), see Fig. 13.

5.2.3 Recommendation Use. We compared the relative use of recommendations versus the static menu to complete actions. When

[4] Given the additional time it would take to re-deploy and monitor an additional batch of runs, we opted to stop at 98% participant coverage.

a. The visual preview for the aggregated interface combines multiple command recommendations into a single image. The preview can be accepted, dismissed, or edited.

b. Selecting "Edit" on the visual preview displays an aggregated panel of the individual commands with a summary in the top right corner. The set of commands can be interacted with in the same manner as in Experiment 1 (see Fig. 2).

c. The summary tile in the top right reflects changes in commands. In this example it reflects an addition of a blue fill color command.

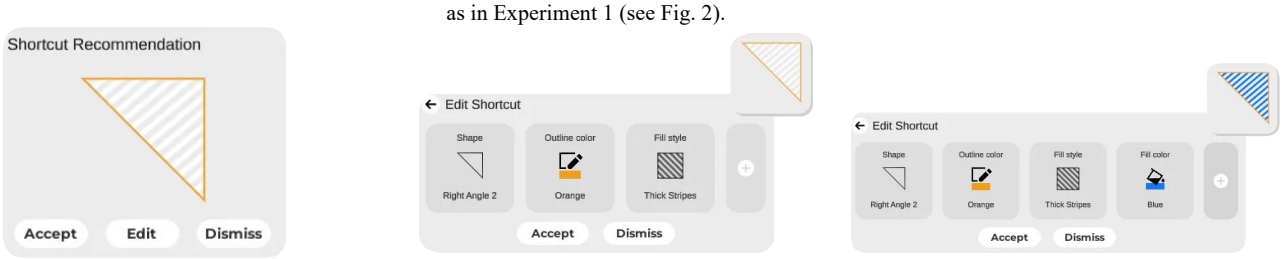

Figure 10: Aggregated interface for Experiment 2. (a) Participants are presented with a visual preview of the effect of the recommended set of commands. (b) Selecting "Edit" on the preview displays the aggregated interface from Experiment 1, with an additional visual preview. (c) Adding a new command updates the visual preview.

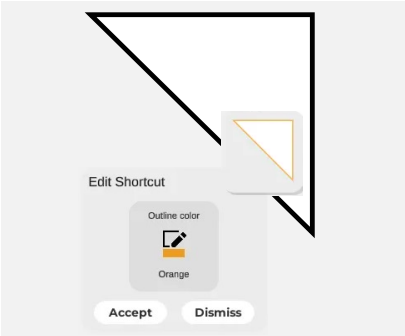

Figure 11: Sequential interface for Experiment 2. Command recommendations are provided individually. A visual preview shows the effect of executing the current command on the current state of the drawing area.

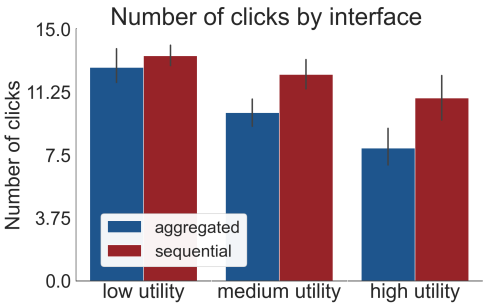

Figure 12: Results from Experiment 2. Participants completed the task in significantly fewer clicks when receiving aggregated recommendations (p = 0.002).

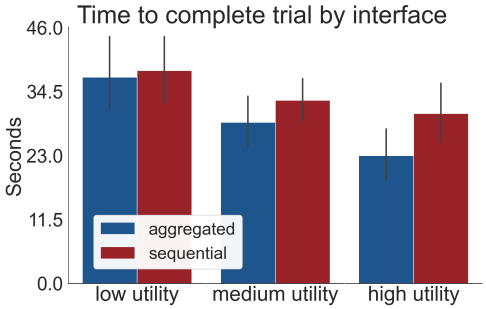

Figure 13: Results from Experiment 2. Participants trended towards completing the task in less time when receiving aggregated recommendations (p = 0.054).

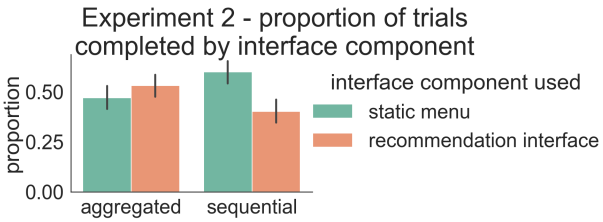

Figure 14: Results from Experiment 2. Participants using the aggregated interface completed a higher proportion of the image using the recommendation interface.

command recommendations were aggregated participants completed actions through the recommendation interface significantly more often than when recommended commands were presented sequentially (53% vs 40%, t(173) = 3.11, p = 0.002). Unlike in Experiment 1, we did not observe a difference between conditions in the number of dismissed command recommendations (t(173) = 0.5, p = 0.615).

5.2.4 Deliberation Time. As in Experiment 1, we analyzed the deliberation time between subsequent interactions with recommendations. We replicated the steady decrease in deliberation time over subsequent edit actions observed in Experiment 1, as confirmed by a regression of the edit session on deliberation time (r(466) = 0.37, p < 0.001), see Fig. 15.

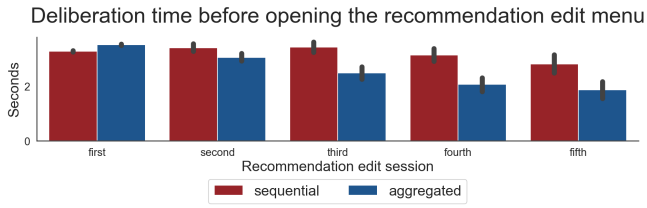

Figure 15: Results from Experiment 2. As in Experiment 1, deliberation time between subsequent recommendation clicks steadily decreased for aggregated recommendations, but not for sequential recommendations.

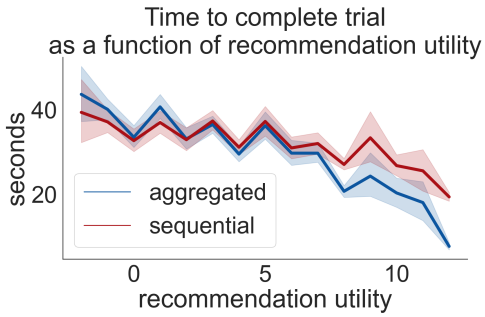

Figure 16: Results from Experiment 2. Participants receiving aggregated recommendations completed trials more quickly when those recommendations had high utility. Note that a recommendation with 12 utility is perfect: accepting it with no edits completes the goal image.

5.2.5 Perfect Recommendation Effect. We hypothesized that the relative benefit of aggregated recommendations over sequential would increase with the utility of the set of commands being recommended, and that a preview mechanism could enhance this by reducing the time to evaluate the aggregated set of commands. We tested this hypothesis by comparing the time to complete a trial based on interface and per-trial recommendation utility (i.e., the number of clicks saved over completing the image through the static menu). There was a significant interaction between interface and recommendation utility on time to complete the trial ($F(6275, 1) = 141.14$, $p < 0.001$). This supports our hypothesis. Furthermore, we observe that the effect was largely driven by significantly faster trial completion times for recommendations with utility of 8 or higher, as can be seen in Fig. 16. This suggests that the benefit comes from enabling users to rapidly recognize high utility aggregated recommendations.

5.3 Discussion

Participants receiving aggregated command recommendations with a preview mechanism completed their task with fewer clicks than when those same recommendations were presented sequentially. Aggregated command recommendations also resulted in a higher proportion of tasks being completed through the recommendation interface rather than the static menu, as compared to sequential command recommendations.

Unlike Experiment 1, we observed a marginal benefit to the aggregated interface in the time taken to complete trials. This effect was driven by a dramatic benefit for aggregated recommendation interface when the recommendations closely matched the goal image. Participants completed trials in less than half the time when perfect recommendations were presented in the aggregated interface (8 vs 19 seconds). Moreover, this benefit for trials where recommendations had high utility did not come with decreased performance in low utility trials, perhaps because of the aggregated interface's benefit in decreased deliberation time when correcting errors in recommendations.

Collectively, results from Experiment 2 suggest that aggregated presentation of command recommendations can help users accomplish their tasks in fewer clicks and encourage the user to interact with the system through the command recommendation interface. Moreover, the benefit of the aggregated presentation appears to come from enabling users to recognize and take advantage of high utility recommendations.

6 GENERAL DISCUSSION

Experiments 1 and 2 investigated two approaches to recommending a set of commands. A sequential approach involves presenting command recommendations one at a time, holding future command recommendations until the participant has accepted or dismissed the current one. In contrast, an aggregated approach presents all command recommendations together, permitting the user to accept or dismiss them as a group. We investigated how these approaches to command recommendation presentation affect user experience in an image recreation task across two studies.

Experiment 1 showed that participants were able to use command recommendations to accomplish tasks with fewer interface actions than when they only had access to a static menu. Experiments 1 and 2 built on this finding by investigating how different methods of presenting identical sets of command recommendations influenced use of those recommendations. Across both experiments, participants required fewer clicks to accomplish their tasks when command recommendations were presented through an aggregated interface than when the same commands were presented sequentially. Aggregated command recommendation also increased participant engagement with system recommendations, with participants in the aggregated condition completing a higher proportion of trials through the recommender interface than the static menu.

While Experiment 2 showed only a marginal improvement in time to complete tasks for the aggregated interface (and no differences were observed in Experiment 1), there were strong effects of aggregating command recommendations on other measures of task performance. Participants in both experiments exhibited a significant decline in deliberation time between corrections to recommendations in the aggregated condition, suggesting that the aggregated presentation enabled users to efficiently plan corrections to errors in the recommended set of commands. The visual preview introduced in Experiment 2 also allowed participants to rapidly identify high quality recommendations and accept them, resulting in task completion in less than half the time for perfect sets of commands recommended through the aggregated interface,

as compared to the sequential interface. Importantly, these benefits of did not come with a drawback in task completion time for aggregated recommendations of imperfect sets of commands.

Together, these findings suggest that aggregation be an effective approach for presenting sequences of command recommendations. Across two experiments, participants using an aggregated interface completed tasks with fewer clicks, used system recommendations more often, and efficiently corrected errors in recommendation. Aggregated recommendations were particularly effective when the recommendations had high utility, and when the recommendations were represented in a visual manner easily understood by users.

6.1 Generalization and Limitations

The image recreation task used in our experiments was designed to mimic a task where multiple atomic commands must be performed to reach a goal. For example, editing a photo may require cropping, adjusting brightness, and color correction. However, in many applications performing specific commands may involve a nested set of actions. For example, a color correction command may require manipulating multiple parameters for color balance, saturation, exposure, etc. Future work is required to investigate how to represent and support interactions with commands that include more complex and nested parameters, and how this influences the benefits of aggregated vs sequential recommendations.

Experiment 2 showed that aggregated command recommendation was particularly beneficial with a visual preview mechanism. Visual previews were easy to generate for the image recreation task, but may be more complex to generate in other domains. For example, recommendations in an interface like Microsoft Word may involve changing header font sizes, correcting spelling and grammar, and creating a Table of Contents. These edits may be difficult to combine into a concise and easy-to-consume visual representation. Similarly, in our studies executing actions had immediate consequences that could be visually checked against the goal. A user may not find out until much later if there was a mistake in a spell check or if an automatic reference incorrectly formatted a citation. The results of this paper point to the value of designing interfaces that provide easily graspable representations of sets of action command sets, but how to achieve this in additional domains is an interesting open question.

In our task, executing an incorrect command could be fixed by using the static menu, but this may not be possible in all domains. For example, making an erroneous purchase or acting on incorrect navigation directions may require very different actions and resources to resolve. Future investigations are needed to investigate the effect of different classes of errors on user behavior and use of command recommender interfaces.

We also acknowledge that there are a number of limitations of the presented experiments, which should be addressed in future work. First, the command recommendations in our studies were simulated and pre-generated, to tightly control their utility. It remains an open question to study the aggregated vs sequential approach when powered by a working command recommendation system, embedded in a more realistic high level task. Second, it would be interesting to investigate the ideal number of commands to include in a recommendation, and different criteria or approaches for assembling aggregated recommendations. Third, our studies investigated the choice between static menus versus command recommender interfaces for only one static menu design. It's an open question whether alternative static menu designs (e.g., a right-click contextual menu) might influence the choice to use command recommender interfaces. Finally, the reported experiments were quantitative, and it would be valuable to supplement this with qualitative feedback on the sequential and aggregated recommender interfaces. For example, observational or think-aloud studies could provide deeper insights into how and when users choose to use these different types of command recommender interfaces.

7 CONCLUSION

This paper has contributed an interface for aggregated command recommendation, and through two experiments has empirically established several benefits to the aggregated recommendation approach. This is a first step toward understanding the cognitive mechanisms involved when users assess and use aggregated command recommendations, and toward answering the broader question of how best to design command recommendation mechanisms to best support users. Together, we see this work as a step toward harnessing advances in artificial intelligence to improve the user experience of graphical user interfaces.

REFERENCES

[1] Saleema Amershi, Dan Weld, Mihaela Vorvoreanu, Adam Fourney, Besmira Nushi, Penny Collisson, Jina Suh, Shamsi Iqbal, Paul N. Bennett, Kori Inkpen, Jaime Teevan, Ruth Kikin-Gil, and Eric Horvitz. 2019. Guidelines for Human-AI Interaction. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems (CHI '19)*. Association for Computing Machinery, New York, NY, USA, 1–13. https://doi.org/10.1145/3290605.3300233

[2] Lawrence Bergman, Vittorio Castelli, Tessa Lau, and Daniel Oblinger. 2005. DocWizards: a system for authoring follow-me documentation wizards. In *Proceedings of the 18th annual ACM symposium on User interface software and technology (UIST '05)*. ACM, New York, NY, USA, 191–200. https://doi.org/10.1145/1095034.1095067

[3] Floraine Berthouzoz, Wilmot Li, Mira Dontcheva, and Maneesh Agrawala. 2011. A Framework for content-adaptive photo manipulation macros: Application to face, landscape, and global manipulations. *ACM Transactions on Graphics* 30, 5 (Oct. 2011), 120:1–120:14. https://doi.org/10.1145/2019627.2019639

[4] Minsuk Chang, Ben Lafreniere, Juho Kim, George Fitzmaurice, and Tovi Grossman. 2020. Workflow Graphs: A Computational Model of Collective Task Strategies for 3D Design Software. In *Proceedings of Graphics Interface 2020 (GI 2020)*. Canadian Human-Computer Communications Society / Société canadienne du dialogue humain-machine, 114 – 124. https://doi.org/10.20380/GI2020.13 Place: University of Toronto.

[5] Pei-Yu Chi, Sally Ahn, Amanda Ren, Mira Dontcheva, Wilmot Li, and Björn Hartmann. 2012. MixT: Automatic generation of step-by-step mixed media tutorials. In *Proceedings of the 25th annual ACM symposium on User interface software and technology (UIST '12)*. ACM, 93–102.

[6] Matthew JC Crump, John V McDonnell, and Todd M Gureckis. 2013. Evaluating Amazon's Mechanical Turk as a tool for experimental behavioral research. *PloS one* 8, 3 (2013), e57410.

[7] Jonathan D. Denning, William B. Kerr, and Fabio Pellacini. 2011. MeshFlow: Interactive visualization of mesh construction sequences. *ACM Trans. Graph.* 30, 4 (July 2011), 66:1–66:8. https://doi.org/10.1145/2010324.1964961

[8] Kevin C Engel, Martha Flanders, and John F Soechting. 1997. Anticipatory and sequential motor control in piano playing. *Experimental Brain Research* 113 (1997), 189–199. Issue 2. https://doi.org/10.1007/BF02450317

[9] Martin H Fischer, David A Rosenbaum, and Jonathan Vaughan. 1997. Speed and sequential effects in reaching. *Journal of Experimental Psychology: Human Perception and Performance* 23 (1997), 404–428. Issue 2. https://doi.org/10.1037/0096-1523.23.2.404

[10] C. Ailie Fraser, Mira Dontcheva, Holger Winnemöller, Sheryl Ehrlich, and Scott Klemmer. 2016. DiscoverySpace: Suggesting Actions in Complex Software. In *Proceedings of the 2016 ACM Conference on Designing Interactive Systems (DIS '16)*. Association for Computing Machinery, New York, NY, USA, 1221–1232. https://doi.org/10.1145/2901790.2901849

[11] Floraine Grabler, Maneesh Agrawala, Wilmot Li, Mira Dontcheva, and Takeo Igarashi. 2009. Generating photo manipulation tutorials by demonstration. *ACM Trans. Graph.* 28, 3 (July 2009), 66:1–66:9. https://doi.org/10.1145/1531326.1531372

[12] Tovi Grossman, Justin Matejka, and George Fitzmaurice. 2010. Chronicle: capture, exploration, and playback of document workflow histories. In *Proceedings of the 23nd annual ACM symposium on User interface software and technology (UIST '10)*. Association for Computing Machinery, New York, NY, USA, 143–152. https://doi.org/10.1145/1866029.1866054

[13] Philip J. Guo, Sean Kandel, Joseph M. Hellerstein, and Jeffrey Heer. 2011. Proactive wrangling: mixed-initiative end-user programming of data transformation scripts. In *Proceedings of the 24th annual ACM symposium on User interface software and technology (UIST '11)*. Association for Computing Machinery, New York, NY, USA, 65–74. https://doi.org/10.1145/2047196.2047205

[14] David Hauser, Gabriele Paolacci, and Jesse Chandler. 2019. Common concerns with MTurk as a participant pool: Evidence and solutions. In *Handbook of research methods in consumer psychology*. Routledge, 319–337.

[15] Jeffrey Heer. 2019. Agency plus automation: Designing artificial intelligence into interactive systems. *Proceedings of the National Academy of Sciences* 116, 6 (Feb. 2019), 1844–1850. https://doi.org/10.1073/pnas.1807184115 Publisher: National Academy of Sciences Section: Colloquium Paper.

[16] Oliver Herbort and Martin V Butz. 2009. Anticipatory planning of sequential hand and finger movements. *Journal of Motor Behavior* 41 (11 2009), 561–569. Issue 6. https://doi.org/10.3200/35-09-003-RA

[17] Eric Horvitz. 1999. Principles of Mixed-initiative User Interfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '99)*. ACM, New York, NY, USA, 159–166. https://doi.org/10.1145/302979.303030 event-place: Pittsburgh, Pennsylvania, USA.

[18] Takeo Igarashi and John F. Hughes. 2007. A suggestive interface for 3D drawing. In *ACM SIGGRAPH 2007 courses (SIGGRAPH '07)*. Association for Computing Machinery, San Diego, California, 20–es. https://doi.org/10.1145/1281500.1281531

[19] Sean Kandel, Andreas Paepcke, Joseph Hellerstein, and Jeffrey Heer. 2011. Wrangler: interactive visual specification of data transformation scripts. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '11)*. Association for Computing Machinery, New York, NY, USA, 3363–3372. https://doi.org/10.1145/1978942.1979444

[20] Nicholas Kong, Tovi Grossman, Björn Hartmann, Maneesh Agrawala, and George Fitzmaurice. 2012. Delta: a tool for representing and comparing workflows. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '12)*. ACM, New York, NY, USA, 1027–1036. https://doi.org/10.1145/2208516.2208549

[21] David Kurlander. 1993. Chimera: example-based graphical editing. In *Watch what I do: programming by demonstration*. MIT Press, Cambridge, MA, USA, 271–290.

[22] Gierad Laput, Eytan Adar, Mira Dontcheva, and Wilmot Li. 2012. Tutorial-based interfaces for cloud-enabled applications. In *Proceedings of the 25th annual ACM symposium on User interface software and technology (UIST '12)*. Association for Computing Machinery, New York, NY, USA, 113–122. https://doi.org/10.1145/2380116.2380132

[23] Gilly Leshed, Eben M. Haber, Tara Matthews, and Tessa Lau. 2008. CoScripter: automating & sharing how-to knowledge in the enterprise. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '08)*. Association for Computing Machinery, New York, NY, USA, 1719–1728. https://doi.org/10.1145/1357054.1357323

[24] Wei Li, Justin Matejka, Tovi Grossman, Joseph A. Konstan, and George Fitzmaurice. 2011. Design and evaluation of a command recommendation system for software applications. *ACM Transactions on Computer-Human Interaction* 18, 2 (July 2011), 6:1–6:35. https://doi.org/10.1145/1970378.1970380

[25] Justin Matejka, Wei Li, Tovi Grossman, and George Fitzmaurice. 2009. CommunityCommands: command recommendations for software applications. In *Proceedings of the 22nd annual ACM symposium on User interface software and technology (UIST '09)*. Association for Computing Machinery, New York, NY, USA, 193–202. https://doi.org/10.1145/1622176.1622214

[26] Mengqi Peng, Li-yi Wei, Rubaiat Habib Kazi, and Vladimir G. Kim. 2020. Autocomplete Animated Sculpting. In *Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology (UIST '20)*. Association for Computing Machinery, New York, NY, USA, 760–777. https://doi.org/10.1145/3379337.3415884

[27] Mengqi Peng, Jun Xing, and Li-Yi Wei. 2018. Autocomplete 3D sculpting. *ACM Transactions on Graphics* 37, 4 (July 2018), 132:1–132:15. https://doi.org/10.1145/3197517.3201297

[28] Quentin Roy, Futian Zhang, and Daniel Vogel. 2019. Automation Accuracy Is Good, but High Controllability May Be Better. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems (CHI '19)*. Association for Computing Machinery, New York, NY, USA, 1–8. https://doi.org/10.1145/3290605.3300750

[29] Michael Terry and Elizabeth D. Mynatt. 2002. Side views: persistent, on-demand previews for open-ended tasks. In *Proceedings of the 15th annual ACM symposium on User interface software and technology (UIST '02)*. Association for Computing Machinery, New York, NY, USA, 71–80. https://doi.org/10.1145/571985.571996

[30] Rolf Ulrich, Markus Giray, and Roland Schäffer. 1990. Is it possible to prepare the second component of a movement before the first one? *Journal of Motor Behavior* 22 (1990), 125–148. Issue 1. https://doi.org/10.1080/00222895.1990.10735505

[31] Xu Wang, Benjamin Lafreniere, and Tovi Grossman. 2018. Leveraging Community-Generated Videos and Command Logs to Classify and Recommend Software Workflows. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems (CHI '18)*. Association for Computing Machinery, New York, NY, USA, 1–13. https://doi.org/10.1145/3173574.3173859

[32] Kanit Wongsuphasawat, Dominik Moritz, Anushka Anand, Jock Mackinlay, Bill Howe, and Jeffrey Heer. 2016. Voyager: Exploratory Analysis via Faceted Browsing of Visualization Recommendations. *IEEE Transactions on Visualization and Computer Graphics* 22, 1 (Jan. 2016), 649–658. https://doi.org/10.1109/TVCG.2015.2467191

[33] Jun Xing, Hsiang-Ting Chen, and Li-Yi Wei. 2014. Autocomplete painting repetitions. *ACM Transactions on Graphics* 33, 6 (Nov. 2014), 172:1–172:11. https://doi.org/10.1145/2661229.2661247

[34] Tom Yeh, Tsung-Hsiang Chang, and Robert C. Miller. 2009. Sikuli: using GUI screenshots for search and automation. In *Proceedings of the 22nd annual ACM symposium on User interface software and technology (UIST '09)*. Association for Computing Machinery, New York, NY, USA, 183–192. https://doi.org/10.1145/1622176.1622213

A MENU HIERARCHY

Below we present the full menu hierarchy used in the study system. Top level items are image features (shape, fill color, fill style, ...), with parameter values for those features nested at the next level. The default value for each parameter is indicated in bold text.

- Shapes
  - Circles and Ellipses
    - Circle
    - Wide Ellipse
    - Tall Ellipse
  - Squares and Rectangles
    - **Square**
    - Diamond
    - Wide Rectangle
    - Tall Rectangle
  - Triangles
    - Equilateral
    - Right Angle 1
    - Right Angle 2
    - Right Angle 3
    - Right Angle 4
  - Symbols
    - Star
    - Hexagon
    - Arrow Up
    - Arrow Down
    - Arrow Left
    - Arrow Right
- Fill Color
  - Black
  - **White**
  - Red
  - Orange
  - Yellow
  - Green
  - Blue
  - Purple
- Fill Style
  - **Solid**

  - Thin Stripes
  - Thick Stripes
  - Checkerboard
  - Grid
- Outline Color
  - **Black**
  - Red
  - Orange
  - Yellow
  - Green
  - Blue
  - Purple
- Outline Thickness
  - **Thin**
  - Thick