Photo Editor Mobile Application

Vaida Diana-Laura
Technical University of
Cluj-Napoca
Email: Vaida.So.Diana@student.utcluj.ro

***Abstract*—Write a short abstract of your planned project.**

I. Introduction

What is the problem/task? Why is it important? Give a context to the problem you are trying to solve. Think of the bigger picture and what your solution is trying to solve. Give a brief overview of the proposed solution. Highlight what your contributions will be, what will be implemented by you.

I want to implement a photo editor application for mobiles using Python and the Kivy framework(for simulating the camera). My application should be capable of processing images from the gallery or captured by the camera. Basic operations: flip horizontally and vertically, resize. The user should be able to apply different filters, to enhance the images by changing its contrast. It is important because with photo editing, there is no limit to the effects that can be brought about in a simple photograph. Every picture you see today is a masterpiece of post processing techniques or simply a result of photo editing. No matter how beautifully or aesthetically a picture is captured, it only shows its full potential when it has been through the photo editing process.

II. Bibliographic study

How did others tackle the problem? Here you can introduce the original paper and compare it to other papers.

Filtering is one of the most basic and common image operations in image processing. You can filter an image to remove noise or to enhance features; the filtered image could be the desired result or just a preprocessing step. Regardless, filtering is an important topic to understand.

Local filtering
The "local" in local filtering simply means that a pixel is adjusted by values in some surrounding neighborhood. These surrounding elements are identified or weighted based on a "footprint", "structuring element", or "kernel".
And for changing the lighting, I want to use as source the link presented in the references.

Lighting
"Lighting" in the context of image processing refers to the manipulation of the brightness or luminance of an image. It involves adjusting the intensity of the light present in the image to enhance visibility or emphasize certain features. This adjustment can be applied uniformly to the entire image or selectively to specific regions. For the lighting I used a Slider (to reajust the lighting on the image).
I also implemented resize, flip horizontally/vertically, inversion from the sources above.

Describe the datasets/libraries you want to use.
The Python libraries I want to use are: OpenCV, numpy.
I used numpy to manipulate arrays after I turned the images into arrays. I especially used OpenCV to read and writemages. I also use numpy to transform arrays into images.

Cite papers that you want to use as references (e.g. He et al. [?]).
-"Hands-On-Image-Processing-with-Python" by Sandipan Dey
-https://github.com/PacktPublishing/Hands-On-Image-Processing-with-Python
-https://medium.com/@parasraorane/image-manipulation-in-python-without-external-libraries-8a7996b1155a
-Algorithms implemented in the lab sesssions (HSV, Grayscale, RGB).
-https://kivy.org/doc/stable/guide/widgets.html
-https://www.tutorialspoint.com/kivy/kivy-camera.htm

III. Method

How did you tackle the problem? Here you describe your method in detail and tell us what exactly you have implemented. If your method is in some parts different from the original paper you can explain that here. Include an overview figure that shows your planned processing pipeline.

For the moment, the user can select from a variety of pictures from a gallery. It can select an image, apply algorithms to it or go back and select another one. There is also an option to open up the camera and apply filters to the image.

My project has the option to open the camera, or select an image from the gallery, pick an image, and go back if I want to the main gallery. The options to apply to the image are: flip horizontally, vertically, invert the image (negative), resize, transform the image to RGB and HSV, apply lighting and make the image in grayscale. Tranformation to HSV, Grayscale, RGB are from the lab, and the others I implemented using my sources. (the ones from above)

Thumbnail Class:

The Thumbnail class inherits from the ButtonBehavior and Image imports from the kivy framework. It represent thumbnails images displayed in the application. Here's a breakdown of its attributes and methods:

Attributes:
image_ path: Path to the image file.

Methods:
__ init __(self, image_path, **kwargs): Constructor method that initializes the thumbnail with the provided image path.

on_press(self): Method triggered when the thumbnail is pressed, which displays the full image associated with the thumbnail.

ImageList Class:

The ImageList class represents the main container for managing and displaying images in the application. Here's an overview of its attributes and methods:
Attributes:
current_image_path: Path to the currently displayed image.

full_image: Instance of the full-size image currently displayed.

red_channel_image_path: Path to the image with only the red channel.

green_channel_image_path: Path to the image with only the green channel.

blue_channel_image_path: Path to the image with only the blue channel.

thumbnail_grid: Grid layout for displaying thumbnail images.

Methods:
load_images(self): Loads thumbnail images from the ”images” directory.

This method is used to load the images from the folder and to show the option of opening the camera.

open_camera(self): Opens the camera view for capturing images.

capture_image(self): Captures an image from the camera and displays it.

The capture_image method is part of a class (likely within a Kivy-based application) that allows users to capture images using the device's camera and then display the captured image within the application.

show_full_image(self, image_path): Displays the full-size image associated with the given image path.

The options after picking a photo are declared here (buttons).

clear_split_channels_images(self): Clears the widgets displaying split channel images.

flip_image(self, direction): Flips the currently displayed image horizontally or vertically.

invert_image(self): Inverts the colors of the currently displayed image.

resize_image(self): Resizes the currently displayed image.

on_lightness_change(self, instance, value): Adjusts the lightness of the currently displayed image.

split_channels_image(self): Splits the currently displayed image into its RGB channels.

The method first loads the image and extracts its dimensions. It then creates three empty arrays to store the red, green, and blue channels separately. The nested loops iterate over each pixel in the image, assigning the corresponding color value to the appropriate channel array while setting the other values to 0.

show_channel_images(self, red_array, green_array, blue_array): Displays the RGB channels as separate images.

load_image_to_array(self, image_path): Loads an image from the specified path into a NumPy array.

save_array_to_image(self, img_array, image_path): Saves a NumPy array representing an image to the specified path.
mirror_h(self, img): Mirrors the given image horizontally.

mirror_v(self, img): Mirrors the given image vertically.

convert_to_grayscale(self): Converts the currently displayed image to grayscale

grayscale(self, img): Converts an RGB image to grayscale.
First, convert the image to RGB and then apply a mathematical formula to get the grayscale version of the image.

convert_to_hsv(self, instance=None): Converts the currently displayed image to HSV color space.

show_hsv_images(self, h_array, s_array, v_array): Displays the H, S, and channels of the HSV image.

ImageViewerApp:
This class starts the application.

IV. Evaluation and Results

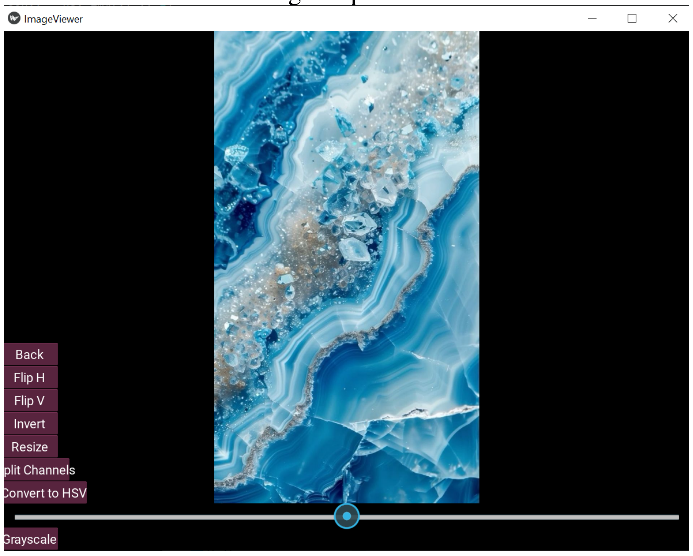
How well does your approach work? Here you show results and explain the experiments you did. Ideally you should have done some evaluation or ablation study that you would present here.
Run the algorithm on a set of images. Include 1-2 visual results. Also, evaluate the algorithm quantitatively (define or search for performance metrics that show how well your algorithm performs).
For many real-world applications, Python's performance is “good enough,” especially when ease of development, readability, and maintainability are prioritized. Many performance-critical applications use Python as a glue language, with performance-intensive parts written in C, C++, or other faster languages.

V. Conclusion

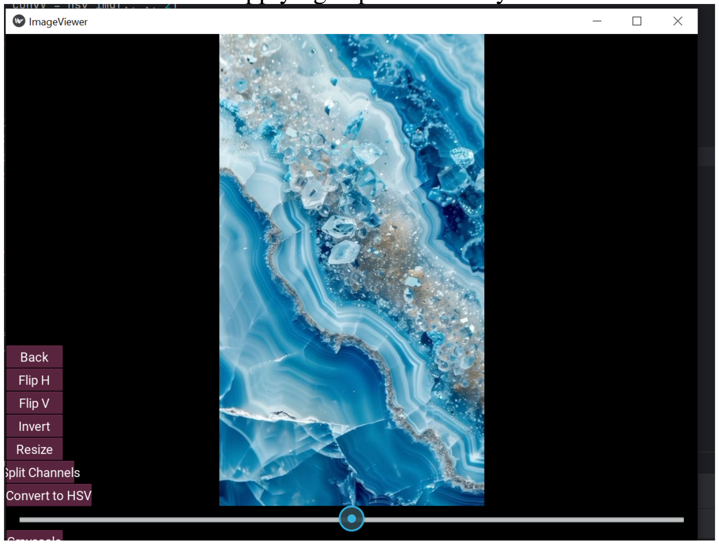
Conclude your work. Restate what problem you have tried to solve, what was original in your work and how well did you manage to achieve the results. You can also suggest future improvements.

I tried to put my project on a mobile phone/emulating it on a phone but I did not succeed. Also when I capture and image, and then go back, it shows the same image again, but the filters are applied on the new image. I didn't manage to achieve the results as much as I wanted. Future improvements: make the project run on a phone, make a collage, make more filters, solve bugs.

Original picture



After applying flip horizontally



After applying grayscale filter

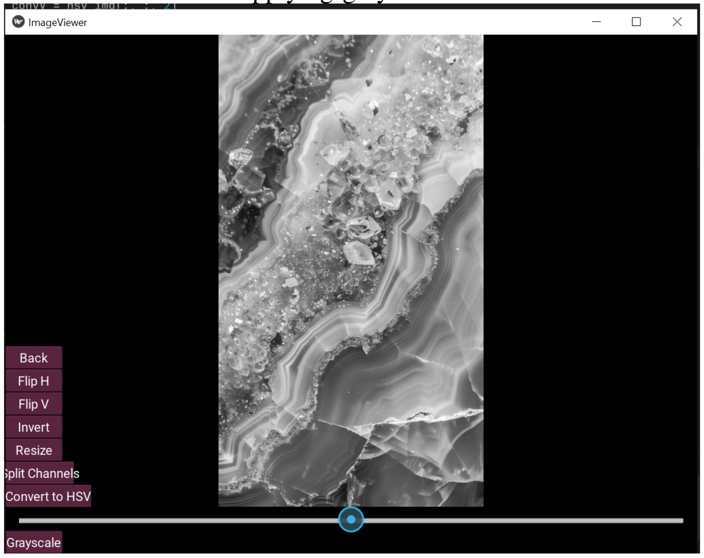


Fig. 1. Caption