Data-driven Multi-level Segmentation of Image Editing Logs

Zipeng Liu
University of British Columbia
Vancouver, BC, Canada
zipeng@cs.ubc.ca

Zhicheng Liu
Adobe Reseasrch
Seattle, WA, USA
leoli@adobe.com

Tamara Munzner
University of British Columbia
Vancouver, BC, Canada
tmm@cs.ubc.ca

ABSTRACT

Automatic segmentation of logs for creativity tools such as image editing systems could improve their usability and learnability by supporting such interaction use cases as smart history navigation or recommending alternative design choices. We propose a multi-level segmentation model that works for many image editing tasks including poster creation, portrait retouching, and special effect creation. The lowest-level chunks of logged events are computed using a support vector machine model and higher-level chunks are built on top of these, at a level of granularity that can be customized for specific use cases. Our model takes into account features derived from four event attributes collected in realistically complex Photoshop sessions with expert users: command, timestamp, image content, and artwork layer. We present a detailed analysis of the relevance of each feature and evaluate the model using both quantitative performance metrics and qualitative analysis of sample sessions.

Author Keywords

Log segmentation; image editing logs; interaction history; multi-level hierarchy

CCS Concepts

•**Human-centered computing** → **User models;** *Graphical user interfaces;*

INTRODUCTION

Analyzing and modeling user interaction logs can address challenges posed by complex user interfaces in professional creativity tools. In particular, segmenting logs into semantically meaningful pieces would benefit many downstream use cases. A primary use case is *smart undo* [4]: if we can automatically segment the log into low-level tasks, users will be able to navigate editing history more easily by undoing or redoing a group of related actions instead of one action at a time. Other use cases that also reply on accurate identification of key decision or transition points in the log data include smart version control [3], automatic generation of tutorials from interaction logs, and recommendation of alternative design ideas or workflows [6].

CHI'20, April 25–30, 2020, Honolulu, HI, USA

ISBN 978-1-4503-6708-0/20/04…$15.00

DOI: https://doi.org/10.1145/3313831.3376152

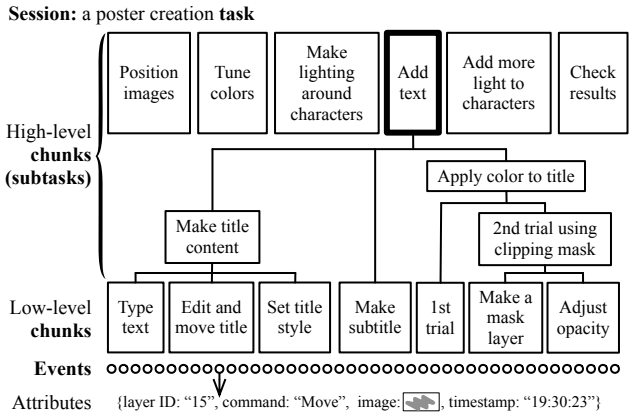

Figure 1. Concrete example of session segmentation. High-level chunks consist of low-level chunks, which consist of logged events, which have attributes. The single expanded chunk shows the non-balanced nature of the hierarchy.

In this paper, we investigate the segmentation of image editing logs from Adobe Photoshop (PS). Figure 1 shows a concrete example of a segmented image log, introducing key terms used in this paper. Designers and artists use PS for a variety of **tasks**, such as composing a poster, retouching a photo, and applying special effects to an image. A task is accomplished in a number of steps or **subtasks**, for example positioning images and tuning colors during poster creation. A subtask may be further decomposed into smaller subtasks, for example adding text involves making text content and applying text color. The hierarchy of subtasks in a complex PS task is not balanced; they may have different **granularities**. At the lowest level, a subtask is accomplished by a group of **events**, namely individual operations triggered through key strokes or mouse clicks. Each event has a number of **attributes**, including the **command** invoked, such as `Move`, and **timestamp**. The entire sequence of events logged in the process of accomplishing a PS task comprises a **session**.

We define **segmentation** as grouping consecutive events within a session into a hierarchy of semantically meaningful **chunks**. A perfect segmentation method will produce chunks that correspond to the subtasks. The desired subtask granularity depends on the downstream use cases: for example, smart undo and redo will require finer granularity, while creating an overview of the whole session can be more coarse-grained.

Despite the wide range of potential applications, many questions about segmenting image editing logs remain unanswered.

First, image editing tools support diverse tasks such as photo retouching, image composition, special effects creation and user interface design. These different tasks result in very different workflows. However, existing segmentation methods usually are limited to one type of editing task [4, 37], and provide no evidence on whether segmentation models can generalize across task types.

Secondly, many segmentation approaches (e.g. hierarchical clustering [27], sequence progression models [35], infinite hierarchical hidden Markov models [16]) assume that command name and timestamp are the only two relevant event attributes. Other event attributes such as image content and editing area overlap have been included in segmentation models [3, 4]. The relevance of these event attributes for segmentation has not been thoroughly analyzed, and the suitability of other potentially relevant attributes has been ignored.

Finally, real-world user behaviors are complex due to individual differences. Collecting realistic workflow datasets with rich event attributes and labelling them as ground truth is expensive and time-consuming. As a result, the log datasets collected in previous work either enforce homogeneous workflows with predefined subtasks [4], or are unrealistically clean without user mistakes or digressions [3]. It is unclear if certain kinds of user behavior may pose challenges in creating unambiguous labels and building robust segmentation models.

In this paper, we take a first step in addressing these problems, targeting *smart undo* [4] as a primary use case. Our goal is three-fold: build a reasonable model, understand the relevance of features, and investigate the performance of the model in the context of user interface applications given the complexity of heterogeneous tasks and individual variation in user behavior.

We collect ecologically valid log data of PS expert users' workflows for different types of image editing tasks. For each session, we record four event attributes (commands, layers, image contents, and timestamps) that are potentially relevant for model building, and label chunk boundaries in all collected sessions as ground truth. We derive five features from these for use in machine learning (ML) models: command similarity, layer similarity, image diff, working region overlap, and duration. We then develop a two-stage segmentation model: a supervised machine learning model to predict the boundaries for low-level chunks, and two alternative approaches to construct high-level chunks from the low-level ones – multi-tier thresholds or agglomerative clustering. This two-stage scheme allows for the flexible choice of levels to support various downstream use cases.

Our primary contributions are: 1) a simple and interpretable low-level segmentation model that works reasonably well for smart undo, and 2) an analysis of feature relevance to segmentation, where we find that command, timestamp, and layer are relevant event attributes, but image content is not as useful. We are the first to use layers for segmentation and confirm their relevance. We validate the model with quantitative evidence showing that our model generalizes across three image editing tasks for low-level chunks. Our secondary contribution is an adjustable high-level segmentation model built on top of the low-level one. We conduct a preliminary qualitative analysis of the suitability of higher-level chunks for these three tasks. We also provide observations on the complexity of real-world user behavior and note the challenges it raises for multi-level segmentation.

RELATED WORK

We briefly introduce the related work on event sequence visualization and graphical histories, then focus on computational models of event logs.

Event Sequence Visualization

Event sequences have been frequently in the visualization community for application domains including e-commerce, program execution traces, and medical records, where researchers focus on visualizing large-scale collections of event sequences to get data insights such as identifying common patterns or outliers [9, 14, 20]. In contrast, with image editing logs we are dealing with one sequence (session) at a time, so visualizing many sessions at once does not help with segmentation.

Workflows and Graphical Histories

Researchers have proposed graphical histories of workflows in domains ranging from general scientific analysis [2] to specific biological experiments [22] to the standalone visualization application Tableau [15, 21]. However, these domain-specific approaches do not adequately address the interface problems of creativity tools such as image editing.

Work targeted at creativity tools addressed the question of how to represent and manipulate application state to undo and redo in single-user [10] and multi-user collaborative settings [28]. Other approaches used annotations on the graphical interface to inform user interaction history [24, 30], or supported the capture, exploration, and playback of creative document workflow histories [13]. The Delta system visually represented and compared a small amount of short workflows [17]. Grabler et al. [12] and Chi et al. [5] proposed an automatic generation of multimedia tutorials from logged sessions in GIMP (an alternative to Photoshop). The focus of these papers are either on data models or visualization of interaction history. Some of them tried to group interactions into chunks, as we do, but their goal was to support selective undo on the canvas, whereas our goal is to group events into subtasks that might or might not relate to spatial proximity on the canvas.

Event Log Models

The previous work on segmentation models for general event logs using approaches such as dynamic programming [31], hidden Markov models [16], and probabilistic generative models [35] does not perform well in our setting of image editing logs with rich attributes. Moreover, some of them require large-scale data which we do not have. The work on non-segmentation models for general logs, such as bias detection in visual analytics logs [32], is more distant from our own; models targeted at creativity tool logs are more relevant.

Creativity Tool Logs

We take inspiration from many non-segmentation models of logs. Lafreniere et al. studied and characterized large-scale

real-world image editing command logs [18], providing us with interesting patterns of command usage in image editing tools. Adar et al. proposed CommandSpace to model the relationships between commands, subtasks, and text descriptions of subtasks [1], using machine learning models for natural language. Our command similarity feature is computed in a similar way. Two other approaches to learning usage semantics from command logs solidified our interest in ML approaches: Yang et al. proposed a deep learning model for representation of PS users using command logs [36], and Wang et al. proposed a two-step log classifier to recommend frequent patterns [33]. Causality, a conceptual model by Nancel and Cockburn [25], models causal relationships between events, which helps us think about structure within a session.

The few papers on segmentation models of creativity tool logs are most relevant to our work. Denning et al. proposed to use regular expressions to retrieve a multi-level segmentation interaction logs in a mesh construction application [7, 8], which does not apply in image editing case as there is more variation in our tasks. Chen et al. proposed a non-linear revision control system to model, aggregate, and visualize image editing logs [3]. However, the non-linear model is not intuitive for PS users, who can already achieve such non-linear edits with non-destructive editing techniques [26]. Moreover, their dataset seems unrealistically short and clean. Chen et al. later proposed to use a support vector machine to predict a single-level segmentation for portrait retouching logs [4], which specifically targets at one task with predefined subtasks. They also compared to a baseline from Li et al. that uses duration for segmentation [19], and argued that duration is not effective. We were inspired by this approach to use support vector machines ourselves. Zhao et al. developed Sketcholution system to get a high-level segmentation of sketching logs using an agglomerative algorithm [37], but it is not clear that the specific solution generalizes to other tasks. However, we were inspired to try agglomerative approaches as well. In this paper, we are dealing with multiple common PS tasks instead of one, and we collect more realistic sessions than previous work. The use of layer as a feature is absent from all of this previous work, as is the systematic investigation of the relevance of each feature.

APPROACH

We provide an overview of our approach and its rationale for data generation, model selection and evaluation methods.

Data Generation

Based on literature review and our interaction with Photoshop users, we identify four potentially relevant event attributes: command, timestamp, image content, and layer. Command, timestamp and image content have been used in previous work [3, 4, 37]. Layer has not been considered in previous work, but we introduce it as a potentially important attribute because of our observation that layers are essential in any realistically complex PS tasks.

No existing logs captured all four of these attributes, so we created a logging PS plugin to record this information for each event during normal usage sessions. We conducted user studies to collect data from experts carrying out multiple tasks, capturing these logs and also think-aloud recordings. We used these records to manually create labels for chunks at multiple levels of granularity. For low-level chunks, we found good inter-coder agreement between two independent coders. However, for high-level chunks there was little agreement even between human coders. We decided that only the low-level chunk labels were reasonable to consider as ground truth, a decision that in turn affected our choices of models and evaluation strategies. We also concluded that deeper investigation of the relevance of segmentation attributes was merited.

Model and Feature Selection

The goal of the segmentation model is to predict whether an event e is the starting point of a new chunk (similar to Chen et al. [4]). Considering that applications such as smart undo operate on streaming data during interactive sessions, we ruled out building models that required knowledge of both the events that precede and those that succeed e. Instead, our model makes predictions based on only the events that precede e. We further choose to build on similarity (or distance) measures between events according to their attributes, under the assumption that similar events are likely to be in the same chunk.

Our finding that only the low-level chunk labels are reliable led us to develop a two-stage segmentation model. For the low-level chunks, we use a supervised approach built on the human labels as ground truth; we apply an unsupervised approach for segmenting higher-level chunks. We frame the low-level segmentation problem as an instance of binary classification in machine learning (ML), where the classes are 1 for the boundary and -1 for the non-boundary. We also conducted early experiments with multiple hand-designed rule-based models for low-level segmentation (e.g. segment when a user switches to a different layer), which we documented in Supp. Sec. 5, but found that they were substantially outperformed by a simple and popular ML model, a support vector machine (SVM). Since our focus is to gain a deeper understanding of the segmentation problem and we do not have large-scale data for training, we prioritize the interpretability of a simple model over more complex and powerful ones like random forests or neural networks. We use a linear kernel, again for simplicity. During development, we also tried SVM with a quadratic (RBF) kernel, but since it did not improve performance we stayed with the simpler approach.

The simple SVM model achieves good performance, as we show below in Results, with a small number of features (five) that we derive from the four logged attributes: command similarity, layer similarity, image diff, working region overlap, and duration.

Evaluation Methods

With reliable labels on low-level chunks, we are able to perform quantitative evaluation on the SVM model by analyzing feature relevance and the model's performance on low-level segmentation. The lack of ground truth for the high-level segmentation led us to a qualitative-only evaluation: we constructed visualizations compact enough to show entire sessions

with event-level detail, so that we could inspect the relationship between feature values and compare the predictions with human labels.

DATA COLLECTION AND CHARACTERIZATION

We report our data collection procedure and characterize event attributes.

Data Collection

Our study with expert users was conducted in two rounds of data collection, early and late in the project timeline. The data collected from the first round was used for exploratory analysis and model building, while data from the second round was mainly used for model validation and testing.

We compiled a pool of PS tasks (9 in round one and 11 in round two) from popular PS tutorial websites to use in the studies, falling into three types: poster creation, portrait retouching and special effects creation. We recruited 13 PS expert users with many years of in-depth professional experience (ranging from 2 to over 10 years), with 7 participants in each round; one person participated in both rounds. The study was run for each participant via online video conference, lasting around one hour in total. We first introduced the participant to the goal of the study, and helped them install the PS plugin. We showed them the pool of PS tasks, and asked them to choose one (or two) tasks that they felt most comfortable and competent working on. We provided all the required source images for the task, and a final image for reference (except for portrait retouching task). We asked the participants to think aloud throughout the session, and recorded the entire video conference with screen and audio information.

We collected a total of 16 sessions, 8 from each round, where one participant in each round did two sessions. Broken down by task type, there were 8 sessions for poster creation, 6 for portrait retouching, and 2 for special effect creation. There were 5718 events logged in total across all sessions, with an average of 357 per session (min: 29, max: 1064), and each session lasted an average of 33 minutes (min: 8, max: 53). Our PS recording plugin saved command names, timestamps, and layer names & IDs in a text file, and the image content was saved as JPEG screenshots.

We manually segmented each session into multi-level chunks by labelling the starting and ending events of each chunk. We built an interface showing all the event attributes in a table and used it in conjunction with the video recordings to perform the labelling, as shown in Supp. Fig. 7.

The labelling process involved subjective judgements about which events should be grouped as a chunk at the lowest level, and whether low-level chunks should be grouped into a higher-level chunks; for example, whether to group "Make title" and "Make subtitle" together in Figure 1. To assess inter-coder reliability, two coders independently labelled the starting events of lowest-level chunks in two sessions (a total of 351 events) and compared their results. We found that the two coders mostly agreed with each other (Cohen's kappa $\kappa = 0.77$), which suggests that the low-level chunks are reliable. However, among the 14 incongruent events, 11 were caused by disagreement on granularity (e.g. one coder labeled a chunk of events 23-44, while the other labeled two chunks 23-39 and 40-44), indicating that there would be large inter-coder differences on high-level chunks. We thus chose not to compare human-generated high-level chunks between coders. This observation led to our decision to only treat the lowest-level labeled chunks as ground truth. One coder then continued to label all the remaining sessions.

Data Characterization

Commands are directives from users. The number of unique commands in PS is large: there are about 1400 menu items in Photoshop 2019, each corresponding to a command. Each command is associated with a meaningful and interpretable name, and is considered highly relevant to segmentation in previous work [3, 4].

Layers are essential for managing visual objects, achieving complex visual effects, and performing non-destructive editing. They are created, ordered, and possibly placed into groups within a hierarchy explicitly by the PS user; three examples are shown in Supp. Fig. 4. During a session, users activate different layers at different points of time to work on, and the layer hierarchy evolves over time. The final count of layers in each session ranged from 3 to 16; the layer hierarchy was always shallow, with final depth ranging from 1 to 3. We consider layer a potentially relevant signal because a change in active layer could indicate a shift in subtask, and the changes in layer hierarchy reflect the user's mental models of the artwork composition.

The **image content** of an event refers to the pixel state of the artwork after the operation has been carried out. The changes in image content after an event could be a useful indicator of the active working area, and may be relevant to segmentation.

The **timestamp** is used to derive duration. Duration between consecutive events could be relevant to segmentation, if rapid-fire events are more likely to occur within a chunk, whereas a long pause between two events might signal a boundary between chunks.

FEATURE COMPUTATION

We derive features for our segmentation model from the four event attributes collected in the expert user sessions: command, layer, image, and timestamp. The five features that we derived are specific instances of a similarity (or distance) measure between events: command similarity, layer similarity, image diff, working region overlap, and duration.

Command Similarity

A conventional approach to quantify command similarity is to group commands according to the menu hierarchy in the Photoshop interface [4]. This approach fails to differentiate commands within the same cluster, and might not capture how commands are used in reality. An alternative approach, validated in the CommandSpace previous work [1], is to learn the semantics of the words using machine learning tools for large natural language datasets. The rationale is that commands are analogous to words and a PS session is equivalent to a document, where the proximity of logged commands within

a session expresses meaningful semantics that capture typical usage patterns. Inspired by this idea, we obtained access to a command log database internal to Adobe, which only recorded commands for PS events from opt-in non-enterprise users. We extracted 100 million commands from this database and reconstructed 169,387 sessions by grouping according to associated session identifiers. We ran word2vec [23] to obtain a vector representation with 100 dimensions for each unique command. We then computed the similarity score between two commands as the cosine similarity between their corresponding vector representations. We conducted a sanity check of the learned command vectors using the Google Embedding Projector [29] and found the results to be plausible, as discussed in Supp. Sec. 1.

Layer Similarity

We compute layer similarity using rules that we manually generated based on the combination of observations of participant sessions and additional conversations with PS experts about their use of layers. We classify the strength of the relationship between two active event layers A and B, into five ranked categories and assign heuristic values accordingly, as shown in Table 1. Users frequently operate on the same layer consecutively for a while before switching to a different one, and we assign the maximum similarity value of 1 between these identical layers. The user often copies a layer to a new one to make repeating objects, edit repetitively, or back up a layer, so we assign an 0.8 similarity value between a duplicate and its source layer. An adjustment layer applies color and tonal adjustments to one or more main layers without permanently changing pixel values, a common technique for non-destructive image editing. Layers stack up on each other in pixel space, and although technically all layers below another could be affected we determined that users often consider only directly above/below pairs, so we assign a measure of 0.5 similarity between the two. For layers in the same group, We assign a value ranging from 0 (infinitely distant layers) to 0.5 (direct sibling layers) depending on layer proximity in the hierarchy: layers separated by d hops have similarity $1/2^{d-1}$. Finally, all other layers that differ are assigned with the minimum similarity 0. We summarize this description with detailed formulae in Supp. Sec. 1 and provide annotated examples of layer hierarchies in Supp. Fig. 4.

Relationship	Description	Similarity
Same layer	A = B	1.0
Duplicate layer	A is a copy of B	0.8
Adjustment layer	A is an adjustment layer of B	0.5
Grouped layer	A and B are located in the same layer group	<=0.5
Other diff. layer	none of the above	0.0

Table 1. The five kinds of relationships between two layers A and B and their assigned similarity values. Each relationship is commutative.

Image Diff

The image difference (diff) of two events is the percentage of different pixels between the two corresponding images out of the total number of pixels in an image. The implied conjecture in previous work is that in an image editing session the amount of change, as quantified using image diff, is relevant to segmentation where a larger image diff implies greater difference between two events. Although there exist more advanced image diff scores such as the structural similarity index (SSIM) [34], we do not choose them because they are potentially biased to a particular type of image edits: for example, SSIM is sensitive to structural changes but not color tone changes.

Working Region Overlap

We also compute a second potentially relevant image-related feature, to assess which of the two is a better indicator. The working region overlap measures the degree of continuance in pixels between two events. It is defined as the percentage of overlapping changed pixels between two events out of the total number of pixels of an image. We conjecture that two events are similar if they operate on a similar pixel area, and thus that they would be likely to belong to the same chunk. We compute it as a "diff of a diff": count the different pixels between the diff image of two events down-sampled by a factor of 32, where a diff image of an event is the difference image between itself and the previous one. We down-sample to check for overlap in coarser regions rather than individual pixels.

Duration

The duration is simply the time between consecutive events.

SEGMENTATION MODEL

Our two-stage model predicts low-level chunk boundaries with an SVM classifier, and has two alternatives for computing higher-level chunks from those results.

Low-level Segmentation

To classify events as boundary or non-boundary with an SVM, we must construct a feature vector, partition the data, consider the appropriate point in the precision-recall trade-off space, and tune the hyperparameters.

Feature Vector Construction

To determine whether an event is the starting point of a chunk, a reasonable way is to compare it against its previous events in terms of the computed similarity features. If the difference is big enough, it is likely to belong to a new chunk. We consider a window of k previous events for comparison, where the window size k as a hyperparameter that will be tuned. For each previous event in the window, we use the five similarity (or distance) measures, described in the previous section. We then concatenate all measures of comparison to all previous events in the window into a feature vector, whose size is $k \times 5$.

Data Partition

We split our collected dataset (5718 events in 16 sessions) into training, validation, and test sets. By the second round of data collection we had finalized what features and model to use; we set aside 5 of the 8 sessions that second round as the test set, and used the other 11 sessions for training and validation (3 from the second round, and all 8 from the first round). The

5 test set sessions are randomly drawn with stratified sampling for task type: 2 for poster creation, 2 for portrait retouching, and 1 for special effect creation. This procedure results in a total of 1842 events (about 30% of grand total) in the test set. The remaining 3860 events are further split randomly into training (75%) and validation (25%) set.

Precision Recall Trade-off

Our dataset is unbalanced, with many more non-boundary events than boundary events. We consider the trade-off between precision and recall in model training and validation, and deliberately trade precision for higher recall. It is important to correctly identify as many boundaries as possible, and false negatives (boundaries predicted as non-boundaries) are undesirable as it is not easy for end users to identify these missed boundaries. On the other hand, false positives (predicting non-boundaries as boundaries) are less detrimental: although these errors lead to over-segmentation of the logs, users can quickly dismiss them through an interactive interface.

Hyperparameter Tuning

We train the model on the training set, and measure performance on the validation set to tune these hyperparameters, following common practice in machine learning. Supplemental Sec. 3 covers the procedure and results in detail. In brief, the first hyperparameter, window size k, is the number of previous events used in the feature vector. Using receiver operating characteristic (ROC) curves, we find that window size has little influence on the model performance, and thus we choose $k = 1$ to reduce the size of feature vectors. The second one, threshold t, determines the probability value where an event is predicted to be a boundary. We select $t = 0.24$ in consideration of the trade-off between precision and recall.

High-level Segmentation

The higher-level chunks are built based on the predicted lowest-level chunks.The higher-level segmentation should be flexible in granularity, to accommodate different downstream use cases. We describe two methods to construct high-level chunks: multi-tier thresholds in SVM and agglomerative clustering.

Method 1: Multi-tier Thresholds

In the SVM model for lowest-level segmentation, the threshold t can be interpreted as the granularity of segmentation: as t increases, the granularity gets coarser and there are fewer predicted boundaries (chunks). In that case, we use a threshold that achieves the optimal performance on a labeled dataset (the validation set). Intuitively, we can simply use a series of higher thresholds to obtain multiple higher levels of chunks despite the lack of ground truth. Every higher-level boundary will necessarily fall along a lower-level boundary, since the same probability scores are in use.

Method 2: Agglomerative Clustering

Higher-level chunks can also be generated from lowest-level ones using the bottom-up approach of agglomerative clustering [37]. This algorithm uses a predefined similarity metric to iteratively find the most similar pair of adjacent chunks and merges them together, until a single remaining chunk contains the whole session. We define the similarity between two chunks as the average similarity of all pairs of events across the chunks. We compute a weighted sum of the five feature values, where the weights are the linear coefficients of the kernel function in the learned SVM model, since the coefficients indicate feature importance. The output of the agglomerative algorithm is a binary tree, where each node represents a chunk and the distance to root can be considered as its granularity. Therefore, we can make a cut through this tree to obtain chunks of the desired granularity, either a straight cut with uniform distance to the root or a more complex shape to capture subtrees of different resolution.

Comparison

Both methods reuse some parts of the low-level SVM model: the multi-tier threshold method reuses the probability score directly, whereas the agglomerative method leverages the linear coefficients in the kernel function.

We compare the two methods in terms of computational complexity, potential limitations, and usage pattern. For computational complexity, the multi-tier threshold method does not need extra computation: it is $O(n)$, where n is the number of events. The agglomerative method has to compute event and chunk similarity iteratively. Its computational complexity is $O(n^2)$, an acceptable cost since a session contains a few hundred events.

A potential limitation of the multi-tier threshold method is that the SVM model for low-level segmentation only leverages minimal context information (one previous event), which has not been validated for the high-level one. In contrast, the agglomerative clustering computes average values over many event pairs as the chunk similarities, making it more robust against single-event outliers. However, the agglomerative method may yield sub-optimal results due to the greedy merging mechanism [11].

Thresholds in the multi-tier method are straightforward to specify but less flexible as one value is used across all levels of the hierarchy. The cluster hierarchy resulting from the agglomerative clustering method provides the flexibility for an adaptive criterion that can vary within a session to select nodes according to the specific use case needs.

RESULTS: FEATURE RELEVANCE

The performance of segmentation model highly depends on the effectiveness of the selected features, and moreover one of our goals is to understand feature relevance for its own sake. We analyze relevance to low-level segmentation quantitatively and qualitatively, and the results are consistent. Then we discuss the reason of their (ir)relevance and limitations.

Quantitative Analysis

The linear coefficients in the SVM model, listed in Table 2, are usually interpreted as feature importance. The sign of a coefficient indicates whether correlation between the feature value and probability of being a boundary is positive or negative: only duration has a positive correlation, e.g., longer duration indicates higher probability that the event is at a chunk boundary. The absolute value of the coefficient indicates the degree

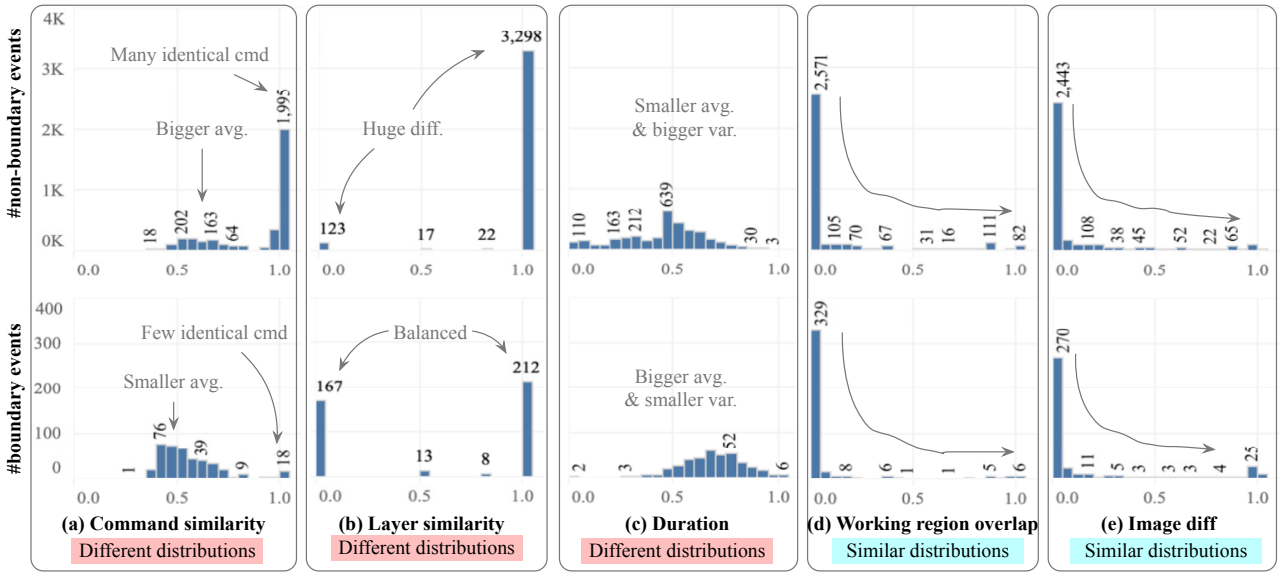

Figure 2. Distributions of the five feature values of non-boundary (top) and boundary (bottom) events. The X axis represents the value of each feature (bin size = 0.05), and the Y axis the counts. Note that the Y scales are 10x different between top and bottom because there are significantly more non-boundary events. We annotate the differences or similarities between the two distributions for each feature.

of importance, and we bin these quantities into three categories: commands as most important, layers and duration as important, and the two image-related features have almost no effect.

Feature	Coefficient	Relevance category
Command similarity	-2.57	Most important
Layer similarity	-1.74	Important
Duration	1.55	
Working region overlap	-0.18	No effect
Image diff	-0.07	

Table 2. Corresponding coefficients to features in SVM model.

Qualitative Analysis

We also conduct a qualitative analysis of feature relevance. We visualize the distributions of feature values using histograms and compare the distributions between non-boundary events and boundary events to see if the difference is salient. Figure 2 shows these distributions, with annotations including our assessment of when they differ and thus the feature is relevant. We can see clearly that layer similarity, command similarity, and duration are relevant, while image diff and working region overlap are not.

Result Interpretation and Limitations

Below we discuss the potential causes for feature (ir)relevance and the limitations of our feature choices.

Command Similarity

It has been shown in the CommandSpace system [1] that the command usage semantics can be learned from large-scale of command logs, analogous to text data, and that the usage semantics are relevant to user goals and subtasks.

However, we see that except for the maximum similarity when commands are identical, the variance in the remaining values is small; most of the range between 0 and 1 is not exploited. A second problem is that a single command in PS such as `BrushTool` can support multiple different functions, which is analogous to the polysemy problem in natural language. Future work could experiment with a context-aware learning model instead of word2vec, or augment the computation by considering the command parameters (such as brush size) that are not captured by our current PS instrumentation plugin. The third potential problem is the information loss in the computation of cosine similarity between the command vectors.

Layer Similarity

Our observations during data collection were that participants keep the layer panel visible at almost all times as they refer to layers frequently, and are able to explain the semantics of each layer explicitly. Different subtasks in a task are often performed on different layers, particularly when the artwork consists of objects from different source images.

However, we heuristically assigned discrete values as layer similarity for our 5 chosen cases, and these values have not been strictly verified. Also, our current approach does not fully exploit all information available in the layer hierarchy. We have not found a good way to use the logged information about layer order, which might have further distinguishing power. Also, some layer attributes are not logged, such as layer mask, blending options, and transparency. Exploration of these issues is an important direction of future work.

Duration

We observed during the data collection study that pauses do indicate thinking time when switching to a new subtask and when comparing current results to a reference image. We also saw clear evidence of short duration values indicating

continuous operations when they are actively involved with a subtask, especially when they have a mental map of how the subtasks should be fulfilled.

However, we think that the duration from our data collection in a lab setting may be artificially clean, since participants are doing an assigned task under observation. We conjecture that in real-world usage, pauses may often arise from external interruption rather than an internal switch of subtasks. Therefore, we conclude that the relevance of duration is debatable.

Working Region Overlap

Our results contradict previous work from Chen et al. [4] that uses working region overlap. We conjecture that the signal from pixels of image content is dominated by the one from layers, which serve as an abstraction of visual object structure, for identifying subtasks. Another possible reason for our opposite finding could be the information loss with pixel counts: we use a single number instead of a vector representation in Chen's paper.

Image Diff

We conclude that the image diff is too noisy for segmentation. A subtask switch (boundary event) does not always come with large change in pixels; for example, adding text in a small font only changes a small number of pixels. Similarly, large changes in pixels sometimes can happen within a chunk, for example moving a large object. We see from the distributions that most events change very few pixels.

RESULTS: LOW-LEVEL SEGMENTATION

With reliable ground-truth labels for the low-level segmentation, we measure model performance on the test set, conduct an preliminary analysis on the variance across different tasks, and compare our performance metrics with a previous paper.

Performance Metrics

We train the SVM model on the training set, tune hyperparameters on the validation set (window size $k = 1$, threshold $t = 0.24$), and measure performance on the test set, which contains 1842 events. The F2 score, which weighs recall twice as much as precision, is 0.76, with a decent recall 0.87 but modest precision 0.51. The numbers are almost identical to the performance metrics on the validation set (provided in Supp. Sec. 3), suggesting that the model does not overfit.

Task Variance

We analyze the variance of model performance across different types of task (poster creation, portrait retouching, and special effect creation) to verify our claim that our model generalizes.

We categorize all 16 collected sessions by task type, and conduct two analyses to measure model performance. In Analysis 1, we use the chosen threshold $t = 0.24$ for all three tasks, and compute the F2, recall and precision scores for each task. In Analysis 2, we tune the threshold hyperparameter for each task. We observed that different participants organize their sessions at different levels of granularity, which is particularly obvious across different task types, so we need to understand how the threshold affects performance for different task types.

We present the performance metrics in Table 3. For both analyses, the difference of F2 scores between any two task types is low (± 0.08). Within each task type, the difference of F2 scores across analysis is also low (± 0.03). In Analysis 2, where the threshold is optimized for each task type, we notice that the threshold for portrait retouching (0.17) is different than the other two $(0.27, 0.25)$, confirming our observation about the influence of task type on granularity. In conclusion, the SVM model can generalize to the three task types, while further research is needed to address the nuances in granularity.

Task type		Poster creation	Portrait retouching	Special effect creation
Data scale	#sessions	8	6	2
	#events	3156	1876	670
Analysis 1	F2	0.71	0.77	0.79
	recall	0.85	0.82	0.97
	precision	0.43	0.63	0.45
	threshold	0.24		
Analysis 2	F2	0.73	0.80	0.79
	recall	0.82	0.92	0.96
	precision	0.49	0.52	0.47
	threshold	0.27	0.17	0.25

Table 3. The performance metrics of sessions in the three task types.

Comparison to Previous Work

The closest work to our model is Chen et al.'s method [4]. They also trained an SVM model, targeting smart undo as a primary use case. We deemed it infeasible to directly compare with their model fairly. It does not make sense to apply their trained model on our dataset, because their training dataset (and hence their model) only has portrait retouching data, whereas we covered three tasks in total. Also, a difference in data collection (they have data on brushing area but we do not) induces technical difficulties in re-implementing their model for retraining on our dataset.

Chen et al. chose window size $k = 5$ for their model and achieved a similar F2 score (0.74) with a lower recall (about 0.7) but much higher precision (0.96) on their dataset with portrait editing task only. That outcome is surprising, since they did make similar arguments to our own for valuing recall over precision. We also note that our segmentation problem is considerably harder: we deal with three types of tasks instead of portrait retouching only, and we did not prescribe a set of pre-defined subtasks for the participants like they did. We argue that our results are reasonable, since we tackle a harder problem. In conclusion, we have achieved reasonable results for a more difficult problem than theirs.

RESULTS: MULTI-LEVEL SEGMENTATION

We present the visualizations of two exemplar sessions to show how our computed low and high-level segmentation aligns with human labels, which are subjective judgements from the authors. Here we show two from five sessions from the test set: one portrait retouching session (S10), and one poster creation session (S15). They are representative sessions in terms of both general model performance and failure cases; Supp. Sec. 4 contains the analysis of the other three test sessions. We also provide high-resolution screenshots for all sessions in Supp. Sec. 4, page 13-17. We first provide

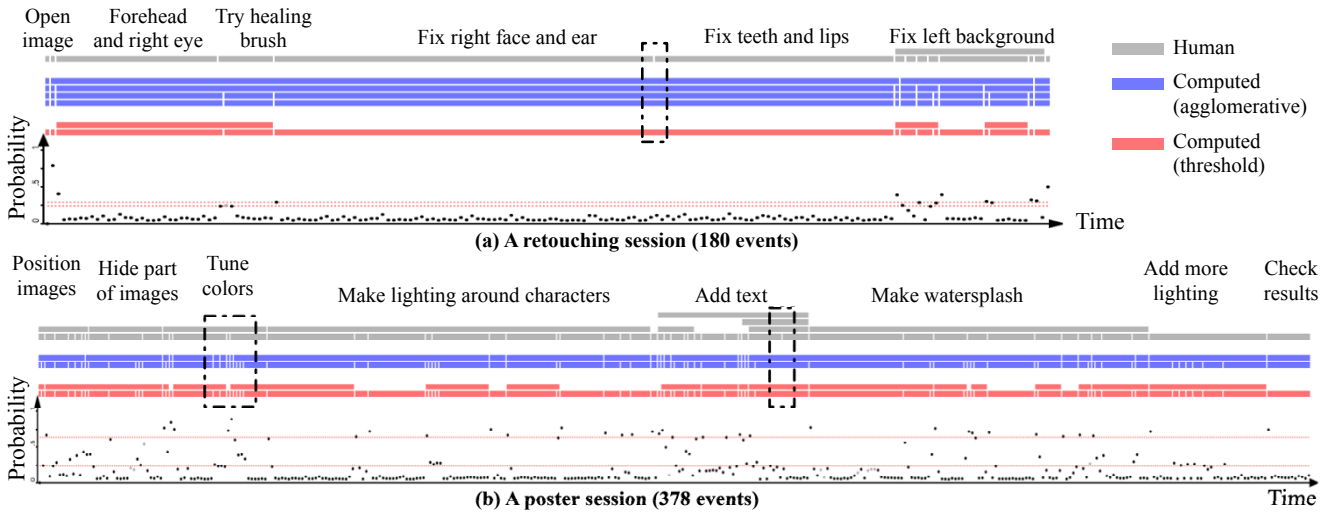

Figure 3. Session visualizations. (a) Retouching. (b) Poster. From bottom to top in each session: a scatterplot showing the probability score of events in sequential order computed by the SVM model; a multi-level computed chunks using multi-tier thresholds (red); the multi-level computed chunks using the agglomerative algorithm (blue); the multi-level human labels (grey); text description of high-level chunks with human-assigned labels. Examples of over-segmentation and missing boundaries are highlighted with dotted rectangles. Thresholds (corresponding to red chunks from low to high) are $0.24, 0.29$ for retouching and $0.24, 0.64$ for poster. Distances to root, corresponding to high-level blue chunks from low to high, are $8, 5, 3$ for retouching and 6 for poster. Figure 1 illustrates an expansion of the "add text" chunk in Poster.

an overview of participant's workflow (high-level segmentation in the human labels), and then inspect how our computed results align with human labels.

Exemplar Retouching Session

The goal of the first exemplar session (Figure 3(a)), with 180 events, is to restore an old photo (a child's portrait) with cracks and stains on it. At the bottom of the figure there is a scatterplot showing the probability scores of events being predicted as a boundary in time order, with red dotted lines to show threshold values. Above the scatterplot are three versions of segmentation (computed chunks by the multi-tier thresholds SVM in red, computed chunks by the agglomerative algorithm in blue, and human labels in gray). Each colored rectangle represents a chunk, and a white space (gap) between two rectangles represents a chunk boundary.

From the grey human labels, we can see that the participant is working on different regions in the image roughly one at a time: forehead and right eye, right face and ear, teeth and lips, and finally the background. Note that the regions (face, eye, background, etc) are only meaningful for a human user. However, there are also outliers subtasks that do not follow this order: e.g, trying out a different command `Healing Brush` to fix the photo. In the last step to fix the left background, they used a more complicated procedure than previous chunks, resulting in a two-level segmentation.

To compare different versions of segmentation, we can inspect the alignment of gaps (boundaries) vertically. In general, the gaps in human labels and those in the computed results align well. One obvious missing boundary happens around the middle between "fix right face and ear" and "fix teeth and lips" (highlighted with a dotted rectangle), due to the high similarities between these events (same layer and command, duration small, and little changes in image content).

Exemplar Poster Session

The goal of the second exemplar session (Figure 3(b)), which is longer with 378 events, is to create a poster featuring two football players. From the human labels in the figure, we can see that the chunks consist of variant numbers of events, and the hierarchy of chunks is unbalanced; that is, different high-level chunks have different subtree depths below them. From the high-level human labels and an expanded part in Figure 1, we can see that the participant did multiple trials for some subtasks, making some mistakes and corrections.

For the low-level chunks between human labels and computed results, most gaps are aligned, indicating a good match between the two. There are very few missed boundaries (highlighted): one example happens when the participant finished making a layer mask and then started adjusting opacity, where they stayed at the same layer and used commands that are semantically close (`Brush Tool` vs. `Master Opacity Change`). There is some but not excessive over-segmentation: the consecutive gaps that separates individual events into chunks. For example, it happens in a human-labeled chunk "tune colors of front player": he actually performed two scaling operations of the image in between color tuning operations, resulting in low similarity between consecutive commands (`Free Transform` vs. `Camera Raw Filter`) and thus over-segmentation in the computed results. We will discuss this particular user behaviour in the next section.

For the high-level chunks, although both computed results aggregate most of the single-event over-segmentation, neither has a good match with the human labels: about half of all high-level boundaries are missed. As we have already noted,

we do not consider that data to represent ground truth; the ambiguity about what granularity to code for led to the two human coders had even higher levels of discrepancies with each other in the sessions they both coded, so that data was not used in the development of the high-level model. Nevertheless, we inspect the probability scores of the missing boundaries in the scatterplot, and find that many of them have a high probability score, but it is hard to draw a horizontal straight line that could serve as a threshold to separate most of them out. This finding indicates that there is some value in reusing the results of SVM model in the high-level segmentation, but it needs further research to improve the performance.

DISCUSSION

We discuss the strengths and weaknesses of our segmentation model, of the dataset we collected compared to previously reported ones, and the nuanced characteristics of real-world user behaviour in image editing revealed by our study.

Strengths and Limitations of Model

Our multi-level segmentation model achieves fast computation, due to its simplicity and the small number of parameters. The low-level SVM model takes less than 1 second to train and validate, and prediction is immediate. The high-level segmentation takes at most a few seconds to handle an 1K-event session with a JavaScript implementation. This speed makes the model suitable for streaming log data, the hallmark of the downstream interactive application use cases.

It is hard to thoroughly evaluate our approach to higher-level segmentation since we do not have appropriate labelled data for it, due to the subjectivity of the granularity assessment. Ground truth labels and a more rigorous evaluation would allow further progress. The complex user behaviours described below do leave considerable room for improvement in model performance. Our precision is currently only 0.5; although we did prioritize recall, both measures need to be improved.

Dataset Comparison

We now compare our dataset to two other datasets from previous work in more depth. In the nonlinear revision control paper by Chen et al. [3], most of their sessions are less realistic than ours, where there is little complex user behaviour such as continuous refinement, mistakes and corrections, and experimentation. The scale of their data is also smaller in terms of number of sessions and number of events per session: a few sessions with less than a dozen events, and a few sessions of sketching with hundreds of events.

The adaptive history paper by Chen et al. [4] targets only the portrait retouching task, with predefined subtasks chosen from a list. The restricted variance in subtasks makes their model hard to generalize to other task types, or even retouching that does not follow the predefined subtasks.

A major difference between our dataset and previous ones is that we capture layer information. However, there is still some available data that our plugin architecture does not capture, such as command parameters (e.g., brush size) and layer attributes (e.g., transparency). Future work could investigate whether it contains useful signal to exploit for segmentation.

Real-world User Behaviour

Our data collection study documented that the real-world usage patterns of creativity support tools are usually messy and complex: users rarely perform precise and effective actions from the beginning of a session to the end. We characterize several of the complex user behaviours that we observed.

First, users make mistakes. Common examples of mistakes are misuse of commands, undesired command parameters, or operations on a wrong layer. The user may or may not correct the mistakes, and corrections may happen immediately after a mistake or later. This situation is challenging as the user usually does not consider these corrective actions as a change of subtask, but the model would only see the a sudden change of events and thus predict them as boundaries. Second, users interleave different subtasks together. This situation may happen when the user has no clue of how to perform a subtask, when the user realizes there is another subtask that is more urgent than the current one, or when the user finds a previously done subtask unsatisfying. This situation is challenging for segmentation as the user's mental model is less organized. Third, users experiment with the tool to achieve a desired outcome through trial and error. This situation is common in users with lower levels of expertise, and the resulting session usually contains many undo actions and layer deletions. The challenge in automatically segmenting this kind of user behaviour is that the difference between trials can be subtle and hard to detect. Fourth, the boundaries between subtasks could be intrinsically fuzzy. Before switching to a new subtask, users often check if results are satisfactory by toggling layer visibility, or make small adjustments of different components in the artwork while thinking about what to do next. These actions result in fuzzy boundaries between subtasks that span multiple events, but our model assumes single-event boundaries.

Our data collection study succeeded in revealing a picture of user behaviour that is far more complex than what previous work had captured, where less attention was paid to ecological validity. Our model makes a significant step towards addressing this complexity, but there are many interesting problems to address in future work.

CONCLUSION

We developed a multi-level segmentation model for real-world image editing logs that works for three image editing tasks. We present evidence for what features are relevant and irrelevant for the segmentation. Results show that command and layer similarity are highly relevant, image-related features are not useful despite claims in previous work, and duration is debatable due to the limitation in data collection. We also present quantitative and qualitative evaluation of our segmentation model, and show that it performs reasonably well for the challenging problem of segmenting realistic logs that capture mistakes, experiments, and subtask switching.

ACKNOWLEDGMENTS

We thank Adobe for funding the research, the Photoshop experts who participated in the study, and many researchers in Adobe for early discussion of ideas. We thank the UBC InfoVis group for their feedback on the manuscripts.

REFERENCES

[1] Eytan Adar, Mira Dontcheva, and Gierad Laput. 2014. CommandSpace: modeling the relationships between tasks, descriptions and features. In *Proc. 27th annual ACM Symp. User interface software and technology*. ACM, 167–176.

[2] Steven P Callahan, Juliana Freire, Emanuele Santos, Carlos E Scheidegger, Cláudio T Silva, and Huy T Vo. 2006. VisTrails: visualization meets data management. In *Proc. 2006 ACM SIGMOD International Conf. Management of data*. ACM, 745–747.

[3] Hsiang-Ting Chen, Li-Yi Wei, and Chun-Fa Chang. 2011. Nonlinear revision control for images. In *ACM Transactions on Graphics (TOG)*, Vol. 30. ACM, 105.

[4] Hsiang-Ting Chen, Li-Yi Wei, Björn Hartmann, and Maneesh Agrawala. 2016. Data-driven adaptive history for image editing. In *Proc. 20th ACM SIGGRAPH Symp. Interactive 3D Graphics and Games*. ACM, 103–111.

[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 *Proc. 25th annual ACM Symp. User interface software and technology*. ACM, 93–102.

[6] Adobe Creative Cloud. 2017. Accelerating Your Creativity - Adobe MAX 2017 - Day 1 Keynote. (18 October 2017). Retrieved September 18, 2019 from `https://youtu.be/4j_pd1awSac?t=8119`.

[7] Jonathan D Denning, William B Kerr, and Fabio Pellacini. 2011. MeshFlow: interactive visualization of mesh construction sequences. In *ACM Transactions on Graphics (TOG)*, Vol. 30. ACM, 66.

[8] Jonathan D Denning, Valentina Tibaldo, and Fabio Pellacini. 2015. 3DFlow: Continuous summarization of mesh editing workflows. *ACM Transactions on Graphics (TOG)* 34, 4 (2015), 140.

[9] K Dextras-Romagnino and T Munzner. 2019. Segmentifier: Interactive Refinement of Clickstream Data. In *Computer Graphics Forum*, Vol. 38. Wiley Online Library, 623–634.

[10] W Keith Edwards, Takeo Igarashi, Anthony LaMarca, and Elizabeth D Mynatt. 2000. A temporal model for multi-level undo and redo. In *Proc. 13th annual ACM Symp. User interface software and technology*. ACM, 31–40.

[11] Brian S. Everitt, Sabine Landau, Morven Leese, and Daniel Stahl. 2011. *Cluster Analysis* (5th ed.). Wiley Publishing.

[12] Floraine Grabler, Maneesh Agrawala, Wilmot Li, Mira Dontcheva, and Takeo Igarashi. 2009. Generating photo manipulation tutorials by demonstration. In *ACM Transactions on Graphics (TOG)*, Vol. 28. ACM, 66.

[13] Tovi Grossman, Justin Matejka, and George Fitzmaurice. 2010. Chronicle: capture, exploration, and playback of document workflow histories. In *Proc. 23nd annual ACM Symp. User interface software and technology*. ACM, 143–152.

[14] Shunan Guo, Ke Xu, Rongwen Zhao, David Gotz, Hongyuan Zha, and Nan Cao. 2017. EventThread: Visual summarization and stage analysis of event sequence data. *IEEE Trans. Visualization & Computer Graphics* 24, 1 (2017), 56–65.

[15] Jeffrey Heer, Jock Mackinlay, Chris Stolte, and Maneesh Agrawala. 2008. Graphical histories for visualization: Supporting analysis, communication, and evaluation. *IEEE Trans. Visualization & Computer Graphics* 14, 6 (2008), 1189–1196.

[16] Katherine Heller, Yee Whye Teh, and Dilan Gorur. 2009. Infinite Hierarchical Hidden Markov Models. In *Proc. 12th International Conf. Artificial Intelligence and Statistics*, Vol. 5. PMLR, 224–231.

[17] Nicholas Kong, Tovi Grossman, Björn Hartmann, Maneesh Agrawala, and George Fitzmaurice. 2012. Delta: a tool for representing and comparing workflows. In *Proc. SIGCHI Conf. Human Factors in Computing Systems*. ACM, 1027–1036.

[18] Benjamin Lafreniere, Andrea Bunt, John S Whissell, Charles LA Clarke, and Michael Terry. 2010. Characterizing large-scale use of a direct manipulation application in the wild. In *Proc. Graphics Interface*. Canadian Information Processing Society, 11–18.

[19] 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 Trans. Computer-Human Interaction (TOCHI)* 18, 2 (2011), 6.

[20] Zhicheng Liu, Bernard Kerr, Mira Dontcheva, Justin Grover, Matthew Hoffman, and Alan Wilson. 2017. Coreflow: Extracting and visualizing branching patterns from event sequences. In *Computer Graphics Forum*, Vol. 36. Wiley Online Library, 527–538.

[21] M Loorak, Melanie Tory, and Sheelagh Carpendale. 2018. ChangeCatcher: Increasing Inter-author Awareness for Visualization Development. In *Computer Graphics Forum*, Vol. 37. Wiley Online Library, 51–62.

[22] Eamonn Maguire, Philippe Rocca-Serra, Susanna-Assunta Sansone, Jim Davies, and Min Chen. 2012. Taxonomy-based glyph designâĂŤwith a case study on visualizing workflows of biological experiments. *IEEE Trans. Visualization & Computer Graphics* 18, 12 (2012), 2603–2612.

[23] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* (2013).

[24] Toshio Nakamura and Takeo Igarashi. 2008. An application-independent system for visualizing user operation history. In *Proc. 21st annual ACM Symp. User interface software and technology*. ACM, 23–32.

[25] Mathieu Nancel and Andy Cockburn. 2014. Causality: A conceptual model of interaction history. In *Proc. SIGCHI Conf. Human Factors in Computing Systems*. ACM, 1777–1786.

[26] Adobe Photoshop. 2018. Techniques for nondestructive editing. (21 November 2018). Retrieved September 18, 2019 from `https://helpx.adobe.com/ca/photoshop/using/nondestructive-editing.html`.

[27] Lior Rokach and Oded Maimon. 2005. Clustering methods. In *Data mining and knowledge discovery handbook*. Springer, 321–352.

[28] Thomas Seifried, Christian Rendl, Michael Haller, and Stacey Scott. 2012. Regional undo/redo techniques for large interactive surfaces. In *Proc. SIGCHI Conf. Human Factors in Computing Systems*. ACM, 2855–2864.

[29] Daniel Smilkov, Nikhil Thorat, Charles Nicholson, Emily Reif, Fernanda B Viégas, and Martin Wattenberg. 2016. Embedding projector: Interactive visualization and interpretation of embeddings. *arXiv preprint arXiv:1611.05469* (2016).

[30] Sara L Su, Sylvain Paris, Frederick Aliaga, Craig Scull, Steve Johnson, and Frédo Durand. 2009. Interactive visual histories for vector graphics. *MIT. Retrieved July* 11 (2009), 2011.

[31] Evimaria Terzi and Panayiotis Tsaparas. 2006. Efficient algorithms for sequence segmentation. In *Proc. SIAM International Conf. Data Mining*. SIAM, 316–327.

[32] Emily Wall, Leslie M Blaha, Lyndsey Franklin, and Alex Endert. 2017. Warning, bias may occur: A proposed approach to detecting cognitive bias in interactive visual analytics. In *IEEE Conf. Visual Analytics Science & Technology (VAST)*. IEEE, 104–115.

[33] Xu Wang, Benjamin Lafreniere, and Tovi Grossman. 2018. Leveraging community-generated videos and command logs to classify and recommend software workflows. In *Proc. CHI Conf. Human Factors in Computing Systems*. ACM, 285.

[34] Zhou Wang, Alan C Bovik, Hamid R Sheikh, Eero P Simoncelli, and others. 2004. Image quality assessment: from error visibility to structural similarity. *IEEE Trans. image processing* 13, 4 (2004), 600–612.

[35] Jaewon Yang, Julian McAuley, Jure Leskovec, Paea LePendu, and Nigam Shah. 2014. Finding progression stages in time-evolving event sequences. In *Proc. 23rd international Conf. World Wide Web (WWW)*. ACM, 783–794.

[36] Longqi Yang, Chen Fang, Hailin Jin, Matthew D Hoffman, and Deborah Estrin. 2017. Personalizing software and web services by integrating unstructured application usage traces. In *Proc. 26th International Conf. World Wide Web Companion*. International World Wide Web Conferences Steering Committee, 485–493.

[37] Zhenpeng Zhao, William Benjamin, Niklas Elmqvist, and Karthik Ramani. 2015. Sketcholution: Interaction histories for sketching. *International Journal of Human-Computer Studies* 82 (2015), 11–20.