Community Enhanced Tutorials: Improving Tutorials with Multiple Demonstrations

Ben Lafreniere[*†], **Tovi Grossman**[*], **George Fitzmaurice**[*]

[*]Autodesk Research, Toronto, Ontario, Canada
{*tovi.grossman,george.fitzmaurice*}@autodesk.com

[†]University of Waterloo, Ontario, Canada
bjlafren@cs.uwaterloo.ca

ABSTRACT

Web-based tutorials are a popular help resource for learning how to perform unfamiliar tasks in complex software. However, in their current form, web tutorials are isolated from the applications that they support. In this paper we present *FollowUs*, a web-tutorial system that integrates a fully-featured application into a web-based tutorial. This novel architecture enables *community enhanced tutorials*, which continuously improve as more users work with them. FollowUs captures video demonstrations of users as they perform a tutorial. Subsequent users can use the original tutorial, or choose from a library of captured community demonstrations of each tutorial step. We conducted a user study to test the benefits of making multiple demonstrations available to users, and found that users perform significantly better using our system with a library of multiple demonstrations in comparison to its equivalent baseline system with only the original authored content.

Author Keywords

Tutorials; Learning; Community; Help.

ACM Classification Keywords

H.5.2. Information interfaces and presentation (e.g., HCI): Graphical User Interfaces.

INTRODUCTION

Software applications for content design and creation can be difficult for new users to learn [3,5,12], particularly when performing unfamiliar tasks. While many means of seeking help exist, a common trend today is to consult web-based tutorials [8,11,19,21], which can help walk a user through the steps necessary to complete a desired task.

However, a number of factors can make tutorials difficult to follow. In some cases, details are omitted or steps are not explained clearly. In addition, a user's content may differ from that demonstrated in the tutorial [21], requiring the tutorial instructions to be adapted. These difficulties have led researchers to explore new techniques for improving the presentation of tutorials, such as combining text with videos [6], or presenting tutorials in the application itself so they can react to the user's context [9,17,28].

CHI 2013, April 27–May 2, 2013, Paris, France.

A design possibility, which has received less attention, is to provide what we define as *community enhanced tutorials*, which continuously evolve and improve as more users work with them. In particular, if a tutorial system could capture and store every user's workflow as they follow a tutorial, a new user could refer to an entire library of examples, instead of just the originally authored tutorial. This could help users understand variations in the presented workflow, identify particularly efficient strategies, or find examples that more closely match their own specific task.

In this paper we present *FollowUs*, a community enhanced tutorial system. FollowUs utilizes an *Application-in-Tutorial Architecture*, integrating a full-featured image-editing application into a web-based tutorial. In doing so, FollowUs can track when users try to follow the tutorial and can capture video of those users' workflows. Subsequent users can then view the author's original tutorial, or choose from a library of captured community demonstrations for each step of the tutorial. Sorting and filtering mechanisms allow the user to identify demonstrations that may be particularly useful, such as those with similar content to their own, or those that use specific tools.

We performed an evaluation of FollowUs to understand a core question related to its design: Can the availability of multiple demonstrations aid user performance when following a tutorial? Our study showed that users perform significantly better when using FollowUs with a library of multiple demonstrations, in comparison to its equivalent baseline system with only the original authored content. Furthermore, users reported that the community-enhanced system was more useful and was subjectively preferred in comparison to the baseline.

In the remainder of this paper, we present a design space for tutorial architectures and how the Application-in-Tutorial Architecture enables community enhanced tutorials. We then describe the FollowUs system, and present results of our evaluation. We conclude with a discussion of the study's limitations and directions for future work.

RELATED WORK

Task-Centric Help Systems

Early HCI research recognized the challenge of providing effective help systems for software applications [3], and established the benefits of minimalist and task-centric documentation as learning resources [4,5].

Today, task-centric documentation is found most commonly in the form of web-based tutorials. Recent work examining how web tutorials are used [21] has shown that they serve as help while performing a task, as resources for refining a user's skill set, or as guides that allow a user to perform a workflow beyond their current ability.

The HCI community has also explored a range of tutorial systems that are presented in the context of the application. Stencils-based tutorials overlay step-by-step instructions on the interface and limit interaction to UI elements related to the current step [17]. Google SketchUp's self-paced tutorials are similar, but do not limit the user's interaction with the interface [28]. Sketch-Sketch Revolution provides in-application tutorials that assist the user in completing drawing tasks to allow the user to experience success while learning [9].

Our approach combines the benefits of web-based tutorials *and* in-application tutorials by embedding a fully-featured application into a web-tutorial system.

Video Demonstrations

Work in the early 90's by Palmiter, Elkerton [25,26], and Harrison [15] have been cited as showing difficulties associated with using animated assistance for learning (e.g. [11,16,17]). More recent studies have started to delineate where animated assistance can be beneficial. ToolClips—short (10–25 sec.) video demonstrations of features displayed as tooltips—have been shown to result in significantly improved task-completion rates as compared to static text+image tooltips [13]. The MixT multimedia tutorial system showed that short, step-specific video demonstrations are useful for demonstrating dynamic actions that are difficult to express via text or static images alone [6]. Pause-and-Play [27] improves a user's ability to follow video demonstrations by automatically pausing and resuming video to stay in sync with the user's progress.

Our work builds upon this research by investigating how the availability of multiple short demonstration videos can be beneficial for users performing steps of a tutorial.

Recording Workflows

A number of systems have been proposed that automatically record a workflow performed by a user. Some uses for recorded workflows that have been explored include: automatically creating step-by-step tutorials consisting of text and images [11], creating multi-media tutorials [6], allowing users to explore and replay the workflow history of documents [7,14], and creating or improving macros [1,11,23]. While work has explored how to *create* tutorials from captured workflows, we are unaware of previous work which *augments* existing tutorials by recording the workflows of users following the tutorial.

Comparing Workflows

The Delta system [19] allows users to explore and compare multiple workflows for accomplishing a task of interest. The authors focus on helping users choose an appropriate tutorial to follow. Our system is also based on variations within a collection of workflows, but augments the tutorial experience so users can view multiple demonstrations at each individual step.

In summary, there exist a number of previous projects related to our system. However, we are unaware of prior research that has studied the design of tutorial systems that improve as more users work with them, or that has studied the value of multiple video demonstrations on how users learn to perform unfamiliar tasks in an application.

In the next section, we present a design space of tutorial system architectures including the Tutorial-in-Application architecture on which our system is based.

DESIGN SPACE OF TUTORIAL ARCHITECTURES

The emergence of cloud services and rich internet application technologies offer an opportunity to rethink how tutorial systems are architected. In particular, new possibilities are opened up by fully-featured in-browser applications, such as Pixlr (http://pixlr.com/editor), and deviantART Muro (http://muro.deviantart.com). We consider a new tutorial architecture enabled by these advances.

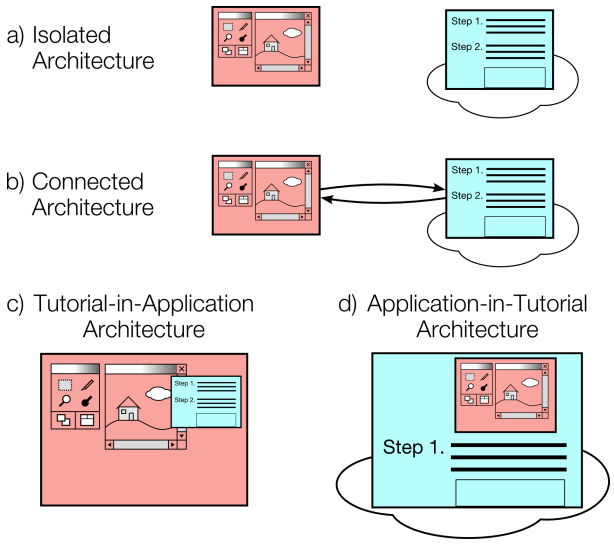

Figure 1. Current and potential architectures for linking web-based tutorials and applications.

Fernquist *et al.* present a design space for tutorial systems [9] consisting of three axes: *Scope*, *Interactivity*, and *Integration*. In this section we introduce a design dimension of tutorial *Architectures*, which was not considered in their design space. The architectures we identify are distinguished based on the degree of *integration* between tutorial and software, and the *domain* in which the tutorial system and application are located: the desktop or the web. The degree of integration between the tutorial and application is important because it affects how much the two can interact with one another. The domain is important because of the qualitative differences between the web and the desktop. The desktop is a personal domain of software maintained by the user, while the web is a social domain of software

maintained by others. A diagram of tutorial architectures can be seen in Figure 1. We discuss each in detail below.

Isolated Architecture

In this architecture, the tutorial and application are in their separate domains (the web and desktop respectively), and cannot communicate with one another (Figure 1a). This is typical for current web tutorials, which users follow in a browser window separate from the application. This architecture is limited in terms of design possibilities.

Connected Architecture

In this architecture, the tutorial and application still reside on the web and desktop respectively, but there is one- or two-way communication between them (Figure 1b). One-way communication could be used to load an initial document into the application, or to customize the application's interface to suit the tutorial [20]. It could also be used to repurpose tutorials as auxiliary input devices for an application, as in Adobe's Tutorial Builder [29]. A two-way communication channel enables *Reactive* tutorials [9] that respond to a user's actions as they follow the tutorial, such as the Pause-and-Play system [27]. Communication between the application and tutorial also enables *tutorial analytics*, where detailed logs of in-application actions are reported to the server hosting a tutorial [20].

This architecture expands the design possibilities for tutorial systems, but it has two disadvantages. First, the tutorial is still displayed in a separate window from the application. This limits interaction possibilities, and is a known source of difficulty when following help [18]. Second, there are technical challenges involved in coordinating a desktop application with a tutorial that resides on the web.

The architectures we discuss next address these issues by embedding the tutorial in the application, or vice-versa.

Tutorial-in-Application Architecture

In this architecture, the tutorial system is integrated into the application (Figure 1c). This allows a tutorial to be presented in-context, and to easily react to user actions. Moreover, the system is implemented in one domain, easing technical challenges. A potential downside is that the tutorials lose the benefits of having a web-presence. This includes the ability to collect web analytics, the ease with which other relevant materials can be accessed through web search [10], and simple mechanisms for sending content to other users. It also removes a key motivation for authors to create tutorials: the potential to earn recognition or ad revenue which are enabled by creating web content [21].

Application-in-Tutorial Architecture

In this paper, we present a new tutorial architecture that contrasts in-application tutorials. Instead of integrating the tutorial into the application on the desktop, the application is integrated into the tutorial system running on the web (Figure 1d). This enables qualitatively new interactions. A user can perform a web search for the task they want to perform, and end up at a page that serves as both a tutorial for the task, and a ready-to-use application tailored to the task. Along these lines, work on the TApps system [22] has explored how existing web-tutorials can be converted into active interfaces for applying tutorial steps and experimenting with settings variations.

Reifying tasks into concrete artifacts also provides a convenient gathering place for a community of users to discuss techniques or experiences with performing those tasks. In this work, we are interested in how an Application-in-Tutorial architecture opens up possibilities for creating tutorials that improve with use and contributions from the user community. We expand on this idea in the next section.

COMMUNITY ENHANCED TUTORIALS

Traditionally, web tutorials are created and do not change once posted on the web. However, the Application-in-Tutorial architecture allows us to explore an alternative model where the initial posting of a tutorial acts as a seed, representing a task in an application. Community content, in the form of demonstrations, comments, answers to questions, etc. then accumulates around the tutorial over time. We see this being particularly useful under two scenarios.

First, the original instructions in the tutorial may not exactly match the goal that the user wants to accomplish. For example, a tutorial on color correction may demonstrate a workflow using a photo with too much red, while the photo that the user wants to correct has too much blue. If the user does not realize this mismatch, they may not understand why the tutorial instructions do not work for their photo. However, if additional demonstrations were available, the user could realize that different photos need to be corrected in different ways.

Second, the quality of a tutorial may be poor, or targeted at users with a higher skill level. For example, a tutorial on creating a collage of images might assume the reader understands how to use selection and transformation tools, and may only provide terse instructions (e.g. "cut out the man from the first image"). If a question-and-answer feature was included with each tutorial step, common sticking points such as "How did the author select the man?" could be documented by the community and benefit all subsequent users of the tutorial.

	High Task Similarity	***Low Task Similarity***
High Tutorial Quality	Following the instructions verbatim will yield satisfactory results.	Instructions must be adapted to match the user's document (e.g. additional commands or settings).
Low Tutorial Quality	Tutorial quality or skill mismatch may make it difficult to follow steps.	User must adapt tutorial instructions, and contend with difficult to follow instructions.

Table 1. Scenarios that may occur when a user is attempting to follow a tutorial.

We formalize these examples into two dimensions, *task similarity*, and *tutorial quality*, that form a 2×2 space of situations that users may encounter when attempting to apply tutorial instructions (Table 1).

We hypothesize that community enhanced tutorials will have the most impact in the low-similarity or low-quality scenarios. In the low-similarity scenario, community features could help generalize the tutorial to address a wider range of user goals. In the low-quality condition, they could smooth flaws in the author's presentation, and adapt the instructions to a wider audience of users.

We now present a description of the features of FollowUs, our community enhanced tutorial system.

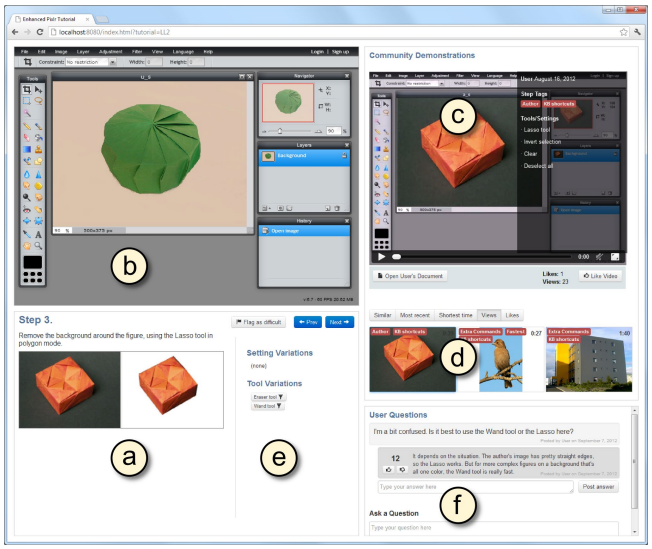

Figure 2. Overview of the FollowUs interface. (a) Tutorial instructions, (b) Embedded application, (c) Demonstration video, (d) Community demonstrations, (e) Tool/Settings Variations, (f) Per-Step Q&A

FOLLOWUS SYSTEM DESCRIPTION

FollowUs is a new type of tutorial system that allows implicit and explicit contributions by the community to improve its tutorials over time. An overview of the FollowUs interface is shown in Figure 2. FollowUs is displayed in a web page, and provides instructions using text and images (Figure 2a). Unlike traditional web tutorials, it includes a fully-functioning built-in application (Figure 2b). A video player is displayed alongside the application (Figure 2c) and users can choose between the author's video, and multiple community demonstration videos (Figure 2d).

Target Application and Implementation

Pixlr is a full-featured image editor implemented in Flash that runs in the web browser. Its feature set is similar to popular Desktop image editing applications such as Adobe Photoshop or GIMP.

Pixlr's source was modified to implement logging of interface events and to take snapshots of open documents. The interface to FollowUs is implemented using standard web technologies (HTML5, JavaScript, CSS, Flash) which easily integrate with one another. Screen recording is implemented using ffmpeg (http://ffmpeg.org) installed on the client system.

Tutorial Instructions

FollowUs provides instructions using text and images (Figure 3). Similar to previous in-application tutorial systems [9,17], FollowUs displays one step at a time, and the user navigates between steps using Next and Previous buttons (Figure 3a). This step-by-step usage model allows the tutorial system to associate content captured in previous interactions with specific steps, and display community content to the user for the currently viewed step.

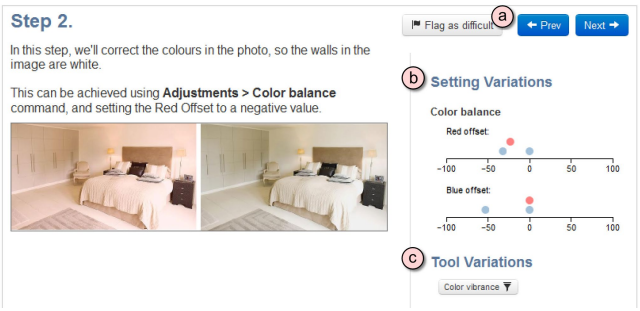

Figure 3. Text+image instructions for a tutorial step. (a) Navigation buttons. (b, c) Settings and tool variations.

Video Demonstrations

At each step of the tutorial, users have access to video demonstrations for that particular step (Figure 2c, d). The author's demonstration is loaded by default, and additional demonstrations from the community are displayed below it.

When not playing, an overlay over the loaded demonstration lists commands and settings used in that video (Figure 4). These are intended to give the user a skimmable overview of the workflow presented in the video. While playing, overlays on the video indicate commands, keyboard shortcuts, and mouse clicks.

Community Demonstrations

Community demonstrations are displayed as a series of thumbnails, each showing the state of that user's document at the start of the step (Figure 5). Hovering over a thumbnail shows the state at the end of the tutorial. Clicking a thumbnail loads the demonstration into the main play area (Figure 4) where the user can play the demonstration for the current step. Moving forward or backward through tutorial steps *follows* the currently selected demonstration, loading the demonstrating user's video for the current step. This allows a user to select a desired demonstration and follow it through the length of the tutorial. The thumbnail view also displays the duration of each demonstration's video for the current step, and the following set of badges that highlight particular attributes of the demonstration (Figure 5a):

Author. The author's demonstration.

Undo. The demonstration includes at least one instance of the Undo command. This may indicate that the demonstrating user made a mistake and corrected it, or experimented with settings.

Extra Commands. The demonstration includes commands different from those used by the tutorial author. This may indicate a variation on the task is demonstrated.

Fastest. The shortest demonstration for the current step.

KB shortcuts. The demonstration includes keyboard shortcuts. This may indicate that the demonstration is by a user with more advanced knowledge of the application.

To remind users of available community videos we implemented *adaptive pop-ups* (Figure 5b). The reminder pop-up is triggered by four seek operations for a given video.

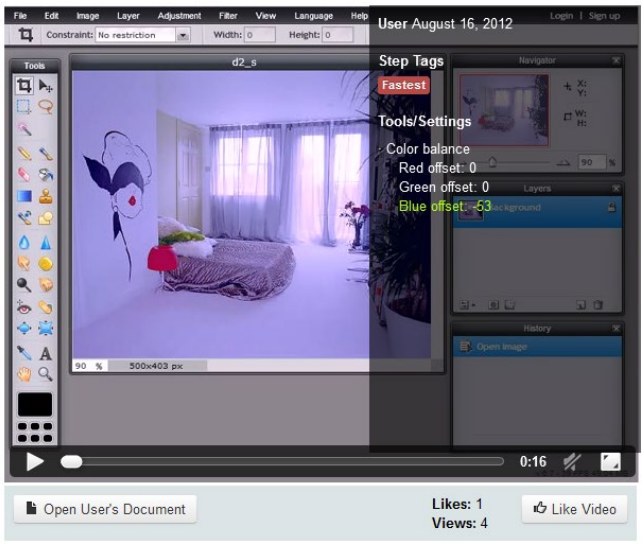

Figure 4. When paused, an overlay over the video player displays tools and settings used in the demonstration.

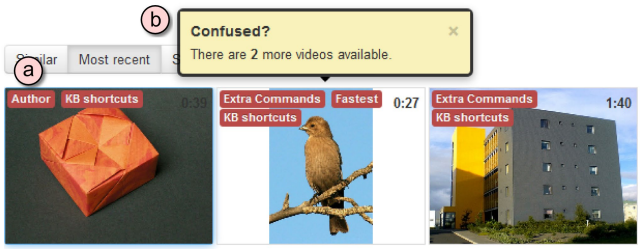

Figure 5. Thumbnails showing available demonstrations. (a) Badges highlight features about each demonstration, (b) Adaptive pop-ups remind users that additional demonstrations are available.

Document Syncing

An additional feature we included is the ability to open the document associated with any demonstration at any step. This allows the user to start a tutorial partway through, skip portions that they aren't interested in, or skip a step and continue the tutorial. While this feature won't help a user to apply tutorial instructions to their own content, it could help them to refine their skill set outside of a current task.

Sorting and Filtering

Users can sort community demonstrations by a range of criteria, including recency, views, likes, and video duration. Users can also sort by image similarity between the document currently loaded in Pixlr and the demonstration thumbnails. This allows the user to find demonstrations on content that matches their own. To compute image similarity a fingerprint is generated for each image consisting of a normalized color histogram separated into red, green, and blue channels. Images are compared by computing the magnitude of the difference between their histograms with the Euclidean norm.

To allow the user to explore variations in how a step was performed by other users, the system includes visualizations of setting and tool variations used at each step (Figure 3b, c). The Settings Variations visualization displays a density plot of setting values used by the tutorial author (marked in red) and other users (blue) for the current step. Selecting a set of data points with the mouse filters the list of community demonstrations to only show those in which the selected range of settings was used. The Tool Variations section lists tools used over and above those used by the author at the current step, and these can be clicked to filter community demonstrations.

Additional Community Feedback Mechanisms

The community enhancements we've described so far are generated implicitly based on screen recording and logs of users' actions as they complete a tutorial. FollowUs also includes a number of explicit community feedback mechanisms.

Per-Step Q&A

Previous work has shown that comment systems on web tutorials play a support role for users, but are hampered because they are separated from tutorial content [21]. In FollowUs, we address this by including a User Questions area for each tutorial step where users can post questions and receive answers from other users (Figure 2f).

Flagging Good and Bad Content

Low-cost feedback mechanisms are included to allow users to flag content they liked or disliked. Users can "Like" video demonstrations, up- or down-vote answers to questions, and flag particular tutorial steps as difficult.

Authoring Tutorials

In our prototype system, an author creates a tutorial by first authoring a series of steps in HTML format (similar to how traditional web tutorials are created), and then records an initial video demonstration by performing the tutorial once in FollowUs while stepping through the tutorial steps. The system segments the raw video based on the time codes for switching between steps, and combines the video data with a log of user interface events from the application to create the author's demonstration video for each tutorial step.

Recording Community Demonstrations

To create the community demonstrations, FollowUs automatically performs a screen capture any time a user starts a tutorial. Interactions with the application are also recorded, to keep track of setting and tool variations. Once the tutorial is completed, the system processes and segments the video and adds it to that tutorial's library of demonstrations. As in the authoring process, video is segmented based on time codes for switching between steps.

To improve the quality of the community demonstration videos, we splice out sections of video where the user is not actively working in the application. If the captured mouse location is outside of the application area for more than four seconds, the corresponding segment of the video is cut.

USER STUDY

We conducted an initial study comparing FollowUs to an equivalent system without community enhancements to see how users would react to the system, and to understand how the availability of multiple demonstrations would impact the use of tutorials to perform unfamiliar tasks.

Participants

We recruited 16 participants (11 male, 5 female), ranging in age from 18 to 63 (median 28.5), via online postings and email. We targeted participants with little or no experience with image editing software, since our interest was in how users would use the tutorial systems to figure out how to perform unfamiliar tasks.

Two additional participants were originally recruited, but later eliminated from the study. One exhibited vision problems that affected his ability to complete the study tasks. The other was a regular user of image manipulation software, and so did not meet our above criteria.

Participants were given a $50 gift certificate to an online retailer in appreciation for their participation.

Conditions

We tested two versions of the FollowUs system. In the baseline *Author Condition*, the community features of the system were removed, and the tutorial consisted only of the author's text/image instructions and the author's demonstration. This condition is similar in spirit to the multimedia tutorials produced by the MixT system [6].

In the *Community Condition*, two additional community demonstrations were included for each task, as well as the described features for filtering and sorting videos. To simplify our evaluation and provide greater control, the Per-Step Q&A sections were not populated with content.

For both conditions, all demonstrations were created by the experimenters, in order to maintain control over the quality of the demonstrations and to simulate a range of specific scenarios, outlined below.

Study Tasks

Eight tasks were designed to simulate applying one step from a larger tutorial. We chose to simulate single tutorial steps because it allowed us greater control over the scenarios faced by participants. The tasks were designed to cover the four quadrants of the task space described in Table 1. In designing the tasks, we varied the quality of the author's text/image instructions and demonstration to attenuate the Tutorial Quality variable, and varied the user's goal document to attenuate Task Similarity. Two sets of four tasks each were designed, with each task set spanning the four quadrants of the task space (Table 2).

Tutorial Quality

For High Quality (*HQ*) tasks the author text/image tutorial provided detailed instructions on how to perform the task, and the author's demonstration shows the procedure clearly, leaving out no details. For Low Quality (*LQ*) tasks, the author's instructions were terse, the author's demonstration was fast, and some detail was left out of video (e.g. the author does not demonstrate how to open a required settings dialog). The additional demonstrations provided in the Community condition were designed to give slower and complete demonstrations of the task.

	High Task Similarity	***Low Task Similarity***
High Tutorial Quality	a) Rotating a photo to correct a tilt. b) Adding a drop shadow to text.	c) Adding a rounded, colored border around text. d) Correcting colors in a blue-shifted photo.
Low Tutorial Quality	e) Removing birds from a skyline scene with the clone stamp tool. f) Applying a photo negative effect to an image.	g) Moving/distorting text into a particular shape. h) Removing the background around a figure.

Table 2. The eight tasks users were asked to perform, arranged by Task Similarity and Author Quality.

Task Similarity

For High Similarity (*HS*) tasks, the user's content was similar in nature to the author's, and the task could be achieved by applying the author's instructions verbatim. For Low Similarity (*LS*) tasks, an additional procedure or alternate command would be required beyond the author's instruction, due to the user's content not matching the author's content. Additionally, for these tasks only one of the additional community demonstrations would be helpful for figuring out how to solve the users' task.

Study Design

A within-subjects experimental design was used. Users performed one complete set of 4 tasks for one condition, and then the other set of 4 tasks for the remaining condition. The order of the conditions and the mapping of system to task sets were fully counterbalanced across participants. The order of the four similarity/quality conditions was counterbalanced in a Latin square, but was the same for the first and second condition for each participant.

Procedure

The experiment was conducted in an enclosed office on a desktop computer running Windows 7. The tutorial system was loaded in the Chrome web browser in full screen mode.

Sessions lasted approximately 1 hour. Each session began with a short questionnaire on the participant's demographic information. The participants was then given an overview of the Pixlr image editor interface. Before each condition, the experimenter showed the participant the associated tutorial system, and walked them through its features.

In each task, the tutorial system displayed a tutorial step, and the user was presented with a printout showing a *before*

and an *after* image of a document. The "before" document was loaded into Pixlr. The user was told that they could make as much or as little use of the features of each system as they wished, and that the main objective was to complete the task by recreating the "after" image.

An experimenter was present throughout the study, to make observations. Assistance was only provided to reset the input image to its default state if this became impossible using Pixlr's Undo and History mechanisms, and to encourage the user to continue. If the user appeared stuck, the experimenter would ask the user "Is there anything else you could try?" or "Is there any way you could make it look more like [the goal image]?" If after several prompts the user was not making any further progress, the user was given the opportunity to go on to the next task. The experimenter also noted instances where the participants showed visible signs of being lost or frustrated.

Between each task, the participant filled out a short questionnaire consisting of six 5-point Likert scale questions about their experience on that task. After completing all tasks, the participant filled out a short post-study questionnaire consisting of 5-point Likert scale questions, and freeform questions about the two tutorial systems and the study as a whole.

Results and Observations

On each of the eight tasks, the experimenter noted whether the participant completed the task, and whether they showed visible signs of frustration during the process.

Participants completed 81% of study tasks, with the observed completion rate for the Community condition (86%) higher than that for the Author condition (77%), though this difference was not statistically significant. We attribute the high overall completion rate to our experimental design, which prompted of participants to continue each task.

Frustration-Free Completion Rates

Instead of the completion rate, we analyze the rate at which tasks were completed *without frustration*. This metric is meaningful for web tutorials, where a frustrated user may be quick to give up or seek alternative help. Participants completed 46% of tasks without signs of frustration, with the observed completion rate for the Community condition (59%), higher than that for the Author condition (34%). Comparing matched pairs of like-typed tasks for each user revealed that frustration-free task completion was significantly higher in the Community condition (McNemar's Chi-squared test with continuity correction, $\chi^2(1, N = 64) = 9.38$, $p < .05$, $\phi = 0.33$).

Figure 6 shows the number of tasks completed without frustration across the two conditions and four task types. Comparing the task types separately with Bonferroni correction, the only significant difference was found for the LS/LQ tasks ($\chi^2(1, N = 16) = 9.09$, $p < .05$, $\phi = 0.15$). However, the trend favored the Community condition for all but the HS/HQ tasks, which may suggest a significant result if we had tested with more participants. The reversal for the HS/HQ case may be because the Author system provided the user with everything required to easily complete the task, and the additional demonstrations and features in the Community condition acted as a source of complexity or frustration.

Overall, these results partially support our hypothesis that multiple demonstrations are particularly useful when the user is contending with a low quality tutorial or applying a tutorial to content that differs from that of the tutorial. We found a significant improvement when users faced both of these challenges and a trend toward an improvement when they faced each separately.

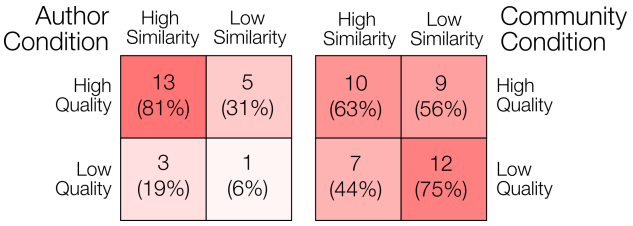

Figure 6. Counts of tasks completed without frustration by task type and system (of 16 for each task type / system).

Videos Watched

Across both conditions, videos were used extensively, with all participants watching at least one video during the study, and 10 of our 16 participants (62.5%) watching a video for all eight tasks.

In 50% of the tasks performed in the Community condition, users watched two or more videos (Table 3). This suggests that users will take advantage of multiple videos when they are available. Unsurprisingly, users viewed multiple videos more often in tasks that were not straightforward. This is the most obvious for the LS/LQ tasks, for which 74% of users watched two or more videos.

	Community				***Author***	
	None	**1**	**2**	**3**	**None**	**1**
HS HQ	1 (6%)	11 (69%)	3 (19%)	1 (6%)	5 (31%)	11 (69%)
HS LQ	2 (12%)	5 (31%)	7 (44%)	2 (12%)	1 (6%)	15 (94%)
LS HQ	1 (6%)	8 (50%)	5 (31%)	2 (12%)	2 (12%)	14 (88%)
LS LQ	0	4 (25%)	10 (62%)	2 (12%)	2 (12%)	14 (88%)
All Tasks	4 (6%)	28 (44%)	25 (39%)	7 (11%)	10 (16%)	54 (84%)

Table 3. Distribution of video viewing behavior. Counts indicate the number of users who viewed the indicated number of videos for each task type.

Questionnaire Results

The post-study questionnaire included three questions on the study as a whole (Figure 7). The majority of users (9) preferred using the Community system, compared with 4 who expressed no preference and 3 who preferred the Author system. Twelve of the participants agreed or strongly agreed to the statement that they could not have completed the tasks without the tutorial system. Likewise, participants'

responses regarding the ease of tasks indicate that the tasks we chose were at a reasonable level of difficulty for the participants that we recruited.

We also asked five questions for each of the two systems (results shown in Figure 8). In all cases, the trend favors the Community system, but only two of the questions showed statistical significance. Specifically, participants rated the Community system as having greater overall quality (Wilcoxon Signed-rank test, $W = 11$, $Z = -2.3518$, $p < .05$, $r = 0.42$), and rated the Community system as less frustrating than the Author system (Wilcoxon Signed-rank test, $W = 91$, $Z = 3.22$, $p < .01$, $r = 0.57$). The result for frustration is notable because it reinforces the experimenter's observations of frustration during tasks, and matches our qualitative analysis.

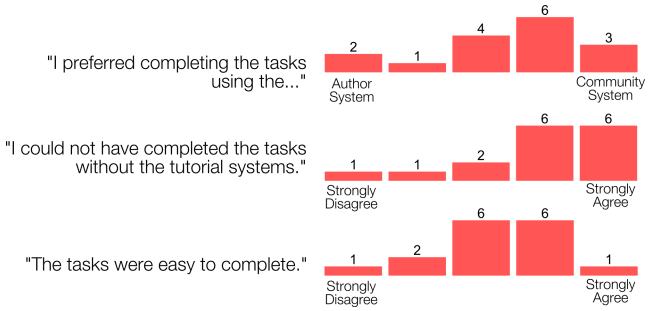

Figure 7. Responses to post-study questions. Numbers indicate the number of participants for each rating.

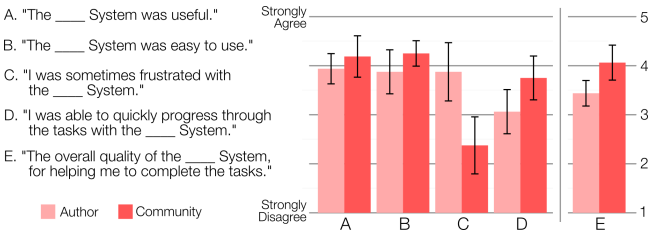

Figure 8. System-specific post-study questions. Error bars show 95% confidence intervals, computed using Morey's method for within-subjects data [24].

Participants also filled out a short questionnaire after each of the eight tasks they performed. This questionnaire consisted of six 5-point Likert scale questions designed to probe three themes: the usefulness of the system for the task, the difficulty or frustration experienced during the task, and whether additional demonstrations were desired. For each theme, a positively- and negatively-phrased version of the question was asked. The aggregated results are shown in Figure 9.

Paired t-tests with Bonferroni correction were used to compare the two Systems for each task type. In the LS/LQ condition, we found the Community system was rated as having significantly higher usefulness (Welch's t-test, $t(15) = 2.20$, $p < .01$), the Author system was rated as being significantly more frustrating (Welch's t-test, $t(15) = 3.78$, $p < .05$), and users desired additional videos for the Author system (Welch's t-test, $t(15) = 4.88$, $p < .01$).

We did not find significant differences for the other task types, but the data for the Author system in Figure 9 suggests a trend for all three questions in which the HS/HQ lies at one extreme, the LS/LQ at the other, and the two remaining task types are somewhere in-between. That this is not the case for the Community condition may indicate that multiple demonstrations make a tutorial less sensitive to issues of tutorial quality and task similarity.

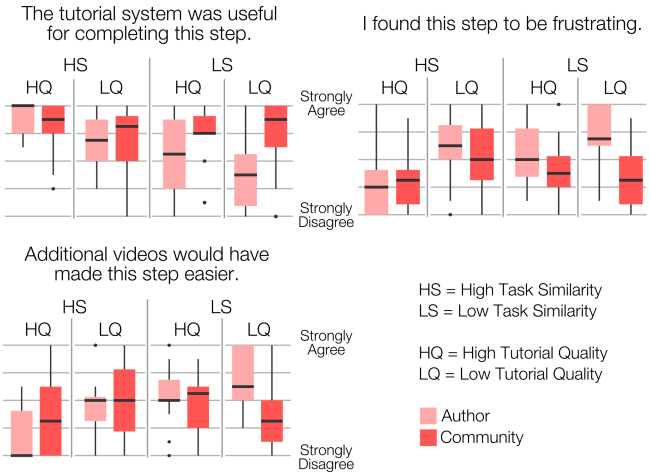

Figure 9. Per-task questionnaire results by task type. Each diagram reports the results of two phrasings of a question.

Observations and User Feedback

In response to our post-study questionnaire, many participants expressed enthusiasm about the community system, and mentioned particular features that they liked about it.

Eleven of our sixteen participants cited multiple demonstrations as a useful feature, as in the following quotes:

> *[P11]: I think it's always useful to have multiple examples of a task. I guess the main reason for that is you're not going to be doing exactly the same thing as every tutorial. So, you know, you pick up on pieces from different tutorials.*

> *[P10]: The Author system wasn't bad, but it's just one person's way of doing it.*

Participants cited a number of ways that multiple demonstrations could be useful, including to show common mistakes, to demonstrate a task at a range of expertise levels, and to show the range of results tutorial instructions could produce. One user also mentioned that multiple demonstrations would be useful because they could be used together, by drawing hints from several demonstrations.

Embedding mistakes in the videos was not an intentional part of our experimental design, but two of the community demonstrations we created included a case where the user experimented while performing an operation. Two users pointed this out as a positive feature. For example:

> *[P8]: In some of the videos they made mistakes which kind of helped me, when they corrected it.*

This is encouraging because it shows that community videos need not be perfect, and in fact there may be some par-

ticular value in videos where users make mistakes and experiment while figuring out tutorial instructions.

In contrast, two users stated that they were only interested in the fastest method. One of these participants said:

> *[P12]: Watching people fail, or do it in a slower manner wasn't really important. It just clogs up the interface. I guess I could say that there's too much extraneous information on it [the community system].*
>
> *Because everyone ends up, or I guess I just end up, just watching the fastest response. The only of the community videos I watched was the person that did it in the quickest time.*
>
> *So maybe you could just have one video on the community screen, the best, as opposed to everyone who's done it.*

This user's feedback provides validation for our surfacing of the "Fastest" attribute as a badge. These comments also validate our decision to show only three videos at a time that match the current sort/filter criteria. These users may be expressing that too much information is overwhelming.

Users also identified areas for improvement. Two users felt that the video demonstrations were too fast, though they may have been reacting to author demonstrations in the LQ tasks, which were purposefully demonstrated quickly.

Two users mentioned that adding audio to the demonstrations would be helpful. Because video is automatically recorded, and then segmented into individual tutorial steps, adding audio to videos is a non-trivial problem. Some of the issues involved in this, and potential solutions, have been discussed in work on the MixT system [6].

Challenges encountered

Even when additional demonstrations were available, we observed many cases where a user would get *demonstration fixation* while watching a particular video. When this occurred the participants appeared to be extremely focused on the video, and ignored the adaptive pop-up notifying them that other videos were available. We observed this phenomenon even for videos that were particularly unhelpful.

We hypothesize that this phenomenon is an instance of the "paradox of the active user" identified by Carroll and Rosson, where a user will resist spending time learning and instead attempt to apply their current, incomplete knowledge to solving a problem [3]. Techniques for addressing this problem for multiple demonstration videos are an interesting subject for future work.

DISCUSSION AND FUTURE WORK

The results of our initial study are encouraging. Users took advantage of multiple demonstrations when they were available, and the presence of multiple demonstrations allowed users to complete tasks with less frustration as compared to a control system. Users also rated our system subjectively better than the control system, and expressed enthusiasm about its features, including identifying a number of potential uses for multiple demonstrations. Moreover, our study revealed issues surrounding the use of multiple demonstration videos, such as the phenomenon of demonstration fixation, where a user becomes stuck watching a particular video.

These are encouraging results, but they only scratch the surface of research questions raised by community enhanced tutorials. In this section, we discuss limitations of our study, and outline opportunities for future work.

A potential limitation of our study is that all demonstrations were created by the experimenters, and the study tasks were on the granularity of individual tutorial steps rather than complete tutorials. This allowed us to test the 2×2 task matrix and the impact of multiple demonstrations in a tightly controlled manner, but leaves open the question of the external validity of our findings. Ideally, this could be addressed by deploying our system and studying the community demonstrations recorded by the system, and usage patterns for community demonstrations. However, we feel it is reasonable to expect that high-quality demonstrations would be captured in a deployed system, because our approach of capturing demonstrations from every user of a tutorial would yield a large set of candidate demonstrations to choose from. Moreover, our study has shown that demonstrations need not be perfect to be helpful to users.

This implicit, *capture-everything* approach to community contribution has the advantage that it requires no additional effort on the part of users to contribute; it is enough that they use the tutorial for their own needs. However, it places additional responsibility on the system to identify and surface useful demonstrations. This is a challenging problem for future work. We anticipate that a fully working system would use of a number of complementary techniques in concert with one another. We outline these below.

First, automatic filtering heuristics could be used to identify good demonstrations (e.g., those that make minimal use of undo operations) from the set of candidates recorded by the system, and conversely eliminate poor quality demonstrations. Given a large number of candidate demonstrations, these heuristics could afford to be conservative about the demonstrations allowed into a tutorial's library.

Second, collaborative filtering could be used to take advantage of viewing patterns and low-cost explicit feedback from users to surface demonstrations that are popular with the user community. This motivated our inclusion of the ability to "Like" demonstrations in FollowUs.

Finally, visualization and presentation techniques could be used to help users recognize and find demonstrations that are relevant to their own needs. The badges on video thumbnails, and the image similarity sort features in FollowUs are examples of features in this vein.

Investigating and evaluating the techniques and approaches outlined above are all areas for future work.

An alternate to the capture-everything approach would be to create mechanisms that allow users to author and explicitly

contribute demonstrations or other improvements to tutorial content. The Per-Step Q&A feature included in FollowUs is a simple explicit contribution mechanism in this vein. Explicit mechanisms require effort on the part of users, who would have to be motivated to contribute. However, they could allow for more targeted and specific refinement of tutorial content by users. This approach has been explored to good effect by Berthouzoz *et al.* [2] in the context of creating content-adaptive photo manipulation macros.

While our work was focused on the tutorial following experience, community enhanced tutorials provide interesting opportunities for tutorial authors as well. A *tutorial dashboard* component could to track open Per-Step Q&A questions, or visualize how users progress through a tutorial, including where they encounter trouble or give up. Video recordings of user experiences would enhance this by allowing a tutorial's author to see how users performed the tutorial at each step.

CONCLUSION

We have presented FollowUs, a new system which leverages an Application-in-Tutorial architecture to bring enhanced community features to the tutorial experience, including additional demonstrations of each tutorial step. An initial evaluation suggests that the presence of multiple demonstrations helps users to complete tasks with less frustration, particularly for tasks with poor author quality and where the user's task does not precisely match that of the tutorial.

ACKNOWLEDGEMENTS

We would like to thank the reviewers for their thorough reviews and thoughtful suggestions. They have greatly improved the final version of this paper.

REFERENCES

1. Bergman, L., Castelli, V., Lau, T., and Oblinger, D. DocWizards: a system for authoring follow-me documentation wizards. *Proc. of UIST '05*, ACM (2005), 191–200.
2. Berthouzoz, F., Li, W., Dontcheva, M., and Agrawala, M. A Framework for content-adaptive photo manipulation macros: Application to face, landscape, and global manipulations. *ACM Trans. Graph. 30*, 5 (2011), 120:1–120:14.
3. Carroll, J.M. and Rosson, M.B. Paradox of the active user. In *Interfacing Thought: Cognitive Aspects of Human-Computer Interaction*. MIT Press, Cambridge, MA, USA, 1987, 80–111.
4. Carroll, J.M., Smith-Kerker, P.L., Ford, J.R., and Mazur-Rimetz, S.A. The minimal manual. *Hum.-Comput. Interact. 3*, 2 (1987), 123–153.
5. Carroll, J.M. *The Nurnberg funnel: designing minimalist instruction for practical computer skill*. MIT Press, Cambridge, MA, USA, 1990.
6. Chi, P.-Y., Ahn, S., Ren, A., Dontcheva, M., Li, W., and Hartmann, B. MixT: Automatic generation of step-by-step mixed media tutorials. *Proc of UIST '12*, (2012), 93–102.
7. Denning, J.D., Kerr, W.B., and Pellacini, F. MeshFlow: Interactive visualization of mesh construction sequences. *ACM Trans. Graph. 30*, 4 (2011), 66:1–66:8.
8. Ekstrand, M., Li, W., Grossman, T., Matejka, J., and Fitzmaurice, G. Searching for software learning resources using application context. *Proc. of UIST '11*, (2011),195–204.
9. Fernquist, J., Grossman, T., and Fitzmaurice, G. Sketch-sketch revolution: an engaging tutorial system for guided sketching and application learning. *Proc of UIST '11*, (2011), 373–382.
10. Fourney, A., Mann, R., and Terry, M. Characterizing the usability of interactive applications through query log analysis. *Proc. of CHI '11*, ACM (2011), 1817–1826.
11. Grabler, F., Agrawala, M., Li, W., Dontcheva, M., and Igarashi, T. Generating photo manipulation tutorials by demonstration. *ACM Trans. Graph. 28*, 3 (2009), 66:1–66:9.
12. Grossman, T., Fitzmaurice, G., and Attar, R. A survey of software learnability: metrics, methodologies and guidelines. *Proc. of CHI '09*, ACM (2009), 649–658.
13. Grossman, T. and Fitzmaurice, G. ToolClips: An investigation of contextual video assistance for functionality understanding. *Proc. of CHI 2010*, (2010), 1515–1524.
14. Grossman, T., Matejka, J., and Fitzmaurice, G. Chronicle: Capture, exploration, and playback of document workflow histories. *Proc. of UIST '10*, ACM (2010), 143–152.
15. Harrison, S.M. A comparison of still, animated, or nonillustrated on-line help with written or spoken instructions in a graphical user interface. *Proc. of CHI '95*, (1995), 82–89.
16. Huang, J. and Twidale, M.B. Graphstract: minimal graphical help for computers. *Proc. of UIST '07*, (2007), 203–212.
17. Kelleher, C. and Pausch, R. Stencils-based tutorials: Design and evaluation. *Proc. of CHI 2005*, (2005), 541–550.
18. Knabe, K. Apple guide: a case study in user-aided design of online help. *CHI '95 Conference Companion*, ACM (1995), 286–287.
19. Kong, N., Grossman, T., Hartmann, B., Agrawala, M., and Fitzmaurice, G. Delta: a tool for representing and comparing workflows. *Proc. of CHI '12*, ACM (2012), 1027–1036.
20. Lafreniere, B., Bunt, A., Lount, M., Krynicki, F., and Terry, M.A. AdaptableGIMP: designing a socially-adaptable interface. *UIST '11 Adjunct*, ACM (2011), 89–90.
21. Lafreniere, B., Bunt, A., Lount, M., and Terry, M. *"Looks cool, I'll try this later!": Understanding the faces and uses of online tutorials*. University of Waterloo, Technical Report CS-2012-01, 2012.
22. Laput, G., Adar, E., Dontcheva, M., and Li, W. Tutorial-based Interfaces for Cloud-enabled Applications. *Proc of UIST '12*, ACM (2012), 113–122.
23. Leshed, G., Haber, E.M., Matthews, T., and Lau, T. CoScripter: automating & sharing how-to knowledge in the enterprise. *Proc of CHI '08*, ACM (2008), 1719–1728.
24. Morey, R.D. Confidence intervals from normalized data: A correction to Cousineau (2005). *Tutorials in Qualitative Methods for Psychology 4*, (2008), 61–64.
25. Palmiter, S., Elkerton, J., and Baggett, P. Animated demonstrations vs written instructions for learning procedural tasks: a preliminary investigation. *Int. J. Man-Mach. Stud. 34*, 5 (1991), 687–701.
26. Palmiter, S. and Elkerton, J. Animated demonstrations for learning procedural computer-based tasks. *Hum.-Comput. Interact. 8*, 3 (1993), 193–216.
27. Pongnumkul, S., Dontcheva, M., Li, W., et al. Pause-and-play: automatically linking screencast video tutorials with applications. *Proc of UIST '11*, ACM (2011), 135–144.
28. Google SketchUp Training. http://sketchup.google.com/intl/en/training/index.html.
29. Adobe Labs Tutorial Builder. http://labs.adobe.com/technologies/tutorialbuilder/.