Astrid: An AI-powered Image Editing App

Zhenyu Wei Jiaxin Guo Yujie He Zhengze Hu

Macau University of Science and Technology

Abstract

Currently, the high complexity of editing images on mobile devices prohibits users from modifying images at will. Motivated by this situation, our group has developed a mobile application for image editing that allows users to edit images by simply entering a description. We also integrate super-resolution and steganography, as well as some personalization features, into our app.

Introduction

Our group observed the challenges mobile users experience with editing images. Motivated by the desire to make image editing easy and accessible for everyone, we have created a user-friendly mobile app. Figure 1 summarizes main features of our app.

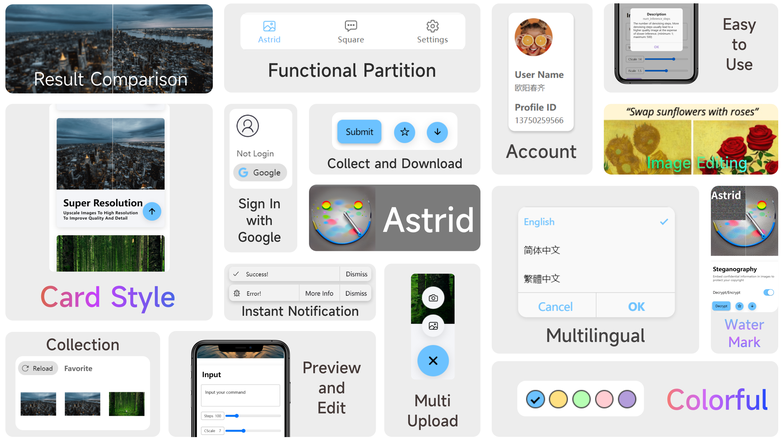

Figure 1. Main features of our app.

Traditional photo editing software can be cumbersome and frustrating to use on a small screen. It often requires a series of technical manipulations that demand advanced skills and can take a long time to complete.

Based on this observation, our group has developed an innovative app that can simplify the image editing process, which allows users to edit images by providing descriptions of the desired effect. Our aim is to provide a user-friendly and accessible platform for anyone to enhance their images on the go. We believe that this app will offer a much more enjoyable experience to mobile users, regardless of their level of technical expertise.

In addition, our app employs algorithms to allow users to super-resolve their images. Furthermore, it has a steganography tool that enables users to embed or extract hidden messages in their images, providing an approach for users to protect their copyright.

Our key contributions are summarized as follows:

- Image tools, including editing, super-resolution and steganography in one app.
- User profiles and customization (language and color themes), as well as a favorite list for the user.

Methods

Some challenges in the process of developing our app, along with their solutions, are listed below.

1. Image Processing Models
   For the model part, the computation burden is a huge challenge for developing and testing. InstructPix2Pix [2] and Real-ESRGAN [3], the models we select for text-based image editing and image super-resolution, are deep learning-based models. Thus their computations cannot finish in an acceptable speed without the help of hardware accelerators (like GPU). Also, InstructPix2Pix consumes 3-4 GB of memory even for a small (512×512) image, and its memory consumption grows quickly as the size of the input image increases.
   To address this challenge, we use a common *ImageService* abstract class for all our image processing services (see Figure 2). And in addition to the 3 services we provide in the app, we implement a *BlurImageService* that acts as a stub module when doing daily development. In this way, developers of our team can conduct coding tasks without having a powerful GPU machine.

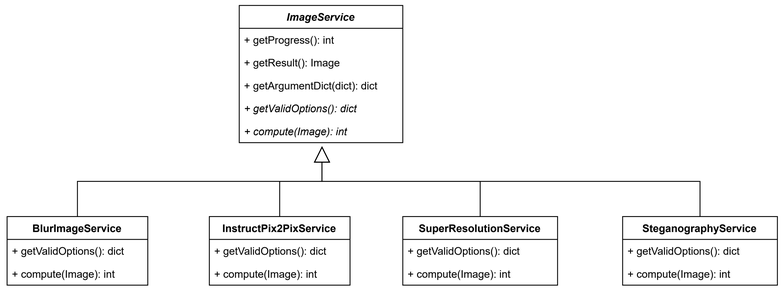

Figure 2. The *ImageService* part of our app's UML class diagram.

2. Frontend UI State Management
   The UI of our app can be categorised as a *multi-tab* design. With a proper division of our features into tabs, most states and actions can be implemented internally inside the source file of each page. However, some information, such as user preferences (language and color mode) and user credential, must be shared across pages.
   It is neither convenient nor safe to store these state information inside one of the tabs. To ease access while avoiding tight coupling, we use *Pinia*, a store library for Vue.js, to provide global getters and setters of properties. It is similar to a singleton design pattern, despite not having an explicit class declaration.
3. Code Reuse
   Our app has multiple image editing tools, which have similar interfaces with different detailed requirements. This leads to a good chance for reusing some code in both frontend and backend, to maintain consistent behaviors across features. For frontend, each image editing tool is implemented as a component, consisting of the card for entering and the main UI. For backend, the inheritance relationship shown in Figure 2 is used to implement different services. Some example code segments are listed below.

```
<Tab1MainCard ... :description='[$t("message.editing") ...' />
<Tab1MainCard ... :description='[$t("message.superResolution") ...' />
<Tab1MainCard ... :description='[$t("message.steganography") ...' />
```

Results

Figure 3 shows our app's user interface for typical use cases. We can see that functions of our app are smoothly connected with proper prompt messages, navigation bars and good response to user operations.

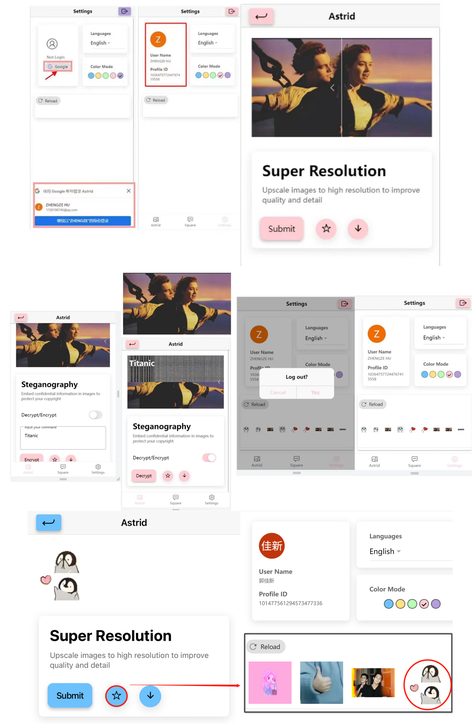

Figure 3. Some screenshots of our app. Above: user logging in (by Google Account) and super-resolution. Middle: steganography, logging out and favorite image list. Below: Adding to favorite by clicking on the star.

Future Work

In the future, more functions, such as *Square* for sharing the user's work, may be added. Besides, the architecture of our app's server can be refactored to increase concurrency. Also, some details for user experiences can be further improved.

References

[1] Fdsig. `https://github.com/MidoriYakumo/FdSig`. Accessed on May 9, 2023.

[2] Tim Brooks, Aleksander Holynski, and Alexei A. Efros. Instructpix2pix: Learning to follow image editing instructions, 2023.

[3] Xintao Wang, Liangbin Xie, Chao Dong, and Ying Shan. Real-esrgan: Training real-world blind super-resolution with pure synthetic data. In *International Conference on Computer Vision Workshops (ICCVW)*, 2021.