COLiER: Collaborative Editing of Raster Images

Ulrike Bath, Sumit Shekhar, Jürgen Döllner, Matthias Trapp

Hasso Plattner Institute, Faculty of Digital Engineering, University of Potsdam, Germany
ulrike.bath@student.hpi.de, sumit.shekhar@hpi.de, juergen.doellner@hpi.de, matthias.trapp@hpi.de

Abstract—**Various web-based image-editing tools and web-based collaborative tools exist in isolation. Research focusing to bridge the gap between these two domains is sparse. We respond to the above and develop prototype groupware for real-time collaborative editing of raster images in a web browser. To better understand the requirements, we conduct a preliminary user study and establish communication and synchronization as key elements. The existing groupware for text documents, presentations, and vector graphics handles the above through well-established techniques. However, those cannot be extended as it is for raster graphics manipulation. To this end, we develop a document model that is maintained by a server and is delivered and synchronized to multiple clients. Our prototypical implementation is based on a scalable client-server architecture: using WebGL for interactive browser-based rendering and WebSocket connections to maintain synchronization. We evaluate our work qualitatively through a post-deployment user study for three different scenarios.**

Keywords-**Human-centered computing, Collaborative interaction, Image processing, Web-based interaction**

I. INTRODUCTION

Collaboration between visual artists dates back to as early as late 16th century (Fig. 1a). In the modern era, this practice continued resulting in various masterpieces [1]. However, its adaptation in the digital domain is progressing only slowly (Fig. 1b). Even though there exist collaborative applications mimicking a shared whiteboard – allowing for doodling and/or simple manipulations of a shared image. A system for real-time collaborative editing of raster-based images at different levels of functionality or control – similar to common image editing desktop applications (e.g., Adobe Photoshop or GIMP) – does not exist to the best of our knowledge [2].

In general, multi-user systems where the actions of one user must quickly be propagated to the other collaborator are referred to as real-time groupware [3]. During recent years, various instances of such systems emerged into today's distributed, collaborative working environments. For example, systems such as Google Workspace or Microsoft Office 365 Online edition, various massively multiplayer online games, or NVIDIA's Omniverse platform for 3D contents. All such systems have in common that (1) a document instance or a shared context is hosted by a server(s), is (2) synchronized using a service, and (3) can be manipulated by multiple participants. The most relevant characterizing aspects of a

(a) Madonna in Floral Wreath   (b) Collaborative Collage

Figure 1: (a) An early example of collaboration between *Jan Brueghel the Elder* and *Peter Paul Rubens* approx. 1617. (b) A collage created collaboratively using our web-based system. It comprises the blending of multiple layers, vector strokes, and image processing operations (e.g., vignetting and pixelation).

real-time groupware system, according to Ellis and Gibs [4], are as follows:

- **Interactive and real-time (Aspect-1):** i.e., response times must be short and notification times must be comparable to response times.
- **Distributed (Aspect-2):** i.e., in general, one cannot assume that the participants are all connected to the same machine or even to the same local area network.
- **Volatile and Ad-hoc (Aspect-3):** i.e., participants are free to come and go during a session and generally are not following a pre-planned script. It is not possible to tell a priori what information will be accessed.
- **Focused (Aspect-4):** i.e., during a session there is high degree of access conflict as participants work on/modify the same data.

A popular application that fulfills all of the above criteria is Google Docs. The cloud-based service provided by Google has revolutionized the way people edit documents collaboratively. However, when editing text, most standard algorithms do not consider the complete structure of the document and make use of per-line diffing and merging.

Challenges: The above approach cannot be extended for images in a straightforward manner: while Aspect-1 and Aspect-2 largely pose specific technical challenges (e.g., undo/redo functionality and latencies), Aspect-3 and Aspect-4 reflects on the spatial, structural, and temporal features

of collaborative raster-image editing. Existing collaborative whiteboard applications maintain their state by tracking the brush strokes of individual clients. They allow users to doodle/sketch on top of the image but hardly provide any tools for image editing itself. In particular, these are missing an integrated approach for the manipulation of the raster data using different image filtering operations. On the other hand, existing web-based image editing tools are not collaborative. A system that is quite close to what we aspire is the Google Draw, a functionality provided as part of Google Workspace. Even though it allows users to collaboratively edit attributes of a shared image, the range of per-pixel edits is limited.

We adopt a human-centered design process to identify the challenges associated with a real-time collaborative image editing system. To this end, we design and conduct a preliminary user study with twenty-seven participants through a questionnaire. The answers to the above questionnaire identify key design principles mainly focused on communication and synchronization. We prototype and iterate on the design of our collaborative system. To understand whether our strategies achieve the design goals, we perform a post-deployment user study with six different groups (two or three persons each). Our system was successful to a large extent. Moreover, participant's experiences and perspectives offer further guidance for improvement.

Approach & Contributions: We aim to create a web-based collaborative image editing application that provides a wide range of edits. To this end our contributions are:

1) A web-based application that allows multiple users to collaboratively edit images, while satisfying Aspect-1 to Aspect-4 properties. The Web-App consists of a responsive Graphical User Interface (GUI) which makes it possible for users to access the application via a smartphone or a tablet.

2) A browser-based rendering framework that enables a wide range of image manipulations along with sketching/doodling functionality.

3) Results of our preliminary and post-deployment user studies that identify key design principles for a real-time collaborative image editing system.

For it, we choose the following approach. Sec. II reviews and analyzes related work and existing tools on collaborative editing of graphics. Based on these, a preliminary user study on the current state of real-time collaborative image editing and associated tasks is conducted (Sec. III). Sec. IV describes a system overview of our prototypical implementation of basic server and client functionality. Sec. V evaluates the implementation through a post-deployment user study. We summarize our findings and potentials for future work and research in Sec. VI.

II. BACKGROUND & RELATED WORK

The challenges associated with collaborative image editing has two aspects: the conceptual/design level and imple-mentation level, which nowadays demands web-based approaches using services. The existing web-based applications mainly address sketching and/or designing functionalities.

Collaborative Graphics Editing: One of the earliest study towards the desired characteristics of a collaborative graphics editing system was performed by Sun and Chen [5]. They propose a formal specification for conflict resolution, versioning, and consistency maintenance for such systems. Design analysis of visual analytic tools has been explored by Heer and Agrawala [6], where they propose techniques to improve shared context and awareness, and provide suggestions to increase engagement. As a specific instance, Salvati *et al.* [7] and Calabrese *et al.* [8] analyze collaborative mesh manipulation by robustly sharing and merging version histories in real-time. In recent work, Gao *et al.* [9] map the two-dimensional drawing area into the linear structure and correspondingly transform the two-dimensional graphical operations to linear operations for collaborative editing. In order to prevent consistency conflicts, Wu *et al.* [10] propose the Common Graphics Collaborative Editing (CGCE) algorithm. Both Gao *et al.* and Wu *et al.* implement their solution using the latest web-based technologies, however, their system only allows for sketching or primitive geometric figure manipulation. For the purpose of cooperative image editing, Zhai *et al.* [11] develop a method using wireless communication over mobile phones. Nevertheless, they only consider simple atomic operations of *import, export, update*, and *commit*. Novakova *et al.* [12] developed a tool specifically for collaborative sketching, suitable for architectural communication. In comparison, our layer-based rendering framework can handle a variety of image edits and also provides brushing and sketching functionality.

Web-based Sketching and Designing: Web-based collaborative whiteboards allow users to ideate and collaborate visually, e.g., Aggie.io or Draw.chat. In comparison, collaborative design tools are more recent and focused on creating new designs by arranging images as multiple layers, e.g., Figma, Canva, or AdobeXD. However, both types of applications have hardly any image editing functionality. The existing web-based image editing applications can be used for per-pixel processing, but are not collaborative in nature, e.g., Photopea or Pixlr. In a very recent development, Adobe now allows for asynchronous collaboration for raster and vector images. Nonetheless, our goal is to provide a real-time synchronous collaborative environment. Tab. I compare existing web-based photo-editing and whiteboard applications regarding the following aspects:

- **Layer (Yes/No):** The application does support layering of multiple images. This allows for an increased function scope and assumes a complex data model.
- **Direct Manipulation (Yes/No):** The application does support direct manipulation of image contents, e.g., using brushing or transform functionality.
- **Undo/Redo (Yes/No):** The support of undo/redo func-

Table I: Comparison of various web applications for the editing of raster images with respect to different features.

Application	Layer	Direct Manip.	Undo/Redo	Image Filtering	Data Type	Resp. GUI	Collaboration Type
canvaspaint.org	No	Yes	Yes	None	Raster	Yes	None
pixlr.com	Yes	Yes	Yes	Destructive	Raster	No	None
photopea.com	Yes	Yes	Yes	Destructive	Raster	No	None
draw.chat	No	Yes	Yes	None	Vector	No	Synchronous
aggie.io	Yes	Yes	Yes	None	Raster	Yes	Synchronous
Google Draw	No	No	No	Non-destructive	Raster	Yes	Synchronous
Adobe Creative Suite	Yes	Yes	Yes	Destructive	Both	Yes	Asynchronous

tionality facilitates error-tolerance while using direct manipulation metaphors.

- **Image Filtering (None/Destructive/Non-destructive):** An application supports the usage of single or multiple destructive/non-destructive image filtering operations.
- **Data Type (Raster/Vector/Both):** The application can handle raster, vector, or both types of input data.
- **Responsiveness (Yes/No):** The GUI of the application support responsive layout of components, thus supports desktop and mobile devices with varying screen sizes.
- **Collaboration Type (None/Synchronous/Asynchronous):** A collaborative application enables multiple clients to modify image data simultaneously. This requires communication between clients and modeling of messages reflecting the editing process.

Our web-based collaborative system provides sketching and designing functionality along with image manipulations. Moreover, we enable synchronous collaboration among users. It provides all features compared in Tab. I while operating on layered raster images.

III. Analysis and Preliminary Considerations

This section reports on a preliminary user study (Sec. III-A), conducted to analyze the major requirements for real-time collaborative image editing and elaborate potential conflicts to be addressed in collaborative editing (Sec. III-B).

A. Preliminary User Study

To better understand the design requirements of a real-time collaborative image-editing application, we designed and conducted a preliminary user study using a questionnaire. Subsequently, we analyzed results and major findings, identified main use cases, and created GUI design sketches.

Participants and Study Design: We selected participants who have experience in collaborative image editing with existing technologies. They belong to a broad range of background and have performed image editing: for casual creativity and/or as a professional activity. A total of 27 participants answered the 17 questions. The questions broadly addressed the following aspects: (i) *How do you perform collaborative image editing tasks?* and (ii) *What are the challenges associated with it?*. The challenges thus identified are used as the basis for designing our system.

Summary on Challenges in Collaboration: The foundation of any collaborative task is efficient *communication*, which also reflects in our survey answers: "*Communication is everything, it is sometimes hard to get an artistic idea thru*" (P7). "*Communicating who edits what and how*" (P9). The lack of communication is not only restricted to high-level requirements and task sharing but also low-level details such as data/edits synchronization: "*Staying in sync, keeping a history of changes, knowing what the partners already have done*" (P5). "*Handling data conflicts, know on which parts or region my teammates are currently working on, handling different versions...*" (P10). To mitigate the above problems, users make use of existing communication channels. However, such an approach seems to be quite inefficient in terms of both data and time: "*It takes a lot of time sending images back/forward and see when progress is made*" (P11). "*Sharing huge files of raw pictures with the team and keeping them in sync*" (P6).

Design Inference: Concerning the above challenges, we choose an integrated messaging functionality for our system. Our WebGL-based rendering framework allows for image edits that are visible in real-time among the collaborators, further enabling low-level communication. The user edits are maintained as part of session management providing for a consistent editing environment. Data conflicts are handled with complementary update processes on server and client-side (Sec. IV-A). Similar to the variety of challenges and its coping strategy, there is a range of application scenarios where collaborative image editing can be used: "*logo creation*", "*poster designing*", "*creative editing*", etc. We support such varying application scenarios by providing Visual Computing Asset (VCA)-based image editing along with a mouse/pen/touch-based sketching interface.

B. Potential Conflicts in Collaborative Editing

There are several potential conflicts arising in real-time collaborative image editing systems, especially if these support a variety of tools being applied to multiple layers. Specific to our approach this concerns challenges arising from (i) limited attention-bandwidth and (ii) synchronous as well as (iii) asynchronous editing conflicts. Considering users operating in the same sessions, our system offers tools to approach the above challenges.

Limited Attention-bandwidth: While performing editing tasks, such as painting or designing, a user focuses on the immediate effect of the current tool. Thus, the user has
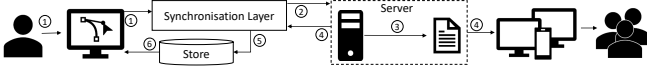
Figure 2: Sequence of client-server communication: (1) user modifies project, (2) modified parameters are processed in synchronization layer, (3) a change request is sent to server, (4) server updates document (5) if successful, updates are sent to all clients, (6) synchronization layer updates the local store, (7) changes are applied in the GUI.
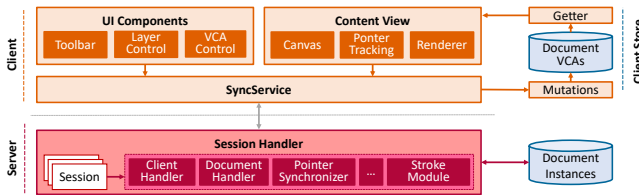

Figure 3: Overview of the client-server architecture used for our system. The server synchronizes and propagates project modifications among clients through different session handlers. A user can modify the project through the GUI components and the content view. The cyclic update process on the client-side prevents version conflicts.

a limited attention-bandwidth and is usually not aware of the changes performed by other users in the document. For example, adding new layers impacts the layer order and often interrupts the user's workflow, and can quickly lead to confusion. To counterbalance this, a document version history is offered that enables users to comprehend the performed changes over time.

Synchronous Editing Conflicts: There are various causes for conflicts in synchronous data editing, e.g., users use the same/different tool on a given layer or a layer is about to be removed that is used by others users. Instead of making tools modal, we choose to raise awareness by indicating that another user is using the same tool or has selected the same data using visual feedback. For this purpose, colored hints (lines) visualize which user(s) currently select a layer and tool respectively (Sec. IV-C3). Moreover, the cursors of all users are depicted in the respective avatar color (Sec. IV-A2).

Asynchronous Editing Conflicts: These conflicts are often caused if several users simultaneously edit the same layer or due to interruptions of unfinished tasks, e.g., a user is interrupted in its current workflow but wants to continue his/her work later on. This can cause conflicts if other users are not aware of this state and meanwhile perform a task on the same data. To approach this, we introduced an *exclusive-lock* functionality for a layer, i.e., a user can forbid editing of a layer for everyone except himself (Sec. IV-C4). To avoid deadlock scenarios, a layer can be *exclusive-unlocked* by others users. In this case, the user who initiates the exclusive-lock is notified accordingly.

IV. SYSTEM OVERVIEW

We develop our system as a Single-Page Application (SPA) that can be used on desktop systems and on mobile devices. It enables sketching, image adjustments, and creation of image collages among multiple clients in real-time. An overview of our system with a depiction of a client to server communication and vice versa is presented in Fig. 2. To achieve this distributed structure of a real-time groupware system, we develop an extensible client-server architecture (Fig. 3). The server is mainly responsible for session handling and synchronization, and maintaining communication among clients (Sec. IV-A). The client transmits and consumes messages (Sec. IV-B) which represent editing actions and perform client-side rendering (Sec. IV-C).

A. Server Components and Functionality

The main task of the server component is to maintain the session documents, manage and provision its state, and handle the communication between the clients.

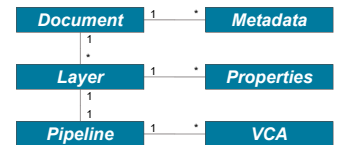*1) Session Document:* The document structure (Fig. 4) is inspired by the OpenRaster file format. A document consists of metadata about the project, e.g., creation date, version, resolution, etc. All used images in a project are stored as layers in an array

Figure 4: High-level structure of a session document comprising multiple layers with multiple VCAs.

in the document. These layers contain property information about the image, e.g., transformations, visibility, etc. Images can be edited using pipelines, i.e., VCAs are applied consecutively. The effect name of each used VCA and its parameters are stored in an array (pipeline) in the layer object. This enables several customization possibilities on a per-layer level and per-VCA level.

2) Session Handling: At the initial state, all stored documents are read and each one is assigned a unique identifier. These sessions contain different handlers for broadcasting client information, e.g., pointer positions and actions, and/or updating the document and notifying clients. When a client connects to the server, the server responds with a session overview. After registering for a session, the user's socket is subscribed to all handlers of the specific session on the server-side. By registering to the server, the client receives its unique server-socket Identifier (ID) that is stored in the local storage of the browser. If the client disconnects, it re-sends its assigned ID when reconnecting to the server and thus is

```
{"module": "drawing",
 "message": {
 "newPath": {
  "timeStamp": "1617804631471",
  "clientId": "m82pY9bvAeIAAAH",
  "color":    "#795EB3",
  "width":    "10",
  "path": [["M",446.99,38],
           ["Q",447,38,448,38],
           ["Q",449,38,449.5,38],
           ["Q",450,38,451,38.5],
           ["Q",452,39,452.5,39],
           ["L",453.01,39]]}}}
```

Listing 1: Exemplary message structure of a newPath action in the drawing module.

recognized again. Moreover, a unique color is assigned to the client, which also serves as the default brush color.

Since several users can work simultaneously in one session, we have a high degree of access-conflict. The server treats incoming changes as "first come, first serve" and, hence, defines the order of updates which is then sent to all subscribed clients. The main logical conflicts are resolved at server-side, e.g., if a user deletes a given layer while another user edits it, the latter change request is dropped. Remaining access-conflicts, which are not mutually exclusive, are then handled at the client-side, i.e., the last executed update will define the modified session state. Thus, session-handling is important to maintain synchronization among clients, a key requirement (Sec. III) for such a system.

### B. Protocol for Client-Server Communication

For the communication between the server and multiple clients, we design a simple protocol that suffices the following requirements: (i) it has a simple yet extensible message structure to facilitate easy development and allows the integration of future features; (ii) it is suitable for fast message (de)serialization to reduce the run-time overhead for clients and server. The clients employ a WebSocket connection to send events to the server, which are then broadcasted to the remaining clients. Both client and server listen for events and process the incoming data accordingly. The sent data includes information about the applied project changes as well as other aspects such as timestamp and client ID that allow for change-history maintenance and enable change traceability among the users. An exemplary message structure (based on JavaScript Object Notation (JSON) standard format) for a `newPath` event is depicted in List. 1. The above allows for efficient communication among clients, which is a necessity (Sec. III-A) for our system.

### C. Client Components and Functionality

The rendering of the raster images is performed entirely on the client-side using WebGL 2.0. The front-end is developed using Vue and the Vuex framework is employed for global storage. Moreover, we make use of Fabric.js to facilitate layer control for canvas rendering.

1) Update Logic: Since all users can potentially work on the session document simultaneously, resulting data-conflicts are required to be handled properly. For it, we propose the communication process among clients as depicted in Fig. 2. After the user changes a parameter, a change-request is sent to the server over a synchronization service. The above service also listens to all change-events from the server. If the request is accepted by the server, it will notify all clients. The service then modifies the store and the GUI is updated accordingly. The user is not allowed to update the local store directly to prevent version conflicts. A small update delay is barely noticeable because of quick socket communication. In case of a parameter-update conflict, when two users update

the same value simultaneously, the last request processed by the server is considered the final version. However, only one user at a time should be able to directly manipulate or transform a layer. For it, during such operations, the layer will be implicitly locked w.r.t. its transformation properties.

2) GUI Structure/Schematics: We assume that the target audience is familiar with some raster-image editing software and therefore decided to re-use GUI concepts from common image-editing applications. Thus, tools such as brush and selection are located in a vertical icon toolbar on the left with additional control parameters on an upper horizontal bar (Fig. 5). The object property panels (e.g., of layers or VCAs) are both located on the right side of the raster graphic. A user can directly interact with the canvas by drawing on a layer or transforming it. Since the image takes up most of the available space for direct editing, the remaining GUI components are arranged compactly with informative icons to ensure intuitive usability. Most applications for raster image editing do not support a responsive design. For smaller screen sizes, e.g., mobile or tablet, this is problematic, because many operations must be clearly represented with large buttons for easy access. Therefore, we hide certain components, which are displayed via a responsive layout if required (Fig. 6). Since the components themselves do not differ between screen sizes, the user can easily switch between desktop and mobile devices without adapting to a new GUI. The generic project tool buttons for downloading the final image ⬇, sharing the project ⤴, or messaging other users working on the project 💬, are placed on top of the editing components.

3) User-specific Visual Feedback: For a coordinated workflow among the clients, the respective selected layer, VCA and the tool of each user is highlighted (Fig. 5b). This allows a user to reproduce canvas changes made by another user. Moreover, this potentially avoids editing conflicts or parameter overwrites as the user can see if someone else has selected the same layer or VCA. Similar to other collaborative web-apps, an overview of currently active users is depicted in the upper right corner. On hovering over the user's icon, the respective username is displayed. We can also get an overview of the user's working area by clicking on the user icon. The cursor position on the canvas is broadcasted to the remaining clients and is displayed with the client's unique color identifier, assigned by the server. The displayed cursor depends on the selected tool, e.g., a pointer or a brush. Additionally, if a user selects or transforms a layer, it is highlighted with the respective client's color. This way, all participants obtain an overview of the active objects of other users.

4) Basic Editing Features: Within a project, layers can be added, deleted, and reordered with simple button clicks in the layer control panel. For each layer, visibility can be set 👁 and the layer itself can be locked/unlocked 🔒. To further enable collaboration we introduce an *exclusive-lock*

(a) Editing Tools and Views

Social and Sharing Tools | Editing Tools | Layer Control | Layer & VCA Properties

(b) User-specific Visual Feedback

User-specific Visual Feedback | Chat Window | Current Work Areas of Users | Cursor Positions of Users
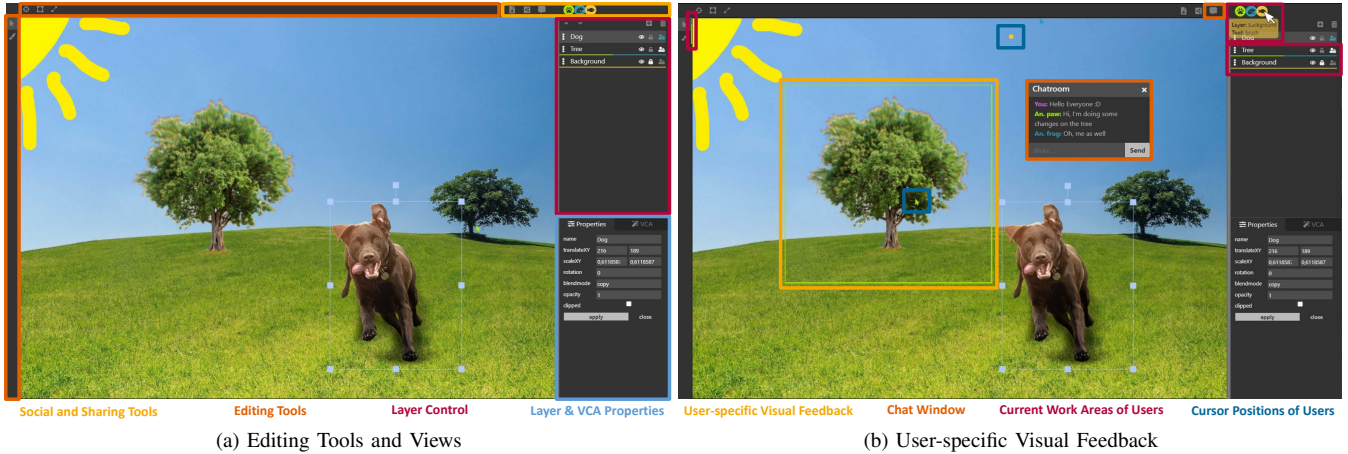
Figure 5: Our GUI provides a variety of (a) editing tools and (b) user specific visual feedback to visually communicate the tool and objects currently operated by other users in order to mitigate the risk for potential editing conflicts.
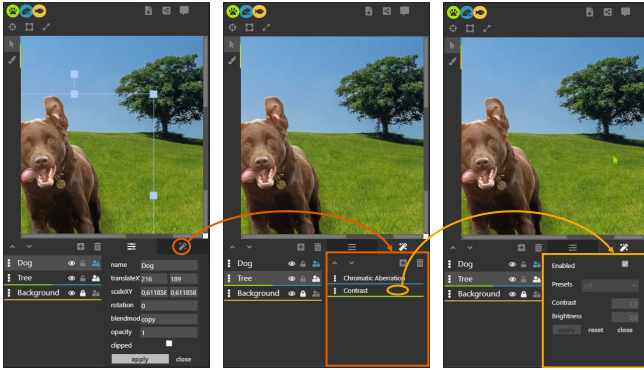

Figure 6: The responsive GUI layout hides editing components if the screen size is too small. The functionality can be easily expanded by pressing the respective button.

button 👥🔒. Analogous to the lock functionality, a user can disable a layer via this button. However, the layer will be locked for everyone except this user. Other users can see who exclusively locked a layer and when. By unlocking this layer, the original user gets a notification. This way, a user can personally lock a layer and signal that he/she does not want interference from other users. Depending on whether the user himself/herself exclusively locked the layer or not, this button is highlighted in a different color. Thus, the user also has a visual overview of which layers he/she is currently working on. Furthermore, for each layer, additional information is displayed in the panel below. A user can switch between the different control settings through tabs, e.g., the layer properties or VCA. Thus, this panel can be easily extended with additional editing features later on by adding new tabs, e.g., viewing the respective layer version history or comments. The main layer properties, e.g., scale, rotation, opacity, are located in the properties tab. In the

VCA tab, the user can add, delete, and reorder VCAs in the pipeline of the layer. Each VCA is adjustable and can be enabled or disabled. All changes are applied to the image in real-time. Depending on the selected tool in the left toolbar, the user has different options to manipulate layers. The corresponding horizontal bar above it is generic and can thus be extended with further tools and settings. So far, the following tools are available:

- **Select:** A layer can be selected and transformed with the respective handles. Additional buttons for resizing and centering facilitate the use.
- **Brush:** One can draw with customizable brush size and color on the selected layer. Performed brush strokes can be undone/redone using the respective tool buttons.

V. POST-DEPLOYMENT USER STUDY

Additional requirements on functionality and user experience are often identified after a prototype is deployed and users have had a chance to try the software and provide feedback. This valuable feedback will be used to improve the future iterations of our prototype. For the post-deployment study, we focused on the following three aspects: (i) do users understand the general structure of the GUI, (ii) do users understand the visualization metaphors to avoid editing conflicts, and (iii) are users satisfied with the prototype.

A. Participants & Apparatus

We recruited 16 volunteers (8 male, 8 female) in 6 different groups. The above participants use our system for the first time and were not part of the preliminary user study to avoid any inherent bias. Each group had a variable number of participants between 2 to 3 and volunteers were aged between the ages 21 and 34. While all of them had experience with computers, 5 had no or only little experience with image

(a) Task-1 Input   (b) Task-2 Input   (c) Task-3 Input

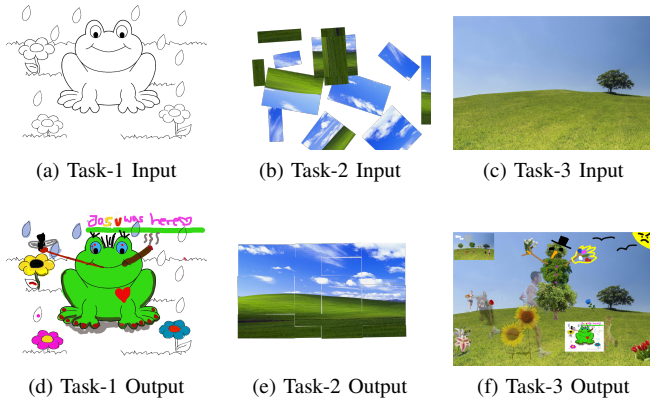(d) Task-1 Output   (e) Task-2 Output   (f) Task-3 Output

Figure 7: Exemplary results obtained with our system during sessions of the post-deployment user study.

editing applications. All of them had normal or corrected-to-normal vision and no known visual impairments. All the participants (except for one, who used an iPad) accessed our SPA on a desktop/laptop system with a single monitor using standard web-browsers (Google Chrome: 7, Mozilla Firefox: 5, Apple Safari: 2, Microsoft Edge: 2) and a computer mouse (two participants used trackpads).

We conducted a supervised/observed study in remote sessions, each with a group of participants. We were connected with them via an online Zoom meeting as they were guided and monitored at the same time. Each session had a length of approx. $60\,\mathrm{min}$ having the following structure. First, each group received a brief introduction into the GUI covering only editing tools as well as layer and VCA controls ($5\,\mathrm{min}$). Following this, each group is asked to collaboratively solve three tasks in sequence.

B. User Tasks

The three tasks performed by each participant group cover the full potential of our editing system. The tasks are ordered by increasing difficulty and took $15\,\mathrm{min}$ to $20\,\mathrm{min}$ respectively for completion. Fig. 7 shows selected results obtained during the study.

Sketching (Task-1): We provide a blank sketch as a background layer (Fig. 7a) and the participants are asked to color the sketch using the brush tool on the empty top layer (e.g., Fig. 7d). The users are encouraged to use multiple brush colors and also create their own doodle using an additional layer. The task objective is to test if users are able to work with layers, use the brush tool effectively, and detect potential synchronous conflicts. We stopped this exercise once the users were familiar with the brush tool and working with layers; this task took $10\,\mathrm{min}$ to $15\,\mathrm{min}$.

Puzzle (Task-2): We provide the users with a set of disarranged pieces of a test image (Fig. 7b). Each piece is represented in the form of a single layer. The task is to rearrange these layers using rotation and translation in order
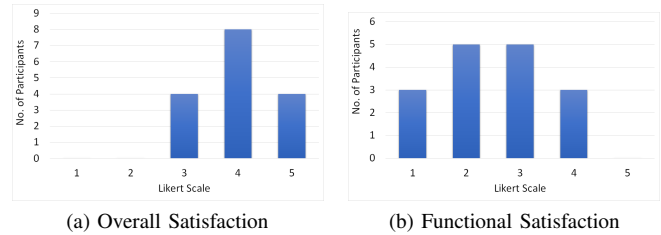

(a) Overall Satisfaction   (b) Functional Satisfaction

Figure 8: The (a) overall and (b) functional satisfaction of the participants during the post-deployment user study on a Likert-Scale of 1 to 5, with 5 being the best.

to solve the puzzle (Fig. 7e). The task objective is to test if users are able to use layer transformation tools effectively. We also provide the puzzling image as a guide. On an average, it took between $15\,\mathrm{min}$ to $20\,\mathrm{min}$ to complete.

Collage Creation (Task-3): Given a set of images, i.e., one background image and various foreground images with alpha matte (Fig. 7c), the users should create and layout respective layers – comprising as many foreground images as they like – in order to create a collage collaboratively. In addition thereto, they are encouraged to apply different image effects (using VCAs such as contrast enhancement, pixelation, chroma-zoom, or chromatic aberration) to each layer. The task objective is to test if users are able to reuse their learning from the previous tasks and also test familiarity with blending and layer modification via VCAs. The time for this task was limited to $15\,\mathrm{min}$.

C. Data Collection and Analysis

The online session of the above tasks is followed by a subjective interview (of approx. $15\,\mathrm{min}$) with questions focusing on performance, collaboration, and potential applications. In addition thereto, the entire online session was video recorded to analyze groups' collaborative practices and also to record their feedback. After the interview, each participant is asked to file a post-study questionnaire based on QUIS and CSUQ without any time constraints.

All the participants were able to perform Task-1 quite easily and were satisfied with the system performance. It indicates that even in the current state our system can be used for a collaborative coloring-book application. For Task-2, the major difficulty was maintaining the control of a particular layer. Participants reported that the user-specific visual feedback regarding layer selection was too subtle. Thus, it happened that two participants were trying to move the same layer and faced unexpected results. However, in the subjective interview, they confirmed that such editing conflicts could have been avoided with the layer locking functionality. For Task-3, the major challenge was in terms of adding effects to layers, most of the users were not able to figure out this functionality on their own. Overall the user feedback can be summarized into the following two categories.

Collaboration: As expected, the novel collaborative aspect of our system was appreciated by most of our participants (Fig. 8a). They showed a great interest in having this collaborative functionality integrated into the image editing tool of their choice. Our participants from different background suggested a broader utility of our system in domains of engineering, architecture, teaching, entertainment, academia, etc., thus indicating a wide user base. However, further improvements for collaboration was suggested mainly in terms of (i) an integrated voice communication channel, (ii) hiding layers created by other team members, and (iii) functionality known from collaborative document editing, e.g., tagging and commenting.

Editing: Our prototype does not offer all the editing functionality generally available in a common image-editing application. Most of the participants who are familiar with such tools noticed the lack of such functionality (Fig. 8b), e.g., an eraser tool, a flood fill tool, or selective layer manipulation (applying VCAs only on a selected region of a layer). However, the integration of collaborative versions of these tools is supported by our architecture.

To answer the initial questions as part of the post-deployment user study: (i) the users understood the general structure of the GUI, (ii) the visualization metaphors, to avoid editing conflicts, were not intuitive in the beginning but were easy to use after guidance, and (iii) users were quite satisfied with our prototype, especially with respect to its collaborative nature.

VI. Conclusions

In this paper, we designed and evaluated a web-based system for real-time collaborative editing of raster images. To the best of our knowledge, ours is the first system that provides such a wide variety of image-edits in a collaborative fashion. In order to better understand the needs for such a system, we conducted a preliminary user study. Our prototype leverages the power of WebGL for interactive browser-based rendering, while synchronization is maintained via WebSocket connections. Our interface re-uses and extends GUI concepts from common image-editing applications. The post-deployment user study indicates a substantial demand for such a system. As part of future work, we would like to address the existing limitations.

Acknowledgments

References

[1] B. R. Lee, "Analysis of digital art content created through collaboration," *Archives of Design Research*, vol. 30, no. 4, pp. 17–25, 2017.

[2] T. Isenberg, "Interactive npar: What type of tools should we create?" in *Proceedings of the Joint Symposium on Computational Aesthetics and Sketch Based Interfaces and Modeling and Non-Photorealistic Animation and Rendering*, ser. Expressive '16. Goslar, DEU: Eurographics Association, 2016, p. 89–96.

[3] W. K. Edwards, "Flexible conflict detection and management in collaborative applications," in *Proceedings of the 10th Annual ACM Symposium on User Interface Software and Technology*, ser. UIST '97. New York, NY, USA: Association for Computing Machinery, 1997, p. 139–148.

[4] C. A. Ellis and S. J. Gibbs, "Concurrency control in groupware systems," in *Proceedings of the 1989 ACM SIGMOD International Conference on Management of Data*, ser. SIGMOD '89. New York, NY, USA: Association for Computing Machinery, 1989, p. 399–407.

[5] C. Sun and D. Chen, "Consistency maintenance in real-time collaborative graphics editing systems," *ACM Trans. Comput.-Hum. Interact.*, vol. 9, no. 1, p. 1–41, Mar. 2002.

[6] J. Heer and M. Agrawala, "Design considerations for collaborative visual analytics," *Information Visualization*, vol. 7, no. 1, pp. 49–62, 2008.

[7] G. Salvati, C. Santoni, V. Tibaldo, and F. Pellacini, "Meshhisto: Collaborative modeling by sharing and retargeting editing histories," *ACM Trans. Graph.*, vol. 34, no. 6, 2015.

[8] C. Calabrese, G. Salvati, M. Tarini, and F. Pellacini, "Csculpt: A system for collaborative sculpting," *ACM Trans. Graph.*, vol. 35, no. 4, 2016.

[9] L. Gao, D. Gao, N. Xiong, and C. Lee, "Cowebdraw: a real-time collaborative graphical editing system supporting multi-clients based on html5," *Multimedia Tools and Applications*, vol. 77, no. 4, pp. 5067–5082, Feb 2018.

[10] C. Wu, L. Li, C. Peng, Y. Wu, N. Xiong, and C. Lee, "Design and analysis of an effective graphics collaborative editing system," *EURASIP Journal on Image and Video Processing*, vol. 2019, no. 1, p. 50, Mar 2019.

[11] J. Zhai, Q. Li, X. Li, and L. Wenyin, "A cooperative image editing tool over mobile phones," in *Proceedings of the 11th International Multimedia Modelling Conference*, ser. MMM '05. USA: IEEE Computer Society, 2005, p. 264–270.

[12] K. Nováková, V. Jakubal, H. Achten, and D. Matejovska, "Collab sketch: Case study on collaborative sketching," in *Fusion - Proceedings of the 31st eCAADe Conference*, 2013, pp. 213–218.