BRNO UNIVERSITY OF TECHNOLOGY
FACULTY OF INFORMATION TECHNOLOGY

Excel@FIT 2026
http://excel.fit.vutbr.cz

Web Application for Image Editing for Technical Documents

Pavol Humený*

Abstract
Images are an integral part of academic texts and significantly contribute to the clarity of presented information; however, their editing is often time-consuming and requires the use of complex tools. Based on these requirements, a web application *Figurio* was designed, enabling simple and fast image editing directly in the browser through a clear and intuitive user interface and specialized tools. The application implements a model based on a sequential list of operations and a layered image representation, allowing the combination of both raster and vector edits while continuously rendering the result. The outcome is a tool that simplifies the image editing process, improves output consistency, and preserves user privacy through local data processing.

*xhumenp00@stud.fit.vut.cz, *Faculty of Information Technology, Brno University of Technology*

1. Introduction

Images play a crucial role in academic texts, as they allow information to be presented more clearly and effectively than text alone. However, their preparation is often time-consuming, as it requires multiple edits performed in complex graphic editors, whose efficient use demands considerable time, which slows down the overall workflow.

This thesis focuses on the design and implementation of a tool for efficient image editing intended for academic texts. The tool should be easy to use, accessible directly in a web browser, and support common operations such as cropping, highlighting, and adding presentation frames. At the same time, it should ensure the protection of processed data and allow export into suitable formats.

Existing solutions include advanced editors such as Adobe Photoshop[1] or GIMP[2], which are unnecessarily complex for simple tasks. Online editors such as Photopea[3], Pixlr[4], Canva[5], or Fotor[6] are more accessible, but they often do not provide adequate tools or support for processing images in PDF format. Current tools therefore fail to effectively combine simplicity, accessibility, and functionality, which are key requirements for this type of use.

The proposed solution is the application *Figurio*, focused on efficient preparation of images for academic texts using tools specifically designed for this purpose. It enables users to perform common editing operations efficiently within a unified environment, eliminating complex workflows while emphasizing clarity, workflow smoothness, and reduced editing time.

The main contributions of the solution include the design of specialized tools such as edge cropping, detail highlighting, area blurring, and adding frames. Emphasis is placed on usability validated through iterative user testing and on local data processing, ensuring user privacy. The solution thus simplifies the preparation of visual materials and improves the quality of the resulting work.

2. Image Data Processing and Architecture

The *Figurio* application is a web-based tool designed for editing images intended for academic texts, accessible directly in a web browser without the need for installation. It focuses on fast execution of typical edits of screenshots and diagrams through a clear and intuitive user interface. The functionality of the application

[1] https://adobe.com/products/photoshop.html
[2] https://gimp.org
[3] https://photopea.com
[4] https://pixlr.com
[5] https://canva.com
[6] https://fotor.com

is designed with an emphasis on minimizing the number of steps required to achieve the desired result.

2.1 Image Loading

The application supports common input formats, including PNG, JPG, WebP, and PDF. After loading a file, its format is automatically detected, and the appropriate processing method is selected accordingly. Raster images are loaded into a `canvas`, while PDF documents are processed using the *pdf.js*[7] library, where a specific page of the document is selected and then loaded as an image (Figure 1).

2.2 Internal Image Representation

The central state of the application is implemented as a *Pinia*[8] store, which maintains the current image, its metadata, and a list of applied operations. This model provides data for both processing and rendering of the image and enables reactive updates of the output.

The resulting image is represented as a combination of multiple layers, enabling independent modification of individual parts (Figure 2):

- **Frame layer** — presentation frame
- **Vector layer** — annotations and text
- **Raster layer** — brush-based drawing
- **Base layer** — original image

2.3 Applying Operations

Individual tools store the definition of an operation together with its parameters. Operations are maintained in a sequential list representing the image processing pipeline. When applying operations, the current image data are loaded, the appropriate processing module is selected based on the operation type, the modification is applied, and the result is stored back. This approach enables re-rendering after each change and easy modification of individual steps.

2.4 Image Export

The user selects the output format and export parameters, after which individual layers are merged into the final image. For raster formats, the `canvas` element is used, while PDF output is generated using the *jsPDF*[9] and *pdf-lib*[10] libraries (Figure 3).

3. Selected Tools

The application provides basic tools for editing images intended for academic texts.

[7] https://mozilla.github.io/pdf.js
[8] https://pinia.vuejs.org/
[9] https://github.com/parallax/jsPDF
[10] https://pdf-lib.js.org/

3.1 Crop

The Crop tool is used to remove unnecessary parts of an image, including its margins. It supports both manual cropping and automatic edge cropping. The *fit crop* function automatically determines a suitable region based on content detection, which utilizes edge detection based on the *Canny* algorithm [1] (Figure 4).

3.2 Magnify Area and Blur Area

These tools are used to highlight details by zooming in or to blur a selected part of an image. Highlighting is implemented by magnifying a specific area, while blurring relies on local filtering applied to the raster representation of the image (Figure 5). Both operations are applied to a defined region and rendered as a separate layer above the image, based on the current raster representation that already includes previously applied modifications.

3.3 Frame

This tool adds a presentation frame to an image, for example in the form of an application window or a mobile device. The frame is generated as a combination of independent vector elements in the SVG format and rendered as a separate layer (Figure 6).

4. Conclusion

The *Figurio* application represents a solution for fast and simple editing of images intended for academic texts, combining specialized tools, a clear user interface, and an architecture based on sequential operation processing and layered representation. The solution enables users to perform typical edits within a unified environment without the need for complex tools, reduces the time required for image editing, and ensures user privacy through local data processing. Future development may focus on extending the functionality and further optimizing performance.

Acknowledgement

I would like to thank the supervisor of my bachelor's thesis, Prof. Ing. Adam Herout, Ph.D., for his professional guidance, valuable advice, and for testing the application throughout the entire development process of this work.

References

[1] John Canny. A computational approach to edge detection. *IEEE transactions on pattern analysis and machine intelligence*, PAMI-8(6):679–698, 1986.